

Contents

Contents

Contents

#include <Preprocessor Directive>	1
Operators in C programming	12
What are Operators?	12
Arithmetic Operator In C programming	15
RELATIONAL OPERATORS AND CONTROL STRUCTURE IN C	19
Control Structure	19
1. Sequence Structure.....	19
2. Selection Structure	20
3. Iteration Statement	26
Logical Operators In C Programming	31
Logical OR () operator	32
Logical NOT(!) operator	33
Bitwise Operators In C Programming.....	34
Bitwise AND operator (&).....	35
Bitwise OR operator ()	36
Bitwise XOR(^) operator.....	37
Bitwise Complement Operator (~).....	38
Shift Operators	40
Left Shift Operator (<<).....	41
Right Shift Operator	42
Sizeof Operator.....	43
Break and Continue Statement in C programming.....	45
Break Statement	45
Syntax of break statement	45
continue Statement	48

goto Statement	49
Example	51
Example 1: Program to check Even or Odd	51
Example 2: Program to Check Vowel or consonant	52
Example 3: Finding a largest number.....	52
Example 4 Find all Roots of a Quadratic Equation	53
Example 5 Floyd Triangle	56
Example 6 Lottery Game Part 1	58
Function ရေားပုံ	59
Avoidable Error.....	60
Random Function	61
Example 3	62
Example 4	63
Example 5	66
Example 6	67
Example 7 Dice Game.....	68
Lesson7_1.....	69
အကိုးအကားများ.....	69
Scope Rules(variables).....	69
Variable တစ်ခုရဲ့ scope rule ဆိတာဘာလဲ?	69
Local Variables.....	69
2.Global Variables.....	72
Calling Function Argument.....	74
Example 5 Function တစ်ခုအား value ဖြင့်ခေါ်ဆိုခြင်း	74
Example 6 Function တစ်ခုအား reference ဖြင့်ခေါ်ခြင်း	75
Storage Classes.....	76
1.auto storage class	76

Example 7 using auto Storage class	77
2.extern storage class	78
Example 8 using normal variable and extern storage class	79
Example 9 Using extern storage class with two program file	79
3.Static storage class	80
Different between static variables and normal variables.....	80
4. register storage class.....	81
Array.....	83
One Dimensional Array	83
Two Dimensional Array.....	84
Declaration and Use of Two Dimensional Array.....	84
Example 1 Printing Two Dimensional Array	85
Example 2	86
Example 3	87
Accessing Array	87
Example 4 Accessing Array	88
Limitations Of Array.....	89
Data Structure and Memory allocation	89
For Extra Knowledge.....	90
Static Data Structure and Dynamic Data Structure.....	90
Different Data Type	90
Inserting Data In The Array	90
Example 5 Inserting Data In The Array	91
Example 6 Inserting Data In The Array and change array size	95
Deleting Data In The Array	97
Example 7 Deleting Data in the array.....	97
Example 8 Deleting Data In The Array In Another Way	99
Example 9 array's elements are stored at contiguous locations in C programming	101
Dynamic Memory Allocation in C	102

How Memory Management in C works?.....	102
Malloc or Memory Allocation.....	104
next example program to test.....	105
Adding data to our dynamic memory	105
Free().....	108
Realloc()	109
C File Processing	110
Creating a File.....	111
Modes of file processing	112
Writing to a File	113
Example Programs for Writing to a File.....	113
fputs Example program	115
fprintf example program	116
Reading Data From a File	116
fgetc Sample Program.....	117
fscanf Sample Program	118
Reading Data with Format Sample Program	119
feof function.....	121
Pass by reference in c++	122
Sample program with c	122
Sample Program Using Pointer	123
C Structure	123
Self-Referential Structures	124
Declaring Structure Variables	124
Structure Initialization.....	126
Array as a Structure Variable.....	126
Nested Structure	129
Pointer to a Structure.....	130
Accessing Structure's member using Pointer	131

Using Indirection (*) Operator and Dot (.) Operator	131
Using Arrow Operator (->)	132
Starting C++ Programming	134
Variables	137
What is Variable?.....	137
Variable Declarations and Initialization	138
Fundamental Types	139
Memory Concepts Of Variables.....	142
Class and Object in C++	143
Sample Program : Accessing and Creating Properties.....	144
Sample Program : Private and Public	146
Defining Methods	147
Protected.....	149
Sample program : protected.....	149
Constructor.....	151
Sample program : constructor	151
Type of Constructors.....	153
Parameterized Constructor	153
Copy Constructor	155
Sample Program : Copy Constructor	155
Data Structure And Algorithms	158
What is Data Structure?	158
Famous Data Structures(commonly used)	158
What is an Algorithm?	158
Abstract Data Types (ADTs)	159
STL Containers	160
Vector	161
Example Program: Input Data to Vector and Access element From Vector	161
Vector Functions	163

Sample Program : Begin and End.....	163
Rbegin and Rend Function	164
Sample Program : rbegin() and rend()	165
cbegin() and cend()	165
Sample program : cbegin() and cend()	165
Vector :: insert().....	167
Example Program :: Vector::insert().....	167
Vector :: size() and pop_back()	168
Sample Program :: size() , push_back() , end() , begin() and pop_back().....	168
Vector:: resize() and shrink_to_fit()	169
Sample Program :: resize() and shrink_to_fit()	170
Vector :: emplace() function.....	172
Sample Program :: emplace() function.....	172
Vector :: erase	173
Sample Program :: erase()	173
Linked Lists	174
Singly linked list	175
Searching in Single Linked List.....	196
Doubly Linked List	199
Algorithm For Append Function.....	200
Pointer To Pointer	202
Templates	205
Binary Search	209
Linear Search.....	213
Jump Search	214
Bubble Sort	217
Insertion Sort	220
Selection Sort	221
Merge Sort	224

C-typedef	224
Stack Data Structure	225
Stack Data Structure with LinkedList	233
Queue DS implementation with array	243
Queue implementation with Linked List	253
Dequeue with Linked list	258
TREE	262
Inorder Traversal , Preorder Traversal , Postorder Traversal	266
Exercise	273
Full Binary Tree	274
Perfect Binary Tree	282
Complete Binary Tree	290
Balanced Binary Tree	296
Binary Search Tree ရေးရန်	305
AVL Tree	305
AVL tree အသေးစိတ် ရှင်းလင်းချက်	311
Software installation	316
C programm စတင်ရေးသားခြင်း	322
Project Structure	339
DataBase for Bank	352
Loading From File	355
Space Counter	358
Debugging Space_counter	361
Printing_ALL_Data_From_File	368
Email Validation	371
Email_Exit_Checking	373
Nrc_validation	379

Strong_Password.....	383
Phone_Validation.....	388
Encryption_key.....	390
Login Function	402
User_sector function	404
Money_transcation	408
User_withdraw.....	410
Checking Program.....	414
Error checcking	418
Get_Time	418
Transication_record.....	422
Adding_Time_To_Transication_record.....	427
Integer_To_Char_array.....	430
Time_class.....	435
Amount_transcation_record.....	445
Transcation_amount_limit_per_day	449

```
#include <Preprocessor Directive>
```

C preprocessor ဆိုတာ C programming language နဲ့ ရေးသားထားတဲ့ source code file တွေကို compiler ကနေ compile မလုပ်ခင်မှာ modifies လုပ်ပေးခြင်း ဖြစ်ပါတယ်။ တစ်နည်းပြောရရင် program ကို run တဲ့အခါ execute မလုပ်ခင်မှာ build လုပ်ပေးရပါတယ်။ build လုပ်ခြင်းအောင်မြင်မှသာ run လို့ရမှာ ဖြစ်ပါတယ်။ အဲဒီလို build လုပ်တဲ့နေရာမှာ C Program ရဲ့ ထိပ်ဆုံးမှာ ကြေညာထားတဲ့ header file တွေကို ကျွန်တော်တို့ ရေးသားထားတဲ့ program ထဲကို လာပေါင်းထည့်ပေးခြင်းကို ဆိုလိုတာ ဖြစ်ပါတယ်။
preprocessor ကို နေရာ 3 ခုမှာ အသုံးပြုပါတယ်။

1. directives
2. constants
3. macros

ကျွန်တော် အခုပြောပြုမှာကတော့ directives အကြောင်းဖြစ်ပါတယ်။ (အောက်ပါ program ထဲက line number 4 ကိုကြည့်ပါ)။ Directives တွေက အမြဲတမ်း sharp sign(#) နဲ့ ရေးကြပြီး သူက preprocessor ကို program ထဲမှာ include <stdio.h> ဆိုတဲ့ standard input output header တွေ ပါဝင်မယ်ဆိုပြီး ထွန်ကြားပါတယ်။ ထို <stdio.h> ဆိုတဲ့ header ထဲမှာ ပါဝင်တဲ့ အချက်အလက်တွေကို compiler ကနေ compile လုပ်တဲ့အချင်မှာ လုမ်းခေါ်သုံးပါတယ်။ ဘယ်လို အချက်အလက်တွေလဲ ဆိုတော့ standard input/output library function ဖြစ်သော (printf_s) ဆိုတဲ့ function ကို program ရဲ့ Line number 8 မှာသုံးထားပါတယ်။ line number 5 မှာ ပြောထားတာကတော့ <conio.h> console input/output header ပါဝင်မယ်ဆိုပြီး ထွန်ကြားထားပါတယ်။ သူက line number 9 က _getch() ဆိုတာနဲ့ သက်ဆိုင်ပါတယ်။

line number 6 ကတော့ program ကိုပြန်ဖတ်ရလွယ်ကူအောင် တစ်ကြောင်း ကျော်လိုက်ခြင်း ဖြစ်ပါတယ်။ line number 7 က main function ကတော့ Program ရဲ့ (Entry Point) အစဗ်တော် ဖြစ်ပါတယ်။ program ကိုစတင် run တဲ့အခါ ထို main ကနေ စတင်အလုပ်လုပ်မှာ ဖြစ်ပါတယ်။ အသေးစိတ်ကိုတော့ နောက်သင်ခန်းစာတွေမှာ ဆက်လက် ရှင်းပြပေးပါမယ်။

အပေါ်မှာ ကျန်ခဲ့တဲ့ line number 1 မှာ // ဆိုတဲ့ forward slash လေး ရှိပါတယ်။ single line comment လိုခေါ် ပြီး သူကတော့ ကျွန်တော်တို့ရေးတဲ့ program မှာ comments တွေ ရေးချင်တဲ့အခါမှာ သုံးပါတယ်။ line number 2 ကတော့ forward slash တစ်ခုရယ် (*) asterisk တစ်ခုနဲ့စပြီး အဆုံးမှာ * တစ်ခုနှင့် / တစ်ခုကိုပြန်ပြီး ပိတ်ပေးရပါတယ်။ သူ့ကိုတော့ ကိုယ်ရေးတဲ့ program မှာ comments တွေ အများကြီးရေးချင်တဲ့ အချိန်မှာ သုံးပါတယ်။ သူ့ကို multi-line comments လိုလည်း ခေါ်ပါတယ်။ ကျွန်တော်တို့က program ကို execute လုပ်တဲ့အခါ သို့မဟုတ် run တဲ့အခါမှာ comments တွေကို အလုပ်မလုပ်ပဲ Compiler က (ignore) ကျော်သွားပေးပါတယ်။ ထိုကဲ့သို့ ကျော်သွား ပေးတဲ့အတွက် comments တွေသည် မိမိတို့ program အပေါ် မသက်ရောက်စေပဲ ကိုယ်ပြန်ဖတ်ရတာ လွယ်ကူသလို နောက်လူက ပြန်ဖတ်ရင်လည်း အဆင်ပြေစေမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် ကျွန်တော်တို့ ရေးသားတဲ့ program တွေမှာ comments ရေးထားသင့်ပါတယ်။

```
//single line comments
/*multi-line comments */
#include <stdio.h> //standard input output header
#include <conio.h> // console input/output header

int main()
{
    printf_s ("Welcome From NCC !");
    _getch();
}
```

Simple C program with Integer(int) and Variable (Memory Location)

```

#include<stdio.h>
#include<conio.h>

int main(void) {
    int number1 = 0;
    int number2 = 0;
    int total = 0;
    printf_s("Please enter first number:\n");
    scanf_s("%d", &number1);
    printf_s("Please enter second number:\n");
    scanf_s("%d", &number2);
    total = number1 + number2;
    printf_s("Total number is %d\n",total);

    getch();
    return 0;
}

```

C:\Users\WinHtut_GreenHackers\source

Please enter first number:
15
Please enter second number:
16
Total number is 31

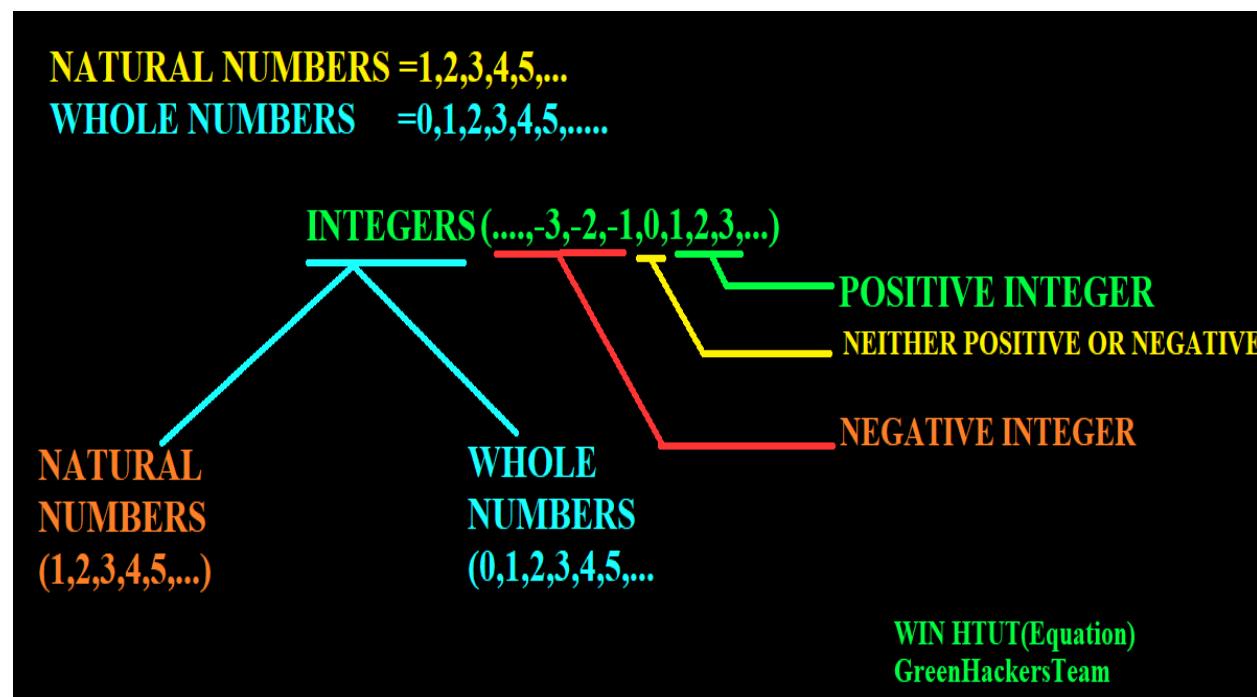
**WinHtut
GreenHackersTeam**

1. **#include <stdio.h>**
2. **#include <conio.h>**
3. **int main(void) {**
4. **int number1 = 0;**
5. **int number2 = 0;**
6. **int total = 0;**
7. **printf_s("Please enter first number:\n");**
8. **scanf_s("%d", &number1);**
9. **printf_s("Please enter second number:\n");**
10. **scanf_s("%d", &number2);**
11. **total = number1 + number2;**
12. **printf_s("Total number is %d\n",total);**
13. **_getch();**
14. **return 0;**
15. }

Line number 1,2,3 က ကျွန်တော် အရင်သင်ခန်းစာများ ရှင်းပြထားပြီးသားဖြစ်တဲ့အတွက် အခု line number 4 ကနေပြီးတော့ စရှင်းပြပါမည်။

Line number 4 မှာ int number1=0; ဆိုပြီးတော့ ရေးထားပါတယ်။ သူမှာ ရှင်းပြရမယ့် အချက် ၂ ချက် ရှိပါတယ်။ int ဆိုတဲ့ integer data type ရယ်၊ number1 ဆိုတဲ့ variable name ရယ် ဖြစ်ပါတယ်။ Integer data type ကို မရှင်းပြခင်မှာ ကျဉ်တော်တို့တွေ တစ်ခုသိထားဖို့ လိုပါတယ်။ အဲဒါကတော့ What are integers?။ Integer ဆိုတာ ဘာလဲပေါ့။

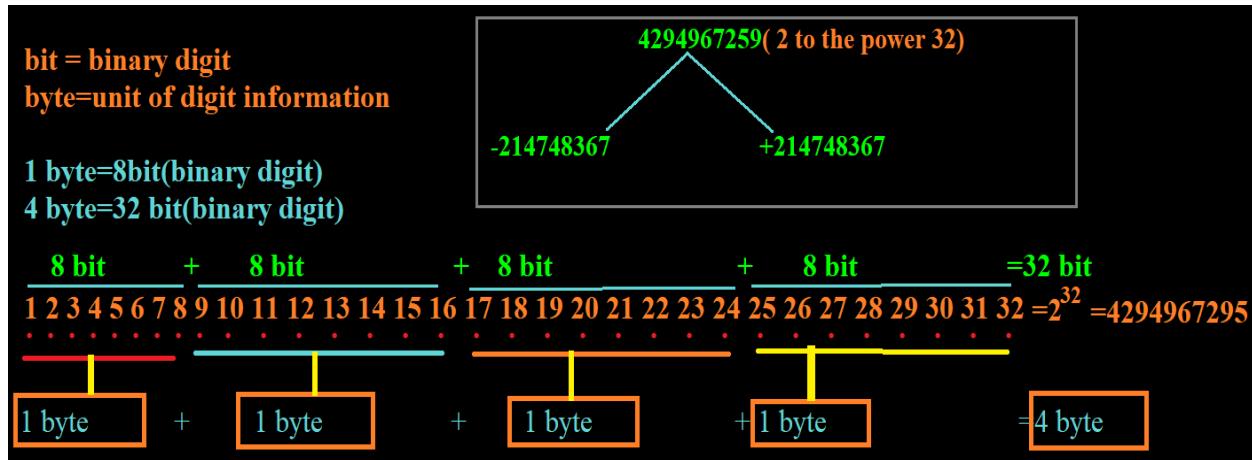
Integer ဆိုတာကို မသိခင် Natural Numbers နဲ့ Whole Numbers တွေကို သိဖို့လိုပါတယ်။ Natural Numbers ဆိုတာကတော့ 1,2,3,4,5,...,etc တို့ ဖြစ်ပါတယ်။ Whole Numbers တွေကတော့ 0,1,2,3,5,..... စသည်တို့ပဲ ဖြစ်ပါတယ်။ integer ဆိုတာကတော့ သူညှာ နှင့် အပေါင်း ကိန်းပြည့်တွေ၊ အနုတ်ကိန်းပြည့်တွေ ပါဝင်မှာ ဖြစ်ပါတယ်။,-3,-2,-1,0,1,2,3,..... စသည်တို့သည် integers များ ဖြစ်ပါတယ်။ ..., -3, -2, -1 စသည်တို့သည် negative integers များ ဖြစ်ပါတယ်။ 1,2,3,... စသည် တို့သည် positive integers များ ဖြစ်ပါတယ်။ Zero (0) ကတော့ Neither Positive Nor Negative ဖြစ်ပါတယ်။ တကယ်လို့ရှုပ်သွားရင် အောက်ကပုံလေးမှာ math concept လေးနဲ့ ဖော်ပြထားပါတယ်။



ဆက်လက်ပြီး Computer programming ဘက်ကနေ သွားပါမယ်.....။ Data type တွေမှာ ဆိုရင် Standard C အရ အခြေခံ data type လေးခုရှိပါတယ်။ int, char, float, double တို့ဖြစ်ပါတယ်။ data

type 4 မျိုးမှာ သူတို့၏ ကိုယ်ပိုင်ကိုင်တွယ်ပြီး သိလောင်တွက်ချက်နှင့်တဲ့ ပမာဏတန်ဖိုးရှိပါတယ်။ အောက်ကပိုလေးကို ကြည့်ပါ။

Integer data type ကို မရင်းပြခင်မှာ bit နဲ့ byte ကိုကဲ့ပြားစွာ သိထားဖို့လိုပါတယ်။ bit ဆိုတာက binary



digit ကို ပြောတာဖြစ်ပြီး binary ဆိုတာကတော့ 0 နှင့် 1 ကိုနဲ့နှစ်လုံးသာ ပါဝင်တဲ့ base 2 ရှိတဲ့ numbering system ကိုပြောတာဖြစ်ပါတယ်။ byte ကတော့ unit of digit Information ကိုပြောတာဖြစ်ပါတယ်။ သူကတော့ digit တွေရဲ့ unit ပါ။ 1 byte မှာ 8 bit ရှိပါတယ်။ ကျွန်ုတ်တို့ အခုပြောသွားမှာက 32 bit ပါ။ 32 bit ဆိုတာ 2 to the power 32 ကိုပြောတာပါ (ပုံထဲမှာကြည့်ပါ)။ တဲ့ အဲဒီတော့ 2 to the power 32 ရဲ့တန်ဖိုးဟာ 4294967295 ဖြစ်ပါတယ်။ ဒါပေမယ့် ကျွန်ုတ်တို့က သူကို အပေါင်းကိန်းရယ် အနုတ်ကိန်းရယ် ယူချင်တာပါ။ ထို့ကြောင့် 32 ထဲကနေ 1 နှုတ်ပေးရပါတယ်။ အဲဒီဆိုရင် 2 power က 31 ဖြစ်သွားပြီး သူတန်ဖိုးက 214748367 ဖြစ်ပါတယ်။

Positive integer ရယ် negative integer ရယ်ပြန်ခဲ့တဲ့အချိန်မှာ (Positive integer = +214748367) ဖြစ်ပြီးတော့ (Negative integer = -214748367) ဖြစ်သွားပါတယ်။ အခုဆိုရင် ကျွန်ုတ်တို့ Integer data type ထဲမှာ အပေါင်းကိန်း +214748367 အထိ သိလောင်နှင့်သွားပြီး အနုတ်ကိန်း -214748367 ထိ (From -214748367 to +214748367) သိလောင်ကိုင် တွယ်နှင့်သွားမှာ ဖြစ်ပါတယ်။ ဒါဆိုရင် ပုံလေးနဲ့ တွဲကြည့်ပြီး 32 ထဲကနေ 1 ကို ဘာကြောင့် နှုတ်လိုက်ရတယ် ဆိုတာကို နားလည်လိမ့်မယ်လို့ မျှော်လင့်ပါတယ်။ ဥပမာ- ပေးရရင် int ဆိုတဲ့ ဂျက်ဖော်မှူး ထဲမှာ စာလုံးပေါင်း -214748367 ကနေ +214748367 အထိ နှစ်ခုပေါင်း 4294967295 လောက်ကို ထည့်ထားနိုင်တယ်လို့ ဆိုလိုချင်တာ ဖြစ်ပါတယ်။

အထက်မှာ ရှင်းပြထားတာက 32 bit က int data type ကိုပဲ ရှင်းပြပေးထားတာ ဖြစ်ပါတယ်။ 64 bit မှာ 2 power 64 ဆိုရင်တော့ သူတန်ဖိုးက 18,446,744,073,709,551,616 ပါ။ အပေါင်းကိန်းရယ် အနုတ်ကိန်းရယ် ပြန်လုပ်ဖို့ 2 to the power 63 ကတော့ 9,223,372,036,854,775,808 ဖြစ်ပါတယ်။ negative integer -9,223,372,036,854,775,808 ကနေ +9,223,372,036,854,775,808 အထိ အားလုံးပေါင်း 18,446,744,073,709,551,616 ကို ကိုင်တွယ် သို့လောင်နိုင်သွားမှာ ဖြစ်ပါတယ်။ ဒါဟာ data type တွေ အများကြီးရှိတဲ့ထဲက integer data type အကြောင်းကို အကြမ်းဖျင်း ပြောပြတာပါ။ အခုဆက်လက်ပြီး variable ဆိုတာကို ပြောပြပါမယ်။

variable ဆိုတာ memory ထဲမှာရှိတဲ့ location တစ်ခုကိုလှမ်းပြီး ညွှန်ပြထားတာ ဖြစ်ပါတယ်။ အဲဒီ location ထဲမှာ သူနဲ့သက်ဆိုင်တဲ့ value တစ်ခုကို သိမ်းထားတာ ဖြစ်ပါတယ်။ အဲဒီ သိမ်းထားတဲ့ value ကိုမှ ကျွန်တော်တို့ program အလုပ်လုပ်တဲ့အချိန်မှာ ခေါ်သုံးမှာ ဖြစ်ပါတယ်။

The screenshot shows a debugger interface with three main panes:

- Memory 1**: Shows a memory dump starting at address 0x0116FE74. A yellow arrow points from the address 00 00 00 00 in the dump to the variable declaration in the source code.
- Source.cpp**: Displays the C++ source code for a program that adds two numbers. A red arrow points from the variable declaration `int number1 = 0;` to the corresponding memory address in the dump.
- Watch 1**: Shows a table with one entry: `number1` has a value of `0` and a type of `int`.

Annotations in the screenshot:

- "ဒါကတော့ number1 ဆိုတဲ့ variable ရဲ့ memory address ပါ"
- "00 00 00 00 ဆိုတာ ကတော့ သူရဲ့ value ပါ"

ပုံမှာ ကြည့်မယ်ဆိုရင် မြေးအဝါနဲ့ ပြထားတဲ့ 0x0116FE74 ဆိုတာ number1 ဆိုတဲ့ Variable name ရဲ့ memory address ပါ။ မြေးအနီရောင်နဲ့ ပြထားတာကတော့ သူရဲ့ value ဖြစ်ပါတယ်။

Variable တွေမှာ အချက် (၃) ချက် မှတ်ထားရပါမယ်။ အဲဒါကတော့ name ရယ်၊ type ရယ်၊ value ရယ် ဖြစ်ပါတယ်။ line number 5 မှာ int number1 = 0; မှာ number1 ဆိုတာက variable name ဖြစ်ပါတယ်။ integer data type နဲ့ number1 ဆိုတဲ့ variable တစ်ခုကို ကြော်လှားခြင်း ဖြစ်ပါတယ်။ 0 ဆိုတာကတော့ number1 ဆိုတဲ့ variable ရဲ့ value ဖြစ်ပါတယ်။ line number 5 and 6 မှာ ကျဉ်းတော်တို့ Program ထဲမှာ သုံးချင်တဲ့ variable ကို သုံးချင်တဲ့ data type နဲ့ ကြော်လှားပါတယ်။ တစ်ခုမှတ်ထား ရမှာက program ထဲမှာ ခေါ်မသုံးခင် ကိုယ်သုံးချင်တဲ့ variable တွေကို ကြိုပြီး ကြော်လှားရမှာ ဖြစ်ပါတယ်။

ကျဉ်းတော်တို့ variable တွေကို int number1 = 0; and int number2 = 0; လို့မကြော်လှာဘဲ int number1, number2; ဆိုပြီးတော့လဲ ကြော်လှားလို့ ရပါတယ်။ ဒါပေမယ့် ကျဉ်းတော်တို့တွေ int number1=0; လို့ မကြော်လှားခဲ့ဘူးဆိုရင် int number1; ဆိုပြီး ရှိုးရှိုးပဲကြော်လှာခဲ့ရင် visual studio နဲ့ watch လုပ်ကြည့်တဲ့အချိန်မှာ သူ့ထဲမှာ ဘယ်ကမှန်းမသိတဲ့ random value တွေ ဝင်နေတာကို တွေ့ရပါတယ်။ ဒါကတော့ မိမိတို့ သဘောအတိုင်း အဆင်ပြုသလို ကြော်လှု့ ရပါတယ်။ ကိုယ်တိုင်လေ့လာပြီး အသုံးပြုတာကတော့ အကောင်းဆုံးဖြစ်မှာပါ။

အပေါ်က program လေးကို စပြီးတော့ ရှင်းကြရအောင်...။

3. int number1 = 0;
4. int number2 = 0;
5. int total = 0;

integer data type တွေနဲ့ variable တွေ ကြော်လှားလိုက်ပါတယ်။ အဲဒါ variable တွေကိုမှာ program ထဲမှာ ခေါ်သုံးမှာ ဖြစ်ပါတယ်။

8. printf_s("Please enter first number:\n");
9. scanf_s("%d", &number1);
10. printf_s("Please enter second number:\n");
11. scanf_s("%d", &number2);

Number 8 ကတော့ printf_s ဆိုတဲ့ function ကိုသုံးပြီး user ဆိုကို Screen ပေါ်မှာ ပေါ်အောင် please enter first number: ဆိုပြီး message တစ်ကြောင်းရေးခိုင်းလိုက်တာပါ။ prompt လုပ်တယ်လို့လဲ ခေါ်ပါတယ်။

Please enter a number: ဆိုတဲ့နောက်မှာ \n ဆိုတာ ပါ,ပါသေးတယ်။ \n ဆိုတာ ကိုသိဖို့ ဆိုရင် escape sequence ဆိုတာကို သိဖို့လိုပါတယ်။ c programming မှာဆို အခြေခံအနေနဲ့ escape sequence (၅) မျိုး ရှိပါတယ်။

\n ဆိုတာကတော့ new line ပါ။ ဆိုလိုချင်တာကတော့ program ထဲမှာ \n ကို တွေ့တာနဲ့ အောက်ကို တစ်ကြောင်း ဆင်းသွားမှာ ဖြစ်ပါတယ်။

\t ဆိုတာကတော့ Horizontal tab ပါ။ ဆိုလိုချင်တာကတော့ စာကြောင်းထဲမှာ tab တစ်ချက် ခုနှစ်ခုနှင့် ပြောချင်တာပါ။ အောက်ကပုံမှာ ကြည့်ပေးပါ။

```

int total = 0;
printf_s("Please enter first\t number:\n");
scanf_s("%d", &number1);
printf_s("Please enter second number:\n");
scanf_s("%d", &number2);
total = number1 + number2;
printf_s("Total number is %d\n", total);

_getch();
return 0;

```

C:\Users\WinHtut_GreenHackers\source\r

Please enter first number:
\t ကြောင့် tab ခုနှစ်ခုနှင့် ပြောချင်တာပါ

```
(Global Scope)
int total = 0;
printf_s("Please enter first\t number:\n");
scanf_s("%d", &number1);
printf_s("Please enter second\\number:\n");
scanf_s("%d", &number2);
total = number1 + number2;
printf_s("Total number is %d\n",total);

_getch();
return 0;

_getch();
return 0;
```

C:\Users\WinHtut_GreenHackers\source\repos

Please enter first number:
5
Please enter second\\number:

backslash လေးပါလာတာကို တွေ့ရပါမယ်

အနားလေး ရောက်ရင် ကွန်ပူးတာက အသံ တစ်ခု မြှည်ပါလိမ့်မယ်

\a ဆိုတာကတော့ Alert ပါ။ ဆိုလိုချင်တာကတော့ အသံ သို့မဟုတ် သိသာတဲ့ အရာတစ်ခုခုနဲ့ ရောက်နေတဲ့ cursor နေရာမှာ alert လုပ်တာပါ။ အောက်ကပုံမှာ ကြည့်လို့ရပါတယ်။

\\" ဆိုတာကတော့ Backslash ပါ။ ဆိုလိုချင်တာကတော့ ကျွန်တော်တို့ စာကြောင်းထဲမှာ \ ဆိုတဲ့ character ကို ထည့်မယ်လို့ ပြောတာပါ။ အောက်ကပုံမှာ ပြထားပါတယ်။

```
(Global Scope)
int total = 0;
printf_s("Please enter first\t number:\n");
scanf_s("%d", &number1);
printf_s("Please enter second\\*number:\n");
scanf_s("%d", &number2);
total = number1 + number2;
printf_s("Total number is %d\n",total);

_getch();
return 0;

_getch();
return 0;
```

C:\Users\WinHtut_GreenHackers\source\repos\3\|

Please enter first number:
5
Please enter second*number:

* လေးပေါ်လာတာကို သတိပြုမှာပါ

* ဆိုတာကတော့ ကျွန်တော်တို့စာကြောင်းထဲမှာ * ဆိုတဲ့ character ကို ထည့်မယ်လို့ ပြောတာပါ။ အောက်ကပုံမှာ ကြည့်ပါ။

```

int total = 0;
printf_s("Please enter first\t number:\n");
scanf_s("%d", &number1);
printf_s("Please enter second\n*number:\n");
scanf_s("%d", &number2);
total = number1 + number2;
printf_s("Total number is %d\n", total);

_getch();
return 0;

```

The screenshot shows a C code editor with the following code:

```

int total = 0;
printf_s("Please enter first\t number:\n");
scanf_s("%d", &number1);
printf_s("Please enter second\n*number:\n");
scanf_s("%d", &number2);
total = number1 + number2;
printf_s("Total number is %d\n", total);

_getch();
return 0;

```

The output window shows the following interaction:

```

Please enter first      number:
5
Please enter second*number:
* လေးပေါ်လာတာကို သတိပြုမှာပါ

```

An orange arrow points from the line `scanf_s("%d", &number2);` to the asterisk (*) in the output.

ဒီလောက်ဆိုရင် c programming ရဲ့ escape sequence အကြောင်းကို အနည်းငယ် သိသွားမယ်လို့
ထင်ပါတယ်။ ကျွန်တော်တို့ program ကို ဆက်သွားရအောင်။

line number 8 က နောက်ဆုံး \n ကို တွေ့တာနဲ့ Cursor က အောက်တစ်ကြောင်းကို ဆင်းသွားပါတယ်။
ဒီနေရာမှာ သတိတစ်ခုထားရမှာက ကျွန်တော်တို့ prompt လုပ်ချင်တဲ့စာကြောင်း အစမှာရော
အဆုံးမှာရော double quote (".....") ကိုထည့်ဖို့ မမေ့ပါနဲ့။ ပြီးရင်တော့ Line အဆုံးမှာ (;) semicolon
ထည့်ပေးရပါတယ်။

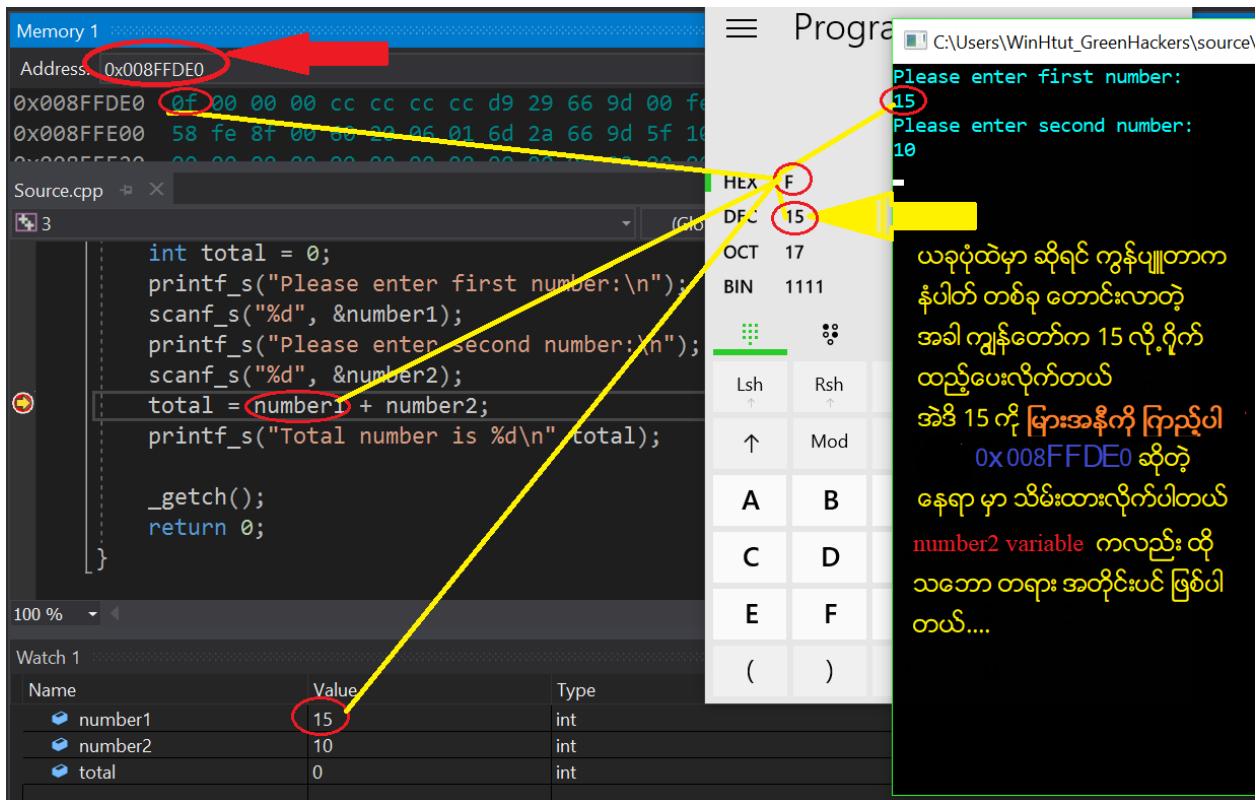
9 `scanf_s("%d", &number1);`

Line number 9 မှာ `scanf_s` ဆိုတာကတော့ scan formatted ကိုဆိုလိုတာပါ။ `scanf_s` ဆိုတဲ့ function
မှာ argument နှစ်ခုပါပါတယ်။ ပထမ argument ကတော့ %d ဒုတိယတစ်ခုကတော့ ampersand (&)
သူကို address operator လိုလဲ ခေါ်ပါတယ်။ သူက user ဆီကနေပြီး ထည့်လိုက်တဲ့တန်ဖိုး (standard
input, eg- keyboard) ကို ဖတ်လိုက်တာပါ။ %d ဆိုတာ ကတော့ integer နှဖတ်မယ်လို့ ပြောတာပါ။ d
ဆိုတာကတော့ decimal ကိုပြောချင်တာ ဖြစ်ပါတယ်။ သူကိုလည်း double quote လေးတွေထဲ
ထည့်ပါ။ ပြီးရင်တော့ (,) လေးထည့်ဖို့ မမေ့ပါနဲ့။ နောက်တစ်ခုကတော့ &number1 (address of
number1) သူပြောချင်တာက user ဆီက ရှိက်ထည့်လိုက်တဲ့ တန်ဖိုးကို %d ဆိုတဲ့ integer နဲ့ ဖတ်ပြီး
number1 ဆိုတဲ့ Address ရှိတဲ့နေရာမှာ သိမ်းမယ်လို့ ပြောလိုက်တာပါ။

အခုဆိုရင် line number 8 and 9 ကို နားလည်သွားလိမ့်မယ်လို့ ထင်ပါတယ်။ အခုဆက်ဖတ်လိုက်ရင် 10
and 11 ကိုလည်း နားလည်သွားပါလိမ့်မယ်။

10 `printf_s("Please enter second number:\n");`

11 `scanf_s("%d", &number2);`

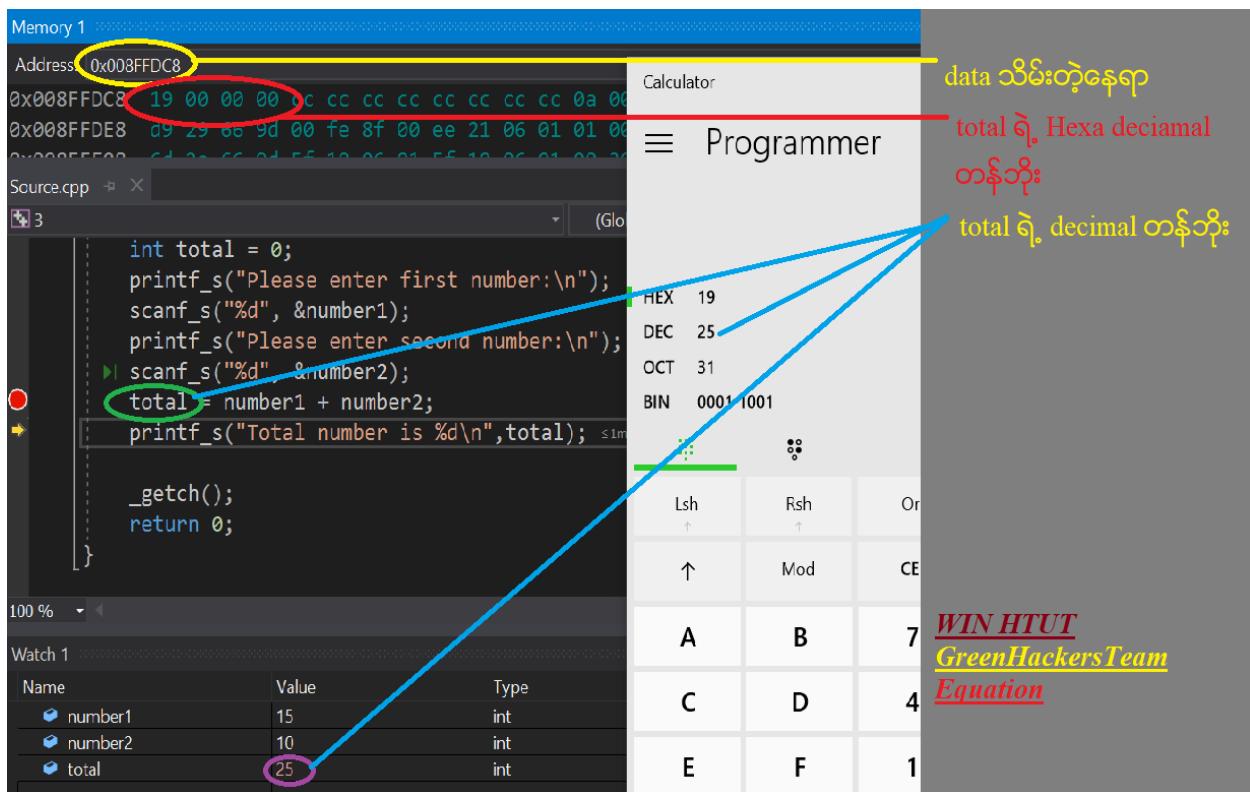


အထက်မှာ ပြထားတဲ့ပုံကတော့ ကျွန်တော်တို့ ပေးလိုက်တဲ့ number ကို Hexadecimal value နဲ့ ဖော်ပြထားတာဖြစ်ပါတယ်။ မြားအနီရောင်နဲ့ ပြထားတဲ့ 0x008FFDE0 ဆိုတာကတော့ သူရဲ့ address ဖြစ်ပါတယ်။

12 `total = number1 + number2;`

13 `printf_s("Total number is %d\n",total);`

Line number 12 ကတော့ `number1 + number2` ပေါင်းလိုက်လာတဲ့ တန်ဖိုးကို assignment operators ဖြစ်တဲ့ (=) ကိုသုံးပြီး `total` ဆိုတဲ့ variable ထဲကို ထည့်လိုက်တာ ဖြစ်ပါတယ်။ ပြီးရင်တော့ line number 13 မှာ `total` ရဲ့ value ကို prompt လုပ်လိုက်ပါတော့တယ်။ အောက်ကပုံတွင် ကြည့်ပါ။



Operators in C programming

What are Operators?

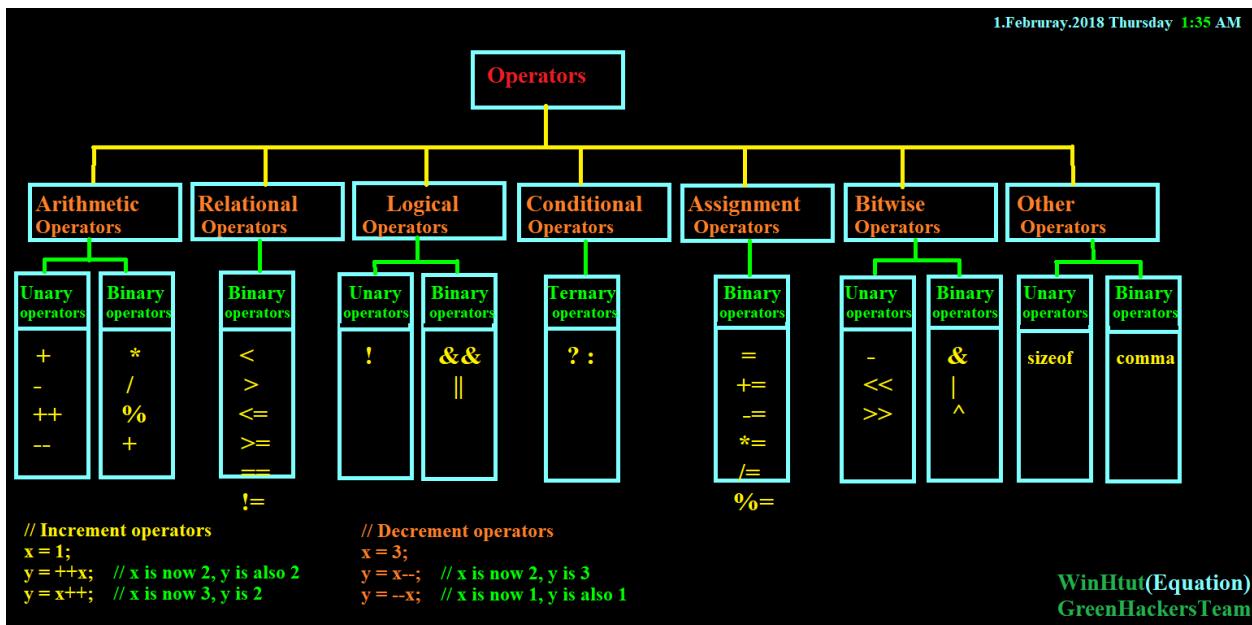
Operators တွေဆိုတာ သက်တလေးတွေပါ ။ ဘယ်လို သက်တလေးတွဲလဲ ဆိုတော့ compiler က mathematical ဒါမ္မဟုတ် logical နဲ့ဆိုင်တဲ့ တွက်ချက်မှုတွေကို တိကျသေချာစွာ လုပ်နိုင်အောင် သူတို့က ကူညီပေးပါတယ်။ အဲတော့ program data တွေ၊ variable တွေကို တိကျစွာ ကိုင်တွယ်နိုင်ဖို့ operator တွေကို သုံးပါတယ်။ Operators က program တစ်ပုဒ်မှာ အမြဲတမ်းလိုလို ပါတတ်ကြပါတယ်။ C Programming Language မှာလည်း operators တွေ အများကြီး ပါဝင်ပါတယ်။

C language ရဲ့ Operators တွေကို ဖော်ပြုပေးပါမယ်။

1. Arithmetic operators
2. Relational operators
3. Logical operators
4. Assignment operators
5. Increments and Decrement operators
6. Conditional operators
7. Bitwise operators

8. Special operators တို့ ဖြစ်ပါတယ်။

အထက်ပါ Operators တွေအကြောင်းကို မရှင်းပြခင် Operators တွေအကြောင်းကို နောက်ထပ် အနည်းငယ် သိတော်းသင့်တာလေး ရှိပါသေးတယ်။ အဲဒါကတော့ Operator တွေမှာ Assignment operator, conditional operator နှင့် Relational operator တွေကလဲပြီး တစ်ခြားကျွန်တဲ့ operators တွေမှာ Unary Operation နဲ့ Binary Operation ဆိုပြီး နှစ်ခုရှိပါသေးတယ်။ အောက်က ပုံလေးမှာ ကြည့်ပါ။



ကျွန်တော်တို့ ဒီနောက် တစ်ခုသတိတော်းရမှာက binary operator ဆိုတာ element နှစ်ခု ပါတယ်လို့ ရည်ညွှန်းချင်တာပါ။ base two ဖြစ်တဲ့ binary number တွေကို ဆိုလိုချင်တာ မဟုတ်ပါဘူး။ operand နှစ်ခုနဲ့ အလုပ်လုပ်တယ်လို့ ဆိုလိုချင်တာပါ။

ဥပမာ - ကျဉ်တော်တို့ Arithmetic Operators တဲ့ binary operation ဖြစ်တဲ့ (*) Multiplication (မြောက်ခြင်း) မှာဆိုရင် operand နှစ်ခုနဲ့ အလုပ်လုပ်ပါတယ်။ အောက်ကပုံမှာ တစ်ချက်ကြည့်ပေးပါ။

```

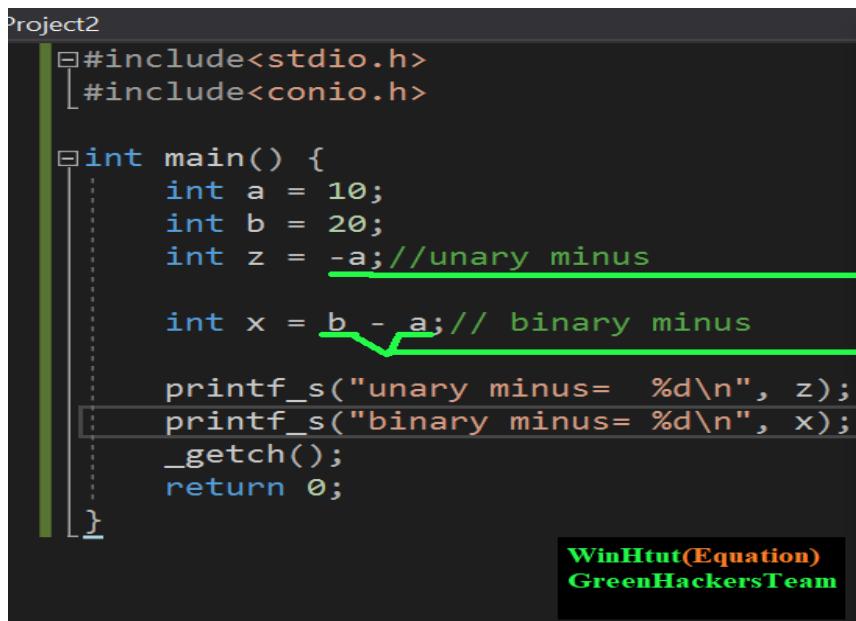
Project2                                (Global Scope)
#include<stdio.h>
#include<conio.h>

int main() {
    int a = 15;
    int b = 10;
    int z = a * b;                         binary operation
    printf_s("a and b multi is %d", z);   a နှင့် b နှစ်ခုတို့ operand နှစ်ခုလို့.
    getch();
    return 0;
}

```

WinHtut(Equation)
GreenHackersTeam

ဒါဆိုရင် binary operation ဆိုတာ ဘာကိုပြောချင်တာလဲ သဘောပေါက်သွားလိမ့်မယ်လို့ ထင်ပါတယ်။ အခုဆက်လက်ပြီး unary operation အကြောင်းကို ပြောပြပါမယ်။ Arithmetic operator တဲ့ unary operation မှ (-) unary minus operator နဲ့ binary minus operator ကို အောက်ကပုံထဲမှာ ကြည့်လိုက်ရင် ရှင်းသွားမယ်လို့ ထင်ပါတယ်။ Binary ၁ ၏ operand နှစ်ခု ပေါ်ပါမယ်။ unary ၁ ၏ operand တစ်ခုနဲ့ operation လုပ်တယ် ဆိုတာကိုတော့ မှတ်ထားဖို့ လိုပါမယ်။



```

Project2
#include<stdio.h>
#include<conio.h>

int main() {
    int a = 10;
    int b = 20;
    int z = -a;//unary minus

    int x = b - a;// binary minus

    printf_s("unary minus= %d\n", z);
    printf_s("binary minus= %d\n", x);
    getch();
    return 0;
}

```

WinHut(Equation)
GreenHackersTeam

ဥပမာ- Unary minus operator တစ်ခုအနေနဲ့

A = -6 ;

B = 7

C = a - b;

C = -13

အခုခံရင် unary minus နဲ့ binary minus အကြောင်းကို အနည်းငယ်သိသွားမယ်လို့ မျှော်လင့်ပါတယ်။

[Arithmetic Operator In C programming](#)

Arithmetic operator ဆိတာကာ

+ (addition) ပေါင်းခြင်း

- (subtraction) နှုတ်ခြင်း

* (Multiplication) ငြောက်ခြင်း

/ (Division) စားခြင်း

% (Remainder) အကြောင်း တိုဖြစ်ပါတယ်။

ကျွန်တော်တို့ တစ်ခုသတိထားရမှာက programming မှာ equation (=) တစ်ခုရေးတော့ မယ်ဆိုရင် Algebraic (သချုပ်အကွဲပွဲ) ရေးသလို ရေးလိုမရပါဘူး။ ဥပမာ a နဲ့ b ငြောက်မယ်ဆိတာကို c = ab လို့ ရေးလိုမရပါဘူး။ c = a*b လို့ ရေးမှသာ မှန်ပါတယ်။ (*) ကို asterisk လို့ ခေါ်ပါတယ်။

ဥပမာ(2) - A နဲ့ B ကို စားတွေ့မယ်ဆိုရင်လည်း ရှိုးရိုးသချို့မှာလို့ A%B ဆိုပြီး ရေးလို့ မရပါဘူး။ ကျွန်တော်တို့ programming မှာ အားလုံးကို straight-line form နဲ့ပဲ ရေးပါတယ်။ ဒါကြောင့် a/b ဆိုပြီးတော့ ရေးရပါတယ်။ အောက်ကပ်ကတော့ Arithmetic Operators တွေကိုမှာ Algebraic (သချို့အကွ္ခရာ) ပုံစံ နဲ့ဖော်ပြထားတာ ဖြစ်ပါတယ်။

```

int main() {
    int a = 10;
    int b = 5;
    int x, y, z, i, j;
    x = a + b;
    y = a - b;
    z = a * b;
    i = a / b;
    j = a % b;

    printf_s(" a and b addition is = %d\n\n", x);
    printf_s(" a and b subtraction is = %d\n\n", y);
    printf_s(" a and b multiplication is = %d\n\n", z);
    printf_s(" a and b division  is = %d\n\n", i);
    printf_s(" a and b remainder  is = %d\n\n", j);
    getch();
    return 0;
}

```

C:\Users\WinHtut_GreenHackers\source\re
a and b addition is = 15
a and b subtraction is = 5
a and b multiplication is = 50
a and b division is = 2
a and b remainder is = 0

WinHtut(Equation)
GreenHackersTeam

photo credit
[http://slideplayer.com/slide/
5134027/](http://slideplayer.com/slide/5134027/)

Arithmetic Operators

operation	Arithmetic operator	Algebraic expression	expression
Addition	+	$a + 7$	$a + 7$
Subtraction	-	$p - q$	$p - q$
Multiplication	*	xy	$x * y$
Division	/	x/y or $x \div y$	x / y
Remainder (modulus)	%	$r \bmod s$	$r \% s$

6

ဒါက ကျွန်တော်တို့ equation တစ်ကြောင်းမှာ Arithmetic Operator တစ်လုံးပဲပါလို့ ဖြစ်ပါတယ်။ ဥပမာ - equation တစ်ကြောင်းမှာ Arithmetic Operators တွေ အများကြီး ပါလာရင် ဘယ်လိုစပြီး ဖြေရှင်းမလဲ။ ဖြေရှင်းပုံ ဖြေရှင်းနည်းကို သိတားမှ ကျွန်တော်တို့ code ရေးတဲ့အခါ အမှားနည်းစေမှာ ဖြစ်ပါတယ်။

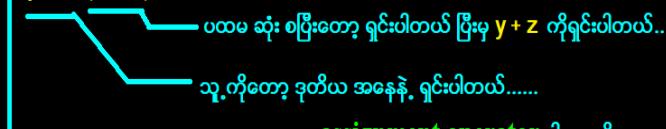
Example:

$$x = a + b + c - d * y / q \% y$$

အထက်ပါ equation မှာဆိုရင် ကျွန်တော်တို့ ဘယ်လိုစရှင်းမလဲပေါ့။ အဲဒါကို ရှင်းဖို့ဆိုရင် math concept လေးသိုးဖို့ လိုမယ်ပျော်။ အဲဒါ ဘာလဲဆိုတော့ precedence (ဦးစားပေး) ဖြစ်ပါတယ်။ အောက်က ပုံလေးမှာ တစ်ချက်ကြည့်ပေးပါ။ ကျွန်တော် အခုပုံထဲမှာ precedence in arithmetic operators ကိုပဲ ပြောဦးမယ်။ Relational Operation တွေရဲ့ precedence တွေ ကျွန်ပါသေးတယ်။ နောက်သင်ခန်းစာတွေမှာ example လေးတွေနဲ့ ဆက်ပြေပေးသွားပါမယ်။

Precedence in Arithmetic Operator(C programming)

$$x = y + z(i * k)$$



* ကျန်တော်ထိ၊ ပြီးမှုခြင်းတစ်စုံတည်ပါ အငြောက်ရယ် အား ယင် အဖြွေ့စွဲရာတဲ့ ဟာ ထဲးနိုင်ပါလဲ နဲ့မယ် ဆုံးရေးတော်ထိ အငြောက်ဂို့ အရင် ရှင်းပါယ်
/ ပြီးမှုများများ အပြွေ့စွဲရာတော်ထိ ရှင်းမှုခြင်းဝါတယ် တကယ်လို့ အငြောက်နှစ်ခု ပါလာရင်တော့ ဘယ်ဘက်က အငြောက် တို့ အရင်
% ရှင်းပါယ်

x = y + z (a+b) + (i+j) * k *c *e % f / g % h / 10 - 5
13 10 1 11 2 3 4 5 8 6 9 7 12

အထက်ပါ ဉာဏ်ခြင်းမှာဆိုရင် () * / % + - တို့၏ precedence များ၏ဖော်ပြခေါ်သားပါတယ်....
Left to Right ဘားဘေးလေးမှတ်တီ သိတိပါပြီ။ကြော်ခေါ်ပါတယ်ရင်ကဲ့

WinHtut(Equation)
GreenHackersTeam
Feb 7.2018 2:28AM

RELATIONAL OPERATORS IN C

Equality operators

==	x==y	x ወ y ቁለጥታውን ላይ ማስቀመጥ ተችል፡፡ assignment operator (=) ቁጥር፡፡ ከተሰጠውን ሁኔታውን ላይ ማስቀመጥ ተችል፡፡
!=	x != y	x ቁጥር፡፡ y ቁጥር፡፡ ላይ ማስቀመጥ ተችል፡፡

RELATIONAL OPERATORS AND CONTROL STRUCTURE IN C

Relational operators တွေကို အထက်မှာ ပုံနဲ့ဖော်ပြပေးထားပါတယ်။ ကျွန်တော်တို့ ငယ်ငယ်တုန်းက သင်ခဲ့ကြတဲ့ ၁ ကြီး c ငယ်ဆိုတာကို မှတ်မီးမယ် ထင်ပါတယ်။

Relational operators နဲ့ Equality operators တို့ဟာ သူတို့ချဉ်းပဲ ဆိုတာထက် Decision Making လုပ်တာတွေ Control လုပ်တာတွေနဲ့ ပေါင်းသုံးလိုက်ရင် အစွမ်းထက် လာပါတယ်။ ဥပမာ- decision making ဖြစ်တဲ့ if ဆိုတာကို program ထဲမှာ ထည့်သုံးပြပါမယ်။

```

int main() {
    int first = 0;
    int second = 0;
    int total = 0;
    printf_s("Please enter a first number:");
    scanf_s("%d", &first);
    printf_s("Please enter a second number:");
    scanf_s("%d", &second);
    total = first + second;

    if (total == 15)
    {
        printf_s("your total number is equal to 15\n");
    }
    if (total >= 15)
    {
        printf_s("your total number is greater than or equal to 15\n");
    }
    if (total < 15)
    {
        printf_s("your total number is less than to 15");
    }
    getch();
}
  
```

The terminal window shows the following output:

```

C:\Users\WinHtut_GreenHackers\source\repos\if condicton\Debug\if condicton
Please enter a first number:5
Please enter a second number:5
your total number is less than to 15
  
```

Annotations in the callout box:

- Yellow bracket over the assignment operator (`=`) in `total = first + second;`: "total တို့ ထည့်မယ်လို့ ပြောတာပါ အခါး အောင်မြင်မှုများပါ။"
- Yellow bracket over the relational operator (`==`) in `if (total == 15)`: "ယက လဲလဲ relational operator ပါ total က 15 နဲ့ ညီညီ အောင် စာကို ဖော်ပြပါလို့ ပြောပြုပါ။ Equality Operator ပါ။"
- Yellow bracket over the relational operator (`>=`) in `if (total >= 15)`: "ယက လဲလဲ relational operator ပါ total က 15 တို့မယ် သို့မဟုတ် ညီညီလို့ ဆိုတာပါ....."

အခုဆက်လက်ပြီး Control Structures တွေအကြောင်းကို ပြောသွားပါမယ်။ Control Structures ဆိုတာ ကျွန်တော်တို့ program တစ်ပုဒ်မှာ statement (ဖော်ပြချက်) တွေကို တစ်ခုပြီးမှ တစ်ခု လုပ်သွားတာကို ပြောချင်တာပါ။

Control Structure

Control structures မှာ အရေးကြီးတဲ့ အပိုင်း (၃) ပိုင်း ပါဝင်ပါတယ်။

Sequence Structure

Selection Structure

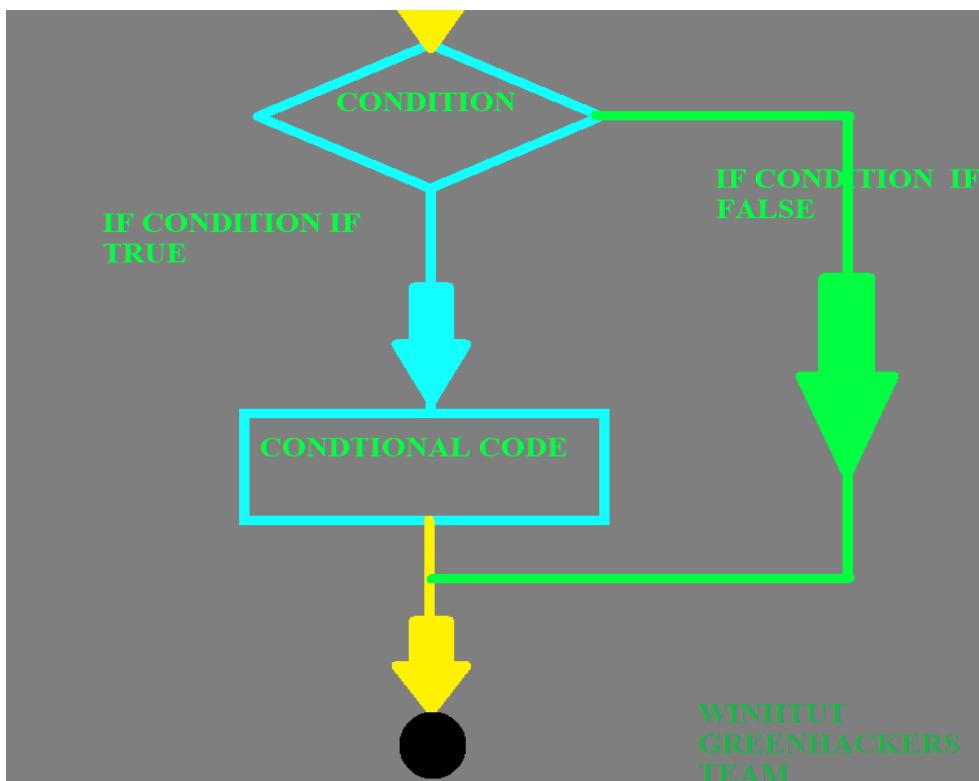
Iteration Structure တို့ဖြစ်ပါတယ်။

1. Sequence Structure

Sequence Structure ဆိုတာကတော့ program ရေးထားတဲ့အတိုင်းပဲ အစဉ်လိုက် computer ကနေ execute လုပ်သွားတာကို ဆိုလိုခြင်းဖြစ်ပါတယ်။

2. Selection Structure

Selection Structure မှာတော့ selection statement ဆိုပြီး (၃) မျိုး ထပ်ခွဲပါတယ်။



Selection Statement

If selection statement

သူကတော့ ရှင်းပါတယ်။ ကျွန်တော် အပေါ်က ပုံထဲမှာလည်း ထည့်ထားပါတယ်။ သူက if နောက်က (...) ကွင်းထဲက ဖော်ပြချက်တွေကို စစ်ပါတယ်။ (...) ထဲကဟာကို Condition လို့ ခေါ်ပါတယ်။ မှန်ရင် သူနောက်က {.....} တွန်ကွင်းထဲက ကောင်ကို ဆက်လုပ်တယ်။ (အဲဒီ {...} ထဲကကောင်ကိုတော့ conditional code လို့ခေါ်ပါတယ်)။ မမှန်ဘူး ဆိုရင် ကျော်သွားပေးပါတယ်။ အထက်မှာ ဖော်ပြထားတဲ့ပုံကို ကြည့်ရင် ရှင်းသွားမှာပါ။

If ...else statement

သူကတော့ if ထက်တစ်ခုပို့သွားပါတယ်။ အဲဒါကတော့ else ပါလာတာပါ။ if (...) နောက်က ကွင်းထဲက အခြေအနေကို စစ်တယ်။ ဥပမာ- if ($x == 1$) ဆိုရင် x ဟာ 1 နဲ့ညီသလားပေါ့။ ညီတယ်ဆိုရင် {.....} တွန်ကွင်းထဲက ဖော်ပြချက်တွေကို ဆက်လုပ်တယ်။ မညီဘူးဆိုရင် else ထဲက code ကို အလုပ်လုပ်ပါတယ်။

```
If (x == 1)
{
    //x ဟာ 1 နဲ့ညီခဲ့ရင် အလုပ်လုပ်မည့် code ဖြစ်ပါတယ်။
    printf(" x is equal to 1");
}
else
{
    //x ဟာ 1 နဲ့မညီခဲ့ရင် အလုပ်လုပ်မည့် code ဖြစ်ပါတယ်။
    printf( "x is not equal to 1");
}
```

```

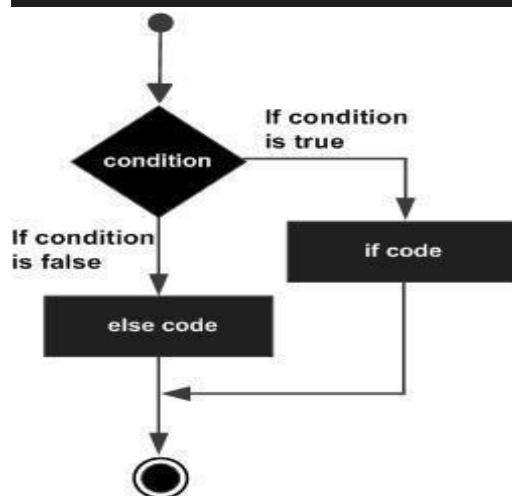
#include<stdio.h>
#include<conio.h>

int main() {
    int n1 = 0;
    int n2 = 0;
    printf_s("enter a number\n");
    scanf_s("%d", &n1);

    if (n1 > 5) {
        printf_s("your number is 5\n");
    }
    else
    {
        printf_s("your number is not equal to 5");
    }
    _getch();
    return 0;
}

```

WinHtut(Equation)
GreenHackersTeam



switch Statement

သူကတော့ ကျွန်တော်တို့စစ်ချင်တဲ့ အခြေအနေတစ်ခုကို () ထဲမှာ ထည့်လိုက်ပြီး လုပ်ဆောင်ချက်တွေ အများကြီး လုပ်ဆောင်ပါတယ်။

ဥပမာ -

`switch(x)` ဆိုပြီး ကိုစစ်ချင်တဲ့ variable ကို () ကွင်းထဲမှာ ထည့်လိုက်ပါတယ်။

{

`case 1:` တကယ်လို့ x က 1 ဖြစ်ခဲ့ရင် အောက်က စာကိုရေးပါ။

```
printf("x is 1");
break; အပေါ်က စာကို ရေးပြီးရင် ရပ်မယ်။
```

`case 2:` တစ်ကယ်လို့ x က 2 ဖြစ်ရင် အောက်က စာကိုရေးပါ။

```
printf("x is 2");
break; အထက်က စာကို ရေးပြီးရင် ရပ်မယ်။
```

`case 3:` တစ်ကယ်လို့ x က 3 ဖြစ်ရင် အောက်က စာကိုရေးပါ။

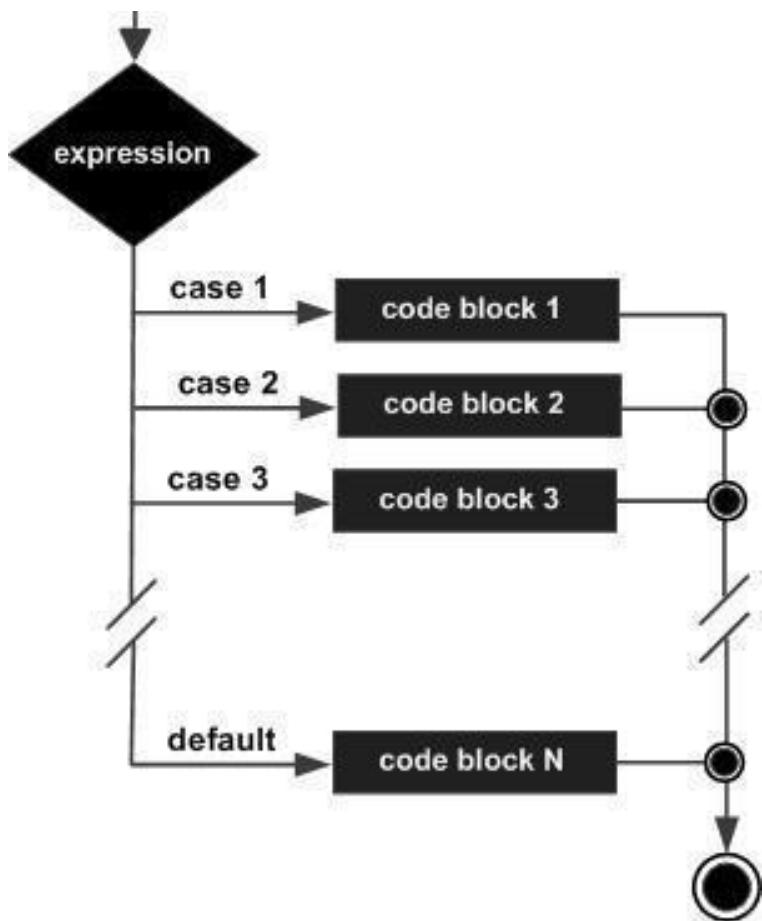
```
printf("x is 3");
break; အထက်က စာကို ရေးပြီးသွားရင် ရပ်မယ်။
```

`default:` သူ့ကတော့ အထက်က ဖော်ပြထားတဲ့ ဟာတွေတစ်ခုမှ မဟုတ်ဘူးဆိုရင် default အောက်က ဖော်ပြချက်တွေကို အလုပ်လုပ်မယ်လို့ ပြောတာပါ။

```
printf("x is not equal to 1,2,3");
break; အထက်က စာကို ရေးပြီးသွားရင် ရပ်မယ်။
```

}

အောက်က ပုံလေးတွေကို သင်ခန်းစာအနေနဲ့ လုပ်ကြည့်ပါ။ အဲဒါဆိုရင် သေချာနားလည် သွားပါလိမ့်မယ်.....။



```
test resistor
C:\Users\WinHtut_GreenHackers\source\resistor> Please enter a number:
2
this your number: 2
```

WinHtut(Equation)
GreenHackersTeam

```
Source.cpp  X
test resistor
(Global Scope)
#include<stdio.h>
#include<conio.h>

int main() {
    int x = 0;
    printf_s("Please enter your password:\n");
    scanf_s("%d",&x);
    switch (x) {
        case 123:
            printf_s("welcome mg mg %d", x);
            break;
        case 456:
            printf_s("welcome kyaw kyaw %d",x);
            break;
        default :
            printf_s("your password is wrong ");
            break;
    }
    _getch();
    return 0;
}
```

WinHtut(Equation)
GreenHackersTeam

if selection statement ကို single selection statement လိုလည်း ခေါ်ပါတယ်။ ဘာလိုလည်းဆိုတော့
အလုပ်တစ်ခုပဲ လုပ်လိုပါ။

if...else selection statement ကိုတော့ double-selection statement လိုခေါ်ပါတယ်။ သူက
မတူညီတဲ့လုပ်ဆောင်ချက် နှစ်ခုကြားမှာ အလုပ်လုပ်လို့ဖြစ်ပါတယ်။

switch selection statement ကိုကျတော့ multiple-selection statement လိုလည်း ခေါ်ပါတယ်။ သူက
မတူညီတဲ့ အခြေအနေတွေ အများကြီးနဲ့ အလုပ်လုပ်ပါတယ်။

3. Iteration Statement

ဆက်လက်ပြီး ကျွန်တော်တို့ Iteration Statement အကြောင်းကို သွားကြရအောင်။ Iteration
Statement ကို Repeat လုပ်တဲ့ Statement လိုလည်း ခေါ်သလို Loop လုပ်တဲ့ Statement လိုလည်း
ခေါ်ပါတယ်။ သူမှာလည်း (3) မျိုးပါဝင်ပါတယ်။

1. while Statement
2. do...while Statement
3. for Statement

သူတို့ကို while loop, do...while loop, for loop လို့ ခေါ်ပါတယ်။ program ရေးတဲ့နေရာမှာ
အမြတ်များလိုလို သုံးပါတယ်။

while loop ဆိုတာက

```
int a = 10;
While (a<20) // a ရဲ့တန်ဖိုးက 20 အောက်ငယ်တဲ့အထိ အလုပ်လုပ်မယ်လို့ပြောတာ ဖြစ်ပါတယ်။
{
    printf ("this is called while loop");
    a++; //a = a +1
    // တစ်ကြိမ် loop ပတ်တိုင်း a ထဲကို 1 ပေါင်းသွားမယ်လို့ပြောတာ ဖြစ်ပါတယ်။
}
```

အဲဒီတော့ ကျွန်တော်တို့ စဉ်းစားကြည့်ရအောင်။ ပထမ a တန်ဖိုးကို 10 လို့ကြော်ပြာထားတယ်။ 10
ကနေ 20 အောက်ငယ်တဲ့ 19 အထိဆိုရင် ဘယ်နှစ်ကြိမ် အလုပ်လုပ်သလဲပေါ့။ 10, 11, 12, 13, 14, 15,
16, 17, 18, 19 စုစုပေါင်း (၁၀) ဆယ်ကြိမ် အလုပ်လုပ်သွားမှာ ဖြစ်ပါတယ်။ ပထမတစ်ခါ a က 10 ကနေ
loop စပတ်တယ် printf ကိုရေးတယ်။ ပြီးရင် a ကို 1 ပေါင်းတယ်။ အဲတော့ a က နိုင်က 10

ရှိယားတာဆိုတော့ 11 ဖြစ်သွားတယ်။ နောက်တစ်ကြမ်လည်း အဲဒီလိုပဲ printf ရေးတယ်။ a ကို 1 ပေါင်းတယ်။ အဲဒီတော့ a ရဲ့တန်ဖိုး 11 ကို 1 ထပ်ပေါင်းထည့်တော့ 12 ဖြစ်သွားတယ်။ အဲလိုနဲ့ a က 20 ထိ ရောက်သွားတော့ ရပ်သွားတယ်။ ဘာလို့လည်းဆိုတော့ **while (a < 20)** a က 20 အောက်ငယ်တဲ့ အထိလို့ ပြောယားတယ်လဲ။ အောက်ကပုလေးကို ကြည့်ပြီး သင်ခန်းစာ လုပ်ကြည့်ပါ။ အဆင်ပြသွား မယ်လို့ ထင်ပါတယ်။

```

while loop
#include<stdio.h>
#include<conio.h>
int main()
{
    int a = 10;
    while (a<20)
    {
        printf_s("this is called while loop\n");
        a++;
    }
    getch();
    return 0;
}

```

(Global) C:\Users\WinHtut_GreenHackers\source\repos\while loop\Debug\while loop.exe

this is called while loop
this is called while loop

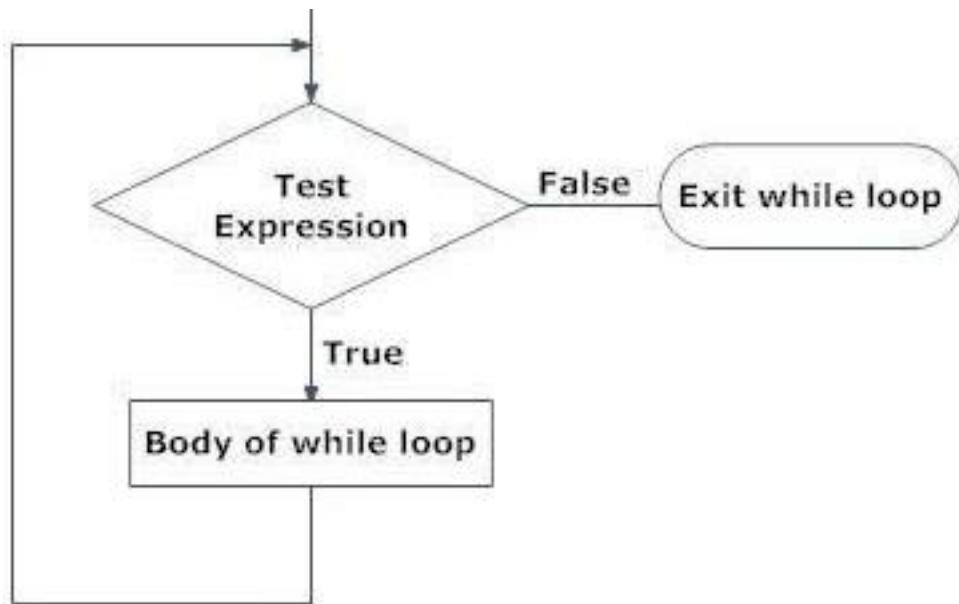


Figure: Flowchart of while loop

နောက်တစ်ခုအနေနဲ့ do....while loop ကတော့ အောက်ကပုလေးမှာ ကြည့်ပေးပါ။

The screenshot shows a code editor with a file named `doWhileApp.cpp`. The code contains a `while` loop that prints "this is do while loop" 15 times. The output window shows the same text repeated 15 times. The copyright notice at the bottom right reads "WinHtut(Equation) GreenHackersTeam".

```

#include<stdio.h>
#include<conio.h>
int main() {
    int a = 5;
    do{
        printf_s("this is do while loop\n");
        a++;
    } while (a < 20);

    getch();
    return 0;
}

```

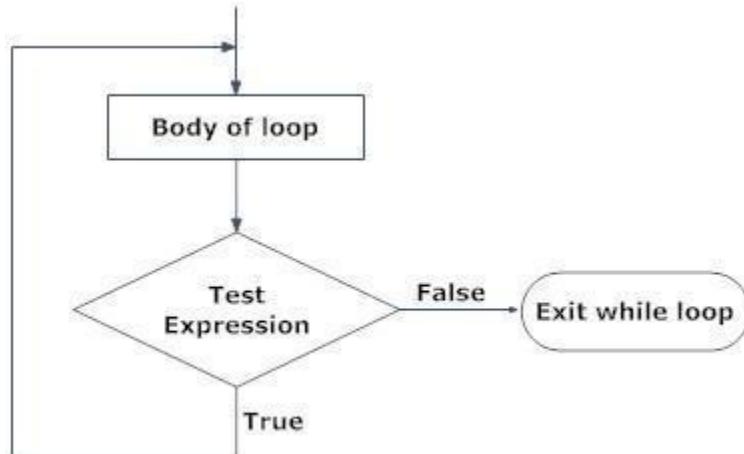


Figure: Flowchart of do...while loop

While loop သင်ခန်းစာကို သေချာလုပ်ထားရင် အခါး `do while loop` ကလည်း ရေးပုံလေးသာ ပြောင်းလဲသွားတာပါ။ `do` ဆိုတဲ့ အလုပ်လုပ်မဲ့ကောင် ကိုရေးတယ်။ ပြီးမှ `while` ထဲမှာ ဆုံးရမဲ့နေရာကို ထည့်ပါတယ်။ နောက်ဆုံးတစ်ခုအနေနဲ့ `for loop` အကြောင်းကို ပြောပါမယ်။

```

int i=0;
for(i=0; i<10 ; i++)
{
    printf("Hello for loop\n");
}

```

အထက်ပါ program ကို run ကြည့်မယ်ဆိုရင် Hello for loop ဆိုပြီး ဆယ်ကြာင်းထုတ်ပေးပါလိမ့်မယ်။ စာနဲ့ ရှင်းပြရမယ်ဆိုရင် For ရဲနောက်က ကွင်းထဲကနေ စရှင်းပြရပါမယ်။ For loop ကို ဘေးလုံးကွင်းတစ်ကွင်းကို အပြေးပြိုင်တယ်လို့ မြင်ကြည့်လိုက်ပါ။ i=0 ဆိုတာက စရမဲ့ အမှတ်ဆိုပါမျို့။ i<10 ဆိုတာက ပတ်ရမဲ့ အကြိမ်အရေအတွက်၊ i++ ဆိုတာက တစ်ကြိမ်ပတ်တိုင်း i တန်ဖိုးကို 1 ပေါင်းမယ်လို့ ပြောတာ ဖြစ်ပါတယ်။

For (i=0 ; i<10 ; i++) တွင် i<10 ဖြစ်တဲ့အတွက် 1000 ပတ်ရမယ့် အကြိမ်အရေအတွက်က 10 ဖြစ်ပါတယ်။ ဘာလို့လဲဆိုတော့ စရမဲ့အမှတ်က 0 ကနေ 10 အောက်ငယ်တဲ့ 9 အထိ ဆိုတော့ 0,1,2,3,4,5,6,7,8,9 အလုံးရေ 10 လုံးပါ။ ဒါကြောင့် 10 ကြိမ်ပတ်မယ်လို့ ပြောတာပါ။ တကယ်လို့ for (i=1 ; i<10 ; i++) ဆိုရင် 1 ကနေ 9 အထိ ဆိုတော့ 1,2,3,4,5,6,7,8,9 စုစုပေါင်း 9 လုံး ဖြစ်တဲ့အတွက် ကိုးကြိမ်ပတ်မယ်လို့ ပြောတာ ဖြစ်ပါတယ်။ အောက်က ပုံလေးကို သင်ခန်းစာအနေနဲ့ လုပ်ကြည့်ပါ နားလည်လာပါလိမ့်မယ်။ တစ်ကြိမ်တည်းနဲ့တော့ နားမလည်နိုင်ပါဘူး။

1 Team (Global Scope)

```
#include<stdio.h>
#include<conio.h>

int main() {
    int i = 0;
    int j = 0;
    int total = 0;

    printf_s("Please enter a number\n");
    scanf_s("%d", &i);
    printf_s("Please enter a number\n");
    scanf_s("%d", &j);
    total = i + j;
    printf_s("Your total number is %d\n", total);

    for (int k = 0; k < total; k++) {
        printf_s("Hello Myanmar\n");
    }

    getch();
    return 0;
}
```

WinHtut(Equation)
GreenHackersTeam


```
#include<stdio.h>
#include<conio.h>
int main() {
    int a = 10;
    for (int i = 5; i < a; i++) {
        printf_s("Hello coder:\n");
    }
    getch();
    return 0;
}
```

WinHtut(Equation)
GreenHackersTeam

အခု ပုံမှာဆိုရင် စတောက 5 ကနေ စပါတယ်။ ဆုံးတောက 10 ထက်ငယ်တဲ့ 9 မှာပါ။ အဲတော့ ဘယ်နှစ်ခေါက် Loop ပတ်သွားသလဲ ဆိုတောကို ကြည့်ပါ။ တစ်ခေါက်ပတ်တိုင်း မှာ Hello coder ကို

တစ်ခါရေးမယ်ဆိုတော့ 5 ခါပတ်ရင် Hello coder ဆိုပြီး 5 ကြောင်း ပေါ်လာပါလိမ့်မယ်။ အခုဆက်လက်ပြီး ဥက္ကာလုပ်မှုများလေးတွေ ရေးကြည့်ရအောင်။

```

1 Team (Global Scope)

int main() {
    int i = 0;
    int j = 0;
    int total = 0;

    printf_s("Please enter a number\n");
    scanf_s("%d", &i);
    printf_s("Please enter a number\n");
    scanf_s("%d", &j);
    total = i + j;
    printf_s("Your total number is %d\n", total);

    for (int k = 0; k <= total; k++) {
        for (int x = 0; x < k; x++) {
            printf_s("*");
        }
        printf_s("\n");
    }

    _getch();
    return 0;
}

```

WinHtut(Equation)
|GreenHackersTeam

အထက်ပါ ပုံမှုဆိုရင် for loop ထဲမှာပဲ နောက်ထပ် for loop တစ်ကြောင်းထပ်ရေးထားတာ ပါ။ ပြီးတော့ k တန်ဖိုးကို သေချာမှတ်ထားပါ။ သူက တစ်ကြိမ်ပတ်တိုင်း 1 ပေါင်းထားတယ်။ အဲ k တန်ဖိုးကိုပဲ နောက်ထပ် for loop မှာ အဆုံးမှတ်အနေနဲ့ ပြန်သုံးထားပါတယ်။ code လည်းရေးပါ။ စာအုပ်နဲ့ ဘေးပင် ကိုလည်း ဘေးမှာချထားပြီး တန်ဖိုးတွေကို မှတ်ထားပါ။ အဲဒါဆိုရင် အဆင်ပြုပါလိမ့်မယ်။

Logical Operators In C Programming

Logical operators တွေကို programming Language တိုင်းမှာ အမြဲတမ်းသုံးပါတယ်။ သူကို Relational Operators တွေဖြစ်တဲ့ $>$ (greater than), $<$ (less than), \geq (greater than or equal to), \leq (less than or equal to) တွေနဲ့ လည်းတွဲသုံးပါတယ်။

Logical Operators သုံးမျိုးရှိပါတယ်။

1. Logical AND (&&)
2. Logical OR (||)
3. Logical NOT (!)

အထက်ပါ သုံးမျိုးထဲက `&&` ဆိုတဲ့ Logical AND operator ကို အရင်ဆုံးရှင်းပေးပါမယ်။ ဥပမာ-ကျွန်တော်တို့ ကျောင်းသားတစ်ယောက်ရဲ့ အသက်ရယ်၊ အမှတ်ရယ်ကို if statement ထဲမှာ ထည့်စစ်မယ်ပေါ့။ condition က lesson 5 ကလို တစ်ခုတည်းမဟုတ်တော့ဘူး။ နှစ်ခုဖြစ်သွားပြီ ဆိုရင် ကျွန်တော်တို့လိုချင်တဲ့ ပုံစံကကျောင်းသားဟာ အသက် 10 နှစ်ထက်လည်း ကြီးရမယ် (သို့) ညီလည်း ညီရမယ်။ ပြီးတော့ (`&&`) AND သူ့ရဲ့ အမှတ်ကလည်း 99 ထက်ကြီးရမယ် (သို့) ညီလည်း ညီရမယ်ဆုံးရင် အဲလိုမျိုး condition နှစ်ခုလုံးမှန်မှုသာ သူ့ နောက်က expression တွေကို အလုပ် ဆက်လုပ်မယ်လို့ ဆိုလိုတာပါ။ မှားနောက်တော့ အလုပ်ဆက် မလုပ်တော့သလို output လည်း ထွက်လာမှာမဟုတ်ပါဘူး။

```
int student_age = 10;
int student_mark = 99;
if ((student_age >= 10) && (student_mark >= 99)) {
    printf_s("You must go next Grade!");
}
```

OUTPUT:: You must go next Grade!

မှတ်ထားဖို့လိုတာက Logical AND (`&&`) operator က နှိုင်းယှဉ်တဲ့ condition အားလုံး မှန်မှုပဲ အလုပ်လုပ်ပါတယ်.....။

Logical OR (||) operator

သူကကျေတော့ condition နှစ်ခုထဲက တစ်ခု မှန်ရင်ကို အလုပ်လုပ်ပါတယ်။ ဥပမာ ကောင်လေးတစ်ယောက်ရဲ့ အသက်ဟာ 18 နှစ်ကျော်ရမယ်။ သူ့ရဲ့ အရပ်ကတော့ 6 ပေ ကျော်ရမယ်။ အဲ condition နှစ်ခုထဲက တစ်ခုမှန်ရင်ကို အလုပ်လုပ်မယ်လို့ ဆိုလိုခြင်း ဖြစ်ပါတယ်။

Example

```
int main() {
    int boy_age = 18;
    int boy_high = 6;
    if ((boy_age >= 18) || (boy_high <= 5)) {
        printf_s("He is a tall boy!");
    }
}
```

OUTPUT => He is a tall boy!

Condition နှစ်ခုထဲက နောက်က condition ဟာ မှားနေသံလည်းပဲ output ထွက်မှာဖြစ်ပါတယ်။ ရှိခဲ့က condition သို့မဟုတ် နောက်က condition ဘယ်တစ်ခုပဲမှန်မှန် အလုပ်လုပ်သွားမှာဖြစ်ပါတယ်။

Logical NOT(!) operator

သူ့ကတော့ မဟုတ်ဘူး(မတူဘူး)လို့ဆိုလိုချင်တာပါ။ ဥပမာ- a=5, b=10, c=15; မှာဆိုရင် (a != b) ဒါဆိုရင် မှန်ပါတယ်။ ဘာလိုလဲဆိုတော့ a=5 ဖြစ်ပြီးတော့ b=10 ဖြစ်နေလို့ သူတို့ နှစ်ခုက မတူဘူးလို့ပြောထားတဲ့ အတွက်ကြောင့်ပါ။

```
int a = 5, b = 10, c = 15;
if (c == (a + b)) {
    printf_s("Hello");
}
```

အထက်ပါ program မှာဆိုရင် Hello ဆိုတဲ့ output ထွက်ပါလိမ့်မယ်။ အောက်က program မှာဆိုရင်တော့ ထွက်မှာ မဟုတ်ပါဘူးခင်ဗျာ။

```
int main() {
    int a = 5, b = 10, c = 15;
    if (c == !(a + b)) {
        printf_s("Hello");
    }
    _getch();
    return 0;
}
```

နောက်ထပ်တစ်မျိုး ထပ်ရေးကြည့်မယ်ဆိုရင် ပိုရှင်းသွားပါလိမ့်မယ်။

```
#include<stdio.h>
#include<conio.h>
int main() {
    int a = 5, b = 10, c = 15;
    if (c == !(a + b)) {
        printf_s("Hello");
    }
    else {
        printf_s(" C is not equal to a+b is false");
    }
    _getch();
    return 0;
}
```

OUTPUT=> C is not equal to a+b is false

အခါ program လေးကို run ကြည့်ပြီး output ကိုကြည့်မယ်ဆိုရင် ရှင်းသွားပါလိမ့်မယ်။

Bitwise Operators In C Programming

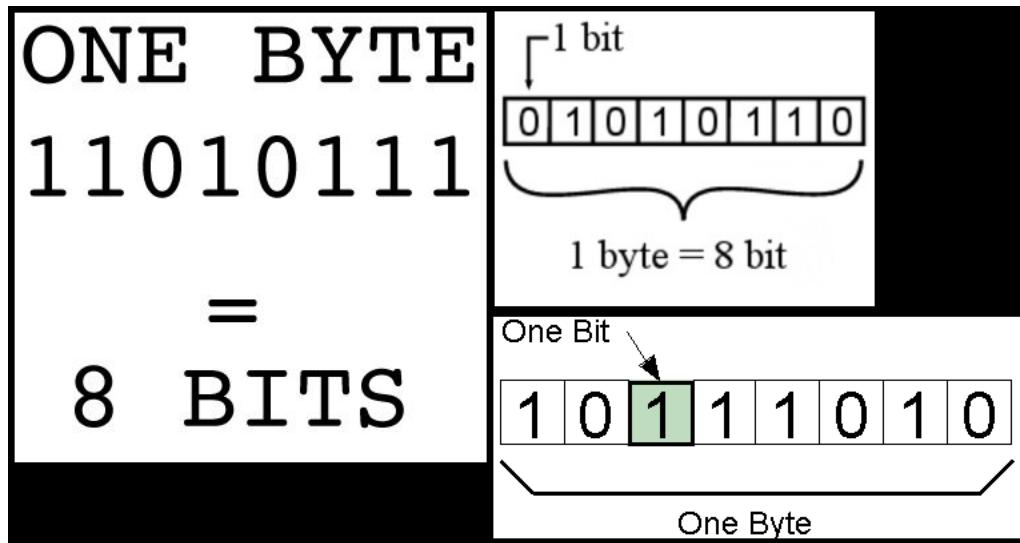
Bitwise operators တွေဟာ အရေးပါပါတယ်။ ဘာကြောင့်လဲဆိုတော့ computer ဟာ Mathematical တွက်ချက်မှုတွေ လုပ်ဆောင်တဲ့အခါမှာ bitwise တွေကို သုံးပါတယ်။ ဘယ်လို တွက်ချက်မှုတွေလဲဆိုတော့ arithmetic operators တွေဖြစ်တဲ့ ပေါင်း၊ နှုတ်၊ ဧည့်က်၊ စား တွေပါ။ ထိုကဲ့သို့ တွက်ချက်မှုတွေကို bit-level အထိ ပြောင်းပြစ်လိုက်ပြီးတော့ တွက်ပါတယ်။ အဲလို ပြောင်းလဲလိုက်ခြင်းအားဖြင့် တွက်တာ ချက်တာ ပိုမိုမြန်ဆန်လာမယ်။ ပြီးတော့ power စားတာ တွေလည်း သက်သာသွားစေပါတယ်။ bit ဆိုတာက 0 နဲ့ 1 (binary digit) ကို ပြောတာပါ။ ကျွန်တော်

Simple C Program With Integer (int) and Variable (Memory Location) lesson 6

ပြောပြထားပါတယ်။ အောက်က ပုံလေးတွေမှာ ကြည့်ပါ။

Bitwise Operators 6 ମୁଖ୍ୟ ପିତାଙ୍କ

- 1. **(&)** Bitwise AND
 - 2. **(|)** Bitwise OR
 - 3. **(^)** Bitwise XOR



~ Bitwise complement

<< Shift left

>> Shift right

Bitwise AND operator (&)

AND (&) အကြောင်းကို အထက်မှာလည်း အတော်အသင့်ရှင်းပြထားထားပါတယ်။ ထပ်ပြီး ပြောပြပါ၌ဦးမယ်။ 0 & 1 ဆိုရင် အဖြေက 0 ပါ။ ဘာလို့လည်းဆိုတော့ မတူတဲ့အတွက်ပါ။ 1 & 1 ဆိုရင်တော့ အဖြေက 1 ပါ။ 0 & 0 ဆိုရင်တော့ အဖြေက 0 ပါ။

[စကားချုပ်...> programming ကို လေ့လာသောသူများအနေဖြင့် Number System လေးမျိုးကို လေ့ထားသင့်ပါတယ်။ Number System လေးမျိုးကို သိတားမှသာ computer ရဲ့တွက်ချက်ပုံ တွေကို ကောင်းမွန်စွာ နားလည့်မှာ ဖြစ်ပါတယ်။

Number system ලේඛීම් සහිත

- ### 1. Binary Number System වාග base 2 (0 and 1) අනුබිතයි॥

2. Decimal Number System වුගැනී base 10 පි|| (0,1,2,3,4,5,6,7,8,9)||
 3. Octal Number System වුගැනී base 8 පි|| (0,1,2,3,4,5,6,7)||
 4. Hexadecimal number system වුගැනී base 16 පි|| (0,1,2,3,4,5,6,7,8, 9,A,B,C,D,E,F)
- පි ඔතු නූත්‍ය තොරු Arithmetic ලද්ධුවේ තො Conversion ලද්ධුවේ තො ගි
computer අයුරුදු මුද්‍රණය ඇත්තා පෙන්වනු ලබයි]

Example in Binary

$12=00001100$ (in binary number)

$25=00011001$ (in binary number)

ත්‍යාක්‍රම අනු පිළිගි & operation ලද්ධුව

00001100	
& 00011001	

$00001000 = 8$ (in decimal number)	

Example in C Programming

```
#include <stdio.h>
int main()
{
    int a = 12, b = 25;
    printf("Output = %d", a&b);
    return 0;
}
```

Output = 8

Bitwise OR operator (|)

OR operator ගි ත්‍රේඛුත් ත්‍රේඛු ලද්ධුවාතා ප්‍ර්‍රතිතයි|| අත්‍යුත්‍ය lesson තොමුලයුෂ් නෑත්‍ය නෑත්‍ය පිතයි|| වුගැනී 1 and 0 ගි (|) OR operation ලද්ධුවයේදී පිළිගි 1 පි||

1 and 1 ကို (|) OR operation လုပ်မယ်ဆိုရင်လည်း အဖြေက 1 ပါပဲ။ 0 and 0 ကို (|) OR operation လုပ်မယ်ဆိုရင် အဖြေက 0 ပါ။

Example in Binary

12=00001100 (in binary number)

25=00011001 (in binary number)

ထိန်စ်ခုကို | operation လုပ်ရင်

00001100
00011001

00011101= 29 (in decimal number)

Example in C programming

```
#include<stdio.h>
#include<conio.h>
int main() {
    int x = 12, y = 25;
    printf_s("output=%d", x|y);
    getch();
    return 0;
}
output=29
```

Bitwise XOR(^) operator

သူကတော့ တူရင် 0 မတူရင် 1 ပါ။ ဥပမာ 0 and 1 ကို XOR (^) operation လုပ်မယ်ဆိုရင် အဖြေက 1 ပါ။ နောက်ထပ် 0 and 0 ကို XOR (^) operation လုပ်မယ်ဆိုရင် အဖြေက 0 ပါပဲ။

Example in Binary

12=00001100 (in binary number)

25=00011001 (in binary number)

ထိန်စ်ခုကို XOR(^) operation လုပ်ရင်

00001100
^ 00011001

00010101 = 21 (in decimal number)

Example in C Programming

```
#include<stdio.h>
#include<conio.h>
int main() {
    int x = 12, y = 25;
    printf_s("output =%d", x^y);
    getch();
    return 0;
}
output =21
```

Bitwise Complement Operator (~)

Complement မှာ နှစ်မျိုးရှိပါတယ်။

1. 1's complement
2. 2's complement

1's complement သည် မူလ number ကို invert (ပြောင်းပြန်) လုပ်ခြင်း ဖြစ်ပါတယ်။ $35=00100011$

သူကို 1' complement လုပ်မယ်ဆိုရင် (35 ကိုမှတ်ထားပါ)။

~ 00100011

11011100 ဒါကတော့ 1's complement လုပ်တာပါ။ နှာက်တစ်ခု 2's complement လုပ်ကြရအောင်။ ** တစ်ခု မှတ်ထားဖို့ လိုပါတယ်။ 2's complement ကို အပေါင်းကိန်းကနေ အနှစ်ကိန်းကို ပြောင်းချင်တဲ့ အခါမှာ သုံးပါတယ်။ ** Assembly programming ကိုလေ့လာတဲ့ အခါမှာ အသုံးဝင်ပါလိမ့်မယ်။ အပေါင်းကိန်း 35 ရဲ့ 1's complement တန်ဖိုးက ~ 11011100 အနှစ်ကိန်း - 35 ရဲ့ တန်ဖိုးက ဘယ်လောက်လဲပေါ့။ အပေါင်းကိန်းကနေ အနှစ်ကိန်းပြောင်းချင်ရင် 2's complement လုပ်ရမယ်။

35 ရဲ့ 1's complement 11011100

$$\begin{array}{r} + \quad 1 \\ \hline \end{array}$$

35 ရဲ့ 2's complement $11011101 = -35$

အထက်မှာ ဖော်ပြထားတာတွေက 1's complement နဲ့ 2's complement လုပ်တာတွေ ဖြစ်ပါတယ်။ အခုဆက်လက်ပြီး မည်သည့် number ကို မဆို Bitwise complement (\sim) လုပ်ကြရအောင်။ equation လေး တစ်ခြားရှိပါတယ်။ ဝါသနာပါတဲ့ သူတွေတော့ တွေက်ကြည့်လို့ရပါတယ်။

N=number

N ကို Bitwise complement(\sim) လုပ်မယ်ဆိုရင် ?

$$\sim N = -(\sim(\sim N) + 1) = - (N + 1)$$

35 ကို Bitwise complement(\sim) လုပ်ချင်တယ်ဆိုရင်

$$35 = - (35 + 1)$$

အဲဒီတော့ 35 ကို Bitwise complement လုပ်တာက -36 ရပါတယ်။ အောက်ပါ program လေးကို run

ကြည့်ရင် ပိုရှင်းသွားမယ်လို့ ထင်ပါတယ်။

Example => Bitwise Complement (\sim) In C programming

```
#include<stdio.h>
#include<conio.h>
int main() {
    int x = 35;
    printf_s("output=%d", ~x);
    _getch();
    return 0;
}
```

output =-36

အထက်မှာ ဖော်ပြခဲ့တာက အပေါင်းကိန်း 35 ကို Bitwise complement(\sim) လုပ်တာပါ။ အခု အနုတ်ကိန်း -35 ကို Bitwise complement(\sim) လုပ်ကြရအောင်။ ခုနက equation ကို ပြောင်းပြန် ပြန်လုပ်ရပါမယ်။

$$\sim N = -(\sim(\sim N) + 1) = - (N + 1)$$

$$35 = - (35 + 1) = -36$$

$$-35 = ?$$

```
#include<stdio.h>
#include<conio.h>
int main() {
    int x = -35;
    printf_s("output = %d", ~x);
    _getch();
    return 0;
}
```

output = 34

= 35-1

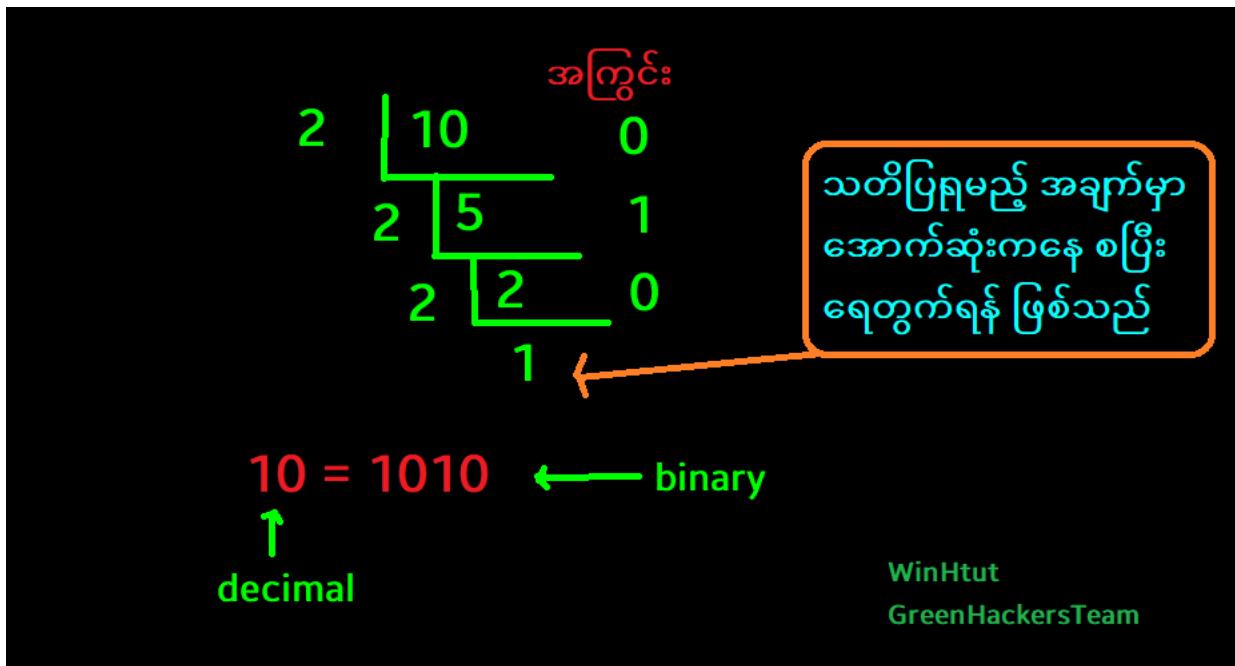
=34

```
lesson 6
└#include<stdio.h>
  #include<conio.h>
└int main() {
    int x = -35;
    printf_s("output = %d", ~x);
    _getch();
    return 0;
}

C:\Users\WinHut_GreenHackers\source\repos\lessor
output = 34_          WinHut
                           GreenHackers
```

Shift Operators

Shift operators နှစ်မျိုးရှိပါတယ်။ Right shift operator နဲ့ Left shift operator ပါ။ Shift operators တွေ အကြောင်းကို ပြောမယ်ဆိုရင် Binary Number System ကို သေချာပေါက် သိကို သိရပါမယ်။ ကျွန်တော်တို့ program ရေးတဲ့နေရာမှာ သုံးနေတာတွေက Decimal Number တွေပါ။ ဥပမာ int i=10; လို့ပြောလိုက်ရင် 10 ဆိုတာက decimal number ပါ။ binary number ဆိုတာကတော့ 1 နဲ့ 0 ပဲရှိတာပါ။ ခုနာက i=10 ဆိုတဲ့ decimal number ကနေ binary number ပြောင်းချင်ရင် အဲဒီ 10 ကို 2 နဲ့ စားပေးရပါတယ်။ အောက်က ပုံလေးမှာ တစ်ချက်ကြည့်ပေးပါ။



Left Shift Operator (<<)

```
x = 50 ;
```

```
printf(" Left shift by 1 bit to x is ( x<< 2^1) = %d", x<<1);
```

အထက်ပါ program မှာဆိုရင် မူလတန်ဖိုး $50 << 1$ bit နဲ့ shift လုပ်မှာပါ။ သူဆိုလိုချင်တာ က 50 ကို 2 to the power 1 နဲ့ ပြောက်မယ်လို့ ဆိုလိုခြင်းနဲ့ တူပါတယ်။

$50 * 2 = 100$ တကယ်လို့ 2 bit နဲ့ left shift လုပ်မယ်ဆိုရင် 2 to the power 2 နဲ့ ပြောက်ပေးရမှာပါ။

$50 * (2 * 2) = 200$ ဖြစ်သွားမှာပါ။ တကယ်လို့ 3 bit နဲ့ left shift လုပ်ခဲ့မယ်ဆိုရင်တော့ မူလတန်ဖိုးကို 2 to the power 3 နဲ့ ပြောက်ပေးရမှာပါ။ $50 * (2 * 2 * 2) = 400$ ဖြစ်သွားမှာပါ။ တိုးလာတာက left shift ပါ။

ဘယ်ဘက်ကိုဆိုရင် မူလတန်ဖိုး တိုးလာမှုလို့ မှတ်သားထားရပါမယ်။ 1 bit တိုးတိုင်း 2 power တစ်လုံး ပိုသွားပါလိမ့်မယ်။

```
#include<stdio.h>
#include<conio.h>
int main() {
    int x = 30;
    printf_s("Left shift By 1 bit ( x*2 ) = %d\n\n", x<<1);
    printf_s("Left shift By 2 bits(x*2*2) = %d\n\n", x << 2);
    printf_s("Left shift By 3 bits(x*2*2*2)= %d\n\n", x << 3);
    printf_s("Left shift By 4 bits(x*2*2*2*2) = %d\n\n", x << 4);
    getch();
    return 0;
}
```

Left Shift Operator IN C programming

C:\Users\WinHtut_GreenHackers\source\repos\lesson 6\Debug\lesson 6.exe

```
Left shift By 1 bit ( x*2 ) = 60      left shift by 1 bit  မူလ တန်ဘိုး ကို 2 နဲ့ တစ်ခါ ပြောက်ခြင်းပါ
Left shift By 2 bits(x*2*2) = 120    left shift by 2 bits  မူလ တန်ဘိုး ကို 2 နဲ့ နှစ်ခါ ပြောက်ခြင်းပါ
Left shift By 3 bits(x*2*2*2)= 240   left shift by 3 bits  မူလ တန်ဘိုး ကို 2 နဲ့ သုံးခါ ပြောက်ခြင်းပါ
Left shift By 4 bits(x*2*2*2*2) = 480 left shift by 4 bits  မူလ တန်ဘိုး ကို 2 နဲ့ 4 ခါ ပြောက်ခြင်း
```

60 = 111100	မူလ တန်ဘိုး ကို 2 တစ်လုံး တိုးပြောက်လိုက်တိုင်း 0 တစ်လုံး ပိုတိုးလာပါတယ်
120 = 11110000	binary number အနေ decimal number ပြောင်း decimal number အနေ binary number ပြောင်းပြီး Left Shift Operator ကို ကောင်းမွန်စွာ သဘောပေါက် နားလည် လာပါလိမ့်မယ်....
240 = 111100000	
480 = 1111000000	
960 = 11110000000	

WinHtut (GreenHackersTeam)

Right Shift Operator

Right shift Operator (>>) သူ့ကတော့ စားသွားမှုဖြစ်ပါတယ်။ X = 50; right shift ကို 1 bit နဲ့ လုပ်မယ်ဆိုရင် 50 ကို 2 power 1 နဲ့ စားပေးရမှာပါ။

```
printf("Right shift by 1 bit to x is ( x / 2^1)= %d",x>>1);
//output
//Right shift by 1 bit to x is (x/2^1) = 25
```

```
#include<stdio.h>
#include<conio.h>
int main() {
    int x = 500;
    printf_s("Right shift by 1 bit to x (x/2^1) = %d\n\n", x >> 1);
    printf_s("Right shift by 2 bits to x (x/2^2) = %d\n\n", x >> 2);
    printf_s("Right shift by 3 bits to x (x/2^3) = %d\n\n", x >> 3);
    printf_s("Right shift by 4 bits to x (x/2^4) = %d\n\n", x >> 4);

    getch();
    return 0;
}
```

C:\Users\WinHtut_GreenHackers\source\repos\lesson 6\Debug\lesson 6.exe

```
Right shift by 1 bit to x (x/2^1) = 250
Right shift by 2 bits to x (x/2^2) = 125
Right shift by 3 bits to x (x/2^3) = 62
Right shift by 4 bits to x (x/2^4) = 31
decimal      binary
500 = 111110100
250 = 11111010
125 = 1111101
62   = 111110
31   = 11111
```

right shift by 1 bit ဆိတာ မူလ တန်ဘိုးကို
2 to the power 1 နဲ့ စားခြင်းဖြစ်ပါတယ်
right shift by 2 bits ဆိတာက မူလ တန်ဘိုး
ကို 2 to the power 2 နဲ့ စားခြင်းဖြစ်ပါတယ်

right shift by 2 bits = $x/(2^2)$

WinHtut (Equation)

GreenHackersTeam

Sizeof Operator

သူကတော့ unary operator အမျိုးစားထဲက ဖြစ်ပါတယ်။ unary operator နဲ့ binary Operator တွေ အကြောင်းကို ရှေ့သင်ခန်းစာတွေမှာ ရေးသားခဲ့ပြီးပါပြီ။ sizeof operator က ဘယ်လို လုပ်ဆောင် ပေးနိုင်လဲဆိုတော့ variable သို့မဟုတ် data type ရဲ့ size ကို ဖော်ပြပေးနိုင်ပါတယ်။ ပြောချင်တာကတော့ sizeof operator က သူ့ operand ရဲ့ memory storage space ကို တွက်ချက်ပေးနိုင်တယ်ပေါ့။

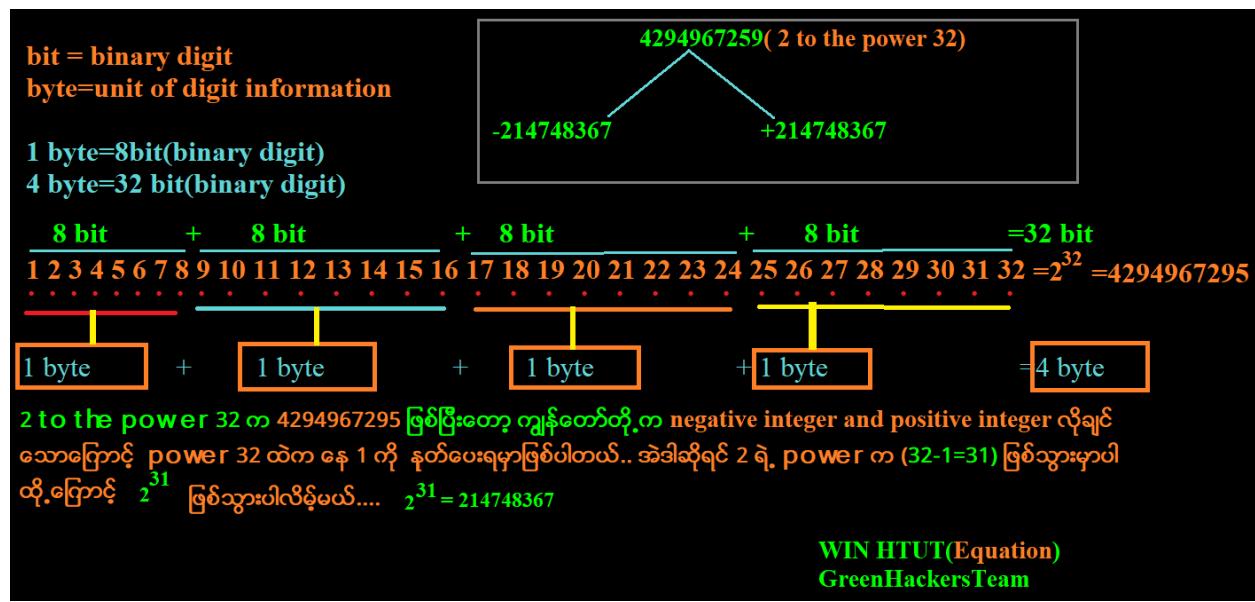
```
#include<stdio.h>
#include<conio.h>
int main() {
    char x;
    printf_s("Memory storage space of char is = %d\n", sizeof x);
    printf_s("Memory storage space of int data type is = %d\n", sizeof (int));

    getch();
    return 0;
}
```

C:\Users\WinHtut_GreenHackers\source\repos\lesson 6\Debug\lesson 6.exe

Memory storage space of char is = 1
Memory storage space of int data type is = 4

အထက်ပါ program လေးကို run ကြည့်မယ်ဆိုရင် အစိမ်းရောင်နဲ့ မြင်နေရတဲ့ output ကို မြင်နေရမှာ ဖြစ်ပါတယ်။ အထက်ပါ program ကိုနားလည်ဖို့ဆိုရင် bit တို့ byte တို့ကို သိထားဖို့ လိုပါတယ်။ 1 byte မှာ ဘယ်နှစ် bit ရှိလည်းဆိုတာ သိထားရပါမယ်။ အရင်သင်ခန်းစာတွေမှာ ဖော်ပြုဖို့ပါတယ်။ အောက်ကပူလေးကို တစ်ချက်ကြည့်ရင် အတော်အသင့် နားလည်လာပါလိမ့်မယ်။



အဲတော့ 1 byte မှာ 4 bit ရှိပါတယ်။ char data type ရဲ့ memory storage space ဟာ 1 byte ဖြစ်တဲ့အတွက် 4 bit ဖြစ်ပါတယ်။ 4 bit ဆိုတာကို မသိသေးရင် အထက်ပါပုံကို သေချာကြည့်ပေးပါ။

Int data type ရဲ့ memory storage space ဟာ 4 byte ဖြစ်ပါတယ်။ အဲတော့ သူက 32 bit ဖြစ်ပါတယ်။ နောက်ထပ် program လေးတစ်ခုလောက် ရေးကြည့်ပြီးရင် ပိုပြီး သဘောပေါက်သွားမှာပါ။

```
#include<stdio.h>
#include<conio.h>
int main() {
    int integerType;
    float floatType;
    double doubleType;
    char charType;

    printf("Size of int: %d bytes\n", sizeof(integerType));
    printf("Size of float: %d bytes\n", sizeof(floatType));
    printf("Size of double: %d bytes\n", sizeof(doubleType));
    printf("Size of char: %d byte\n", sizeof(charType));

    getch();
    return 0;
}
```

Sizeof
operator
ကို

```
C:\Users\WinHtut_GreenHackers\source\repos\lesson 6\Debug\lesson 6.exe
Size of int: 4 bytes
Size of float: 4 bytes
Size of double: 8 bytes
Size of char: 1 byte
```

သုံးတဲ့အခါမှာ data type တွေ ရေးတဲ့အခါကျရင် parentheses ထဲမှာ ထည့်ရေးပေးရပါမယ်။

Break and Continue Statement in C programming

Break and continue statement တွေကို loop ထဲမှာရှိတဲ့ အချို့သော statement တွေကို ကျော်သွားစေချင်တဲ့အခါ မျိုးတွေမှာ သုံးတတ်ကြပါတယ်။ ပြီးတော့ break and continue statement နောက်တစ်နေရာဖြစ်တဲ့ test expression ကို ဆက်မစစ်ဆေးတော့ဘဲ ရှုတ်တရက် ထွက်သွားချင်တဲ့ အခါမျိုးတွေမှာလည်း သုံးတတ်ကြပါတယ်။

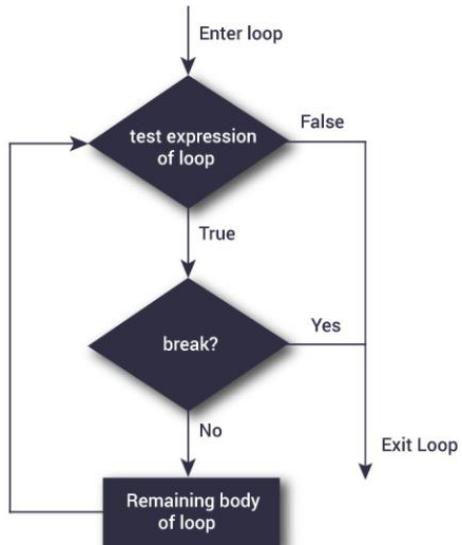
Break Statement

Syntax of break statement

`break;` (break ရဲ့ syntax form ကိုပြောတာပါ)

break statement ၊ loop ပတ်နေရာကင် ထွက်သွားမယ်။ ဥပမာ - for loop , while loop and do....while loop ကင် ထွက်သွားမယ်ပေါ့။ နောက်တစ်ခုက decision making ပြလုပ်တဲ့ if...else statement တွေမှာလည်း သုံးပါတယ်။

Flowchart of break statement



How break statement works?

```

while (test Expression)
{
    // codes
    if (condition for break)
    {
        break;
    }
    // codes
}
  
```

```

for (init, condition, update)
{
    // codes
    if (condition for break)
    {
        break;
    }
    // codes
}
  
```

```

#include<stdio.h>
#include<conio.h>
int main() {
    while (test Expression)
    {
        // another codes
        if (condition for break)
        {
            break;
        }
        // another codes
    }
    getch();
    return 0;
}
  
```

အောင်ပြုပါ program မှာဆိုရင် while ဆဲ
က expression ၊ စစ်ဆေးပြီး အဲဒဲ
while ထဲမှာ ပဲ if ထဲမှာ condition
တစ်ခု ထည့်သွေးပြီး တစ်ကယ်လို့ အဲဒဲ
condition ဖြစ်လာခဲ့ရင် break ကြောင့်
while loop ထဲက အခြားသော code
တွေကို အလုပ်ဆက် မလုပ်တော့ဘဲ
program က ထွက်သွားမှာဖြစ်ပါတယ်

WinHtut
Equation
GreenHackersTeam

```

int main()
{
    int i;
    int number;
    int sum = 0;
    for (i = 1; i <= 6; ++i) {
        printf_s("Please enter subject mark %d: ", i);
        scanf_s("%d", &number);
        if (number < 30)
        {
            break;
        }
        sum = sum + number;
    }
    printf_s("Sum = %d", sum);
    getch();
    return 0;
}

```

ဒဲ program လေးကို ဤညွှန်ပစ်ခိုင် ကျွန်တော်တိ.ကျောင်းမာရီ
major 6 ဘာသာ ရှိတယ် ထားပါတယ်
program ကို စုပ် လိုက်တာ နဲ့ အူကျ ပထမ စာမေ major အဖွတ်တောင်
မယ် ပြီးရင် အခို အမှတ်ကို စစ်ဆေးမယ် စစ်လိုက်လို့ 30 ထက် နည်
းနေရင် သေချာပြီ သုတေသနများ ကျပြောပါ အခါးဖို့ရင် အော် ကို အက်
အလုပ် ပုဂ္ဂိုလ်တော်ဘူး၏ အော် အပြင်ကို ထွက်ပြီး တစ်ခြားသော
code တွေကို ဆက်ပြီး အလုပ် လုပ်သွားမှာ ပါ။
ဒါလေးကတော့ project base လေးပါ။ တစ်ကယ်လို့
ဒဲproject လေးကို တစ်ခြားသော tutorial အဖွတ်တွေ ဘာတွေ
ထပ်ပေါင်းမယ်ဆိုရင် သူအောင်တာ နိုင်လားပါ။ အဲလိမ့်မြတ် ရှုန်း
တွေကို beginner level အနေနဲ့ ရေးကြည့်လိုရပါတယ်
ဒဲနေရာမှာ အဓိက ပြောချင်တာက break ကို သုံးသွားလေးပါ....

ထပ်ပြီးရေးရှင်သေးရင် ပထမ နှစ်ဝက် ကတိယ နှစ်ဝက် အမှတ်တွေ
ကို ပါထည့်ပေးပြီး code လေးလေး နည်နည်းနဲ့ တစ်ကယ် အသုံးပါင်
တဲ့ project လေးတစ်ရုံ စာစစ်တဲ့ လူတွေ အတွက် ထွက်လာမှာ ပါ
အေားသော code တွေက တုံးဆုံးတာ မပါလို့ ဒဲနေရာမှာ တွင်ပဲ
ရပါရဘေး.....

WinHtut(GreenHackersTeam)

နောက်ထပ် switch case break ဆိုပြီးတော့လည်း ကျွန်တော်တိ သုံးနေခဲ့ပါသေးတယ်။ Switch case
break ကတော့ အရင်သင်ခန်းစာတွေမှာ ဖော်ပြုခဲ့ပြီး ဖြစ်ပေသည်။ အဲတော့ switch....case...break
ကိုပဲသုံးပြီး ရှိုးရှင်းတဲ့ calculator လေး တစ်ခုလောက် ရေးကြရအောင်။

```
#include<stdio.h>
#include<conio.h>

int main() {
    char opera;
    double firstNumber, secondNumber;
    printf_s("Enter an operator ( +,-,*,/ ): ");
    scanf_s("%c", &opera);
    printf_s("Please enter Two Numbers:");
    scanf_s("%2lf %2lf", &firstNumber, &secondNumber);

    switch (opera)
    {
        case '+':
            printf("%.2lf + %.2lf = %.3lf", firstNumber, secondNumber, firstNumber + secondNumber);
            break;
        case '-':
            printf("%.2lf - %.2lf = %.3lf", firstNumber, secondNumber, firstNumber - secondNumber);
            break;

        case '*':
            printf("%.2lf * %.2lf = %.3lf", firstNumber, secondNumber, firstNumber * secondNumber);
            break;

        case '/':
            printf("%.2lf / %.2lf = %.3lf", firstNumber, secondNumber, firstNumber / secondNumber);
            break;
        default:
            printf_s("Error! Operator is not correct");
    }
    getch();
    return 0;
}
```

char ဆိတာကတော့ character တွေကို သိလောင်ဖိုပါ။ double ဆိတာကတော့ အသမ ကိန်းတွေကို သိလောင်ဖိုပါ။ If ဆိတာကတော့ အသမကိန်းတွေကို ဖတ်ဖိုပါ။ printf လုပ်တဲ့အချင် မှာသုံးတဲ့ ဒါ If ဆိတာ အသမသုံးနေရာထိ ဖော်ပြုမယ်လို့ ပြောတာပါ...။

continue statement

သူရဲ့ syntax form ကတော့ continue; ပါပဲ။ သူကလည်း loop ထဲမှာရှိတဲ့ statement အချို့ကို ကျော်ဖို့သုံးတယ်။ continue statement ကို if....else ဆိုတဲ့ decision making statement တွေနဲ့ တွဲသုံးကြပါတယ်။ break statement ကကြတော့ ရပ်သွားတယ်။ သူနဲ့ဆန့်ကျင်ဘက် continue

statement ကြေတော့ ဆက်ပြီး အလုပ်လုပ်တယ်ပေါ့။ for..loop ထဲက if ထဲမှာ break ကို
ထည့်ခဲ့သလိုပျိုး continue ကိုလည်း ထည့်ပြီး စမ်းကြည့်ရအောင်။

```
#include<stdio.h>
#include<conio.h>
int main()
{
    int i;
    int number;
    int sum = 0;
    for (i = 1; i <= 6; ++i) {
        printf_s("Please enter subject mark %d: ", i);
        scanf_s("%d", &number);
        if (number < 30)
        {
            continue;
        }
        sum = sum + number;
    }
    printf_s("Sum = %d", sum);
    getch();
    return 0;
}
```

အုပ် program မှာ for loop ထဲမှာ 6 ခါံ ထည့်သွေးထားတယ်။ ထို့ကို program ကို ပြန်ပြင်ရေး စိန်တယ် break မှာ တိုးကဗျာ user input ၁၃၀ ထက် ငါးနေရာ၏ loop ထဲကတော့ ထွက်သွားတယ် အုပ် continue မှာ ကြတော့ program က ထွက်မသွားဘူး။ သတ်မှတ်ထားတဲ့ အမြဲအနေထိ အလုပ် ဆက်လို့

သွားတော့ program ကို run ကြည့်ရင် သိလာပါ လို့မယ်... break statement လုပ် ထွက်လို့ အတော် အသင့် နားလည် သဘောပါက်ပြီလို့ထင်ပါတယ်...

sum=sum + number လို့မရေးချွဲရင်
sum += number လို့ရေးလည်း ရပါတယ်
ရလဒ်တော့ အတွက်ပါပဲ ဘယ်လို့ပဲ ရေးရေး sum လဲကို ပဲ ပေါ်ပေါ်ထည့်သွားမှာပါ

WinHtut Equation GreenHackersTeam

goto Statement

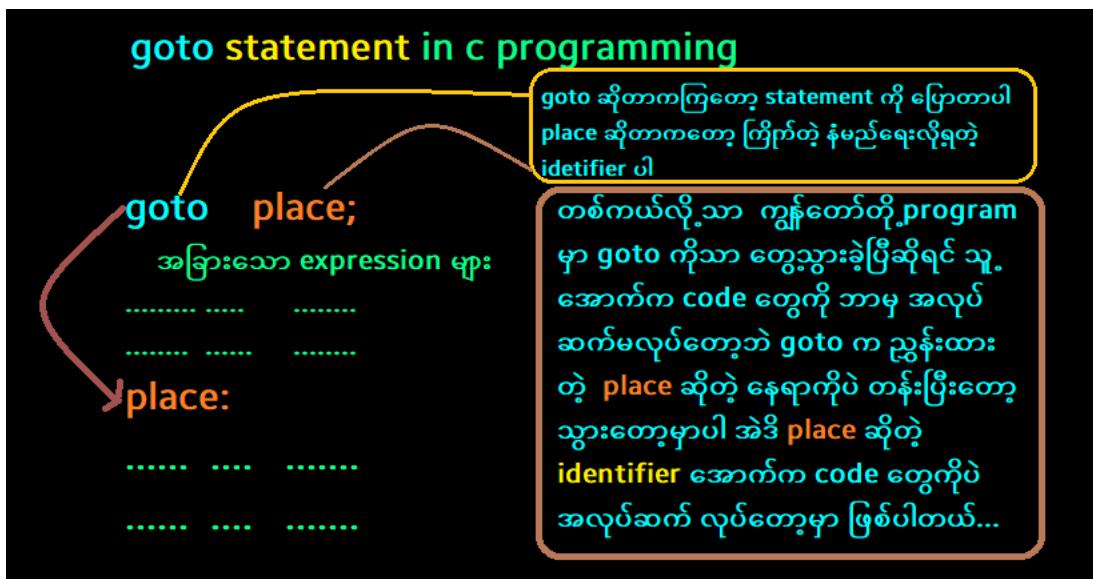
goto Statement အကြောင်းကို ရှင်းပြဖို့ဆိုရင် c programming ရဲ့ keywords နဲ့ identifiers တွေကို သိဖို့လိုပါတယ်။ keywords ဆိုတာ ကြိုပြီး ကြော်ကြော်တွေ ဖြစ်ပါတယ်။ ဥပမာ- int MgMg; မှာဆိုရင် int ဆိုတာက keyword ကိုပြောတာဖြစ်ပြီး MgMg ဆိုတာက variable ဖြစ်ပါတယ်။ ဘယ်လို့ variable လဲဆိုရင် integer data type နဲ့ variable ကို ပြောတာပါ။ c programming က sensitive ဖြစ်တဲ့ language ပါ။ အဲထဲက တစ်ခုကတော့ keywords တွေကို အမြဲတမ်းအသေးနဲ့ပဲ ရေးရတာပါ။ အောက်မှာ ANSI C (American National Standards Institute) က ခွင့်ပြုထားတဲ့ c keywords တွေကို ဖော်ပြထားပါတယ်။ အခြားသော ဒီထက်ပိုတဲ့ keywords တွေလည်း ရှိနိုင်ပါသေးတယ်။ အဲဒါကတော့ compiler ပေါ်မှုမှုတည်ပါတယ်။

Keywords in C Language Allow in ANSI C			
auto	double	int	struct
break	else	long	switch
case	enum	register	typedef
char	extern	return	union
continue	for	signed	void
do	if	static	while
default	goto	sizeof	volatile
const	float	short	

Keywords ကိုသိသင့်သလေက် သိသွားပြီဆိုတော့ identifiers ဘက်ကို ဆက်သွားရအောင်။ identifiers ဆိုတာ နာမည်တွေကို ပြောတာပါ။ ဘယ်လိုနာမည်တွေလဲဆိုရင် function တွေ၊ variables တွေ၊ structures တွေမှာ ပေးတဲ့ နာမည်တွေကို ပြောတာပါ။

ဥပမာပြောရရင် int boy; နဲ့ double number; အခုရေးထားတဲ့ အထဲမှာ boy နဲ့ number ဆိုတဲ့ နာမည်တွေက identifier တွေပါ။ ဒေယားကွက်ထဲက keywords ဖြစ်တဲ့ စားလုံးတွေကိုတော့ identifier တွေအဖြစ် အသုံးပြုလို့မရပါဘူး။

goto statement အကြောင်းကို ပြောတဲ့နေရာမှာ သိတားရမဲ့ keywords တွေနဲ့ identifiers တွေကို သိသွားပြီဆိုတော့ goto statement ကို program တစ်ခုရဲ့ပုံမှန် အလုပ်လုပ်နေတဲ့ sequence ကိုပြောင်းလဲဖို့အတွက် သုံးပါတယ်။



goto statement ဟာ beginner တွေအတွက် program ကိုပြန်ပြီး trace လုပ်ရတာ ခက်ခဲ့စေပါတယ်။
မိမိတို့အဆင်ပြေသလို အသုံးပြုနိုင်ပါတယ်။ အခုဆက်လက်ပြီး Exercise လေးတွေ လုပ်ကြရအောင်။

Example

Example 1: Program to check Even or Odd

Even number နဲ့ odd number ဆိုတာ ငယ်ငယ်က သချ်ရမှာ သင်ခဲ့ဖူးကြပါတယ်။ 2 နဲ့စားလို့ ပြတ်ရင် even number ဖြစ်ပြီးတော့ 2 နဲ့ စားလို့မပြတ်ရင် odd number ဖြစ်ပါတယ်။ အောက်က program လေးကို လေ့လာကြည့်ပါ။

```
#include<stdio.h>
#include<conio.h>
int main() {
    int number;
    printf_s("Please enter an integer: ");
    scanf_s("%d", &number);
    if ((number % 2)== 0)
    {
        printf_s("%d it's even number", number);
    }
    else
    {
        printf_s("%d is odd number", number);
    }
    getch();
    return 0;
}
```

$$\begin{array}{r}
 \text{(Quotient)} \\
 5 \\
 \hline
 \text{(Divisor) } 5) \overline{27} \text{ (Dividend)} \\
 \underline{25} \\
 \hline
 2 \text{ (Reminder)}
 \end{array}$$

သရ program မှာ သုံးထားတာက remainder operator ကို သုံးထားပါတယ် မူးတတ်တာက quotient နဲ့ remainder ပါ ကျိုးတော်တို့တိုက စားလိုပြတ်လား ပြတ်လား ဆိတဲ့ ရိုးရိုး integer ကိုပဲ စစ်မှာ ဖြစ်တဲ့ အတွက် သုံးထားခြင်းဖြစ်ပါတယ် user က နံပါတ် တစ်ခု ရိုက်လည်မယ် အဲ number ကို 2 နဲ့စားမယ် remainder(အကြောင်း) က သုည နဲ့ ညီနေခဲ့ရင် even number ဆိုပြီး ယူလိုက်ခြင်းဖြစ်ပါတယ်...

Example 2: Program to Check Vowel or consonant

Vowel စစ်တဲ့ program လေးစမ်းရေးကြည့်ကရအောင်။

```
#include<stdio.h>
#include<conio.h>

int main() {
    char c;
    int smallvowel, bigvowel;

    printf_s("Please enter a alphabet: ");
    scanf_s("%c", &c);
    smallvowel = ( c=='a' || c == 'e' || c == 'i' || c == 'o' || c == 'u' );
    bigvowel = (c == 'A' || c == 'E' || c == 'I' || c == 'O' || c == 'U');

    if (smallvowel || bigvowel) {
        printf_s("%c is a vowel", c);
    }
    else
    {
        printf_s("%c is a consonant", c);
    }

    getch();
    return 0;
}
```

သူ program မှာ အသုတေသန logical or (||) operators တွေကို သုတေသနတဲ့
တယ် if (smallvowel || bigvowel) အခါနရောမှာ တင်ခြားထော
operator တွေကို သုတေသနတဲ့
program ကိစ်စေရန် အသုတေသနပါတယ်

သရုပ် program မှာဆိုရင် logical or (||) operators တွေကို သုံးတေားပါတယ် if (smallvowel || bigvowel) အဒီဇနာဂျာဘာ တစ်ခြားသော operator တွေကို သုံးပြီးတော့လည်း program စိစစ်ပေးဖြောနိုင်ပါတယ်

Example 3: Finding a largest number

Logical AND (`&&`) operator ကို သုံးပြီး user တည့်လိုက်တဲ့ numbers တွေထဲက အကြီးဆုံး number ကို ရှာဖြန့်ရအေ။

```
int main() {
    double first, second, third, fourth;
    printf_s("Please enter first numbers: ");
    scanf_s("%lf", &first);
    printf_s("Please enter second numbers: ");
    scanf_s("%lf", &second);
    printf_s("Please enter third numbers: ");
    scanf_s("%lf", &third);
    printf_s("Please enter fourth numbers: ");
    scanf_s("%lf", &fourth);
    if ((first > second) && (first > third) && (first > fourth)){
        printf_s("%.2f is the largest number.", first);
    }
    if ((second > first) && (second > third) && (second > fourth)) {
        printf_s("%.2f is the largest number.", second);
    }
    if ((third > second) && (third > first) && (third > fourth)) {
        printf_s("%.2f is the largest number.", third);
    }
    if ((fourth > second) && (fourth > third) && (fourth > first)) {
        printf_s("%.2f is the largest number.", fourth);
    }
    getch();
    return 0;
}
```

numbers නෑත් අඟිහිදී
number ගි ඊහිදී if ගි යුහුදී
&& operator තෝක් රෙපුයා
සිතය

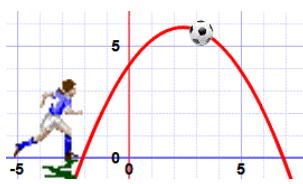
programming ලදාවයුතු
ආක්ෂණික operators අඟ්‍රි
ගොංස්සුම්ප්‍රා පෙන්වනු ලබයි
බැඳීම් සියලු...

Example 4 Find all Roots of a Quadratic Equation

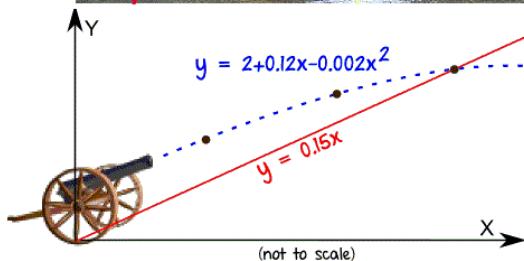
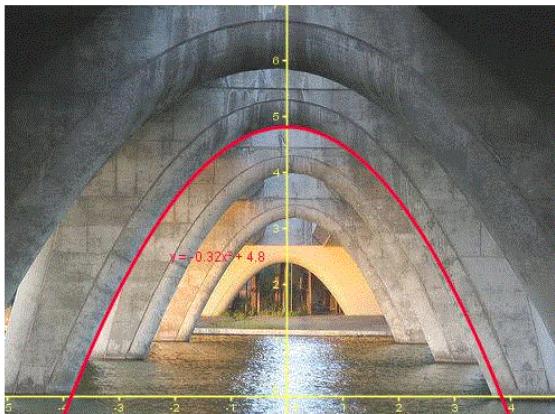
တံတားဆောက်တဲ့ နေရာတွေ၊ ဘေးလုံးကန်တဲ့ နေရာတွေ၊ မျှောင်စင်တွေဆောက်တဲ့ နေရာတွေမှာ
သုံးတဲ့ Quadratic Equation ကို c programming နဲ့အတူ တွဲဖက်ပြီး လေ့လာကြည့်ရအောင်။ အောက်မှာ
ဖော်ပြထားတာကတော့ quadratic equation ဖြစ်ပါတယ်။ quadratic equation အကြောင်းကို
အပြည့်အစုံ သိချင်သေးရင် အောက်ပါ link မှာ သွားရောက်ပြီး လေ့လာနိုင်ပါတယ်။

https://en.wikipedia.org/wiki/Quadratic_equation

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$



Using Quadratic equation in Real Life



$$\begin{aligned} x^2 + \frac{b}{a}x + \left(\frac{b}{2a}\right)^2 &= -\frac{c}{a} + \left(\frac{b}{2a}\right)^2 && \text{Simplify} \\ \left(x + \frac{b}{2a}\right)^2 &= -\frac{4ac}{4a^2} + \frac{b^2}{4a^2} = \frac{b^2 - 4ac}{4a^2} && \text{Square both sides} \\ x + \frac{b}{2a} &= \pm \frac{\sqrt{b^2 - 4ac}}{2a} && \text{Solve for } x \\ x &= \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \end{aligned}$$

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int main() {
    double a = 9;
    printf_s("%lf its square root of a", sqrt(a));

    getch();
    return 0;
}
```

အောက်ပါ program မှာဆိုရင် #include<math.h> ထဲက sqrt() ဆိုတဲ့ square root ရှာတဲ့ function ကို
သုံးထားပါတယ်။

```
#include<stdio.h>
#include<conio.h>
#include<math.h>

int main() {
    double a, b, c;
    double determinant, root1, root2, realPart, imaginaryPart;
    printf_s("Enter coefficients a, b and c: ");
    scanf_s("%lf %lf %lf", &a, &b, &c);
    determinant = b * b - 4 * a*c;
    // condition for real and different roots
    if (determinant > 0)
    {
        // sqrt() function returns square root
        root1 = (-b + sqrt(determinant)) / (2 * a);
        root2 = (-b - sqrt(determinant)) / (2 * a);
        printf_s("root1 = %.2lf and root2 = %.2lf", root1, root2);
    }
    //condition for real and equal roots
    else if (determinant == 0)
    {
        root1 = root2 = -b / (2 * a);
        printf("root1 = root2 = %.2lf;", root1);
    }
    // if roots are not real
    else
    {
        realPart = -b / (2 * a);
        imaginaryPart = sqrt(-determinant) / (2 * a);
        printf_s("root1 = %.2lf+%.2lii and root2 = %.2f-%.2fi", realPart,
imaginaryPart, realPart, imaginaryPart);
    }
}

_getch();
return 0;
}
```

အထက်ပါ program ဟာ programiz.com ဆိုတဲ့နေရာက ဖြစ်ပါတယ်။ သူရေးထားပုံလေး၊ အရမ်းကောင်းလို့ပြန်လည်ဖော်ပြပေးခြင်း ဖြစ်ပါတယ်။

Example 5 Floyd Triangle

```
#include <stdio.h>
#include<conio.h>
int main() {
    int n = 0; user հեղափ քիւթ տաճիւ: Եղանգիւ դաճիւ. Այսու Ֆնաւիսաւաւալ
    int c = 0; զիւթ ացրիւն Լապատիւ ավ լարեթ. Այսու Գնիւթաւաւալ
    int i = 0; սաւ ացրիւն ջուն Լապատիւ ավրիւնաւր Այսու Գնիւթաւաւալ
    int a = 1; առ քարդաւ տաճիւ. Այսու կամաւ այսու Այսու Գնիւթաւաւալ
    printf("Please Enter row numbers For Flyod : \n"); user գր քիւթ ընթիւն Եղանգիւ պատճեն
    scanf_s("%d", &n); զիւթաւ քիւթիւն կամաւ այսու Այսու կամաւ Վերաբարձրաւաւալ
    for (i = 1; i <= n; i++) { user հեղ րասաւ քիւթ ատիւն Լապատիւնաւալ
        for (c = 1; c <= i; c++) { առ մարդ րասաւ ի տաճիւ զիւթ ամիւ լարեթ. Այսու Լապատիւնաւալ
            printf("%d", a); առ տաճիւ գր քամաւալ
            a++; առ տաճիւ գր քամաւալ
        }
        printf("\n"); i տաճիւ 1 քիւթ ա տաճիւ ի տաճիւնաւալ: Այսու 1 վեճուայն ամաւաւալ
    }
    getch(); դաճիւ այսու key տաճիւ գր լաճաւալ
    return 0;
}
```

2 Floyd's Trigle

```
#include <stdio.h>
#include<conio.h>
int main() {
    int n = 0; //number of loop
    int c = 0; // words wirting
    int i = 0; // for loop
    int a = 1; // to show number
    printf("Please Enter row numbers For Flyod :\n");
    scanf_s("%d", &n);
    for (i = 1; i <= n; i++) {
        for (c = 1; c <= i; c++) {
            printf("%d", a);
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

(Global Scope)

c:\users\winhtut_greenhackers\documents\visu

```
Please Enter row numbers For Flyod :
6
1
11
111
1111
11111
```

WIN HTUT(GREEN HACKERS TEAM)

```
#include <stdio.h>
#include<conio.h>
int main() {
    int n = 0;
    int c = 0;
    int i = 0;
    int a = 1;
    printf("Please Enter row numbers For Flyod :\n");
    scanf_s("%d", &n);
    for (i = 1; i <= n; i++) {
        for (c = 1; c <= i; c++) {
            printf("%d", a);
            a++;
        }
        printf("\n");
    }
    getch();
    return 0;
}
```

WIN HTUT(GREEN HACKERS TEAM)

The screenshot shows a Visual Studio code editor with a file named "source.cpp". The code prints Floyd's Triangle. The output window shows the triangle rows:

```

Please Enter row numbers For Flyod :
8
1
23
456
78910
1112131415
161718192021
22232425262728
2930313233343536

```

WIN HTUT(GREEN HACKERS TEAM)

Example 6 Lottery Game Part 1

အောက်မှာ ရေးပြထားတာကတော့ lottery game လေးဖြစ်ပါတယ်။ အရမ်းရှိုးရှင်း ပါတယ်။ Part 1 ကို
မိမိတို့လိုသလို ထပ်ပြီး ပြင်ဆင်ရေးသားလို့ရပါသေးတယ်။

The screenshot shows a Visual Studio code editor with a file named "lottery game". The code is a simple lottery game. The output window shows the game's interaction:

```

Please enter your money:1000
Please enter a number: must be less than 109
Please enter your lottery number:12
Try again.. You are unlucky day
Please enter your lottery number:50
Try again.. You are unlucky day
Please enter your lottery number:10
you are win in lottery:0
1*
2**
3***
4****
5*****
6*****
7*****
8*****
9*****
Get your money: 8000

```

A green callout box highlights the following features:

- Part 1 ဖြစ်တဲ့အတွက် အရေးပို့ရှင်းအောင် ရေးတားပါတယ်
- ဒိုးထိုးလို့ control လုပ်လိုပါ
1. user name and password များဖြင့်ထိုးခွင့်ပေးခြင်း
2. နှစ်လုံးထိုးလို့ ငွေမောက်လို့ ကနိုယ်တော်းမြှင့်
3. နှစ်လုံးထိုးလို့ ငွေမောက်လို့ ကနိုယ်တော်းမြှင့်
4. ကတေသနလို့ နှစ်လုံးထိုးလို့ သူတော်းမြှင့်
5. နှစ်လုံးထိုးလို့ အသင့်မြှင့်တဲ့ သူတော်းထိုးလို့ ငွေမောက်လို့ ကနိုယ်တော်းမြှင့်
6. user တွေအတွက် chance များအောင် ဘယ်လို့ ရေးပေးခြင်း
7. user တွေကို game ကို ဆွဲလမ်းလာအောင် ဘယ်လို့ ဆွဲလမ်းလာ
8. ဘဏ်. ပေါက်သွားခဲ့ရင် ပမာဏ အလိုက် pattern ပြောင်းလဲပေးခြင်း
9. user တွေကို ထိုးကြုံးများဖြင့် ဘယ်နား ပေးပေးမှု
- ဝါဒ္ဒေါ် အရှင်တွေကို ထပ်မံပြည့်စွဲပေးလို့ရပါတယ်....
- နောက် lesson မှာလည်း ရေးသားနည်းကို ပြောပြီးမော်

Win Htut (equation)
GreenHackersTeam

```
#include<stdio.h>
#include<conio.h>

int main() {
    int x;
    int money;
    printf_s("Please enter your money:");
    scanf_s("%d", &money);
    printf_s("Please enter a number: must be less than 10");
    scanf_s("%d", &x);
    while (x < 10) {
        int y;
        printf_s("Please enter your lottery number:");
        scanf_s("%d", &y);
        if (y == 10) {
            printf_s("you are win in lottery:");
            for (int i = 0; i < 10; i++) {
                printf_s("%d", i);
                for (int z = 0; z < i; z++) {
                    printf_s("*");
                }
                printf_s("\n");
            }
            int y = money * 8;
            printf_s("Get your money: %d", y);
            break;
        }
        else
        {
            printf_s("Try again.. You are unlucky day\n");
        }
    }
    getch();
    return 0;
}
```

Function ချေသားပုံ

ပြန်ပေးမယ့် data type အမျိုးအစား function name (parameter-list)

```
{
    ကိုယ်ရေးသားလိုသော အကြောင်းအရာများ
}
```

အောက်က ပုံလေးမှာဖော်ပြထားပါတယ်။

```

1  #include<stdio.h>
2  #include<conio.h>
3
4  int square(int y); //မိအောက်မှာ ရေးသားထားသော ဂုဏ်အတိုင်း function prototype
5  // ကြော်ပေးရမယ် line number 16
6  int main(void)
7  {
8      for (int i = 0; i < 10; i++) // argument
9      {
10         printf_s("%d\n", square(i)); // program က ဒီနေရာကို ရောက်တာနဲ့ line number 16 ကိုလှစ်းခေါ်တယ်
11     }
12     getch(); //နောက်ပေးမယ် data type
13 }
14
15 int square(int y) //ကိုယ်ပေးခြင်သော function name
16 { //function ခဲ့ parameter
17     return y * y; //မိရေးသားထိုသော ဂုဏ်
18 }

```

WinHtut(GreenHackersTeam)

```

1  #include<stdio.h>
2  #include<conio.h>
3
4  int square(int y); // function prototype ကြော်ပေးခြင်း
5
6  int main(void)
7  {
8      //loop 10 ကြိမ်ပတ်ပြီးတော့ i တန်ဖိုးတွေ ထုတ်ပေးမယ်
9      for (int i = 0; i < 10; i++)
10     {
11         printf_s("%d\n", square(i)); // squareဆိုဝဲ့ function ကို
12         //လှစ်းခေါ်တယ် line number 18
13     }
14     getch();
15 }
16
17 // int y ဆိုတဲ့နေရာမှာ i ခဲ့ တန်ဘိုးကို ထည့်ပြီးတွက်လိုက်တယ်
18 int square(int y)
19 {
20     return y * y; // y ကို နစ်ခါမြှောက်ပြီး သူတာန်ဘိုးကိုပြန်ပို့ပေးလိုက်ပါတယ်
}

```

0
1
4
9
16
25
36
49
64
81

↑
OUTPUT

WinHtut

Avoidable Error

function တစ်ခုကို အခြား function တစ်ခုထဲမှာ သွားရောက်ပြီး ကြော်ပြာလျှင် syntax error ဖြစ်နိုင်သည်။

meaningful ဖြစ်တဲ့ function names တွေ meaningful ဖြစ်တဲ့ parameter names တွေ
ပေးခြင်းအားဖြင့် programs ကို ဖတ်ရလွယ်ကူစေတယ်၊ comments တွေ အများကြီးပေးစရာလည်း
မလိုတော့ပါဘူး။

small functions လေးတွေ ရေးသားခြင်းအားဖြင့် programs ပြန်ရေးဖို့ အများရှာဖွံ့ဖြိုး ပြန်လည်
ပြင်ဆင်ဖို့ ပိုကောင်းအောင်ရေးဖို့ လွယ်ကူစေပါတယ်။

```
#include<stdio.h>
#include<conio.h>

int calculate(int x, int y, int z, int zz);
int main(void)
{
    int number1, number2, number3, number4;
    printf_s("%s", "Please enter four integers:_");
    scanf_s("%d%d%d%d", &number1, &number2, &number3, &number4);

    printf_s("The result is: %d\n", calculate(number1, number2, number3,
number4));

    _getch();
}
int calculate(int x, int y, int z, int zz) {
    return (x + y + z) / zz;
}
```

ဒဲ program လေးကိုကြည့်မယ်ဆိုရင် number1 ဆိုတဲ့ argument က x ဆိုတဲ့ parameter ထဲမှာ လာပြီး stored လုပ်ပါတယ်။ number 2 ဆိုတဲ့ argument ကလည်း y ဆိုတဲ့ parameter ထဲ မှာ လာပြီး stored လုပ်ပါတယ်။ထိုကဲ့သို့ပဲ number3, number4 စတဲ့ argument တွေကလည်း z, zz ထဲမှာ အစဉ်အတိုင်း stored လုပ်ကြပါတယ်။ တန်ဖိုးတွေကို ပြန်တွက်ချက်ပြီး သတ်မှတ်ထားတဲ့ data type အတိုင်းပဲ သူ့ကို လုမ်းခေါ်တဲ့နေရာကို ပြန်ပိုပေးလိုက်ပါတယ်။ အောက်ကပုလေးမှာ ဖော်ပြထားပါတယ်။

1 #include<stdio.h>
2 #include<conio.h>
3
4 int calculate(int x, int y, int z, int zz);
5 int main(void)
6 {
7 int number1, number2, number3, number4;
8 printf_s("%s", "Please enter four integers:_");
9 scanf_s("%d%d%d%d", &number1, &number2, &number3, &number4);
10
11 printf_s("The result is: %d\n", calculate(number1, number2, number3, number4));
12
13 _getch();
14 }
15 int calculate(int x, int y, int z, int zz) {
16 return (x + y + z) / zz;
17 }

scanf ဖတ်လိုက်လာတဲ့ argument တစ်ခုပြင်းစီက
အခေါ်ရတဲ့ function ရဲ့ parameter တွေနေရာမှာ
အစဉ်တိုင်း stored လုပ်တယ်ဆိုတာကို သိတားရပါ
မယ်။

တွက်ရှုရန်လိုလာတဲ့ တန်ဖိုးပါ သူတို့ ခေါ်တဲ့
calculate နေရာကို ပြန်ပိုပေးလိုက်ပါတယ်။

C:\Users\WinHtut_GreenHackers\source\repos\Function more parameter\Debug\Function more parameter.exe
Please enter four integers:_10 20 30 10
The result is: 6
WinHtut(GreenHackersTeam)

Random Function

အခုဆက်ပြီး C standard Library function ဖြစ်တဲ့ rand() function အကြောင်းလေးကို ဆက်သွားပါမယ်။ rand() ဆိုတဲ့ function ရဲ့ လုပ်ဆောင်ချက်က သူက random numbers တွေ ထုတ်ပေးတာပါ။ ကျွန်တော်တို့ Microsoft Visual Studio ပေါ်မှာ ရေးတဲ့ အတွက် သူရဲ့ အများဆုံးထုတ်နိုင်တဲ့ maximum value က two-byte ဖြစ်ပါလိမ့်မယ်။ two-byte ဆိုတော့ 16 bit ဖြစ်ပြီး 32767 ဖြစ်ပါတယ်။ rand() function ကို သုံးမယ်ဆိုရင် stdlib.h ဆိုတဲ့ header file ကို အရင်ဆုံး ကြောင့် ပေးထားရပါမယ်။ ဘာလို့လည်းဆိုတော့ rand() ဟာ stdlib.h ထဲမှာ ပါဝင်သောကြောင့် ဖြစ်ပါတယ်။

Example 3

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>

int main() {
    for (int i = 1; i <= 30; ++i) {

        printf_s("%5d", 1+(rand()%6));

        if (i % 5 == 0)
        {
            puts("");
        }
        _getch();
    }
    return 0;
}
```

#include<stdlib.h> rand() ဆိုတဲ့ function ကို သုံးချင်လို့ ရေးထားတာပါ။ for (int i = 1; i <= 30; ++i) counter အကြိမ်သုံးဆယ် ပတ်မယ်လို့ ရေးထားတာပါ။ printf_s ("%5d", 1+(rand()%6)); counter တစ်ကြိမ်ပတ်တိုင်း random number တစ်ခါထုတ်မှာပါ။ 5d လို့ရေးသားထားတာက random number မရေးခင် space ငါးခုခြားမယ်လို့ ပြောထားတာ ပါ။ 1+(rand()%6) ဆိုတာကတော့ remainder operator ကိုသုံးပြီးထွက်လာမယ့် random number တွေကို scaling လုပ်တာပါ။ ဆိုလိုချင်တာက Random number ဟာ 1 ကနေ 6 အထိပဲ ပါချင်တာပါ။

```

1  #include<stdio.h>
2  #include<conio.h>
3  #include<stdlib.h>
4
5  int main() {
6      for (int i = 1; i <= 30; ++i) {
7
8          printf_s("%5d", 1+(rand()%6));
9
10         if (i % 5 == 0)
11         {
12             puts("");
13         }
14     }
15     getch();
16     return 0;
17 }
```

C:\Users\WinHut_GreenHackers\source\repos\Lesson 7_3 random number\Debug\Lesson 7_3 random number.exe

6	6	5	5	6
5	1	1	5	3
6	6	2	4	2
6	2	3	4	1
4	1	3	4	5
5	4	3	3	6

output ကို ဖြည့်လိုက်ရင် 1 ကနေ 6 ကြားထဲမှာ ရှုပါတယ်။
အကြိမ် သုံးပါတယ် counter လုပ်ထားတဲ့ အတွက် random အလုံးပေါင်း 30 ရှုပါတယ်။% နဲ့ d နဲ့ကြားထဲမှာ ကိုယ်ခြား ချင့်တဲ့ space အရေး
အတွက်တိုင်းထည့်လိုရပါတယ်။ line number 10 မှာ 5 နဲ့ စားထားတာ
ဖြစ်တဲ့အတွက် number 5 ရေးပြီးတိုင်း အောက်ကို တစ်ကြောင်းဆင်းပါတယ်။
ပါတယ် အဲအတွက် puts(); ဆိုတဲ့ function ကို သုံးထားပါတယ်။
တစ်ကြောင်းလို့ 6 နဲ့သာ စားခဲ့မယ်ဆိုရင် number 6 လုံးရေးပြီးမှ
အောက်ကို ဆင်းသွားမှာပါ

ဥပမာ - rand() ကို 10 ထွက်လာခဲ့ရင် %6 နဲ့ စားလိုက်တော့ အကြောင်းက 4 ပါ။ အဲဒီ 4 ကို ရှောက 1 နဲ့
ပေါင်းလိုက်တော့ 5 ဖြစ်သွားပါတယ်။ အဲလိုမျိုး အကြိမ်သုံးဆယ် random number
ထုတ်သွားမှာဖြစ်ပါတယ်။ if (i % 5 == 0) တကယ်လို့သာ i ကို 5 နဲ့စားပြီး ရလာတဲ့ တန်ဖိုးဟာ zero
နဲ့ညီနေခဲ့ရင် တစ်ကြောင်း ဆင်းမယ်လို့ ပြောတာပါ။ puts(""); အောက်ကို တစ်ကြောင်း ဆင်းတာပါ။
အောက်မှာ ပုံပြထားပါတယ်။ အထက်ပါ ပုံတွင် return(); ကို puts(); ဟုပြင်ဖတ်ပေးပါ။

Example 4

နောက်ထပ် lesson တစ်ခု အနေနဲ့ counter ကို အကြိမ်ရေ 5000 လောက်ထားပြီး သူရဲ့ scale ကိုတော့ 1
ကနေ 5 အထိပဲ ထားပါမယ်။ အကြိမ်ရေ 5000 ပတ်တဲ့ အချိန်မှာ random number တွေ ဖြစ်တဲ့ 1-2-3-4-
5ကို 1 က ဘယ်နှစ်ကြိမ်ကျသလဲ 2 က ဘယ်နှစ်ကြိမ်ကျ သလဲ ဆိုတာကို ရေတွက် တဲ့ program
လေးရေးကြည့်ပါမယ်။ switch case break ကိုအသုံးပြုပြီး number တွေကို စစ်ပါမယ်။
တကယ်လို့ random number က 1 ဖြစ်နေရင် 1 ဘယ်နှစ်ကြိမ်ကျ သလဲဆိုတာကို ပေါင်းထည့်တဲ့နည်းကို
အသုံးပြုသွားပါမယ်။ အောက်က program လေးကို ကြည့်လိုက်ရင် ပိုပြီးတော့ ရှင်းသွားပါလိမ့်မယ်။


```

    }
}

printf_s(" %s%10s\n", "Number", "Frequent");
    %s မ အောက် မြော ၁- အတွက် ၂ အတွက် အောက် မ printf ၏ ၁ အတွက် အောက်
    %10s မ အောက် မြော ၁- Number ၏၏ Frequent ၏ ၁ အတွက် အောက် မ space 10
    အောက် မြော

printf_s(" 1%10d\n", onetime);
    1၏ အောက် မြော ၁- print ၏ အောက် ၁ - အောက်

printf_s(" 2%10d\n", twotime);
printf_s(" 3%10d\n", threetime);
printf_s(" 4%10d\n", fourtime);
printf_s(" 5%10d\n", fivetime);

_getch();
return 0;
}

```

Output of Example 4

Number	Frequent
1	1012
2	986
3	986
4	1062
5	954

ကျွန်တော်တို့ အထက်ပါ ရေးထားတဲ့ program မှာ random number တွေက တစ်ပုံစုတည်း ကျနေပါတယ်။ အဲလို မကျအောင် user ဆီက input တစ်ခုတောင်းပြီး program ကို ရေးကြည်လို့ရသလို နာက်တစ်မျိုးက time function ကို အသုံးပြုပြီးလည်း ရေးသားနိုင်ပါတယ်။ (ပိုပြီး သိချင်တဲ့သူတွေက အောက်က စာလုံးစောင်းရေးထားတာကို ဖတ်လို့ရပါတယ်။ beginner များအနေနဲ့ မဖတ်ပဲ ကျော်ဖတ် သွားလို့ ရပါတယ်။)

rand() function ဟာ random number တွေကို အနိမ့်ဆုံး zero ကနေပြီးတော့ အမြင့်ဆုံး maximum အထိ ထုတ်ပေးပြီးမှ ကျွန်တော်တို့ လိုချင်တဲ့ပုံစုအတိုင်း scale ပြန်လုပ်တာပါ။ သူကို pseudo-random integral number လိုလဲ ခေါ်ပါတယ်။ သူမှာ Linear Congruential Generator ဆိတဲ့ algorithm ရှိပါတယ်။ အဲဒါ algorithm ကနေပဲ pseudo-random numbers တွေ ထုတ်ပေးပါတယ်။

$X_0 = \text{seed}$ (ပထမဆုံး တန်ဖိုးတစ်ခုပေါ့)

$X_{n+1} = (A * X_n + B) \% C$

Example 5

ယခု program မှာလည်း <stdlib.h> ထဲမှာပါတဲ့ srand() ဆိုတဲ့ function ကို အသုံးပြုသွား ပါမယ်။

နောက်တစ်ခုက <time.h> ထဲမှာပါတဲ့ time() function ကိုလည်း အသုံးပြုသွားပါမယ်။

srand()function ကို သုံးခြင်းအားဖြင့် သူက initialize value တစ်ခုကို ထုတ်ပေးမှာ ဖြစ်ပါတယ်။

```
//source code
int main()
{
    srand(time(NULL));
    int i;

    for (i = 0; i < 20; i++)
        printf("%10u", rand() % 10);
    getch();
}
```

```

#include<stdio.h>
#include<stdlib.h>
#include<time.h> // time function - 用于生成随机数的函数
#include<conio.h>

int main()
{
    srand(time(NULL)); // time function - 用于生成随机数的函数 computer 通过调用此函数来生成随机数
    // second - 用于指定 rand() 函数生成随机数的种子值
    // value - 用于指定 rand() 函数生成随机数的范围
    // change - 用于更改 rand() 函数的种子值
    // srand - 用于生成随机数 / 初始化值
    // random number / initialize value
    // rand() function - 用于生成随机数
    // random numbers
    // run - 用于运行
    int i;

    for (i = 0; i < 20; i++)
        printf("%10u", rand() % 10);
    getch();
    return 0;
}

```

Output 1

9	3	5	1	0	9	0	5	1	7	7	4	2
9	8	2	3	7	9	5	1	1	7	7	4	2

Output 2

1	9	7	8	8	7	8	1	4	4	6	9
7	3	1	3	9	6	3	4	4	4	6	9

Example 6

```

#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
#include<time.h>

int main() {
    int computerNo, humanNo;
    srand(time(NULL));
    computerNo = rand() % 10 + 1;

    do {

```

```

printf_s("%s", "Number 1-10:");
scanf_s("%d", &humanNo);

if (computerNo < humanNo) {
    puts("computerNo number is lower");
}
else if (computerNo > humanNo) {
    puts("humanNo is Lower");
}
} while (computerNo != humanNo);
printf_s("%s", "They are same value");

_getch();
return 0;
}

```

Output

```

Number 1-10:10
computerNo number is lower
Number 1-10:1
humanNo is Lower
Number 1-10:2
humanNo is Lower
Number 1-10:3
humanNo is Lower
Number 1-10:4
humanNo is Lower
Number 1-10:5
humanNo is Lower

```

```

Number 1-10:6
humanNo is Lower
Number 1-10:11
computerNo number is lower
Number 1-10:7
humanNo is Lower
Number 1-10:8
humanNo is Lower
Number 1-10:9
They are same value

```

Example 5 နဲ့ Example 6 ကသိပ်မကွာပါဘူး။ ရှေ့ပိုင်း lessons တွေမှာ အသေးစိတ်ရေး ပြထားပြီးပြီး အခုဆက်ပြီး game လေးတစ်ခု ရေးကြည့်ကြပါမယ်။

Example 7 Dice Game

Enumerations ကို အခု example မှာစပြီး သုံးသွားပါမယ်။ Enumeration ရဲ့ keyword ကတေသ့ enum ပါ။ သူကို programmer တွေက များသောအားဖြင့် integer constants အနေနဲ့ Define လုပ်ပြီး သုံးကြပါတယ်။ enum ကိုသုံးခြင်းအားဖြင့် program ပိုရှင်းမယ်၊ ဖတ်လိုလည်းပို လွယ်သွားမယ်၊ အမှားရှာဖို့တွေ ပြန်ပြင်ဖို့တွေလည်း လွယ်သွားပါမယ်။

enum Game { CONTINUE, WON, LOST }; ကြော်လွှာတဲ့အခါ နောက် constants အကြီးနဲ့ရေးရင် ပိုကောင်းပါတယ်။ မဟုတ်ရင် သူတို့ကို ကိုယ်စားပြုထားတဲ့ Game ဆိုတဲ့ variable နဲ့ ရှုပ်သွားနိုင်ပါတယ်။

Lesson7_1

```
#include<stdio.h>
#include<conio.h>
enum colors {BLACK, BLUE, GREEN, CYAN, RED, YELLOW, WHITE};
enum colors to_out; // to_out ဆိုတဲ့ variable တစ်ခုထပ်ကြော်လွှာလိုက်ပါတယ်။
```

```
int main()
{
    enum colors to_out; // to_out ဆိုတဲ့ variable တစ်ခုထပ်ကြော်လွှာလိုက်ပါတယ်။
```

```
    to_out = YELLOW; // YELLOW ရဲ့ location ကို လျမ်းပြီးတော့ ထုတ်ဖို့အတွက်ပါ။
```

```
    to_out ဆိုတဲ့ variable က YELLOW ရှိတဲ့နေရာကို ကိုယ်စားပြုသွားပါပြီ။
```

```
    printf("%5d",to_out);
    getch();
    return 0;
}
```

Output အနေနဲ့ 6 ဆိုတဲ့ number တစ်ခုတည်းကိုပဲ ထုတ်ပေးမှာဖြစ်ပါတယ်။

အကိုးအကားများ

1 <https://www.geeksforgeeks.org/interesting-facts-default-arguments-c/>

2 https://en.wikipedia.org/wiki/Enumerated_type

Scope Rules(variables)

Variable တစ်ခုရဲ့ scope rule ဆိုတာဘာလဲ?

Program ရေးတဲ့အခါမှာ variables တွေ ပုံစံ 2 မျိုးနဲ့ကြော်လွှာပြီး သုံးကြပါတယ်။

1. Local Variables

2. Global Variables

Local Variables

Local variables တွေဆိုတာ function တစ်ခုအတွင်း သို့မဟုတ် code block ({...} အခုလို bracket အတွင်းမှာ) ကြော်လှာခြင်းဖြစ်ပါတယ်။ Local variables တွေကို သူကို ကြော်လှာထားတဲ့ function or code block အတွင်းကနေပဲ ခေါ်ယူလို့ရပါတယ်။ အခြားသော function တွေကနေ လုမ်းခေါ်လို့မရပါ။ Local variables တွေက သူကို ကြော်လှာထားတဲ့ function or code block ကနေပဲ သိပါတယ်။ အခြားသော function or code block တွေကနေ သိမှာ မဟုတ်ပါဘူး။ Local variable တစ်ခုကို dev c++ လို့ ide နဲ့ရေးကြည့်ပြီး print ထုတ်ကြည့်မယ်ဆိုရင် Garbage values တွေ initialize assign လုပ်ပေးထားတာ တွေ့ရပါလိမ့်မယ်။

Example 1 local and global

```
#include<stdio.h>
#include<conio.h>

int year = 2018;//global variable ဖြစ်ပါသည်။သူကို
                //အပြင်ဘက်ဆုံးမှာကြော်လှာပေးရပါသည်။

int main() {

    int age = 23;// Local variable ဖြစ်ပါသည်။ main ဆိုတဲ့ function
                // တစ်ခုအတွင်းမှာကြော်လှာထားပါသည်။

    printf_s("GlobalVar= %d, LocalVar=%d",year,age);
    int test();
    _getch();
    return 0;
}
```

Example 2 garbage value ထွက်လာတာကို dev c++ ide နှင့်စမ်းပြခြင်း

```
#include<stdio.h>
#include<conio.h>

int x;
int main(){
    int y;
    printf("Glo-%d Loc-%d",x,y);
    getch();
    return 0;
}
```

Select C:\Users\WinHtut_GreenHackers\Documents\Untitled1.c
Glo-0 Loc-54.

Program နဲ့ output ကိုတဲ့ကြည့်ရင် ရှင်းသွားပါလိမ့်မယ် ။just for knowledge ပါ။ ကိုယ်သုံးတဲ့ ide အပေါ်မှာ မူတည်ပြီး result တွေပြောင်းလဲနိုင်ပါတယ်။

Example 3 Using local variable

```
#include<stdio.h>
#include<conio.h>
```

```
float AreaOfRectangle(float length, float width);
/*function prototype ကိုကြောပေးခြင်း */
```

```
int main() {
```

```
    float length, width; /*main function ရဲ့local variables တွေကိုကြောခြင်း*/
```

```
    printf_s("Enter Length and width of Rectangle\n");
```

```
    scanf_s("%f%f", &length, &width);
```

```
    /*user ဆိုက value တွေကို ဖတ်ပြီး variable ထဲကို assign လုပ်ခြင်း*/
```

```
    printf_s("Area of Rectangle = %f",AreaOfRectangle(length,width));
```

```
    /*AreaOfRectangle ဆိုတဲ့ function ထဲကို value တွေ assign လုပ်ခြင်း*/
```

```
_getch();
```

```
return 0;
```

```
}
```

```
float AreaOfRectangle(float len, float wid) { /*function      တစ်ခုကြော်ခြင်း*/
```

```
    float area;
    area = len * wid;
    return area; /*area တန်ဖိုးကို ပြန်ပေးခြင်း*/
}
```

အထက်ပါ program မှာဆိုရင် function နှစ်ခု ရှိပါတယ်။ function တစ်ခုခြင်းစီမှုလည်း သူတို့ရဲ့ Variables တွေ ရှိပါတယ်။ main ထဲက variable တွေကို AreaOfRectangle ကနေ လုမ်းခေါ်သုံးလို့ မရသလို AreaOfRectangle ဆိုတဲ့ function ကနေလဲ main ထဲက variables တွေကို လုမ်းခေါ်သုံးလို့မရပါဘူး။ အဲဒီအချက်က local variables နဲ့ global variables နှစ်ခုကြားထဲက ကွာခြားချက်ပါ။ Global variable ကို program ရဲ့အပြင်ဘက်ဆုံး အပိုင်းမှာ ကြော်ဖြေပြီး သူကိုမည့်သည့် function ကနေမဆို လုမ်းခေါ်သုံးလို့ရပါတယ်။

2.Global Variables

Global variables တွေကို function တွေရဲ့အပြင်မှာ ကြော်ဖြေပြီးတာနဲ့ global variables တွေကို ကြော်ဖြေပြီးတာနဲ့ တို့မည့်သည့်နေရာကနေမဆို လုမ်းခေါ်ပြီးသုံးလို့ရပါတယ်။

Function တွေကနေပြီးတော့ Global variables တွေကိုလုမ်းခေါ်ပြီး လိုအပ်သလို ပြုပြင်သုံးလို့ရပါတယ်။ ဥပမာ - Arithmetic operators များဖြင့် တွဲသုံးခြင်း။

Global variables တွေကို system ကနေပြီးတော့ initialize value ဖြစ်တဲ့ zero ကို assign လုပ်ပေးပါတယ်။ အပေါ်က ပုံနဲ့ program မှာ ပြထားပါတယ်။ int, float, double တွေကိုလည်း သက်ဆိုင်ရာ value တွေ assign လုပ်ပေးပါတယ်။ Pointer variable ဖြစ်နေခဲ့မယ်ဆိုရင်တော့ NULL ကို assignလုပ်ပေးမှာ ဖြစ်ပါတယ်။

Local Variable နဲ့ global variable တွေက နာမည်တူနေခဲ့မယ်ဆိုရင် global variable ရဲ့ value ဟာ မည့်သည့်တန်ဖိုး ဖြစ်နေပါစေ local variable ရဲ့ value အတိုင်းပဲ assign လုပ်ပါတယ်။ အောက်မှာ ပုံလေးပြထားပါတယ်။

```

1 #include<stdio.h>
2 #include<conio.h>
3
4     int x=2;
5     int main(){
6         int x=5;
7
8         printf("%d %d",x,x);
9         getch();
10        return 0;
11    }
12

```

Example 4 Using global variable

```
#include<stdio.h>
```

```
#include<conio.h>
```

```
float AreaOfRectangle(float length, float width);
```

/*function prototype ကိုကြော်ပေးခြင်း */

```
int extra = 10;
```

```
int main() {
```

```
float length, width; /*main function ရဲ့ local variables တော်ကြော်ပြုခြင်း*/
```

```
printf_s("Enter Length and width of Rectangle\n");
```

```
scanf_s("%f%f", &length, &width);
```

/*user ဆိုတဲ့ value တွေကို ဖတ်ပြီး variable ထဲကို assign လုပ်ခြင်း */

```
length = length + extra; /* global variable ဖြစ်တဲ့ extra ကို main function
```

ကနေ လုပ်းခေါ်သုံးခြင်း*/

```
width = width + extra; /* global variable ဖြစ်တဲ့ extra ကို main function
```

ကနေ လုပ်းခေါ်သုံးခြင်း*/

```
printf_s("Area of Rectangle = %f",AreaOfRectangle(length,width));
```

/*AreaOfRectangle ဆိုတဲ့ function ထဲကို value တွေ assign လုပ်ခြင်း */

```
_getch();
```

```
return 0;
```

```
}
```

```
float AreaOfRectangle(float len, float wid) { /*function တစ်ခုကြော်ပြုခြင်း */
```

```

float area;
area = len * wid + extra; /*global variable ဖြစ်တဲ့ extra ကို AreaOfRectangle function ကနေ
လုမ်းခေါ်ခြင်း*/
return area; /*area တန်ဖိုးကိုပြန်ပေးခြင်း*/
}

```

အထက်ပါ program မှာဆိုရင် extra ဆိုတဲ့ Global variable ကို ကျွန်တော်တို့ function တွေကနေ လုမ်းပြီး ခေါ်သုံးထားပုံကို ပြထားတာပါ။

Calling Function Argument

C program တစ်ခုမှာ Function Call ခေါ်တော့မယ်ဆိုရင် ပုံစံနှစ်မျိုးနဲ့ ခေါ်လဲ ရှိကြပါတယ်။

- 1.Call by value
- 2.Call by reference

ပထမဆုံးအနေနဲ့ Formal Parameters and Actual Parametersကို သိထားရပါမယ်။ Formal Parameters ဆိုတာ function တစ်ခုကို ကြော်လှတဲ့အချင်း သူရဲ့ body ထဲမှာ ရေးသားလိုက် arguments တွေဖြစ်ပါတယ်။

ဥပမာ - void formalparameter(int F); အခုရေးထားတဲ့ function မှာ int F ဆိုတာ အဲဒီ function ရဲ့ body မှာရေးထားတဲ့ Formal parameter ဖြစ်ပါတယ်။

Actual Parameters ဆိုတာ function တစ်ခုကို လုမ်းခေါ်တဲ့ အချင်းကြမှ သုံးတဲ့ parameter ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ ဥပမာ - formalparameter(A); မှာ A ဆိုတာက Actual parameter ဖြစ်ပါတယ်။

Example 5 Function တစ်ခုအား value ဖြင့်ခေါ်ဆိုခြင်း

```

#include<stdio.h>
#include<conio.h>

void somevalue(int F);

int main() {
    int A;
    printf_s("Please enter a value\n");
    scanf_s("%d", &A);
}

```

```

/* somevalue ဆိုတဲ့ function ကို value တစ်ခုနဲ့လှမ်းခေါ်လိုက်တာပါ
A သဲ actual parameter ဖြစ်ပါတယ်*/
somevalue(A);

printf_s("A(Actual Parameter) = %d\n", A);

_getch();
return 0;
}

void somevalue(int F) { /*int f သဲ formal parameter ဖြစ်ပါတယ်*/
    F = F * 2;
    printf_s("F(Formal Parameter) = %d\n", F);
}

```

Extra Knowledge (formal parameter နဲ့ acutual parameter သဲ memory မှာ allocated မတူပါဘူး)

Example 6 Function တစ်ခုအား reference ဖွင့်ခေါ်ခြင်း

```

#include<stdio.h>
#include<conio.h>

void somevalue(int *F);

int main() {
    int A;
    printf_s("Please enter a value\n");
    scanf_s("%d", &A);
    /* somevalue ဆိုတဲ့ function ကို referenceနဲ့လှမ်းခေါ်လိုက်တာပါ*/
    somevalue(&A);

    printf_s("A(Actual Parameter) = %d\n", A);

    getch();
    return 0;
}

```

```
void somevalue(int *F) { /*int f က formal parameter ဖြစ်ပါတယ်*/
```

```
    *F = *F * 2;
    printf_s("F(Formal Parameter) = %d\n", *F);
}
```

အထက်ပါ program မှာဆိုရင် ရတဲ့ result တွေ တူညီပါလိမ့်မယ်။ ဘာကြောင့်လဲဆိုရင် ကျွန်တော်တို့ pointer ကို သုံးထားလိုပါ။ pointer အကြောင်းကို နောက်အခန်းမှာ ရှင်းပြပေးသွားပါမယ်။

Extra Knowledge (formal parameter ရော့ actual parameter နှစ်ခုလုံးက တူညီတဲ့ memory location တစ်ခုတည်းကို point လုပ်ထားပါတယ်)

Storage Classes

Generally အားဖြင့် c programming မှာ 5 storage class မျိုးရှိပါတယ်။

1. auto
2. extern
3. static
4. register တို့ဖြစ်ပါတယ်။

storage class တွေဆိုတာ ဘာတွေလဲ သူ့တို့ကိုဘယ်မှာ ဘယ်လို သုံးကြလဲ ဆိုတာကို ဆက်ရှင်းပါမယ်။

Storage Classes တွေဆိုတာ variable or function တစ်ခုရဲ့ feature တွေဖြစ်ပါတယ်။ ဥပမာ Variable တစ်ခုကို ကြော်လှာမယ်။ အဲဒီ variable တစ်ခုရဲ့ရှေ့မှာ feature တစ်ခု ထည့်လိုက်မယ်ပေါ့။ example >> auto a =10; အခုဆိုရင် auto ဆိုတဲ့ keyword က a ဆိုတဲ့ variable ရဲ့ feature လိုဆိုလိုချင်တာပါ။ ထိုကဲ့သို့ variable or function တွေရှေ့မှာ 5storage class တွေ ထည့်လိုက်ခြင်းအားဖြင့် program ကို trace လုပ်ရတာ လွယ်ကူသွားပါတယ်။ အဲဒီ ထက်ပိုပြီး အသုံးချိန်တော့ကို ဆက်လေ့လာကြရအောင်။

1. auto Storage class

auto က function တစ်ခု or code block တစ်ခုအတွင်းမှာသာ ကြော်လှာပြီးသုံးတဲ့ storage class အမျိုးအစားဖြစ်ပါတယ်။ ထို့ကြောင့် အခြားသော function or code block ကနေပြီး လုမ်းခေါ်သုံးလို့မရပါဘူး။ သို့သော် pointer ခန်းရောက်ရင်တော့ variable ရဲ့ memory location ကို pointလုပ်ပြီး အပြင်ကနေ သူ့တန်ဖိုးကို access လုပ်နိုင်ပါတယ်။ program ရေးတဲ့သူက သူ့ value ကို ဘာမှာကြော်လှာ

မထားရင်တော့ garbage value တစ်ခု assign လုပ်သွားမှာဖြစ်ပါတယ်။ သို့သော်လည်း မိမိတို့သုံးတဲ့ ide ပေါ်မှုတည်ပြီး ပြောင်းလဲနိုင်ပါတယ်။

Example : auto a =10;

auto = auto (Storage class) keyword က a ဆိုတဲ့ variable ရဲ့ feature

10 = သူတေနဖိုးကို assign(သတ်မှတ်)ထားတာပါ။ မသတ်မှတ်ထားရင်တော့ garbage

Value တစ်ခု assign လုပ်သွားနိုင်ပါတယ်။

Example 7 using auto storage class

```

1 #include<stdio.h>
2 #include<conio.h>
3
4 int main()
5 {
6     auto num = 20;
7     {
8         auto num = 60;
9         printf_s("nNum : %d\n", num);
10    }
11    printf_s("Num : %d", num);
12    getch();
13    return 0;
14 }
```

line number 4 ကနေ 14 ထိက block
main function ရဲ့ block အဲတဲ့မှာ မှ
auto num ကို ကြော်ထားတယ်

line number 7 ကနေ 10 ထိက
code block တစ်ခုဖြစ်ပါတယ်။

နံမည်ပေးထားပုံခြင်း တို့သော်ပြားလည်း
scope မတွေတဲ့ အတွက် value တွေ
ဟာ ပြောင်းလဲသွားခြင်းမရှိပါဘူး

အထက်ပါ program ကို visual studio 2017 မှာ စမ်းရေးထားခြင်းဖြစ်ပါတယ်။ အခြားသော ide များတွင် error တက် ကောင်းတက်နိုင်ပါတယ်။ အပြင် block က auto ကို အတွင်း block ကနေပြီး
လုမ်းခေါ်လို့ရပါတယ်။ အောက်က program လေးကို စမ်းကြည့်ပါ။

```
int main()
{
    auto name = 20;
```

```

    {
        printf_s("Num : %d\n", name); အပြင်မှာကြော်ပြာထားတဲ့ auto ကို အတွင်း
                                         ကနေ ခေါ်တာ ရပါတယ်။
    }
    _getch();
    return 0;
}

သို့သော အတွင်းဆုံး block မှာရှိတဲ့ auto ကိုတော့ အပြင် block ကနေပြီး လုမ်းခေါ်လို့ မရနိုင်ပါဘူး။
အောက်က program လေးကို စမ်းကြည့်နိုင်ပါတယ်။ error ပြပါလိမ့်မယ်။

int main()
{
    auto name = 20;
    {
        auto num = 60;
        printf_s("Num : %d\n", name);
    }
    printf_s("num %d\n ", num); // အတွင်းက auto ကို အပြင်ကနေခေါ်တာ မရပါဘူး။
    _getch();
    return 0;
}

```

2.extern Storage class

```

#include<stdio.h>
#include<conio.h>

int x;
int main(){
    int y;

    printf("Glo-%d Loc-%d",x,y);
    getch();
    return 0;
}

```

Select C:\Users\WinHtut_GreenHackers\Documents\Untitled1

Glo-0 Loc-54

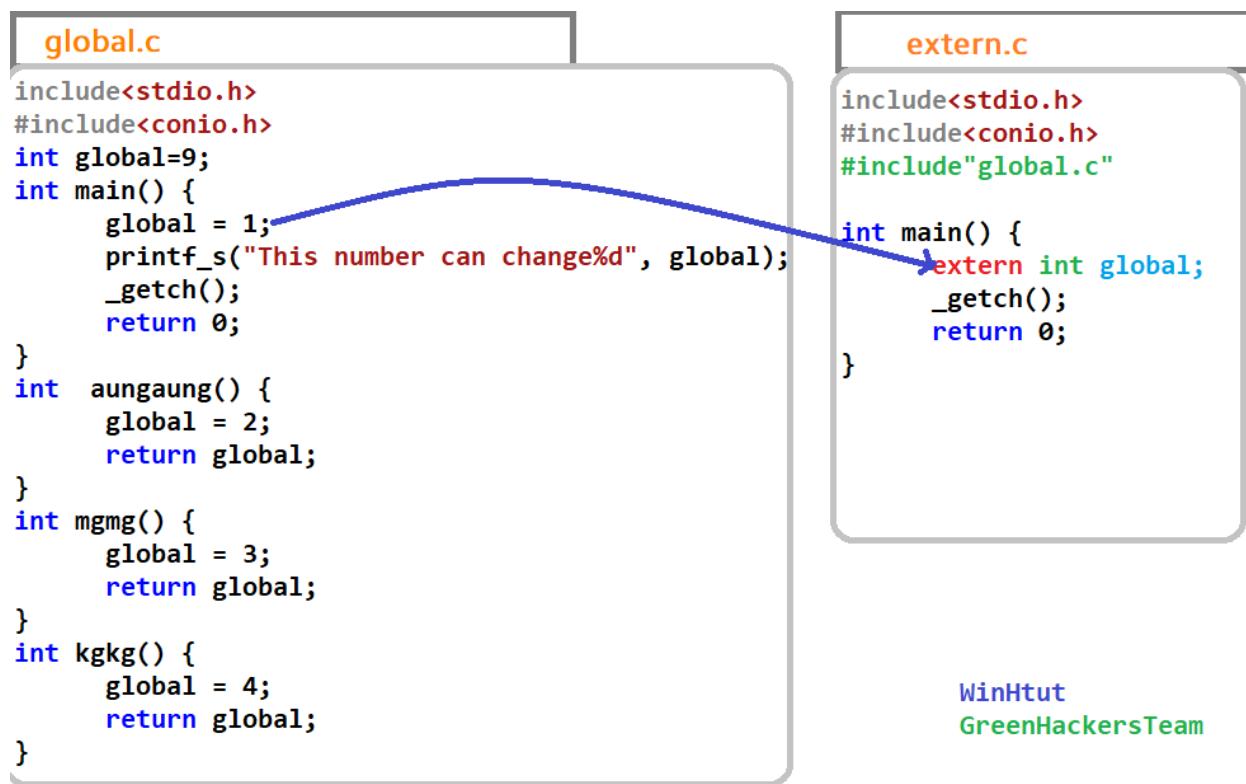
ပုံမှန်အားဖြင့် extern storage class ဟာလည်း global variable လိုပါပဲ။ သူကိုလည်း program ရဲ့ အပြင်ဘက်ဆုံးအပိုင်းမှာ ကြော်ကြာပေးရသလို မည်သည့် function or code block ကနေ မဆို ခေါ်သုံးနိုင်ပါတယ်။ သို့သော် global variable တွေဟာ အခြားသော functions တွေထဲမှာ ခေါ်သုံးရင် သူတို့ value တွေဟာ ပြောင်းလဲနိုင်ပါတယ်။ အောက်က program လေးကို လေ့လာကြည့်ပါ။

Example 8 using normal variable and extern storage class

```
#include<stdio.h>
#include<conio.h>
int global=9;
int main() {
    global = 1;
    printf_s("This number can change%d", global);
    _getch();
    return 0;
}
int aungaung() {
    global = 2;
    return global;
}
int mgmg() {
    global = 3;
    return global;
}
int kgkg() {
    global = 4;
    return global;
}
```

variable တစ်ခု ရဲ့ ရွှေ့မှာ extern ဆိုတဲ့ keyword ကို အသုံးပြုပြီး အခြားသော c program file တစ်ခု ကနေလှမ်းပြီး ခေါ်သုံးလို့ရပါတယ်။ အထက်ပါ program မှ int global=9; ဆိုတဲ့ variable ကို အခြားသော c program file တစ်ခုကနေ extern int global ; ဆိုပြီး လှမ်းခေါ်သုံးလို့ ရပါတယ်။ အောက်ကပုံလေးမှာစမ်းပြီး ရေးပြထားပါတယ်။ header file ကို ပြန်ခေါ်ဖို့လိုပါတယ်။ ပြန်မခေါ်ပေးဘူးဆိုရင် ခေါ်သုံးလို့ မရတော့ပါဘူး။ program နှစ်ခုက သီးသန့်ဖြစ်သွားမှာပါ။

Example 9 Using extern storage class with two program file



ref: www.quora.com/what-is-a-storage-class-in-c/ / www.geekforgeekc-storage-class

3. static Storage class

Different between **static variables** and **normal variables**

static int a; // static variable ရဲ့ syntax ဖြစ်ပါတယ်။

int a; //normal variable ရဲ့ syntax ဖြစ်ပါတယ်။

static variable ရဲ့ initialize value ကို compiler ကနေ zero အဖြစ် သတ်မှတ်ပေးထားပါတယ်။ normal variable ရဲ့ initialize value ကို garbage value တစ်ခုခဲ့ အဖြစ် compiler ကနေ သတ်မှတ်ပေးထားပါတယ်။ အထက်က သင်ခန်းစာတွေမှာလဲ ပြောပြထားပါတယ်။

static int a=0; // static initialize value

int a = (any garbage value)

	static variable	normal variable
syntax	<code>static int a;</code>	<code>int a;</code>
initialize value	0 program ပြီးတဲ့ အစိတ် ကိုထိန်းထားနိုင်တယ်	any garbage value ကြော်လွှာတဲ့ code block အတွင်းမှာပဲ value ကို ထိန်းထား နိုင်တယ်။

8 static.cpp

```

1 #include<stdio.h>
2 #include<conio.h>
3 void counter();
4
5 int main() {
6     for (int i = 0; i < 5; i++)
7     {
8         counter();
9     }
10    _getch();
11    return 0;
12 }
13
14 void counter() {
15     static int count=0;
16     count++;
17     printf("counting %d\n", count);
18 }
```

C:\Users\WinHtut_GreenHackers\Documents\8 static.exe

ပုဂ္ဂိုလ် dev c++ ide တွင် စစ်ဆေးသူး
ခြင်း ဖြစ်သည်

8 static.cpp

```

1 #include<stdio.h>
2 #include<conio.h>
3 void counter();
4
5 int main() {
6     for (int i = 0; i < 5; i++)
7     {
8         counter();
9     }
10    _getch();
11    return 0;
12 }
13
14 void counter() {
15     int count=0;
16     count++;
17     printf("counting %d\n", count);
18 }
```

C:\Users\WinHtut_GreenHackers\Documents\8 static.exe

static နဲ့ကြော်လွှာတဲ့ variable တစ်ခုဟာ ဘူး၊
value ကို program ပြီးတဲ့ အချို့ထိ ဆက်ထိန်းထားပါ
တယ်ပေထမ တစ်ကြိမ် looping ပတ်ပြီး ခေါ်တဲ့ အချို့
မှာ count ရဲ့ value 1 ကို ဆက်ထိန်းထားပါတယ်၍ဒါ
ယ အကြိမ် ပတ်တဲ့ အချို့မှာ count++ လို့ရေးထားတဲ့
အတွက် ပထမ value 1 ရယ် ဒုတိယ count++ ရယ်
ပေါင်းပြီး ရလာတဲ့ count value အသစ်ကိုလည်း ဆက်
ထိန်းထားပါတယ်။ သို့သော် static နဲ့ မကြော်လွှာပဲ ရိုးရိုး
int နဲ့ဟာ ကြော်လွှာတဲ့ ရင်တော့ count value ဟာ
looping ဘယ်လောက်ပတ်ပတ် 1 သာဖြစ်နေမှာပါ။ဘာ
ကြောင့်လဲဆိုတော့ ရိုးရိုးကြော်လွှာတဲ့ variable တစ်ခု
ဟာ ဘူး၊ code block အဆုံးထိပဲ သူ value ရှိနေမှာပါ။

static int count နေရာမှာ int count
ဆိုပြီး ရိုးရိုး variable တစ်ခုကိုရေးလိုက်ပါ
တယ်။ထို့ကြောင့် ဘူးဟာ ရိုးရိုး variable
ဖြစ်သွားတဲ့အတွက် ဘူးကို ကြော်လွှာတဲ့
block ဆုံးသွားတာနဲ့ အရင် value အ
တိုင်းပဲ ဆက်ရှိနေမှာပါ။

dev c++ ide စစ်ဆေးပြီးထားခြင်းဖြစ်ပါ
တယ်။

4. register storage class

```
register int i = 5;
```

```
printf_s("register %d",i);
```

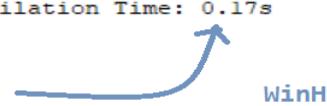
register storage class ဟာလည်း auto နဲ့ တူပြီး သူကိုလည်း local variable အဖြစ်ကြော် ဌာပါတယ်။ သို့သော် မတူတဲ့ အချက်တွေ ရှိပါတယ်။ တကယ်လို့သာ register ထဲမှာ နေရာလွတ်ရှိ နေခဲ့ရင် အခြားသော variable တွေလို့ RAM ထဲမှာ သွားမသိမ်းပဲ register ထဲမှာသာ သွားသိမ်းပါတယ်။ register ထဲမှာ သွားသိမ်းတဲ့အတွက် သူမှာ memory location မရှိပါဘူး။ memory location မရှိသောကြောင့် pointer နဲ့လည်း သွားပြီး ထောက်လို့မရတဲ့အတွက် သူ address ကို မရနိုင်ပါဘူး။ တကယ်လို့ register ထဲမှာ နေရာမရှိဘဲ memory ထဲမှာ သွားသိမ်းမှသာ အခြားသော variables တွေလို့ memory address ရှိပြီး pointer နဲ့ သွားထောက်လို့ရမှာ ဖြစ်ပါတယ်။ ထူးခြားချက် သည် register storage class ကိုသုံးတဲ့အချင်း variable ကို register ထဲမှာပဲ သိမ်းခဲ့မယ်ဆိုရင် program run တဲ့အချင်းဟာ သာမန် memory ထဲမှာပဲ သိမ်းတာထက် ပိုပြီးမြန်ပါလိမ့်မယ်။ ဘာကြောင့်လဲဆိုတော့ register ဟာ CPU(Central Processing Unit) အတွင်းမှာ ရှိနေတာပါ။ သာမန် variable တွေ သိမ်းတဲ့ RAM (Random Access Memory) လို့ CPU ရဲ့အပြင်ဘက်မှာ ရှိနေတာ မဟုတ်ပါဘူး။ ထို့ကြောင့် program ရဲ့ runtime မှာ register ထဲမှာ သိမ်းခြင်းက ပိုပြီးမြန်နေခြင်း ဖြစ်ပါတယ်။ register storage class ကို program တစ်ခုထဲမှာ variable တစ်ခုအား မြန်မြန်ဆန်ဆန် processing လုပ်ချင်တဲ့အခါမျိုးမှာ သုံးကြပါတယ်။ မှတ်သားရန် တစ်ချက်က Hardware တည်ဆောက်ပုံအပေါ်မူတည်ပြီးလည်း variable ကို သိမ်းတဲ့နေရာ ပြောင်းလဲနိုင်သေးတဲ့ အတွက်ကြောင့် runtime ဟာလည်း ပြောင်းလဲနိုင်ပါသေးတယ်။ အောက်မှာ program ရေးပြီးစမ်းပြထားပါတယ်။ dev c++ ide ကို သုံးထားပါတယ်။ အခြားသော ide တွေ သုံးခဲ့မယ်ဆိုရင်တော့ runtime ကြောချိန်ကဲ့ပြားမှာ ရှိနိုင်ပါတယ်။ အောက်မှာ code လည်း စမ်းရေးပြထားသလို microcomputer တစ်ခုရဲ့ block diagram ကို လည်းဆွဲပြထားပါတယ်။

```

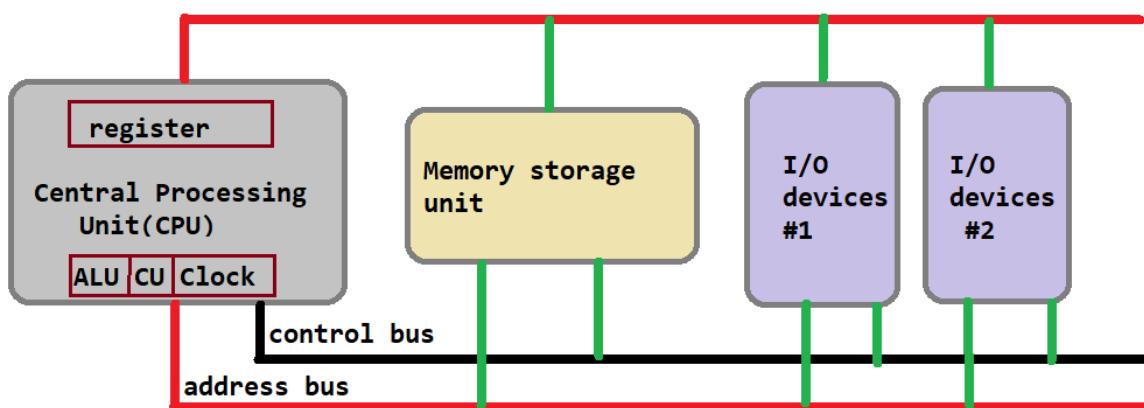
1 #include<stdio.h>
2 #include<conio.h>
3
4 int main(){
5     register int x=10;
6     int y=0;
7     for ( y=0;y<x;y++){
8
9         printf("using register %d\n",x);
10    }
11    getch();
12    return 0;
13 }
```

- Command: g++.exe "C:\Users\WinHtut_Gre...
Compilation results...

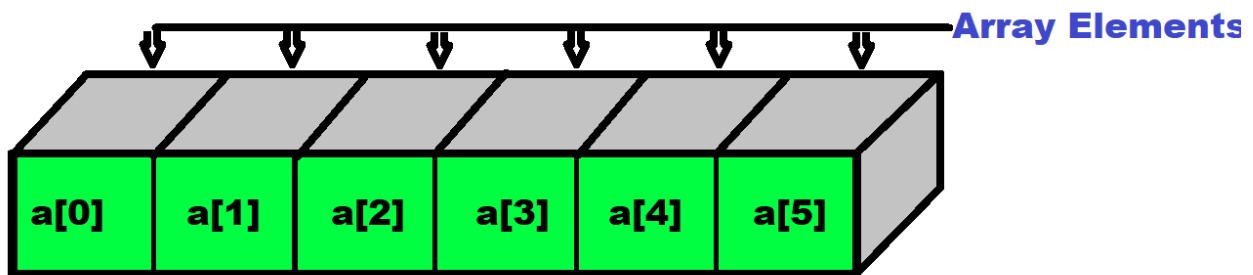
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\WinHtut_Gre...
- Output Size: 101.796875 KiB
- Compilation Time: 0.17s



Data bus, I/O bus



Block Diagram of a Microcomputer

WinHtut
GreenHackerTeam

1 Dimensional Array with 6 Elements

WinHtut
GreenHackersTeam

Array

One Dimensional Array

One dimensional array မှာဆိုရင် subscript variable အနေနဲ့ တစ်ခုပဲ ပါဝင်ပါတယ်။ အထက်ပါပုံမှာဆိုရင် 0,1,2,3,4,5 ဆိုတာတွေက subscript variableရဲ့ နေရာတွေ ဖြစ်ပါတယ်။ ထိုနေရာတွေမှာ i လိုသာ ရေးခဲ့မယ်ဆိုရင် i က subscript variable ဖြစ်မှာပါ။

int x[2];

int y= 1;

x[y] = 5;

 **y is subscript variable**

Array တစ်ခုမှာ subscript variable တွေ တစ်ခုထက်ပိုပြီး ပါဝင်နေခြင်းကို multidimensional array လို့ ခေါ်ပါတယ်။ Multidimensional array ကို နောက်တစ်မျိုးအနေနဲ့ matrix လိုလည်း ခေါ်ပါတယ်။

Two Dimensional Array

Two Dimensional Array

Two

	Column 0	Column 1	Column 2	Column 3	Column 4	Column 5
Row 0	a[0][0]	a[0][1]	a[0][2]	a[0][3]	a[0][4]	a[0][5]
Row 1	a[1][0]	a[1][1]	a[1][2]	a[1][3]	a[1][4]	a[1][5]
Row 2	a[2][0]	a[2][1]	a[2][2]	a[2][3]	a[2][4]	a[2][5]

dimensional array မှာဆိုရင် subscript variable နှစ်ခုပါဝင်မှာဖြစ်ပါတယ်။ Tow dimensional array တွေဟာ values တွေကို matrix ပုံစံနဲ့ သိုလောင်ပါတယ်။ ပထမ subscript variable ဟာ row ဖြစ်ပြီးတော့ ဒုတိယ subscript variable ဟာ column ဖြစ်ပါတယ်။ အထက်ပါ ပုံမှာ ဖော်ပြထားပါတယ်။

Declaration and Use of Two Dimensional Array

Two dimensional array တစ်ခုကို စတင်ကြော်ဖြေကြည့်မယ်။

`int a[3][3] = { {1,2,3}, {4,5,6}, {7,8,9} };`
 int ဆိုတာကတော့ a ဆိုတဲ့ array ရဲ့ data type ပါ။ a ဆိုတာကတော့ array ရဲ့ နာမည်ပေါ့။ သို့သော် မိမိ ကြိုက်တာပေးလို့ ရပါတယ်။ [3] ပထမ 3 က row ရဲ့ အရေအတွက်ပါ။ ဒုတိယ [3] ကတော့ column ရဲ့ အရေအတွက်ပါ။ row သုံးခု column သုံးခုရှိတဲ့ ယေားကွက်လေးတစ်ကွက် ကို မြင်ကြည့် လိုက်ပါ။ သူတို့ကို print ထုတ်ကြည့်ချင်တယ်ဆိုရင် အောက်ပါအတိုင်း print ထုတ်ကြည့်လို့ရပါတယ်။

```
int row = 3;
int col = 3;
for (i = 0; i < row; i++) {
    for (j = 0; j < col; j++) {
        printf_s("%3d", a[i][j]);
    }
}
printf_s("\n");
```

အောက်မှာ ပေးထားတဲ့ အပြည့်အစုံ codes တွေကို လေ့လာကြည့်ပါ။ အပေါက် ပုံနှုန်း
 တွဲကြည့်မယ်ဆိုရင် ကောင်စွာ နားလည်သွားမှာပါ။

	col 0	col 1	col 2
row 0	1	2	3
row 1	4	5	6
row 2	7	8	9

```
int a[0][0]=1;
int a[0][1]=2;
int a[0][2]=3;
int a[1][0]=4;
int a[1][1]=5;
int a[1][2]=6;
int a[2][0]=7;
int a[2][1]=8;
int a[2][2]=9;
```

Two dimensional array
 တစ်ခုကို ယေားဖြင့် ဖော်ပြခြင်း

GreenHackers
 WinHtut

Example 1 Printing Two Dimensional Array

```
#include<stdio.h>
#include<conio.h>
int i, j;
int row = 3;
int col = 3;
```

```

int main() {
    int a[3][3] = { {1,2,3}, {4,5,6}, {7,8,9}};

    for (i = 0; i < row; i++) {
        for (j = 0; j < col; j++)
        {
            printf_s("%3d", a[i][j]);
        }
        printf_s("\n");
    }
    _getch();
    return 0;
}

```

OUTPUT

```

1 2 3
4 5 6
7 8 9

```

Example 2

နောက်ထပ် program တစ်ပုဒ်အနေနဲ့ row 3 ခု column 2 ခုနဲ့ရေးကြည့်ရအောင်။

```

#include<stdio.h>
#include<conio.h>
int main() {
    int i, j;
    int a[3][2] = { { 1, 4 },
                    { 5, 2 },
                    { 6, 5 } };
    // row သိုးခု column နှစ်ခု ရှိတဲ့ array တစ်ခုကို တည်ဆောက်တယ်
    for (i = 0; i < 3; i++) { // row အတွက် loop ပတ်တာပါ
        for (j = 0; j < 2; j++) { // column အတွက် loop ပတ်တာပါ
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
    _getch();
    return 0;
}

```

```
}
```

OUTPUT

```
1 4
5 2
6 5
```

Example 3

နောက်ထပ် program တစ်ပုဒ်အနေနဲ့ row တွေ column တွေကို မိမစိတ်ကြိုက် ရေးကြည့်ရအောင်...။

```
#include<stdio.h>
#include<conio.h>
int main() {
    int i, j;
    int a[3][2] = { { 1 }, // row တွေ column တွေကို မိမစိတ်လို့ data တွေ ထည့်ထားတာပါ
                    { 5, 2 },
                    { 6 } };
    for (i = 0; i < 3; i++) {
        for (j = 0; j < 2; j++) {
            printf("%d ", a[i][j]);
        }
        printf("\n");
    }
    _getch();
    return 0;
}
```

OUTPUT

```
1 0
5 2
6 0
```

Accessing Array

ယခု သင်ခန်းစာမျာတော့ array တစ်ခုထဲမှာရှိတဲ့ data တွေကို access(ရယူ) ကြမယ်။ array ထဲက data တွေကို access လုပ်တဲ့အခါမှာ နည်းလမ်းအနည်းငယ်ရှိပါတယ်။ ဥပမာအနေနဲ့ int arr[] ; ဆိုတဲ့ array တစ်ခုမှာဆိုရင် printf("%d",arr) လို့ရေးခဲ့မယ်ဆိုရင် ထို array ရဲ့ base address ကို access လုပ်နိုင်ပါတယ်။ int arr[] = { 10,11,12,13,14,15}; ဆိုတဲ့ array တစ်ခုမှာဆိုရင် printf("%d", *arr) လို့ ရေးခဲ့မယ်ဆိုရင် ထို array ရဲ့ zeroth element နေရာကို access လုပ်နိုင်ပါတယ်။ output အနေနဲ့ 10 ကိုရရှိမှာ ဖြစ်ပါတယ်။ printf("%d",*(arr+0)) လို့ ရေးခဲ့မယ်ဆိုရင်လည်း array ရဲ့ zeroth element

ရှိတဲ့နေရာကို access လုပ်နိုင်မှာ ဖြစ်ပါတယ်။ `printf ("%d", *(arr+1))` လို့ရေးခဲ့မယ်ဆိုရင် array ရဲ့ first element ရှိတဲ့နေရာကို access ရယူနိုင်မှာ ဖြစ်ပါတယ်။ အောက်ပါ code မှာ output အနေနဲ့ arr ဆိုတဲ့ array မှာဆိုရင် 11 ကိုရရှိမှာ ဖြစ်ပါတယ်။ နောက်ထပ်ရှိုးရှင်းတဲ့ နည်းတစ်ခုအနေနဲ့ subscript variables တွေကို သုံးပြီးလည်း access လုပ်လို့ရပါတယ်။ ဥပမာ `int i=0; printf("%d", arr[i])` လို့ရေးပြီးတော့ လည်း Output ကိုရရှိပါတယ်။ Lesson 10 မှာ အသေစိတ်ပြောခဲ့ပြီးပါပြီ။ နောက်တစ်နည်းအနေနဲ့ `printf("%d", i[arr])` လို့ ရေးခဲ့မယ်ဆိုရင်လည်း array ထဲမှာ ရှိတဲ့ data တွေကို access လုပ်လို့ရပါတယ်။

```
int arr[] = { 10,11,12,13,14,15};
printf("%d",arr) = base address
printf("%d", *arr ) = 10
printf("%d",*(arr+0)) = 10
printf("%d",*(arr+1)) = 11
int i=0; printf("%d",arr[i]) = 10
printf("%d",i[arr]) = 10
```

အောက်မှာ ဖော်ပြထားတဲ့ပုံနဲ့ program ကိုကြည့်မယ်ဆိုရင် ပိုပြီးရှင်းလင်းသွားမှာပါ။

Expression	Description	Value
<code>("%d",arr)</code>	arr ဆိုတဲ့ array ရဲ့ base address ကို ပြန်ရမှာဖြစ်ပါတယ်။	13631048
<code>("%d", *arr)</code>	array ရဲ့ zeroth element နေရာကို access ရမှာဖြစ်ပါတယ်။	10
<code>("%d",*(arr+0))</code>	array ရဲ့ zeroth element နေရာကိုပဲ access ရမှာဖြစ်ပါတယ်။	10
<code>("%d",*(arr+1))</code>	array ရဲ့ first element နေရာကို access ရမှာဖြစ်ပါတယ်။	11
<code>("%d",arr[i])</code>	i ရဲ့ value အတိုင်း array ရဲ့ element နေရာကို access ရမှာဖြစ်ပါတယ်	int i=0; 10
<code>("%d",i[arr])</code>	i ရဲ့ value အတိုင်းပဲ array ရဲ့ element နေရာကို access ရမှာဖြစ်ပါတယ်	int i=1; 11

WinHtut(GreenHackersTeam)

Example 4 Accessing Array

```
#include<stdio.h>
#include<conio.h>
```

```

int main() {
    int arr[] = {10,11,12,13,14,15 };
    int i;
    for (i = 0; i <= 5; i++) {
        printf("\n%d %d %d %d %d", arr,arr[i], *(i + arr),
               *(arr + i), i[arr]);
    }
    getch();
    return 0;
}

```

OUTPUT

```

12778192 10 10 10 10
12778192 11 11 11 11
12778192 12 12 12 12
12778192 13 13 13 13
12778192 14 14 14 14
12778192 15 15 15 15

```

loop တစ်ကြမ်ပတ်လိုက်တိုင်းမှာ ပုံစံလေးမျိုးနဲ့ ထုတ်ပြထားပါတယ်။
 subscript variable တွေကို ကစားပြီး မိမိလိုအပ်သလို ထုတ်ယူလိုရပါတယ်။ array ကို accessing လုပ်ပုံလေးကို ကောင်းမွန်စွာ နားလည်သွားမှာပါ။

Limitations Of Array**Data Structure and Memory allocation**

ကျွန်တော်တို့ အနေနဲ့ array ဘာလည်း ဆိုတာကို အနည်းအကျဉ်းတော့ သိသွားပါပြီ။ array ဟာ အရမ်းအသုံးတည့်သလို data type တစ်ခုတည်းအောက်မှာ တူညီတဲ့ data type တွေ အများကြီး သိလောင်လို့ ရကြောင်းကို လေ့လာခဲ့ပြီးပါပြီ။ နောက်ထပ် သိဖို့လိုတာက Array ရဲ့ limitations တွေ အကြောင်းပါ။

Data Structure တွင် Static Data Structure (SDS) and Dynamic Data Structure (DDS) ဆိုပြီး နှစ်မျိုးခဲ့ထားလို့ ရပါတယ်။ Arrays တွေက Static Data Structure တဲ့မှာပါဝင်ပါတယ်။ Program ကို compile လုပ်တဲ့အချိန်မှာ memory allocated လုပ်ပါတယ်။ memory allocation လုပ်တယ်ဆိုတာက လုပ်ငန်းစဉ်တစ်ခုပါ။ ဘယ်လိုလုပ်ငန်းစဉ်လဲဆိုရင် ကျွန်တော်တို့ရေးလိုက်တဲ့ program တစ်ခု သို့မဟုတ် လုပ်ငန်းစဉ်တစ်ခုဟာ physical memory or virtual memory ပေါ်မှာ သွားရောက်ပြီး နေရာသတ်မှတ်ယူ ခြင်း ဖြစ်ပါတယ်။ ဆိုလိုချင်တာက computer ရဲ့ memory ပေါ်မှာ programs and processes တွေကို execution လုပ်ဖို့အတွက် နေရာယူ သိမ်ဆည်းထားခြင်းကို ဆိုလိုချင်တာပါ။

Memory Allocation မှာလည်း နှစ်မျိုးရှိပါသေးတယ်။ Static Memory Allocation နဲ့ Dynamic Memory Allocation တို့ဖြစ်ပါတယ်။ Static Memory Allocation က compile လုပ်တဲ့အချင်မှာ memory ပေါ်မှာ allocate လုပ်ပြီး Dynamic Memory Allocation ကတော့ run time မှာ allocate လုပ်ပါတယ်။

For Extra Knowledge

Static Data Structure and Dynamic Data Structure

Static Data Structure တွေဟာ program run တဲ့အချင်မှာ သူတို့ရဲ့ size တွေကို ပြောင်းလဲလို့ မရပါဘူး။ Arrays တော်တော်များများဟာ Static Data Structure တွေ ဖြစ်ကြပါတယ်။ သူတို့ကို တစ်ခါ ကြော်ပြီးတာနဲ့ ပြန်ပြောင်းလို့ မရတော့ပါဘူး။ Dynamic Data Structure တွေဟာ သူတို့ရဲ့ size တွေကို program run နေတဲ့အချင်မှာလည်း တိုးလို့ရသလို လျော့လို့လည်း ရပါတယ်။ Advantages of Static Data Structures * Compilation လုပ်နေတဲ့အချင်မှာ compiler က memory allocate လုပ်နိုင်ပါတယ်။ program ရေးသားဖို့ လွယ်ကူပါတယ်။ overflow ဖြစ်စေမယ့် အရာတွေကို အလွယ်တကူ စစ်ဆေးနိုင်ပါတယ်။ ပြီးတော့ array တစ်ခုကိုဆိုရင် အလွယ်တကူ မိမိကြိုက်သလို access လုပ်နိုင်ပါတယ်။ Disadvantages of Static Data Structures * programmer က program ရေးတဲ့အခါ maximum space တစ်ခုကို ကြော်ပြီးရေးခဲ့ရင် memory space တွေကို waste ဖြစ်စေနိုင်ပါတယ်။ လိုအပ်သလောက်ပဲ တိကျွွာ သတ်မှတ်ပြီးရေးရင်တော့ ပိုပြီး အဆင်ပြေစေမှာပါ။ Advantages of Dynamic Data Structures * ဘယ်အချင်မှာမဆို space ကို လိုအပ်တာပဲ သုံးတဲ့အတွက် memory ကို ပိုမိုကောင်းမွန်စွာ အသုံးချလို့ရပါတယ်။ Disadvantages of Dynamic Data Structures * Linked list (နောက်ပိုင်းအခန်းတွေမှာ ရေးသားမှာဖြစ်ပါတယ်)။

Different Data Type

တူညီတဲ့ Data Type ရှိတဲ့ data တွေကိုသာ သိလျောင်လို့ရပါတယ်။ ဘာကြောင့်လဲဆိုတော့ Array Data Structure တစ်ခုဟာ တူညီတဲ့ data တွေကိုသာ ကိုင်တွယ်သိလျောင်နိုင်လိုပါ။

For Example : Character and Integer Values တွေကို separate array ထဲမှာသာ သိလျောင်လို့ရပါတယ်။ တူညီတဲ့ single array တစ်ခုတည်းမှာ သိလျောင်လို့ မရပါဘူး။

Inserting Data In The Array

Array တစ်ခုထဲကို data ထည့်ဖို့ဆိုတာ လွယ်ကူပါတယ်။ သို့သော် array ထဲကို data တစ်လုံးထည့်ချင်တယ်ဆိုရင် ထို array ရဲ့ မူလကြော်ဌာထားတဲ့ size ကို တစ်လုံးတိုးပေးဖို့ လိုအပ်ပါတယ်။ ထိုသို့မကြော်ဌာထားပဲ array size ကို fixed လုပ်ထားမယ်ဆိုရင် မူလ array ထဲမှာရှိတဲ့ data element ကို ဖယ်ထုတ်လိုက်မှာ မို့ပါ။ ဆိုလိုချင်တာ 1,2,3,4,5 ကို array ထဲက data element တွေလို့ ယူဆကြပါစို့။ array size ကိုလည်း 5 လို့ပဲ ကြော်ဌာထားတယ်။ ကျွန်တော်တို့က element သုံးခုမြောက် နေရာမှာ 0 ဆိုတဲ့ data တစ်ခုကို ထပ်ထည့်လိုက်ရင် 1,2,3,0,4 လို့ပဲ ပေါ်လာမှာပါ။ 5 ကို data ထဲကနေ ဖယ်ထုတ်လိုက်မှာ ဖြစ်ပါတယ်။ ဘာကြောင့်လဲဆိုတော့ ကျွန်တော်တို့က array size ကို 5 လို့ ကြော်ဌာထားခဲ့လိုပါ။ အကယ်လို့ array size ကို 6 လို့ဘာ ကြော်ဌာခဲ့မယ်ဆိုရင် 5 ကို ဖယ်ထုတ်လိုက်တော့မှာ မဟုတ်ပါဘူး။ 1,2,3,0,4,5 ဆိုပြီးရလာမှာပါ။ အောက်က program လေးကို ရေးကြည့်လိုက်ရင် ပိုပြီးတော့ နားလည်လာမှာပါ။

(အထူးသတိပြုရမည့်အချက် ။ Code::Blocks 17.12 တွင်ရေးသားထားခြင်းဖြစ်ပါသည်။)

Example 5 Inserting Data In The Array

```
/*Inserting Data in the array*/
#include<stdio.h>
#include<conio.h>
#define SIZE 5
int arr[SIZE], element, number, i, position;
int main() {
    printf_s("enter number of elements:\n");
    scanf_s("%d", &number);
    printf_s("Enter %d elements of Array\n", number);
    for (i = 0; i < number; i++) {
        scanf_s("%d", &arr[i]);
    }
    printf_s("Array data elements are :\n");
    for (i = 0; i < number; i++) {
        printf_s(" %d ", arr[i]);
    }
    printf_s("\nEnter the element to be insert: ");
    scanf_s("%d", &element);
    printf_s("\nEnter the postion of element:");

}
```

```

scanf_s("%d", &position);
i = number - 1;
while (position <= i) {
    arr[i + 1] = arr[i];
    i--;
}
arr[position] = element;
printf_s("After insertion array is:\n");
for (i = 0; i < number; i++) {
    printf_s(" %d ", arr[i]);
}
_getch();
return 0;
}

```

Program ရှင်းလင်းချက်

```

1      /*Inserting Data in the array*/
2      #include<stdio.h>
3      #include<conio.h>
4      #define SIZE 5
5      int arr[SIZE], element, number, i, position;
6      int main() {
7
8          printf_s("enter number of elements:\n");
9          scanf_s("%d", &number);
10         printf_s("Enter %d elements of Array\n", number);
11         for (i = 0; i < number; i++) {
12             scanf_s("%d", &arr[i]);
13         }
14         printf_s("Array data elements are :\n");
15         for (i = 0; i < number; i++) {
16             printf_s(" %d ", arr[i]);
17         }

```

အထက်ပါ program မှာဆို line number 17 အထိက အရှင်သင်ခန်းစာတွေမှာ ရှင်းထားပြီးသား
ဖြစ်တဲ့အတွက် line number 18 ကနေ စပြီးရှင်းပါမယ်။

```

printf_s("\nEnter the element to be insert: ");
scanf_s("%d", &element);
printf_s("\n Enter the postion of element:");
scanf_s("%d", &position);

```

ကျွန်တော်တို့ array ထဲမှာ ထည့်ချင်တဲ့ element ရယ် ထို element ကိုထည့်မယ့် position ရယ်ကို user ဆီက တောင်းလိုက်တာပါ။

```

printf_s("\n Enter the position of element:");
scanf_s("%d", &position);
i = number - 1;
while (position <= i) {
    arr[i + 1] = arr[i];
    i--;
}

```

enter number of elements:
5
Enter 5 elements of Array
5
4
3
2
1
Array data elements are :
5 4 3 2 1
Enter the element to be insert: 0

1 ကျွန်တော်တို့ထည့်ချင်တဲ့ element က 0 ပါ

2 element ထည့်ချင်တဲ့နေရာပါ(ဆိုလိုချင်တာက 3 ခုမြောက် နေရာပါ 5,4,3,0,2 ဖြစ်ချင်တာပါ)

3 5,4,3,0,2 ပြီးရင် 1 က ပျောက်သွားမှပါ ဘာကြောင့်လဲဆိုတော့ array size ကို 5 ပဲကြော်ဖြာထားလိုပါ။

```

i = number - 1;
while (position <= i) {
    arr[i + 1] = arr[i];
    i--;
}

```

ဘယ်နေရာမှာ element ကို သွားထည့်ရမလဲဆိုတာ သိမှို့အတွက် while loop ထဲမှာ ထည့်ထားပါတယ်။

```

while (position <= i) {
    arr[i + 1] = arr[i];
    i--;
}

```

≤1ms elapsed

i | 3 position | 3
arr | 0x00a8a138 {5, 4, 3, 2, 1}

ပထမ တစ်ကြိမ် while loop ပြီးသွားတဲ့ အခါမှာ i တန်ဘိုးက 3 ဖြစ်သွား ပါတယ် ဘာကြောင့်လဲဆိုတော့ i တန်ဘိုးက အစက 4 ပါ i-- ဆိုတာ ပါတဲ့ အတွက် loop ပြီးသွားတဲ့ အခါမှာ 3 ဆိုပြီးဖြစ်သွားတာပါ။

Array ဘယ်လိုပြောင်းလဲသွားလဲဆိုတာကို ထပ်ပြီး ရှင်းကြရအောင်။

```
i = number - 1;  
while (position <= i) {  
    arr[i + 1] = arr[i];  
    i--;  
}
```

The diagram shows two variable declarations. The first, 'i', is enclosed in a red-bordered box and has a blue icon with a 'B' next to it. The second, 'position', is enclosed in a blue-bordered box and has a blue icon with a 'B' next to it. Both variables are followed by their respective sizes in brackets: '4' for 'i' and '3' for 'position'.

```
i = number - 1;
while (position <= i) {
    arr[i+1] = arr[i];
    i--;
}
```

ဒုတိယ တစ်ကြိမ် loop ပတ်ပြီးတဲ့ အချင်မှာ array ရဲ့ အခြေနေဖြစ်ပါတယ်။

`arr[position] = element;` ကျွန်တော်တို့ ထည့်လိုက်တဲ့ element တန်ဖိုးကို ကျွန်တော်တို့ ထည့်လိုက်တဲ့ position နေရာမှာ သွားပြီးတော့ ထည့်ဖို့ပါ။

```
printf_s("After insertion array is:\n");
    for (i = 0; i < number; i++) {
        printf_s(" %d ", arr[i]);
    }
```

array ထဲမှာ ရှိတဲ့ value တွက်၌ loop ပတ်ပြီးထုတ်ဖို့အတွက်ပါ။ output အနေနဲ့ အောက်ပါ အတိုင်း
ထွက်လာပါလိမ့်မယ်။ Code::Blocks မှာ ရေးထားတယ် ဆိုတာကို သတိပြုပါ။ ပြန်ပြီး arrange
လုပ်ထားတဲ့ output ပါ ...။

```
enter number of elements:  
5      printf("%d", arr[i]);  
Enter 5 elements of Array  
5      printf("\nEnter the element to be insert: ");  
4      scanf("%d", &element);  
3      printf("\nEnter the position of element");  
2      scanf("%d", &position);  
1      i=number-1;  
1      while(position <= i){  
Array data elements are :  
5 4 3 2 1  
Eneter the element to be insert: 0  
arr[position]=element;  
Enter the position of element3  
After insertion array is:  
5 4 0 3 0 2 { i++) {
```

အထက်ပါ program ကိုပဲ array size ကို ခီးမိုးပြီး number ကို 1 ပေါင်းထည့်ကြည့်ရအောင်..!!

```
#define SIZE 6
int arr[SIZE], element, number, i, position;
arr[position] = element;
number++;
printf_s("After insertion array is:\n");
```

Codes အပြည့်အစုံကို အောက်မှာ လွှဲလာနိုင်ပါတယ်။

Example 6 Inserting Data In The Array and change array size

```
/*Inserting Data in the array*/
#include<stdio.h>
#include<conio.h>
#define SIZE 6
int arr[SIZE], element, number, i, position;
int main() {
    printf_s("enter number of elements:\n");
    scanf_s("%d", &number);
    printf_s("Enter %d elements of Array\n", number);
```

```

for (i = 0; i < number; i++) {
    scanf_s("%d", &arr[i]);
}
printf_s("Array data elements are :\n");
for (i = 0; i < number; i++) {
    printf_s(" %d ", arr[i]);
}
printf_s("\nEnter the element to be insert: ");
scanf_s("%d", &element);
printf_s("\n Enter the postion of element:");
scanf_s("%d", &position);
i = number - 1;
while (position <= i) {
    arr[i+1] = arr[i];
    i--;
}
arr[position] = element;
number++;
printf_s("After insertion array is:\n");
for (i = 0; i < number; i++) {
    printf_s(" %d ", arr[i]);
}
_getch();
return 0;
}

```

Output အနေဖြင့် အောက်ပါအတိုင်း ရှိမှုမျာပါ။

```

enter number of elements:
5
Enter 5 elements of Array
5
4
3
2
1
Array data elements are :
5 4 3 2 1
Eneter the element to be insert: 0

Enter the postion of element:3
After insertion array is:
5 4 3 0 2 1

```

Deleting Data In The Array

Array တစ်ခုထဲက data ကို ဖျက်ထုတ်ဖို့ဆိုတာ array ထဲသို့ data ထည့်တာထက် ပိုပြီး လွယ်ကူပါတယ်။ အောက်က program ကို ဖော်ပြန်လောက်ည့်ပါ။

Example 7 Deleting Data in the array

```
#include<stdio.h>
#include<conio.h>
#define SIZE 6
int arr[SIZE],number, i, position;
int main() {
    printf_s("enter number of elements:\n");
    scanf_s("%d", &number);
    printf_s("Enter %d elements of Array\n", number);
    for (i = 0; i < number; i++) {
        scanf_s("%d", &arr[i]);
    }
    printf_s("Array data elements are :\n");
    for (i = 0; i < number; i++) {
        printf_s(" %d ", arr[i]);
    }

    printf_s("\n Enter the postion of element you want to delete:");
    scanf_s("%d", &position);
    i = position + 1;
    while (i <= number - 1) {
        arr[i - 1] = arr[i];
        printf_s(" array place is %d", arr[i - 1]);
        i++;
    }
    //number--;
    printf_s("After deleting array is:\n");
    for (i = 0; i < number; i++) {
        printf_s(" %d ", arr[i]);
    }
    getch();
    return 0;
}
```

Program ရှင်းလင်းချက်

```
while (i <= number - 1) {
    arr[i - 1] = arr[i]; //မူလရှိတဲ့ i value ထဲကနေ 1ကိုနှစ်ထားပါတယ်။
    printf_s(" array place is %d", arr[i - 1]);
    //အထက်ပါ printf ထုတ်ထားတာကတော့ program ကို trace လုပ်ချင်လိုပါ။
    i++;
}
//number--; ဒီနေရာလေးက output ရဲ့ အပြောင်းအလဲအတွက် အရေးကြီးပါတယ်။
ဘာလို့လဆိုတော့ ကျွန်တော်တို့ array ထဲမှာ ထည့်ထားတဲ့ အလုံးအရေအတွက် ဘယ်လောက် ကို
ထုတ်မလဲဆိုတာကို control လုပ်ထားတာပါ။ number--; လို့ program ထဲမှာ ထည့်ရေးလိုက်မယ်ဆိုရင်
output မှာ element လေးခုသာ ထွက်လာမှာဖြစ်ပါတယ်။
```

```
enter number of elements:
5
Enter 5 elements of Array
101
111
121
131
141
Array data elements are :
101 111 121 131 141
Enter the position of element you want to delete:3
array place is 141After deleting array is:
101 111 121 141
```

program မှာ number--; ထည့်ပြီး output
ထုတ်ထားတဲ့ပါ။

WinHut
GreenHackers

```
enter number of elements:
5
Enter 5 elements of Array
101
111
121
131
141
Array data elements are :
101 111 121 131 141
Enter the position of element you want to delete:3
array place is 141After deleting array is:
101 111 121 141
```

user ထည့်လိုက်တဲ့ အလုံးအရေအတွက်

user ထည့်လိုက်တဲ့ element များ

user ထည့်လိုက်တဲ့ elements တွေကို ပြန်ပြပေးတာပါ

user က ဖျက်ထုတဲ့ချင်တဲ့နေရာ

ဖျက်ထုတဲ့ပြီးလို့ပြန်ရလာတဲ့ elements တွေပါ။

number--; လို့သာ program ထဲမှာ ထည့်ရေးခဲ့မယ်ဆိုရင်

101 111 121 141 elements လေးခုတဲ့ output ထုတ်ပေး
မှာ အကယ်ကြောင့်ဆိုသော 1oop 4 ကိုမဲ့ အလုပ်လုပ်သွားမှာပါ

Example 8 Deleting Data In The Array In Another Way

যদি program মুাটেও Array তেক element কি আলোচন্য তাত্ত্বিকভাবে প্রিঃ
পুর্ণতাত্ত্বিক পিএড়॥

```
#include <stdio.h>
#include<conio.h>
int main()
{
    int vectorx[10];
    int i, n, pos, element, found = 0;
    printf_s("Enter how many elements\n");
    scanf_s("%d", &n);
    printf_s("Enter the elements\n");
    for (i = 0; i < n; i++)
    {
        scanf_s("%d", &vectorx[i]);
    }
    printf("Input array elements are\n");
    for (i = 0; i < n; i++)
    {
        printf_s("%d\n", vectorx[i]);
    }
    printf_s("Enter the element to be deleted\n");
    scanf_s("%d", &element);

    for (i = 0; i < n; i++)
    {
        if (vectorx[i] == element)
        {
            found = 1;
            pos = i;
            break;
        }
    }
    if (found == 1)
    {
        for (i = pos; i < n - 1; i++)
    }
```

```

    {
        vectorx[i] = vectorx[i + 1];
    }
    printf_s("The resultant vector is \n");
    for (i = 0; i < n - 1; i++)
    {
        printf_s("%d\n", vectorx[i]);
    }
}
else {
    printf_s("Element %d is not found in the vector\n", element);
}
 getch();
return 0;
}

```

Program ရှင်းလင်းချက်

```

printf_s("Enter the element to be deleted\n");
scanf_s("%d", &element); user ဆီက ရလာတဲ့ ဖုန်ခိုင်တဲ့ value ကို element ထဲကို ထည့်သား  
လိုကပါတယ်
for (i = 0; i < n; i++) looping ပတ်ပြီးတော့ array ထဲမှာ ရှိတဲ့ value နဲ့ user ထည့်  
လိုက်တဲ့ value နဲ့ တူလားလို့ တိုက်စစ်ပါတယ်။ user ထည့်သားတဲ့  
value ကို element ထဲမှာ ထည့်သားတော်ကို သတိပြုပါ။
{
    if (vectorx[i] == element) တကယ်လို့ vectorx array ထဲမှာ ရှိတဲ့ value နဲ့ element ထဲက value ကို  
သွားတဲ့အခါ found = 1 ဆိုပြီး သတ်မှတ်လိုက်ပါတယ်။ pos=i ဆိုပြီး  
element နဲ့တူတဲ့ index နေရာကို မှတ်ထားလိုက်ပါတယ်။
    {
        found = 1;
        pos = i;
        break;
    }
}

```

```

if (found == 1) {
    for (i = pos; i < n - 1; i++) {
        vectorx[i] = vectorx[i + 1];
    }
    printf_s("The resultant vector is \n");
    for (i = 0; i < n - 1; i++) {
        printf_s("%d\n", vectorx[i]);
    }
}

```

အထက်ပါ ငါ့မှာ array ထဲက ရှိတဲ့ valueနဲ့ element ထဲက value နဲ့တူသွေးတဲ့ အတွက် found = 1 ဆိုပါ။ သတ်မှတ်လိုက်ပါပြီ။

index value သိမ်းထားတဲ့ pos ကို loop ထဲမှာ ပြန်ထည့်ပြီ။ pos ကသာ 3 ဖြစ်ခဲ့မယ်ဆိုရင် n=5; n-1=4 ဖြစ်တဲ့ အတွက် loop နှစ်ကြမ်းပတ်ပြီး user ဖယ်ထုတ်ချင်တဲ့ element ကို ဖယ်ထုတ်သွားမှာပါ

vectorx array ထဲမှာ ရှိတဲ့ element တွေကို loop ပတ်ပြီး ထုတ်လိုက်ပါတယ်။ n-1 နေရာမှာ n လိုပဲ ရေးပြီး ဆက်လက် စမ်းကြည့်လိုရပါတယ်။ ပိုပြီးစိတ်ဝင်စားဖို့ ကောင်းသွားမှာပါ

Example 9 array's elements are stored at contiguous locations in C programming

နောက်ထပ် program တစ်ပုဒ်အနေနဲ့ c programming မှာ array ရဲ့ element တွေဟာ memory မှာ တစ်ဆက်တည်း allocation လုပ်သွားတာကို ရေးမှာဖြစ်ပါတယ်။ sizeof operator ကို သုံးပြီးရေးသွားမှာပါ။ sizeof operator အကြောင်းကို အရင်သင်ခန်းစာတွေမှာ သေချာ ရေးပေးခဲ့ပါတယ်။

```

#include<stdio.h>
#include<conio.h>
int main() {
    int arr[5], i;
    printf("Size of Integer in this compiler: %d\n", sizeof(int));
    for (i = 0; i<5; i++) {
        printf("Address is [%d] is %u\n\n", i, &arr[i]);
    //array ရဲ့ address ကိုထုတ်ချင်တဲ့အတွက် arr[i] ရှေ့မှာ  &(amp;address of) ဆိုတာလေး ထည့်ပေးထားတာပါ။
    }
    getch();
    return 0;
}

```

```
Size of Integer in this compiler: 4
Address is [0] is 6422168
```

```
Address is [1] is 6422172
```

```
Address is [2] is 6422176
```

```
Address is [3] is 6422180
```

```
Address is [4] is 6422184
```

```
Malloc()
Calloc()
Free()
Realloc()
```

```
Malloc()
Calloc()
Free()
Realloc()
```

DYNAMIC MEMORY ALLOCATION IN C

```
Malloc()
Calloc()
Free()
Realloc()
```

```
Calloc()
Free()
Realloc()
```

```
Malloc()
Calloc()
Free()
Realloc()
```

WIN HTUT
GREEN HACKERS TEAM 12/25/19

Dynamic Memory Allocation in C

How Memory Management in C works?

C programming မှာ variable တစ်ခုကို စတင်ခြော့ဖြေလိုက်သည့်နှင့် တစ်ပြိုင်နှင်း c compiler သည် memory ပေါ်မှာ variable ကို ခြော့ဖြေလိုက်သော data type ပေါ်မူတည်ပြီး နေရာစယူပါတယ်။

example အနေဖြင့် int a=10; လို့ စတင်ရေးလိုက်လျှင် memory ပေါ်မှာ int(integer) ရဲ့ size ဖြစ်တဲ့ 4 bytes ကို နေရာစယူပါတယ်။ 4 bytes သည် ပုံသေမဟုတ်ပါဘူး။ မိမိတို့ အသုံးပြုနေသော platform ပေါ်မှာ မူတည်ပြီး ကွဲပြားနိုင်ပါသေးတယ်။ ထိုကဲ့သို့ variable များသိမ်းဆည်းနေရာယူသော memory ကို stack memory လို့ခေါ်ပါတယ်။

မိမိတို့ platform မှာ ဘယ် data type သည် ဘယ်လောက် size နေရာယူတယ်ဆိုတာကို သိလိုလျှင် sizeof operator ကိုသုံးပြီး ကြည့်နိုင်ပါတယ်။

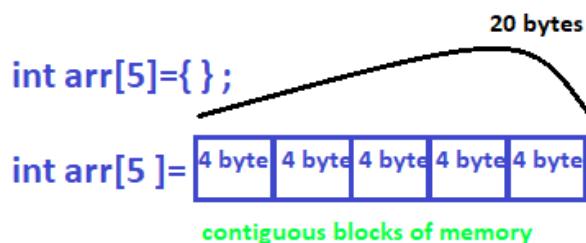
```
int a=10 ;
```

```
int a= 4 bytes
```

```
printf("sizeof int is %d:",sizeof(int));
```

အထက်ပါ code line သည် int (integer) data types ဘယ်လောက် size ယူသလဲ ဆိုတာကို သိစေရန် run ကြည့်နိုင်ပါတယ်။

Array များသည် contiguous blocks of memory ဖြစ်တဲ့အတွက် block တစ်ခုခြင်းစီမှာ size တစ်ခုစီ ရှိပါတယ်။



ဆိုလိုချင်တာက variable or array တစ်ခုကို စတင် ကြော်ဖြေလိုက်တာနဲ့ automatic နေရာယူသွားပါတယ်။ အထက်ပါပုံမှာဆိုရင် array အခန်းကို 5 ခန်းပဲ ယူမယ်လို့ ပြောထားပါတယ်။ တကယ်လို့ program ရဲ့လိုအပ်ချက်အရ 10 ခန်းလိုလာတာမျိုးရှိတတ်ပါတယ်။ ထို့ကြောင့် program runtime မှာ array size ကိုချဲ့ဖို့ လိုလာပါတယ်။ ထိုနည်းတူ array ခန်းကို 5 ခု ကြော်ဖြေထားပြီး

တကယ်တမ်း program မှာ 2 ခုပဲ သုံးထားတယ်ဆိုရင် နေရာ သုံးခုစာအတွက် memory wasteဖြစ်ပါတယ်။

ထိုကဲ့သို့ လိုတဲ့အချိန် မှာ လိုသလို memory ကိုချေယူရန် မလိုတဲ့ အချိန်မှာ မလိုသလို လျှော့ချုပ်လုပ်ဆောင်ခြင်းကို dynamic memory allocation လိုခေါ်ပါတယ်။

C programming မှာ ထိုကဲ့သို့ process များ လုပ်ဆောင်ရန်အတွက် standard library function လေးခုကို support လုပ်ပေးထားပါတယ်။

Malloc()

Calloc()

Free()

Realloc()

အထက်ပါ function လေးခုကို သုံးနိုင်ရန် stdlib.h ကို ကြော်ကြော်ပေးရန် လိုအပ်ပါတယ်။

Malloc or Memory Allocation

Malloc() function သည် memory block တစ်ခုတည်း နေရာယူရာတွင် အသုံးပြုပါတယ်။

Syntax

```
ptr = (cast_type *) malloc (byte_size);
```

Example program

ဥပမာအားဖြင့် 20 bytes လောက် block တစ်ခုတည်းပုံစံနဲ့ နေရာ ယူချင်လိုသောအခါတွင် malloc ကိုသုံးပြီး အောက်ပါအတိုင်းရေးရပါတယ်။

```
int *ptr;  
ptr = malloc(5 * sizeof(*ptr));
```

အထက်ပါ Program အား run ပြီးသော အချိန်တွင် pointer တစ်ခုကို ရပါမည်။ ထို pointer သည် 20 bytes ရှိသော memory address ကို ကိုင်ထားမှာ ဖြစ်ပါတယ်။ ထို pointer ကို အသုံးပြုပြီး data များကို store လုပ်နိုင်ပါပြီ။

```
Int *ptr= malloc(10*sizeof(int));
```

အထက်ပါ program ကို run ပြီးချိန်းမှုတွင် memory ပေးရန်နေရာ ရှိပါက ပိမိလိုချင်သော pointer တစ်ခုကို return ပြန်ပေးပါလိမ့်မယ်။ pointer သည် နေရာ ပေးလိုက်သော memory address ကို point

ထောက်ထားမှာ ဖြစ်ပါတယ်။ အထက်ပါ Program တွင်မူ int type ဖြင့် နေရာယူခြင်း ဖြစ်ပါတယ်။ အကယ်၍ memory ပေးရန် နေရာမရှိပါက NULL pointer ကို return အနေဖြင့် ပြန်ပေးပါလိမ့်မယ်။

next example program to test....

```
#include<stdio.h>
#include<conio.h>
#include<stdlib.h>
int main(){
int *ptr;
ptr=malloc(10 * sizeof(*ptr));
if(ptr!=NULL){
    printf("memory was created at %d",ptr);
} else{
    printf("cannot create memory block something wrongs");
}
}
```

အထက်ပါ program ကို run ပြီးပါက memory block တစ်ခု create လုပ်နိုင်တယ်ဆိုရင် memory address တစ်ခုကို ဖော်ပြပေးမှာ ဖြစ်ပြီး မရပါက မရကြောင်းစာကြောင်း တစ်ကြောင်းထဲတ်ပြပေးမှာ ဖြစ်ပါတယ်။

Adding data to our dynamic memory

ကျွန်တော်တို့ create လုပ်ခဲ့ပြီးသော memory သည် contiguous memory ဖြစ်တဲ့ အတွက် array ကဲ့သို့ လုပ်ဆောင်နိုင်ပါတယ်။ ထိုနည်းတူ arithmetic + ကို သုံးပြီးတော့လည်း data များ ထည့်နိုင်ပါတယ်။ အောက်ပါ program ကို လေ့လာ ကြည့်ပါ။

```
int main(){
int *ptr;
ptr=malloc(10 * sizeof(*ptr));
if(ptr!=NULL){
    printf("memory was created at %d \n",ptr);
    *(ptr+0)=10;
    *(ptr+1)=20;
    *(ptr+3)=30;
    *(ptr+4)=40;
    *(ptr+5)=50;
}
```

```

*(ptr+6)=60;
}else{
    printf("cannot create memory block something wrongs");
}

int i=0;
for(i=0; i<7 ; i++){
    printf(" %d value is %d \n",i,*(ptr+i));
}
memory was created at 6627744
0 value is 10
1 value is 20
2 value is 0
3 value is 30
4 value is 40
5 value is 50
6 value is 60

Process returned 0 (0x0)   execution time : 0.030 s
Press any key to continue.

```

Program ရှင်းလင်းချက်

`*(ptr+0)=10` သည် index 0 နေရာသို့ 10 ဆိုသည့် data ထည့်ခြင်းဖြစ်ပါတယ်။ `ptr + 0` တွင် `ptr` သည် memory address ဖြစ်ပြီး 0 သည် သာမန် number တစ်ခုဖြစ်ပါတယ်။ memory address တစ်ခုနှင့် သာမန် number တစ်ခုကိုပေါင်းပေးရန် နောက်က number ကို type အလိုက် ပြောင်းလဲပေးရပါတယ်။ ဤပေါ်မှု ဒုတိယ code line တွင် `*(ptr+1)=20` ; ဟုရေးထားပါတယ်။ `ptr` နှင့် 1 ကိုပေါင်းရန် ပထမဆုံး နောက်က number ဖြစ်တဲ့ 1 ကို pointer ခဲ့`type` ဖြစ်တဲ့ 4 နဲ့ မြောက်ပေးရပါတယ်။ ထိုကဲ့သို့ 4 နဲ့ မြောက်လို့ရလာတဲ့ `value` ကိုမှ `ptr(memory address)` နဲ့ ပေါင်းပေးရပါတယ်။ ထိုကဲ့သို့ ပေါင်းပေးပြီးလျှင် memory address အသစ် တစ်ခု ရပါမည်။ ထို `address` မှာမှ 20 ဆိုသည့် `value` ကို သို့လောင်ခြင်း ဖြစ်ပါတယ်။

`*(ptr + (x*(data_type)) =20 ;`

အခြားသော code line များသည်လည်း ထိန်းတူ ဖြစ်ပါတယ်။ memory block တဲ့မှာ မည်သည့် data မျှ ထည့်မထားလျှင်မူ zero ကို default အနေဖြင့် compiler က သတ်မှတ်ပေးထားပါတယ်။

Program နောက်ဆုံးတွင်မူ looping ပတ်ပြီး output ထုတ်ထားပါတယ်။ index 2 နေရာတွင် မည်သည့် value မျှ ထည့်မထားသည့်အတွက် 0 ထုတ်ပေးသည်ကို မြင်ရပါမည်။

Calloc()

Calloc ဆိုသည်မှာ contiguous allocation ကိုဆိုလိုခြင်း ဖြစ်ပါတယ်။ malloc သည် memory block တစ်ခုတည်းကိုသာ ဖန်တီးသော်လည်း calloc သည် memory blocks အများကြီးကို ဖန်တီးလိုသော အခါမျိုးတွင် အသုံးပြုပါသည်။ calloc function ကို အသုံးပြုပြီး ဖန်တီးလိုက်သော memory block များမှာ တစ်ခုနှင့် တစ်ခု မှာ size များ တူညီပါသည်။

Syntax:

```
Int *ptr=(cast_type *)calloc(n , size);
```

အထက်တွင် ဖော်ပြထားသော syntax သည် memory blocks အများ ကြီး ယူရန် calloc function ကို သုံးထားပြီး n အရေအတွက်ထိ same size များယူမှာ ဖြစ်ပါတယ်။ Memory block များ ယူပြီးသည့်နှင့် တစ်ပြိုင်နှင့် zero များကို initialized အနေနဲ့ သတ်မှတ်ထားမှာ ဖြစ်ပါတယ်။ Process အောင်မြင်လျှင် pointer တစ်ခုကို return ပြန်ပေးမှာပါ ထို pointer သည် memory ရဲ့ ပထမဆုံး block ကို ထောက်ထားသည့် address ကို ကိုင်ထားမှာပါ။ memory နေရာယူရာတွင် memory မလုံလောက်တာမျိုး သို့မဟုတ် တြေားသော ပြသနာများရှိပါက null pointer ကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။

```
ptr = calloc(10, sizeof(int));
```

10 သည် memory block အရေအတွက် ဖြစ်ပြီး sizeof(int) သည် block တစ်ခု ခြင်းစီရဲ့ size ကို သတ်မှတ်ပေးခြင်းဖြစ်သည်။ example program ကို အောက်တွင် ဖော်ပြထားပါသည်။

```
#include <stdio.h>
#include<stdlib.h>

int main() {
    int i, * ptr, sum = 0;
    ptr = calloc(10, sizeof(int));
    printf("created memory of each size is : %d\n",sizeof(ptr));
```

```

if (ptr == NULL) {
    printf("somethings wrong with memory ");
    exit(0);
}
for (i = 0; i < 10; ++i) {
    * (ptr + i) = i;
    sum += * (ptr + i);
    printf(" %d : value is %d\n", i, sum);
}
printf("Sum of all values at each block = %d", sum);
return 0;
}

```

Program Output

```

created memory of each size is : 4
0 : value is 0
1 : value is 1
2 : value is 3
3 : value is 6
4 : value is 10
5 : value is 15
6 : value is 21
7 : value is 28
8 : value is 36
9 : value is 45
Sum of all values at each block = 45
Process returned 0 (0x0)   execution time : 0.039 s
Press any key to continue.

```

program ကို run ပြီးသော အချိန်တွင် output အနေဖြင့် ပုံပါအတိုင်း ရရှိပါမည်။ *(ptr+i)=i; သည် looping ပတ်ပြီး ရလာသော i တန်ဖိုးကို memory block ထဲသို့ထည့်ခြင်း ဖြစ်ပါတယ်။ sum+=*(ptr+i); သည် သက်ဆိုင်ရာ memory block ထဲမှ value ကို sum ဆိုသည့် variable

ထဲသို့ပေါင်းထည့်ခြင်း ဖြစ်ပါတယ်။ ထို့ကြောင့် နောက်ဆုံးရလဒ်သည် 45 ဖြစ်နေတာပါ။

Free()

Free function သည် အထက်တွင် နေရာယူလိုက်သော memory block များကို program ပြီးဆုံးချိန် မလိုတော့သော အချိန်၌ ယူထားသော နေရာများကို ပြန်လည် ရှင်းလင်းရန် အသုံးပြုပါသည်။ free ကို de-allocation လိုလည်း ခေါ်ပါတယ်။ malloc and calloc function တို့သည် deallocate ပြန်မလုပ်နိုင်တဲ့အတွက် memory waste ဖြစ်ပါတယ်။ ထိုကဲ့သို့ memory waste ဖြစ်သော ပြဿနာကို free() က ဖြေရှင်းပေးပါတယ်။

Syntax :

Free(ptr)

Malloc or calloc သုံးပြီးရလာ သော pointer ကို free function ထဲတွင် ထည့်ပေးလိုက်ပါက ထို pointer ထောက်ထားသော memory blocks များကို ရှင်းလင်း ပေးပါတယ်။

Realloc()

Realloc function သည် memory block အသစ်တစ်ခုကို ဖန်တီးရန် အသုံးပြုပါတယ်။ ရှိပြီးသား memory block တစ်ခုကို ထပ်တိုးလိုသော်လည်းကောင်း သို့မဟုတ် လျော့လိုသောအခါတွင် အသုံးပြုပါတယ်။ return အနေဖြင့် pointer အသစ်ကို လည်ကောင်း သို့မဟုတ် အရင် အဟောင်း address ကိုလည်းကောင်း memory block များအား update လုပ်ပြီး ပြန်ပေးနိုင်ပါတယ်။ မူရင်း သိမ်းဆည်းထားသော data များကို တော့ နောက်ထပ် memory block အသစ် ဖန်တီးပြီးသည့်နှင့် တစ်ပြင်နှင့် အသစ်နေရာမှာ ပြန်လည် copy လုပ်ပေးထားမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် data loss မဖြစ်ပဲ safe ဖြစ်ပါတယ်။ realloc process သည်လည်း successful မဖြစ်ပါက NULL pointer ကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။

```
#include <stdio.h>
#include<stdlib.h>
int main () {
    char *ptr;
    ptr = (char *) malloc(10);
    strcpy(ptr, "Programming");
    printf(" %s, Address = %u\n", ptr, ptr);
    ptr = (char *) realloc(ptr, 20);
    strcat(ptr, " In 'C'");
    printf(" %s, Address = %u\n", ptr, ptr);
    free(ptr);
    return 0;
}
```

Program output

```
Programming, Address = 7938456
Programming In 'C', Address = 7938456

Process returned 0 (0x0)   execution time : 0.031 s
Press any key to continue.
```

Line number 4 တွင် char pointer တစ်ခုကို ကြော်ဖြေပေးထားပါတယ်။

Line number 5 တွင် 10 bytes size ရှိတဲ့ memory block တစ်ခုကို malloc function ကိုသုံးပြီး ဖန်တီးလိုက်ပါတယ်။

Line number 6 တွင် strcpy function ကိုသုံးပြီး programming ဆိုသည့် စာသားကို ptr ဆိုသည့် char pointer ထဲသို့ copy ကူးလိုက်ပါတယ်။

Line number 7 တွင် pointer ထဲတွင် သိမ်းထားသော စာသားနှင့် memory address ကို print ထုတ်ပြထားတာပါ။

Line number 8 တွင် malloc function သုံးပြီး တည်ဆောက်ထားသော memory block ကို realloc function သုံးပြီး bytes ထပ်တိုးလိုက်ပါတယ်။

Line number 9 တွင် In C ဆိုသည့် စာသားကို strcat function သုံးပြီး အသစ်တည်ဆောက် လိုက်သော memory block ထဲသို့ ထပ်ထည့်လိုက်ပါတယ်။ သတိထားရမည့် အချက်သည် realloc လုပ်သော်လည်း မူရင်း ရှိသော data များသည် အသစ်လုပ်လိုက်သော memory block ထဲတွင် copy ကူးထားမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် မူရင်းရှိသော စာသားနှင့် In C ဆိုသည့် စာသားတို့မှာ ပေါင်းသွားမှာ ဖြစ်ပါတယ်။

Line 10 တွင် memory address နှင့် data များကိုဖော်ပြပေးထားတာ ဖြစ်ပါတယ်။

Line number 11 တွင်မူ ယခု program ၌ သုံးထားသော memory block များကို memory waste ဖြစ်ခြင်းမှ ကာကွယ်နိုင်ရန် free(ptr) ဟုရေးပြီး ရှင်းလင်းပေးလိုက်တာ ဖြစ်ပါတယ်။

C File Processing

ကျွန်တော်တို့တွေ program တွေ ရေးကြတယ် data တွေ ထည့်တယ်၊ ထုတ်တယ် calculation တွေလုပ်ကြတယ်။ program ရပ်သွားသည့်နှင့် တစ်ပြိုက်နက် data အားလုံး ပျက်သွားကြတယ်။ မည်သည့် record မှ မကျွန်ခဲ့ပါဘူး။ ထိုသို့သော အခြေအနေများတွင် မပျောက်ပျက်သွားအောင် data များအား file များထဲတွင် သိမ်းဆည်းခဲ့နိုင်ပါသည်။ ထိုနည်းတူ သိမ်းဆည်းထားသော file မှ data များကိုလည်း မိမိတို့ program တွင် ပြန်ခေါ်သုံးနိုင်ပါသည်။ ထိုကဲ့သို့ data ဖတ်ခြင်း၊ data ထည့်ခြင်းများ လုပ်ဆောင်ခြင်းကို file processing ဟုခေါ်ပါသည်။ c programming တွင် file processing ပြုလုပ်ရန် standard library function များစွာကို support လုပ်ပေးထားပါသည်။ အသုံးများသည့် file processing function များကို အောက်တွင် ဖော်ပြထားပါတယ်။



Creating a File

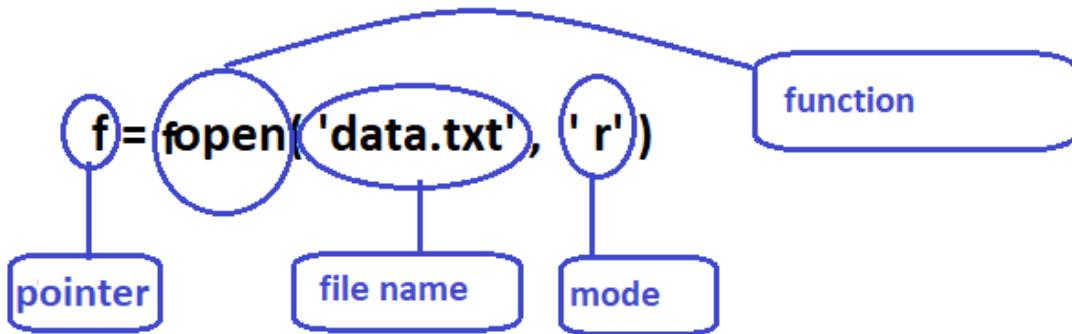
file နဲ့ပတ်သက်သည့် အလုပ်များ processing များ စလုပ်မည်ဆိုလျှင် file တစ်ခု ရှိရပါမည် သို့မဟုတ် create လုပ်ရပါမည်။ ယခု သင်ခန်းစာတွင်တော့ file တစ်ခုကို create လုပ်ပါမည်။ file တစ်ခုအား create လုပ်ရန် fopen() ဆိုသည့် function ကိုသုံးနိုင်ပါသည်။ fopen function သည် parameter နှစ်ခု ယူပါသည်။ ပထမတစ်ခုသည် မိမိ create လုပ်ချင်သော file သို့မဟုတ် မိမိ ဖုန်တီးလိုသော file name ဖြစ်ပါသည်။ ဒုတိယ parameter သည် mode ဖြစ်ပါသည်။ mode ဆိုသည်မှာ file တစ်ခုအား မည်သည့်ပုံစံနှင့် ကိုင်တွယ်မည်ဟု ဆိုလိုခြင်းဖြစ်ပြီး ဥပမာအနေဖြင့် ဒုတိယ parameter နေရာမှာ “r” ကို သုံးထားမည်ဆိုလျှင် file တစ်ခုအား ဖတ်မည်ဟု ဆိုလိုခြင်းဖြစ်ပါသည်။

syntax and sample program

```
FILE *fptr;
fptr= fopen ( " file name " , " mode " );
```

```
FILE *fptr;
fptr= fopen ( " winhtut.txt " , " r " );
```

အထက်ပါ program တွင် winhtut.txt ဆိုသည့် file တစ်ခု အား “r” ဖတ်ရန် ဖွင့်မည်ဟု ဆိုလိုခြင်းဖြစ်ပြီး file ကိုလည်း ဖွင့်လို့ရမည် ဖတ်လို့လည်း အဆင်ပြုမည်ဆိုလျှင် file pointer တစ်ခု ပြန်ပေးပါမည်။ ထို file pointer ကို fptr ဆိုသည့် pointer ထဲသို့ ထည့်မည် ဖြစ်ကြောင်း ရေးထားခြင်းဖြစ်ပါသည်။



File Handling Mechanism

Modes of file processing

r သည် file တစ်ခုအား ဖတ်ရန် ဖြစ်သည်။ အကယ်၍ file မရှိဘူးဆိုလျင် fopen() function သည် return အနေဖြင့် NULL pointer ကို ပြန်ပေးမှာ ဖြစ်ပါတယ်။

w သည် file တစ်ခုအား data များ writing လုပ်ရန်ဖြစ်သည်။ အကယ်၍ file မရှိဘူးဆိုလျင် create လုပ်မှာ ဖြစ်ပါတယ်။

a သည် file တစ်ခုအတွင်းသို့ data များ append လုပ်ပြီး ရှိက်ထည့်မည်ဟု ဆိုလိုခြင်းဖြစ်သည်။ a သည် လည်း file မရှိဘူးဆိုလျင် create လုပ်မှာ ဖြစ်ပါတယ်။

rb သည် file တစ်ခုအား binary mode ဖြင့်ဖတ်ရန် ဖွင့်မည်ဟု ဆိုလိုခြင်းဖြစ်ပြီး အကယ်၍ file မရှိလျင် fopen() သည် NULL pointer ကို return ပြန်မှာ ဖြစ်ပါတယ်။

wb သည် file တစ်ခုအား binary mode ဖြင့် data များ ရှိက်ထည့်ရန် open လုပ်မည်ဟု ဆိုလိုခြင်းဖြစ်ပြီး file မရှိဘူးဆိုလျင် create လုပ်မှာ ဖြစ်ပါတယ်။

ab သည် file တစ်ခုအား binary mode ဖြင့် data များ append လုပ်ရန်ဖွင့်မည်ဟု ဆိုလိုခြင်း စံသည်။

r+ သည် file တစ်ခုအား reading and writing mode နှစ်ခု လုံးအတွက် ဖွင့် မည်ဟု ဆိုလိုခြင်း ဖြစ်ပါသည်။ အကယ်၍ file မရှိဘူးဆိုလျင် NULL pointer ကို return ပြန်ပါမည်။

a+ သည် file တစ်ခုအား reading and appending mode ဖြင့် ဖွင့်မည်ဟု ဆိုလိုခြင်း ဖြစ်ပါသည်။

w+ သည် file တစ်ခုအား reading and writing mode နှစ်ခုလုံးအတွက် ဖွင့်မည်ဟု ဆိုလိုခြင်း ဖြစ်ပြီး အကယ်၍ file မရှိပါက create လုပ်မည်ဖြစ်ပါသည်။

rb+ သည် file တစ်ခုအား binary mode ဖြင့် reading and writing နှစ်ခုလုံး လုပ်မည်ဟု ဆိုလိုခြင်း ဖြစ်ပြီး file မရှိပါက NULL pointer ကို return ပြန်ပါမည်။

wb+ သည် file တစ်ခုအား binary mode ဖြင့် reading and writing နှစ်ခုလုံး လုပ်မည်ဟု ဆိုလိုခြင်း ဖြစ်ပြီး file မရှိပါက create လုပ်မည် ဖြစ်ပါသည်။

ab+ သည် file တစ်ခုအား binary mode ဖြင့် reading and appending နှစ်ခုလုံး လုပ်မည်ဟု ဆိုလိုခြင်းဖြစ်ပြီး file မရှိပါက create လုပ်မည် ဖြစ်ပါသည်။

Writing to a File

File တစ်ခု အတွင်းအား data ပျေားမှတ်သားဖို့ အတွက်ဆိုလျှင် stdio library ထဲမှ fputc , fputs , fprintf တို့ကို သုံးနိုင်သည်။

fputc(char , filePointer) သည် character တစ်လုံးချင်းစီ file တစ်ခုအတွင်းအား ရှိက်ထည့်ရန် အသုံးပြုပါသည်။ fputc သည် arguments နှစ်ခုယူပြီး ပထမတစ်ခုသည် character ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် မိမိရေးထားလိုသော file အား ထောက်ထားသည့် file pointer ဖြစ်ပါသည်။

fputs(str , filePointer) သည် strings အား file တစ်ခု အတွင်းသို့ ရှိက်ထည့်ရန် အသုံးပြုပါသည်။ fputs သည်လည်း arguments နှစ်ခုယူပြီး ပထမတစ်ခုသည် မိမိရှိက်ထည့်လိုသော string ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် မိမိရေးထည့်လိုသော file အား ထောက်ထားသည့် file pointer ဖြစ်သည်။

fprintf(filePointer, str, variableList) သည် strings များအား file တစ်ခုအတွင်းသို့ ရှိက်ထည့်လိုသောအခါမှာ သုံးပါတယ်။ fprintf တွင် arguments သုံးခုပါဝင်ပြီး ပထမတစ်ခုသည် file အား ထောက်ထားသည့် filepointer ဖြစ်ပြီး ဒုတိယတစ်ခုသည် မိမိရေးထည့်လိုသော str format နှင့် တတိယတစ်ခုသည် varaiable များ ရေးထည့်လိုပါက ရေးထည့်နိုင်သည့် variable list ဖြစ်သည်။

Example Programs for Writing to a File

```
#include <stdio.h>
int main() {
    int i;
    FILE * fptr;
    char str[] = "We are Myanmar Crazy Coder\n";
    fptr = fopen("hello.txt", "w");
    for (i = 0; str[i] != '\n'; i++) {
        fputc(str[i], fptr);
    }
    fclose(fptr);
    return 0;
}
```

Program ရှင်းလင်းချက်

line number 4 တွင် File တစ်ခုအား ထောက်ရန်အတွက် file pointer တစ်ခုအား စတင် ကြော်ပြာထားပါသည်။

line number 5 တွင် မိမိ ရေးထည့်လိုသော character array တစ်ခုအား ရေးသားထားပါသည်။ ထိုသို့ရေးထည့်ရာတွင် နောက်ဆုံး၌ \n ကို ထည့်ထားပါသည်။ \n ကို ထည့်ထားရသည့် အကြောင်းရင်း မှာ စာကြောင်းအဆုံးကို သိစေရန် ဖြစ်သည်။

line number 6 တွင် fopen function ကိုသုံးပြီး hello.txt ဆိုသည့် text file ကို ဖွင့်ပါသည်။ အကယ်၍ hellot.txt ဆိုသည့် file မရှိဘူးဆိုလျှင် createလုပ်မည်ဖြစ်ပါသည်။ ဒုတိယ argument နေရာတွင် w ကို သုံးထားသည့်အတွက် ယခု file အား writing mode ဖြင့် ဖွင့်မည်ဟု ဆိုလိုခြင်း ဖြစ်ပါသည်။

line number 7 တွင် for loop ဖြင့် str ဆိုသည့် array ထဲမှာ ရှိသည့် character အားလုံး စာကြောင်းဆုံးသည်အထိ ထုတ်ရန် ဖြစ်သည်။

line number 8 တွင် fputc ဆိုသည့် function ကိုသုံးပြီး str ထဲမှ ရှိသော စာသားများကို fptr ထဲသို့ ရေးထည့်ပါသည်။

line number 10 တွင် Process အားလုံး ပြီးသွားသည့်အတွက် fclose() function ကိုသုံးပြီး file အား ပြန်ပိတ်ပေးလိုက်ပါသည်။

program အားလုံး run ပြီးသွားတဲ့အခါမှာ မိမိယခု program file ရှိသော နေရာတွင်ပင် hell.txt ဆိုသည့် text file တစ်ခု ရှိနေသည်ကို မြင်ရပါမည်။ ထို text file ကို ဖွင့်ကြည့်ပါ။ We are Myanmar Crazy coder ဆိုသည့် စာသားကို မြင်ရပါမည်။

Fputs Example program

```
#include <stdio.h>
int main() {
    FILE * fptr;
    fptr = fopen("1hello.txt", "w+");
    if(fptr==NULL){
        printf("Something wrongs");
    }else{
        fputs("Myanmar Crazy Coders(GreenHackers)\n ", fptr);
        fputs("It is more clear than fputc() \n", fptr);
        printf("All process are successfully");
        fclose(fptr);
    }
    return (0);
}
```

program ရှင်းလင်းချက်

Line number 4 တွင် fopen function ကိုသုံးပြီး 1hello.txt ဆိုသည့် file ကို mode သည် w+ ဖြစ်သည့် အတွက် reading and writing mode နှစ်မျိုးလုံးဖြင့် ဖွင့်လိုက်ပါသည်။ ထို file မရှိပါက အသစ်တစ်ခု create လုပ်မည် ဖြစ်ပါသည်။

line number 5 တွင် file pointer အား NULL ဖြစ်နေသလား ဆိုတာ စစ်တာပါ။ အကယ်၍ line number 4 မှ Process များ မအောင် မြင်ခဲ့ပါက file pointer သည် NULL ဖြစ်နေမှာပါ။ ထိုသို့ ဖြစ်နေပါက something wrongs ဆိုသည့် စာသားကို ထုတ်ပြပြီး program ရပ်သွားမှာပါ။

line number 9 တွင် fputs ကို သုံးပြီး စာသားများအား fptr ဆိုသည့် file pointer ထဲသို့ ရေးထည့်ပါသည်။ fptr သည် 1hello.txt ရဲ့ address ကို ကိုင်ထားသည့်အတွက် fptr သည် 1hello.txt ဆိုသည့် text file ကို ကိုယ်စားပြုထားခြင်း ဖြစ်သည်။

Program အားလုံး run ပြီးပါက ယခု လက်ရှိ c program file ရှိသည့် နေရာတွင်ပင် 1hello.txt ဆိုသည့် file ကို မြင်ရပါမည်။ ထို file ကို ဖွင့်ကြည့်ပါက မိမိ တို့ရေးထားသော စာသားများကို မြင်ရပါမည်။

fprintf example program

```
#include <stdio.h>
int main() {
    FILE *fptr;
    int account=101;
    char name[]="WinHtut";
    double balance=10.1;
    fptr = fopen("1hello.txt", "w");
    if(fptr==NULL){
        printf("Somethings wrong with this process");
    }else{
        fprintf(fptr, "%d %s %.2f\n", account, name, balance);
        fclose(fptr);
        printf("All process are complete\n");
    }
    return 0;
}
```

program ရှင်းလင်းချက်

fprintf function သည် data များအား formatted ပုံစံဖြင့် file တစ်ခုအတွင်းသို့ ရေးထည့် နိုင်ပါသည်။ အထက်ပါ program တွင်လည်း line number 11 တွင် %d %s %.2f စသည့် format များကိုသုံးပြီး နောက်က variable များကို fptr ဆိုသည့် file အတွင်းသို့ data များ ရေးထည့်ပါသည်။

program အားလုံး run ပြီးသွားလျှင် 1hello.txt ဆိုသည့် text file တစ်ခုကို တွေ့ရပါမည်။ ထို text file ထဲတွင် မိမိတို့ ရေးထည့်လိုက်သော variable များကို မြင်ရပါမည်။

Reading Data From a File

File တစ်ခုအတွင်းမှ data များကို ဖတ်လို သော အခါ နည်းလမ်း ၃ မျိုးဖြင့် function သုံးခု ကိုသုံးပြီး ဖတ်နိုင်ပါတယ်။

fgetc(filePointer) သည် file တစ်ခု အတွင်းမှ စာလုံးတစ်လုံးခြင်းစီကို ဖတ်ရန် သုံးပါသည်။ argument အနေဖြင့် filePointer ကို ယူပြီး return အနေဖြင့် file တစ်ခုအတွင်းမှ စာလုံး အားလုံး မကုန်မချင်း

တစ်လုံးခြင်းစီကို return ပြန်ပေးပါသည်။ စာလုံးအားလုံး ကုန်သွားလျှင် EOF ကို return ပြန်ပေးပါသည်။ EOF ဆိုသည်မှာ end of file ကို ဆိုလိုခြင်းဖြစ်ပြီး file တစ်ခုရဲ့ နောက်ဆုံးမှာ ရှိပါသည်။

fgetc Sample Program

ပထမဆုံးအနေဖြင့် c file တစ်ခု တည်ဆောက်ပါ။ ထိုနည်းတူ c file နှင့် same location တွင်ပင် 1hello.txt ဆိုသည် text file တစ်ခု တည်ဆောက်ပါ။ ထို text file ထဲတွင် မိမိ ရေးသားလိုသော စာသားများကို သုံးကြောင်းခန့် ရေးသားထားပါ။ ထို့နောက် c file ထဲတွင် အောက်ပါ code များကို ရေးသားပါ။

```
#include <stdio.h>
int main() {
    FILE *fptr;
    char c;
    printf("__Reading all data From File __\n");
    fptr = fopen("1hello.txt", "r");
    while ((c = getc(fptr)) != EOF) {
        printf("%c", c);
    }
    fclose(fptr);
    return 0;
}
```

Fgets Sample Program

```
#include <stdio.h>
int main() {
    FILE * fptr;
    char data_buffer[40];
    fptr = fopen("1hello.txt", "r");
    printf("__Reading a line From File__\n");
    fgets(data_buffer,50,fptr);
    printf("%s\n",data_buffer);
```

```

fclose(fptr);
return 0;
}

```

အထက်ပါ Program အား run ပြီးလျှင် မိမိတို့ ရေးထားသော file ထဲမှ ပထမ စာကြောင်း တစ်ကြောင်း အား ဖတ်လိုက်သည်ကို မြင်တွေ့ရမှုပါ။ သတိ **ပြရန် အချက် မှာဒုတိယ** parameter ဖြစ်သည့် စားလုံး အရေအတွက်အား များများ ထည့်ပေးထားရန် လိုအပ်သည်။ သို့မဟုတ်ပါက မိမိတို့ ထည့်ပေးလိုက်သည့် စာလုံး အရေးအတွက်အတိုင်းသာ ဖတ်မည် ဖြစ်သောကြောင့် စာလုံး အားလုံးကို မဖတ်ပဲ ကျင့်နေတာ မျိုးဖြစ်တတ်ပါသည်။

fscanf Sample Program

fscanf () သည် arguments သုံးခု ယူပါသည်။ ပထမတစ်ခုသည် file pointer ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် format ဖြစ်သည်။ တတိယတစ်ခုသည် variable and variable's address များ ဖြစ်ပါသည်။ ပထမ argument ရှိ file မှ data များကို ဒုတိယ argument မှ data type format အတိုင်း ဖတ်ပြီး တတိယ argument ရှိ variable များ စီသို့ ထည့်ပေးမှာ ဖြစ်ပါတယ်။ သတိပြရန် အချက်မှာ scanf and fscanf တို့သည့် space or newline character တို့ကို တွေ့လျှင် ဖတ်ခြင်း ရပ်သွားမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် fscanf သည် စကားစုတစ်ခုခြင်းစီကိုသာ ဖတ်နိုင်ပါသည်။

```

#include <stdio.h>
int main() {
    FILE * fptr;
    char data1[20],data2[20],data3[20],data4[20];
    fptr = fopen("1hello.txt", "r");

    printf("__Reading datas parse by parse From File_\n");
    fscanf(fptr,"%s %s %s %s ",data1,data2,data3,data4);
    printf("Reading a String :%s \n",data1);
    printf("Reading a String :%s \n",data2);
    printf("Reading a String :%s \n",data3);
    printf("Reading a String :%s \n",data4);
}

```

```

fclose(fptr);
return 0;
}

```

အထက်ပါ program အား run ကြည့်လျင် output အနေဖြင့် စကားစု တစ်ခုခြင်းစီကို တွေ့ရပါမည်။
အထက်တွင် ဖော်ပြခဲ့သည့် အတိုင်း fscanf သည် space or newline character တို့ကို မြင်လျင်
ဆက်မဖတ်တော့ပဲ ရုပ်သွားသောကြောင့် ဖြစ်သည်။ မိမိ လိုချင်သော format များဖြင့် လည်း
ဖတ်လို့ ရနိုင်ပါသေးတယ်။ sample program ကို အောက်မှာ ဖော်ပြထားပါတယ်။

Reading Data with Format Sample Program

```

#include <stdio.h>
int main() {

    FILE *fptr;

    fptr= fopen("1hello.txt","r+");

    char data1[20],data4[20];
    int code=0;
    float balance=0.0;
    printf("_____Reading data parse by parse from specific file_____ \n");

    fscanf(fptr , "%d %s %f %s",&code,data1,&balance,data4);

    printf("Reading data from file 1 ဆု %d\n",code);
    printf("Reading data from file 2 ဆု %s\n",data1);
    printf("Reading data from file 3 ဆု %.2f\n",balance);
    printf("Reading data from file 4 ဆု %s\n",data4);

    fclose(fptr);
    return 0;
}

```

Example Program

ယခု သင်ခန်းစာမျာတော့ File input output ကို သုံးပြီး company တစ်ခုရဲ့ employee စာရင်း ကို မှတ်တဲ့ program တစ်ပုဒ်ရေးသွားမှာ ဖြစ်ပါတယ်။ နောက်ပိုင်း သင်ခန်းစာတွေမှာ ဆိုရင်တော့ company မှာရှိတဲ့ data တွေကို ပြန်လည် sorting စီခြင်းတွေ လုပ်ဆောင်သွားမှာပါ။ ပထမဆုံး အနေဖြင့် clients.txt ဆိုတဲ့ text file တစ်ခု တည်ဆောက်ပါမယ်။ ထို text file ထဲမှာ user ထည့်ပေးလိုက်တဲ့ company employee စာရင်းတွေကို ထည့်ထားမှာပါ။

```
#include<stdio.h>
int main(){
FILE *fptr;

if((fptr=fopen("clients.txt","w"))==NULL){

    puts("File could not be opened");
}
else{

    puts("Enter the account , age , name , balance");
    puts("Enter EOF to end input.");
    printf(">:");

    unsigned int account;
    int age;
    char name[30];
    double balance;
    scanf("%d%d%20s%lf",&account , &age , name , &balance);

    while(!feof(stdin)){
        fprintf(fptr,"%d %d %s %.2f \n",account ,age, name , balance);
        printf(">:");
        scanf("%d%d%20s%lf",&account ,&age , name , &balance);

    }
}
fclose(fptr);
```

```

return 0;
}

```

Program ရှင်းလင်းချက်

ယခု Program ကို run ကြည့်မည်ဆိုလျှင် ပထမဆုံးအနေဖြင့် user ဆီက data 4 ခုကို တောင်းမှာပါ။ user ကနေ မှ တစ်ဆင့် data လေး ခုလုံးကို space ခြားပြီး အစဉ်လိုက် ထည့်ပေးရမှာပါ။ ထို data များကို feof ဟုတ်သလား မဟုတ်ဘူးလား ဆိုတာကို စစ်မှာပါ။ စစ်လိုက်လို့ feof မဟုတ်ဘူးဆိုရင် data များကို record လုပ်မည့် file အတွင်း ရေးထည့်မှာ ဖြစ်ပါတယ်။

feof function

feof function ကို end of file indicator လို့လည်း ခေါ်ပါတယ်။ ယခု program တွင် feof function ကို stdin ဆိုသည့် argument နဲ့ တွဲပြီး သုံးထားပါတယ်။ stdin ဆိုတာ standard input ကို ဆိုလိုတာပါ။ user ဆီက data များသည် standard input ဖြစ်သလား ဆိုတာကို စစ်လိုတဲ့အတွက်ပါ။ အကယ်၍ user ထည့်လိုက်သည့် data များသည် eof (end of file keys) တွေ ဖြစ်နေခဲ့မယ်ဆိုရင် user ဆီက data ထပ် မတောင်းတော့ပဲ program ကို ရပ်လိုက်မှာပါ။ eof key များသည် မိမိတို့အသုံးပြုသော operating system ပေါ်မှုတည်ပြီး ကွဲပွားနိုင်ပါတယ်။ Linux/Mac OS X/UNIX systems တွေမှာဆိုရင် Ctrl+d ပါ။ windows မှာ ဆိုရင်တော့ Ctrl+z then press enter ပါ။

while(!feof(stdin))

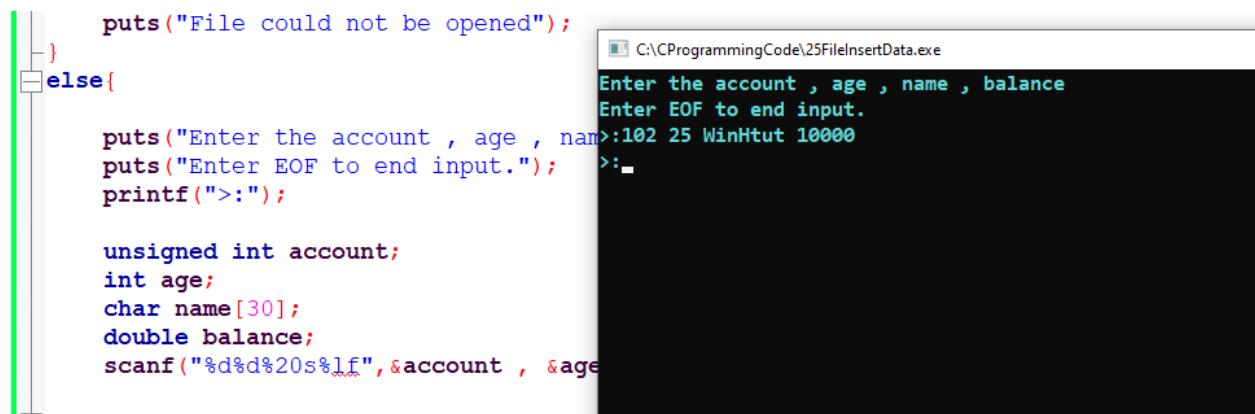
feof function သည် end-of-file indicator keys တွေကိုသာ တွေ့ခဲ့မယ်ဆိုရင် return value အနေဖြင့် nonzero(true) ကို return ပြန်ပါတယ်။ အခြားသော key တွေ ဖြစ်နေခဲ့မယ်ဆိုရင်တော့ zero ကို return ပြန်ပေးပါတယ်။ အထက်ပါ program ထဲမှာ ဆိုရင် !feof(stdin) လို့ ရေးထားတာ ဖြစ်တဲ့အတွက် eof keys တွေ ! မဟုတ်ဘူးဆိုရင် while ထဲက code တွေ အလုပ်လုပ်မယ်လို့ ဆိုလို ထားတာပါ။

```
fprintf(fptr,"%d %d %s %.2f \n",account ,age, name , balance);
```

```
printf(">:");
```

```
scanf("%d%d%20s%lf",&account ,&age , name , &balance);
```

အထက်ပါ code သုံးကြောင်းတွင် ပထမ တစ်ကြောင်းသည် user ထည့်ပေးလိုက်တဲ့ data တွေကို file တစ်ခုအတွင်းမှာ သွားသိမ်းတာပါ။ တတိယ ကြောင်းသည် user ထည့်ပေးလိုက်တဲ့ data လေးမျိုးကို ဖတ်တာပါ။



```

    puts("File could not be opened");
}
else{
    puts("Enter the account , age , name , balance
Enter EOF to end input.");
    puts("Enter EOF to end input."); >:
    printf(">:");

    unsigned int account;
    int age;
    char name[30];
    double balance;
    scanf("%d%d%20s%lf", &account , &age
        ...
        ...
        ...
}

```

Pass by reference in c++

C programming မှာတော့ pass by reference ကို support လုပ်ပေးမထားပါဘူး။ ဆိုလိုချင်တာက variable ရဲ့ reference ကို implicitly နည်းလမ်းနဲ့ pass လုပ်လို့ မရပါဘူး။ reference တစ်ခုကို create လုပ်ပြီးရင်တော့ explicitly နည်းလမ်းနဲ့ pass လုပ်နိုင်ပါတယ်။ သို့သော် explicitly နည်းလမ်းနဲ့ pass လုပ်မည်ဆိုလျှင်တော့ pointer and reference ကို သုံးရမည်ဖြစ်ပြီး ရေးရမည့် code ပိုများလာမှာဖြစ်သလို အနည်းငယ် ပို့ရှုပိုင်ပါတယ်။ ထို့ကြောင့် error တက်နိုင်ချေ လည်း ပိုများ လာပါမယ်။ reference နဲ့ pass လုပ်ချင်ရင် c++ ရဲ့ Pass by reference ကတော့ ပိုရှိုးရှင်းပြီး ရေးရသည့် code လည်းပိုနည်းမှာပါ။

Sample program with c

```

#include <stdio.h>
void function2(int &param) {
    printf("I've received value %d\n", param);
    param++;
}

int main(void) {
    int variable = 10;
    function2(variable);
    printf("pass by value in c++ :%d\n", variable);
    return 0;
}

```

အထက်ပါ program အား refernce.cpp ဟုရေးပြီး run ကြည့်မည်ဆိုလျင် error တက်မည်မဟုတ်ပဲ program ကောင်းမွန်စွာ အလုပ်လုပ်ပါမည်။ အထက်ပါ program ကိုပဲ copy ကူးပြီး refernce.c ဟု name ပေးပြီး run ကြည့်ပါက error ပြနေသည်ကို မြင်ရမှာပါ။ အထက်ပါ နည်းအတိုင်း ရေးသားခြင်းကို cpp တွင် ရသော်လည်း c တွင်မူ မရပါ။ c တွင် အောက်ပါနည်းအတိုင်း pointer ကို သုံးပြီး ရေးသားနိုင်ပါသည်။ function call ခေါ်သော နေရာတွင် address of ကို ထည့်ပေးရမည်ဖြစ်ပြီး ထိုသို့ ထည့်ခြင်းသည် variable ရဲ့ memory address ကို pass လုပ်မည်ဟု ဆိုလိုခြင်းဖြစ်ပါသည်။ အခေါ်ခံရသော function parameter နေရာတွင်လည်း pointer variable တစ်လုံးကို ကြော်ပြာပေးထားရန် လိုအပ်ပါသည်။ ထို pointer နေရာသို့ variable ရဲ့ memory address ဝင်လာမည်ဖြစ်ပါသည်။

Sample Program Using Pointer

```
#include <stdio.h>
void function2(int *param) {
    printf("I've received value %d\n", *param);
    (*param)++;
}

int main(void) {
    int variable = 10;

    function2(&variable);
    printf("pass by value in c++ :%d\n", variable);
    return 0;
}
```

C Structure

Structure ဆိုတာ user-defined datatype တစ်မျိုးဖြစ်ပြီး မတူညီတဲ့ data တွေကို စုစည်ထားနိုင်ပါတယ်။ တစ်နည်းအားဖြင့် primary and derived data type ကို ဆိုလိုခြင်းဖြစ်ပါတယ်။ Structure သည် array နှင့် ဆင်တူသည်ဟု ထင်နိုင်သော်လည်း array တစ်ခုတွင် တူညီသော data များသာစုစည်းထားလို့ရပါတယ်။ သို့သော် structure တွင်မူ မိမိတို့ ကြိုက်နှစ်သိုက်သော မတူညီသည့် data များကို စုစည်းထားနိုင်ပါတယ်။ ထို့ကြောင့် structure ကို data တွေ များစွာ stored လုပ်လိုသော

အခါတွင် သုံးနိုင်သည်။ ဥပမာ- ကျောင်းသား အယောက် ၁၀၀ ရဲ့ information အားလုံးကို structure ကိုသုံးပြီး code အနည်းငယ်ရေးရုံဖြင့် သို့လျှောင်နိုင်ပါသည်။ sample program ကို နောက်ပိုင်းတွင် ဖော်ပြသွားပါမည်။

```
struct info{
    char *name;
    int age;
};
```

အထက်တွင်ရေးသားထားသော ပုံစံသည် structure တစ်ခုကို စတင်ကြော်ပြာသော ပုံဖြစ်သည်။ structure တစ်ခုကို **ကြော်ပြာရာ**တွင် keyword အနေဖြင့် struct ကို သုံးပါသည်။ အထဲတွင် **ကြော်ပြာထားသော** **char *name;** နှင့် **int age;** တို့သည် structure members များဖြစ်ကြသည်။ structure တစ်ခုကို **ကြော်ပြာလိုက်တိုင်း** အဆုံး သတ်တွင် semicolon ထည့်ပေးရမည်။

Self-Referential Structures

Structure တစ်ခုရဲ့ members data name တွေကို structure ရဲ့ name နဲ့ တူအောင် ပေးလို့ မရပါဘူး။ သို့သော် ပုံမှန် variable များ မရသော်လည်း pointer variable ကတော့ ပေးလိုရပါတယ်။ အောက်တွင်ဖော်ပြထားသော code ၏ **struct data work_name;** နေရာတွင် error တက်မည့်ဖြစ်ပါသည်။

```
struct data {
    char name;
    char father_name;
    unsigned int age;
    char gender;
    double payment;
struct data work_name; // ERROR
struct data *teamLeaderPtr; // pointer
};
```

Declaring Structure Variables

structure variable ဆိုတာ မိမိတို့ **ကြော်ပြာလိုက်**သော variable ကို data ထည့်လိုသောအခါ သို့မဟုတ် သုံးလိုသောအခါမှာ ခေါ်သုံးသော Name တစ်ခု ဖြစ်ပါတယ်။

Sample program 1 : Structure Variable

```
#include<stdio.h>
Struct info
{
    char name[25];
    int age;
    char branch[10];
    char gender;
};

int main(){
Struct info sInfo;
sInfo.age=20;

printf("accessing Structure %d",sInfo.age);
return 0;
}
```

အထက်ပါ program တွင် info ဆိုသည့် structure တစ်ခုကို တည်ဆောက်လိုက်ပြီး ထို structure ထဲတွင် member လေးခု ထည့်ထားပါသည်။ ထို structure အား main function မှ access လုပ်လိုသော အခါတွင် ထို structure ရဲ့ variable ကို ကြော်ပြောပေးရပါသည်။ `Struct info sInfo;` ဆိုသည့် code line သည် `sInfo` ဆိုသည့် structure variable ကို ကြော်ပြောပေးခြင်းဖြစ်ပါသည်။ ထို `sInfo` ဆိုသည့် structure variable မှ တစ်ဆင့် `info` ဆိုသည့် structure ထဲမှ structure members များကို access လုပ်လိုရမည် ဖြစ်သည်။ `sInfo.age=20;` ဆိုသည့် code line သည် `sInfo` ဆိုသည့် structure ထဲမှ `age` ဆိုသည့် member ထဲသို့ 20 ဆိုသည့် value ကို လုမ်းထည့်ခြင်းဖြစ်ပါသည်။ `printf` code line တွင် `sInfo.age` ဟု ရေးထားခြင်းမှာ ထို code line အထက်တွင် `age` ထဲသို့ ထည့်လိုက်သော data ကို ပြန်ထုတ်ခြင်း ဖြစ်ပါသည်။ Structure variable ကို အောက်ပါနည်းအတိုင်းလည်း ကြော်ပြနိုင်သည်။

Sample Program 2 : Structure Variable

```
#include<stdio.h>
Struct info
{
    char name[25];
    int age;
}sInfo;
```

```
int main(){
    sInfo.age=20;

    printf("accessing structure %d",sInfo.age);
    return 0;
}
```

သတိပြုရန် အချက်မှာ structure ကို ကြော်လွှာရာတွင် initialization လုပ်လို့ မရပါဘူး။ declaration ပဲလုပ်လို့ ရပါတယ်။ ဆိုလိုသည်မှာ structure members များထဲသို့ data များကို ကြော်လွှာတဲ့ အချိန်မှာ ထည့်လို့ မရပါဘူး။

Structure Initialization

Structure များကို initialization (data ထည့်ခြင်း) လုပ်ရာတွင် မိမိနှစ်သည့်ပုံစံဖြင့် ထည့်နိုင်ပါ တယ်။

```
#include<stdio.h>
struct Student{
    int age;
    int roll;
    float marks;
};

struct Student stu={18,20,100.000}//check here
int main(){

    printf(" printing data from Structure %d\n",stu.age);
    printf(" printing data from Structure %d\n",stu.roll);
    printf(" printing data from Structure %f\n",stu.marks);

    return 0;
}
```

Array as a Structure Variable

Structure variable ကြော်လွှာရာတွင် array ကိုလည်း သုံးနိုင်ပါသည်။

```
#include<stdio.h>
#include <conio.h>
#define STUDENTS 50
struct Student{
```

```

char name[20];
int roll;
float marks;

}stu[STUDENTS];
int main(){
int i=0;
for(i=0; i<STUDENTS; i++){
    stu[i].roll=i+1;
    printf("*****\n");
    printf("please enter for roll number %d\n",stu[i].roll);
    printf("please enter Student name:");
    scanf("%s",stu[i].name);
    printf("please enter Student marks:");
    scanf("%f",&stu[i].marks);
}

for(i=0; i<STUDENTS; i++){
    printf("_____\n");
    printf("Roll Number : %d\n",i+1);
    printf("Name : %s\n",stu[i].name);
    printf("Marks is : %f\n",stu[i].marks);
}
return 0;
}

```

အထက်ပါ program တွင် array အား structure variable အနေဖြင့် သုံးထားပြီး students 50 ခဲ့ data များကို ထည့်စွဲရန် ရေးသားထားခိုင်းဖြစ်ပါသည်။

#define STUDENTS 50
students အယောက် 50 data များကို သိလောင်နိုင်ရန် symbolic constant ကိုသုံးပြီး ကြော်ကြော်ထားခိုင်းဖြစ်ပါသည်။
for(i=0; i<STUDENTS; i++)
students 50 ခဲ့တစ် ယောက်ခိုင်းစီ data များကို ရယူနိုင်ရန် looping ပတ်ခိုင်းဖြစ်ပါသည်။
stu[i].roll=i+1; // stu[0].roll=1;

Stu array ရဲ့ index i roll နေရာတွင် i+1 ဆိုပြီး 1 ကိုပါဝင်ထည့်ပါသည်။ ထိုသို့ရေးရခြင်းမှာ looping တစ်ကြိမ်ပတ်တိုင်း roll number 1 တိုးလိုသောကြောင့်ဖြစ်သည်။ အထက်ပါ code line သည် students ဆိုသည့် structure ထဲမှ roll ဆိုသည့် member varialbe ထဲသို့ i+1 ကို ထည့်ခြင်းဖြစ်ပါသည်။

```
printf("please enter Student name:");
scanf("%s",stu[i].name);
printf("please enter Student marks:");
scanf("%f",&stu[i].marks);
```

ပထမဆုံး အကြိမ် looping ပတ်သောအခါ့် user ထည့်ပေးလိုက်သော name ကို roll number 1 က ဘယ်သူဆိုတာကို stu[0].name= user input name ထည့်ပေးထားမည် ဟုရေးထားခြင်း ဖြစ်သည်။ &stu[i].marks သည်လည်း ထိုနည်းအတိုင်းပင် ဖြစ်သည်။

```
for(i=0; i<STUDENTS; i++){
    printf(" _____ \n");
    printf("Roll Number : %d\n",i+1);
    printf("Name : %s\n",stu[i].name);
    printf("Marks is : %f\n",stu[i].marks);
}
```

အစဉ်လိုက် ထည့်လိုက်သည့် data များ ကို for loop ပတ်ပြီး အစဉ်လိုက် ပြန်ထုတ်ရန် ရေးသားထားခြင်း ဖြစ်သည်။

```

struct student{
    char name[20];
    int roll;
    float marks;
}stu[STUDENTS];
int main(){
    int i=0;
    for(i=0; i<STUDENTS; i++){
        stu[i].roll=i+1;
        printf("*****\n");
        printf("please enter for roll number %d\n", i+1);
        printf("please enter student name:");
        scanf("%s", stu[i].name);

        printf("please enter student marks:");
        scanf("%f", &stu[i].marks);
    }

    for(i=0; i<STUDENTS; i++){
        printf("_____\n");
        printf("Roll Number : %d\n", i+1);
        printf("Name : %s\n", stu[i].name);
        printf("Marks is : %f\n", stu[i].marks);
    }
    return 0;
}

```

Nested Structure

c programming မှာ nested structure ကို ရေးသားအသုံးပြန်ပါတယ်။ Structure တစ်ခုရဲ့ structure's member အနေဖြင့် နောက်ထပ် structure ထပ်ထည့်ထားခြင်းကို nested structure လိုခေါပါတယ်။ အောက်ပါ program ထဲတွင် Student ဆိုသည့် structure ထဲ၌ birth ဆိုသည့် structure တစ်ခုကို ထပ်မံတည်ဆောက်ထားပါသည်။

```

#include<stdio.h>
Struct Student
{
    int rollNumber;
    int age;
    Struct birth
    {
        int year;
        int rollNumber;
        }addr;
}stu;

```

```

int main(){
    Stu.addr.rollNumber=10;
    Stu.addr.year=1994;
    Stu.age=25;
    printf(" printing data from Structure %d\n",Stu.addr.rollNumber);
    printf(" printing data from Structure %d\n",Stu.age);
    printf(" printing data from Structure %d\n",Stu.addr.year);
    return 0;
}

```

structure တစ်ခုထဲမှ structure ကို access လုပ်လိုသောအခါတွင် ပထမ structure variable name ကို ခေါ် ပြီးမှ ဒုတိယ structure variable name ကို ခေါ် ပြီး လုပ်ရသည်။

```
Stu.addr.rollNumber=10;
```

အထက်ပါ code line သည် ငါး ဆိုသည့် Student ဆိုသည့် structure ရဲ့ name ဖြစ်ပြီး addr ဆိုသည့် Student structure ထဲမှ birth ဆိုသည့် structure ရဲ့ name ဖြစ်ပါသည်။

Pointer to a Structure

Pointer အကြောင်းကို အရင်းသင်ခန်းစာတွေမှာ လေ့လာထားပြီးသကဲ့သို့ pointer သည် variable တစ်ခုဖြစ်သလို ထို pointer သည် အခြားသော variable ရဲ့ memory address ကို ကိုင်ထားခြင်း ဖြစ်သည်။ ထိုနည်းတူပဲ pointer သည်လည်း structure variable ကို ကိုင်ထားနိုင်ပါတယ်။ structure pointer ကို ကြော်ဖြေရာမှာ struct keyword ဆိုသည့် keyword ကို structure variable ရွှေ့တွင် ရေးပေးရပါမည်။

```

#include<stdio.h>
struct bird
{
    char name[10];
    char eat[10];
    int age;
    char color[10];
};
int main(){
    struct bird fly;

    struct bird *ptr_bird;

```

```

ptr_bird=&fly;
return 0;
}

```

Program ရှင်းလင်းချက်

`struct bird fly;`

အထက်ပါ code line သည် structure ကို သုံးနိုင်ရန် ကြော်ကြထားခြင်းဖြစ်ပါသည်။

`struct bird *ptr_bird;`

အထက်ပါ code line သည် structure pointer တစ်ခုကို ကြော်ကြခြင်းဖြစ်သည်။

`ptr_bird=&fly;`

အထက်ပါ code line သည် fly ဆိုသည့် structure variable ရဲ့ memory address ကို ptr_bird ဆိုသည့် pointer ထဲသို့ ထည့်ခြင်းဖြစ်သည်။ ထို့ကြောင့် ptr_bird ဆိုသည့် pointer သည် fly ဆိုသည့် bird structure ကို access လုပ်နိုင်သွားပါပြီ။

Accessing Structure's member using Pointer

Structure's members တွေကို access လုပ်ရာမှာ နည်းလမ်း နှစ်ခု ရှိပါတယ်။

indirection (*) operator နှင့် dot (.) operator ကိုသုံးပြီး access လုပ်နိုင်သလို

arrow (->) operator or membership operator ကိုသုံးပြီးလည်း access လုပ်နိုင်ပါတယ်။

Using Indirection (*) Operator and Dot (.) Operator

`ptr_bird=&fly;`

အထက်ပါ code တွင် ဖော်ပြထားသည့် အတိုင်း ptr_bird သည် fly ရဲ့ memory address ကို ကိုင်ထားပြီး ဖြစ်သည်။ ထို့ကြောင့် fly structure ရဲ့ members များ အောက်ပါ အတိုင်း indirection and dot operator တို့ကို သုံးပြီး access လုပ်နိုင်သည်။

`(*ptr_bird).name;`

အထက်တွင် ရေးထားသော code သည် fly structure ထဲမှ name ဆိုသည့် member ကို access လုပ်ခြင်း ဖြစ်သည်။ fly structure ထဲမှ age ကို access လုပ်လိုလျှင်လည်း အောက်ပါအတိုင်း လုပ်နိုင်သည်။

`(*ptr_bird).age;`

`ptr_bird` ဆိုသည့် pointer ကို () parentheses ထဲမှာ ထည့်ပေးဖို့ လိုအပ်ပါတယ်။ အဘယ်ကြောင့်ဆိုသော် . dot operator သည့် indirection operator (*) ထက် precedence မြင့် သော ကြောင့်ဖြစ်သည်။

Using Arrow Operator (->)

Pointer ကိုအသုံးပြုပြီး structure members များကို access လုပ်သည့် နည်းလမ်းဟာ အနည်းငယ် ရှုပ်ထွေးမှ ရှိသလို ဖတ်ရတာလည်း ခက်ခဲစေနိုင်ပါတယ်။ C မှာ arrow operator ကိုသုံးပြီး access လုပ်နိုင်ပါသေးတယ်။ pointer variable နောက်တွင် arrow operator ကိုရေးပြီး arrow operator နောက်မှာမှ structure ရဲ့ member name ကို ရေးခြင်းဖြင့် အသုံးပြုနိုင်ပါတယ်။

pointer variable -> structure's member

```
ptr_bird -> name;
ptr_bird -> age;
```

အထက်ပါ နည်းလမ်း အတိုင်း သုံးမည့်ဆိုလျှင် parentheses , asterisk(*) and dot (.) operator တို့ကို သုံး စရာ မလိုတော့သလို ပိုမိုပြီး ဖတ်ရလွယ်ကူ စေပါတယ်။

Sample Program Using Arrow Operator(->)

```
#include<stdio.h>
struct dog
{
    char name[10];
    char type[10];
    int age;
    char color[10];
};

int main()
{
    struct dog my_dog = {"KyarBo", "WarDog", 5, "Yellow"};
    struct dog *ptr_dog;
    ptr_dog = &my_dog;

    printf("MyDog's name: %s\n", ptr_dog->name);
    printf("MyDog's type: %s\n", ptr_dog->type);
    printf("MyDog's age: %d\n", ptr_dog->age);
```

```

printf("MyDog's color: %s\n", ptr_dog->color);

strcpy(ptr_dog->name, "SoatPhwar");
ptr_dog->age++;

printf("After Change ! \n*****\n");
printf("MyDog's new name is: %s\n", ptr_dog->name);
printf("MyDog's age is: %d\n", ptr_dog->age);

return 0;
}

```

Program ရှင်းလင်းချက်

`struct dog my_dog = {"KyarBo", "WarDog", 5, "Yellow"};`

dog ဆိုသည့် structure ထဲသို့ array ပုံစံဖြင့် data များ တည့်ခြင်းဖြစ်သည်။

```

printf("MyDog's           name:      %s\n",      ptr_dog->name);
printf("MyDog's           type:      %s\n",      ptr_dog->type);
printf("MyDog's          age:      %d\n",      ptr_dog->age);
printf("MyDog's color: %s\n", ptr_dog->color);

Structure ထဲမှ members များကို ထုတ်ပြခြင်း access လုပ်ခြင်းဖြစ်သည်။

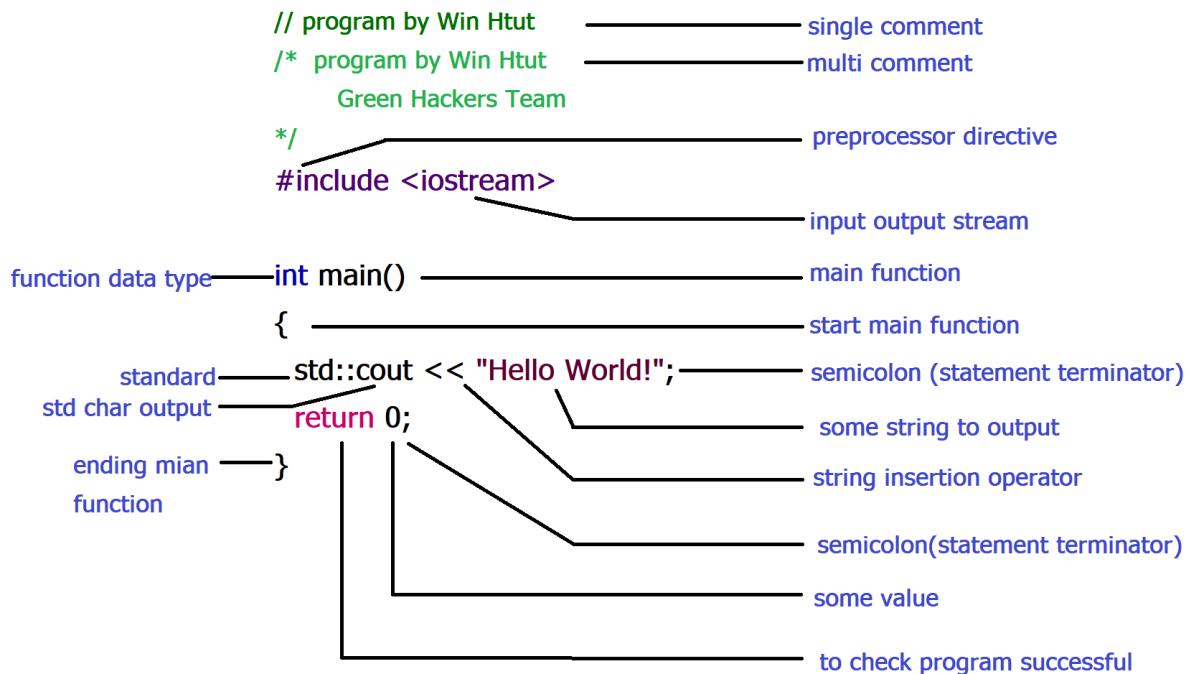
```

`strcpy(ptr_dog->name, "SoatPhwar");`
`strcpy` function ကိုသုံးပြီး name ထဲသို့ နောက်ထပ် နာမည် အသစ်တစ်ခု ထပ်ထည့်ခြင်းဖြစ်သည်။

`ptr_dog->age++;`
`++` ဆိုသည့် increment operator ကိုသုံးပြီး age ထဲမှ data ကို 1 ပေါင်းခြင်းဖြစ်သည်။

Starting C++ Programming

Example C++ Program



WIN HTUT (GREEN HACKERS TEAM)

```

// my first c++ program by Win Htut
#include <iostream>
int main()
{
  std::cout << "Hello World!";
  return 0;
}

```

ယခု program ကို run ကြည့်လိုက်မယ်ဆိုရင် output အနေနဲ့ Hello World ဆိုတဲ့ စာကြောင်း
တစ်ကြောင်းကို computer screen မှာ မြင်ရမှာဖြစ်ပါတယ်။ ယခု program လေးဟာ အရမ်းရှိုးရှင်း
လွယ်ကူသောလည်း c++ program တစ်ပုဒ်မှာ ပါသင့်ပါထိုက်တဲ့ အခြေခံ components တွေ
အားလုံးပါဝင်ပါတယ်။

Line 1: // my first c++ program by Win Htut ပထမဆုံး line မှာ မြင်တွေရတဲ့ မျဉ်းစောင်းလေး နှစ်ကြာင်းဖြစ်တဲ့ (slash signs) ကို အသုံးပြုရခြင်းက မိမိ တို့ program မှာ မိမိမှတ်သားထားလို့သော မှတ်ချက်လေးတွေ (comments) ရေးသားတဲ့ နေရာမှာ အသုံးပြုပါတယ်။ // (slash signs) နောက်က စာင်းတွေက program အတွက် ဘယ်လို အကျိုးသက်ရောက်မှုမှ ရှိမှာ မဟုတ်ပါဘူး။

Line 2: #include <iostream>

#include ဆိုတာ preprocessor Directive ကို ဆိုလိုခြင်းဖြစ်ပါတယ်။ Preprocessor Directive က # နဲ့ စပါတယ်။ program တစ်ပုဒ်ကို compile မလုပ်ခင်မှာ process (ဖြစ်စဉ်) တစ်ခု ရှိပါသေးတယ်။ အဲဒါကို preprocessing လုပ်တယ်လို့ ခေါ်ပါတယ်။ Preprocessing လုပ်ခြင်းဆိုတာ C++ program တစ်ပုဒ်မှာဆိုလျှင် <iostream> ဆိုသည် input/output stream header ကို program မှာ #include ပါဝင်မယ်လို့ ဆိုလိုခြင်းဖြစ်ပါတယ်။ထို process ပြီးမှုသာလျှင် compiler က မိမိရေးသားထားတဲ့ code တွေကို compile လုပ်ပါတယ်။

#include<iostream> iostream ဆိုတဲ့ header file ထဲက functions တွေ classes တွေကို ယခု program ထဲမှာ ထည့်သုံးမယ်လို့ ဆိုလိုခြင်းဖြစ်ပါတယ်။ i= input (ifstream program ထဲသို့ data များ ထည့်ခြင်းကို input ဟု ခေါ်ပါသည်)။ eg: cin (standard input stream) o=output (ostream program မှ data များကို စကားလုံး၊ ရုပ်၊ အသုံးများ ပေါ်ထွက်လာခြင်းကို ဆိုလိုသည်)။ eg: cout,cerr,clog (standard output stream)။ ဥပမာ အနေနဲ့ “Hello World” ဆိုတဲ့ စာင်းတွေကြာင်း computer screen မှာ ပေါ်လာခြင်းသည် Standard Output Stream ဖြစ်ပါသည်။ std သည် namespace တစ်ခုဖြစ်ပြီး Namespaces များသည် C++ နှုပါဝင်သော advanced feature များ ဖြစ်တဲ့အတွက် နောက်ပိုင်း သင်ခန်းစာတွေမှာ ထပ်မံ ဆွေးနွေးသွားပါ၍ မယ်။ စတင်လေ့လာသူများ အနေနဲ့ cin, cout, cerr, clog များရှေ့တွင် std အသုံးပြုရမည်ကို သိရှိထားရပါမည်။

Line 3: int main() ဆိုတာ function တစ်ခုကို စတင်ကြော်ဖြေခြင်းဖြစ်ပါတယ်။ function ဆိုတာက မိမိ အလုပ်လုပ်လိုသော instructions တစ်ခု သို့မဟုတ် တစ်ခုထက်ပိုပါတဲ့ code statements တွေကို စုဝေးထားခြင်းဖြစ်ပါတယ်။ Function နဲ့သက်ဆိုင်တဲ့ အချက်အလက်တွေ သင်ခန်းစာတွေကို နောက်ပိုင်း သင်ခန်းစာမှာ အသေးစိတ်ထပ်မံ ဖော်ပြသွားမှာပါ။ C/C++ program တစ်ပုဒ်မှာဆိုရင်

function တစ်ခု မပါမဖြစ် ပါရပါတယ်။ ထိုမပါမဖြစ်ပါရမဲ့ function သည် main ဆိုတဲ့ function ဖြစ်ပါတယ်။ အဘယ်ကြောင့်ဆိုသော C/C++ program တစ်ပုဒ်သည် ထို main function ထဲမှာပဲ အလုပ်လုပ်ပါတယ်။ အခြားသော function တွေကိုလည်း ထို main function ထဲကနေပဲ လုမ်းခေါယူပြီး အသုံးပြုပါတယ်။ ထို main ဆိုသည့် နာမည်ကို အခြားသော နာမည်များနှင့် အတားထိုး သုံးဆွဲခြင်း ပြုလုပ်လို့မရပါဘူး။ main ဆိုတဲ့ function ရဲ့ ဘယ်ဘက် ဘေးမှာ ရှိတဲ့ int သည် ထို main ဆိုတဲ့ function ရဲ့ data type ဖြစ်ပါတယ်။ int သည် integer data type ကိုဆိုလိုခြင်းဖြစ်ပါသည်။ integer ဆိုတာ (hole number) ကိုဆိုလိုခြင်း ဖြစ်ပါတယ်။ int main () ရဲ့ဆိုလိုရင်းမှာ ယခု C++ program တွင် ပါဝင်သော main function သည် int integer အမျိုးအတား ဖြစ်သောကြောင့် return အနေနှင့် integer value အဖြစ်သာ ရ ရှိမည်ဖြစ်ပါသည်။ မည့်သည့် programming language မှ မလေ့လာဖူးသေးသော beginner များအနေနှင့် function တွေ data type တွေ အကြောင်းကို စိမ်းနေမှာပါ။ ထိုကြောင့် နောက်ပိုင်း lesson တွေမှာ အသေးစိတ် ထပ်မံဖော်ပြသွားမှာပါ။ ယခု ဖော်ပြချက်များသည် introduction များသာ ဖြစ်ပါသည်။

Line 4: { (brace)Function တစ်ခုမှာ သူနဲ့ သက်ဆိုင်တဲ့ စတဲ့နေရာနဲ့ ဆုံးတဲ့နေရာ ဆိုပြီး ရှိပါတယ်။ အစ နဲ့ အဆုံးကို braces တွေနဲ့ သတ်မှတ်ဖော်ပြထားပါပြီ။ code block လိုလည်း ခေါ်ပါတယ်။ အထက်ပါ program မှာဆိုရင် Line 4 “{” (brace)သည် main function အစ ဖြစ်ပြီး Line 7 တွင် ရှိသော “}” သည် main function ရဲ့ အဆုံးဖြစ်ပါသည်။ ထို အစ { နှင့် အဆုံးမှာ } အတွင်းမှာ ပါဝင်သော codes များ သည်သာလျှင် main function နှင့် သက်ဆိုင်ပါသည်။ ထိုကြောင့် main function ထဲတွင် ပါဝင်လိုသော codes(statement) များကို ထိုတွေနဲ့ကွင်းအစနှင့် အဆုံးအတွင်းတွင်သာ ရေးရမည် ဖြစ်သည်။ function အားလုံးသည် ထိုနည်းအတိုင်းသာ ဖြစ်သည်။ Local Variable နှင့် Global Variable သင်ခန်းစာတွင် ထို { } (braces) များအကြောင်းကို ထပ်မံဖော်ပြသွားပါ။ မည်။

Line 5: std::cout << "Hello World!";

std::cout << "Hello World!"; ယခု code line ကိုတော့ statement လို့ခေါ်ပါတယ်။ ဘယ်လို နေရာမှာ အသုံးပြုလဲဆိုရင် မိမိတို့ program ကနေ စာကြောင်းတစ်ကြောင်း ဖော်ပြလိုသောအခါမျိုးမှာ သုံးပါတယ်။ အထက်ပါ statement မှာဆိုရင် အပိုင်း ၃ ပိုင်း ရှိပါတယ်။

No.1 သည် `std::cout` ဖြစ်သည်။ Standard Character Output လို့ ဆိုလိုခြင်းဖြစ်ပါသည်။

No.2 သည် `<< string insertion operator` ဖြစ်သည်။ `<< (string insertion operator)` ရဲ့ ညာဘက်မှာ ရှိတဲ့ operand ဖြစ်တဲ့ double quotes ထဲက "Hello world!" ဆိုတဲ့ စာကြောင်းကို `std::cout` standard output အဖြစ် computer screen မှာဖော်ပြလိုသောအခါမျိုးမှာ သုံးပါသည်။ `std::cout` ရဲ့ ညာဘက်မှာ ရှိတဲ့ double quotes ထဲက စာသားအားလုံးကို computer screen မှာ ဖော်ပြမှာ ဖြစ်ပါတယ်။ သို့သော် "Hello I am \n Win Htut" \n ဆိုတဲ့ စာလုံးကိုတော့ computer screen မှာ ဖော်ပြမှာ မဟုတ်ပါဘူး။ အဘယ်ကြောင့်ဆိုသော် \ (back slash) သည် escape character ဖြစ်ပြီး \n သည် newline ကို ကိုယ်စားပြုပါသည်။ NO.3 semicolon (;) သူကတော့ အမှားအများဆုံးနေရာ ဖြစ်ပါတယ်။ C/C++ program တစ်ပုဒ်မှာဆိုရင် statement တွေ အားလုံးရဲ့ အဆုံးမှာ semicolon(;) ထည့်ပေးရပါတယ်။ သူကို statement terminator လို့လည်းခေါ်ပါတယ်။

Line 6: return 0;

Return statement ကို program အောင်မြင်စွာ ပြီးဆုံးသွားသည်ဟု ဖော်ပြလိုတဲ့နေရာမှာ အသုံးပြုပါတယ်။ return statement ကို main function ရဲ့ အဆုံးမှာသာ အသုံးပြုပါတယ်။ return နေရာက်မှာ 0 (zero) ကို အသုံးပြုနိုင်သလို မိမိကြိုက်နှစ်သက်တဲ့ number ကိုလည်း အသုံးပြုနိုင်ပါတယ်။ program အောင်မြင်စွာ ပြီးဆုံးသွားတယ်ဆိုရင် မိမိ return နေရာက်မှာထည့်ပေးလိုက်တဲ့ value ရဲ့ hexadecimal value ကိုပြန်ပေးမှာဖြစ်ပါတယ်။ return နေရာက်မှာ zero (0) ထည့်ပေးထားပြီး program run သွားပြီးတဲ့နေရာက်မှာ zero ရဲ့ hexadecimal value ဖြစ်တဲ့ zero (0) ကို ပြန်မပေးဘူးဆုံးရင် မိမိ program အောင်မြင်စွာ အဆုံးသတ်သွားတယ်လို့ မဆိုလိုနိုင်ပါဘူး။ return 10; လို့ ရေးကာ program run ကြည့်မယ်ဆုံးရင် 10 ရဲ့ hexadecimal value ဖြစ်တဲ့ "A" ကို ပြန်ပေးတယ်ဆုံးရင် မိမိ program အောင်မြင်စွာ အလုပ်လုပ်သွားကြောင်း သိနိုင်ပါတယ်။

Good Programming Practice 1

Program တိုင်းမှာ comment များ ရေးသားထားသင့်ပြီး ထို comment မှာ program ရဲ့ ကြောင်းအရာတွေ နဲ့ ရည်ရွယ်ချက်တွေကို ဖော်ပြထားသင့်ပါတယ်။

Variables

What is Variable?

1 နဲ့ 2 ပေါင်းပြီး 3 ရတယ်ဆိုတဲ့ equation တစ်ခုကို program အနေနဲ့ ဘယ်လိုပေးရမလဲဆိုတာ
ပထမဦးဆုံး အနေနဲ့ စဉ်းစားကြည့်ရအောင်။

$1+2 = 3$

1 ကို ကိုယ်စားပြုတဲ့ နာမည်လေးတွေကို စိတ်ထဲကနေ မိမိကြိုက်သလို ပေးကြည့်ရအောင်။ ဥပမာ 1 ကို
 x ဟုလည်းကောင်း၊ 2 ကို y ဟုလည်းကောင်း၊ 3 ကို z ဟုလည်းကောင်း။

$1+2 = 3$

$x+y = z$ (ထိုကဲ့သို့ အစားထိုး ပေးလိုက်သော x, y, z တို့ကို variable ဟု ခေါ်ဆိုခြင်းဖြစ်ပါသည်။)

Variable Declarations and Initialization

(variable တွေကို ကြော်ပြေခြင်းနှင့် တန်ဖိုးများ စတင် သတ်မှတ်ပေးခြင်း)

C++ programming language ဟာ strongly-typed language ဖြစ်တဲ့အတွက် variable တွေကို
ကြော်ပြေတဲ့နေရာမှာ သူရဲ့ data type တွေကို ထည့်ပေးရပါတယ်။ variable တစ်ကို ကြော်ပြေဆိုရင်
အခြေခံ အချက်သုံးချက်ကို သိရပါတယ်။ variable ရဲ့ data type , variable ရဲ့ name, variable ရဲ့ value
တို့ဖြစ်ပါတယ်။ variable ရဲ့ name ကိုတော့ အထက်မှာဖော်ပြုခဲ့ပြီး ဖြစ်ပါတယ်။

Syntax: int x{ 0 };

int သည် variable data type ဖြစ်ပါသည်။ x သည် variable name ပဲဖြစ်ပြီး 0 သည် variable ရဲ့ value
ဖြစ်ပါသည်။ int သည် integer ကို ကိုယ်စားပြုထားခြင်းဖြစ်ပြီး number များ
သို့လောင်သိမ်းဆည်းရာတွင် အသုံးပြုသည်။ ဆိုလိုတာက x ဆိုတဲ့ variable ထဲမှာ int ဆိုတဲ့ number
တွေပဲ သို့လောင်သိမ်းဆည်းမယ်လို့ ဆိုလိုခြင်းဖြစ်ပါတယ်။ Data types တွေအကြောင်းကို
နောက်သင်ခန်းစာများ အသေးစိတ် ဆွေးနွေးပါမယ်။

ကြော်ပြာနည်း (declaration) လုပ်တာ သုံးနည်းရှိပါတယ်။

1. int x=0;

2. int x(0);

3. int x{0};

ပထမ နည်းကတော့ C programming မှာ ကြော်ပြာတဲ့နည်းနဲ့ ဆင်တူပါတယ်။ သာမန်
ကြော်ပြာတဲ့နည်းပါ။

ဒုတိယ နည်းကတော့ constructor initialization လို့ ခေါ်ပါတယ်။ C++ language ကျမှုသာ စတင်
အသုံးပြုတာပါ။

တတိယ နည်းကတေသာ uniform initialization လို့ ခေါ်ပါတယ်။ C++ standard 2011 မှာ စတင် အသုံးပြုခဲ့တာပါ။

Very Important: C++ programming language ဟာ case sensitive language ဖြစ်တယ်။ ဆိုလိုချင် တာက programmer က ရေးလိုက်တဲ့ variable name ဟာ keywords တွေနဲ့ တူလိုမရပါဘူး။ ဥပမာ X သည် x နဲ့ မတူပါဘူး။ name လည်းကွာသလို type တွေရော value တွေရော ကဲဖြားနိုင်ပါတယ်။ Variable Name တွေကို ကြော်ပြာတဲ့အခါမှာ အောက်မှာပါတဲ့ keywords တွေနဲ့တူလို့ မရပါဘူး။

Good Programming Practice 2

Variable name တွေကို ကြော်ပြာတဲ့နေရာမှာ meaningful ဖြစ်တဲ့ name တွေကို ပေးခြင်းအားဖြင့် program ကို error တက်ခြင်းမှ ကာကွယ်နိုင်ပါတယ်။ ဥပမာ အရေအတွက် ကို ၁၀ လို့ အတိုကောက် မပေးပဲ count ဆိုပြီး ရေးတာမျိုးပါ။ Variable name တွေ ကြော်ပြာတဲ့နေရာမှာ _ (underscore) တွေ digit တွေနဲ့တဲ့ကပ်ပြီး ပေးလို့ရပါတယ်။ ဥပမာ _count1 ပေါ့။ ဘာကြောင့်လည်းဆိုတော့ intelligent ဖြစ်တဲ့ Integrated Development Environment (IDE) တွေမှာဆိုရင် _ (underscore) ကို ရေးလိုက်ရုံးနဲ့ မိမိအသုံးပြုထားတဲ့ variable တွေ ပေါ်လာနိုင်လို့ပါ။ ဒီနည်းလမ်းဟာ program ရေးတာကို လျင်မြန်စေပြီး အမှားကင်းစေပါတယ်။ သို့မဟုတ် မိမိ အဆင်ပြုသလို မှတ်သားပြီး ရေးသားခြင်းကတေသာ အမြန်ဆုံးနဲ့ အကောင်းဆုံး ဖြစ်စေမှာပါ။

Good Programming Practice 3

Variable name တစ်ခု ကြော်ပြာလိုက်တိုင်း ထို variable ရဲ့ဘေးမှာ ဘာအတွက်ကြောင့် ထို variable ကို ကြော်ညာသလဲဆိုတဲ့ comment ရေးထားမယ်ဆိုရင် program ကို ဖတ်ရတာ ပိုပြီး လွယ်ကူစေသလို အခြားသူများလည်း နားလည်နိုင်မှာဖြစ်ပါတယ်။

C++ Programming မှာ fundamental types အနေဖြင့် 3 မျိုး ရှိပါတယ်။

1. int ကတေသူ integer number တွေကို သိလောင်ဖို့ အတွက်ပါ။
2. char ကတေသူ single lowercase letter တွေကို သိလောင်ဖို့ အတွက်ပါ။ special character တွေဖြစ်တဲ့ \$ or * တို့ကိုလည်း သိလောင်သိမ်းဆည်းပါတယ်။
3. double ကတေသူ ဒသမကိန်းတွေကို သိလောင်ဖို့ အတွက်ပါ။ 3.4 , 0.0 , -11.19

`std:: cin` (Standard input ကို program မှာစတင် အသုံးပြုသွားပါမယ်)

```
/*
C ++ program by Win Htut Green Hackers Team
*/
1. #include<iostream>
2. using namespace std;
3. int main(){
4.     int now=0; // for now time
5.     int birth=0;// for birthday
6.     int age=0;// to get age
7.     cout << "Please enter time1(now) :";
8.     cin >> now;
9.     cout << "Please enter your birth :";
10.    cin >> birth;
11.    age = now - birth;
12.    cout << "This is your age :"<<age;
13. }
```

Program ရှင်းလင်းချက်

```
/*
C ++ program by Win Htut Green Hackers Team
*/
```

အထက်ပါ တွင်ရေးသားထားသော စာများသည် comments သာဖြစ်ပါတယ်။ program ကို နောက်တစ်ကြိမ် ပြန်ကြည့်လျှင် သော်လည်းကောင်း၊ တစ်ခြားသူတစ်ယောက် လွှယ်ကူစွာ ဖတ်နိုင်စေရန် သော်လည်းကောင်း ရေးသားထားခြင်းသာ ဖြစ်ပါသည်။ program ကို compile လုပ်ရာတွင် compiler မှ ထည့်သွင်းပြီး compile လုပ်သွားမည်မဟုတ်ပါ။

1. Line number 1 #include<iostream> input/output stream အကြောင်းကို အစပိုင်းတွင် ရှင်းပြထားပြီးဖြစ်သည်။
2. using namespace std; ကို ကြော်လှပေးထားခြင်းဖြင့် program ထဲတွင် std:: ကို ထပ်ခါထပ်ခါ ရေးသားရန် မလိုတော့ပါ။ namespace အကြောင်းကို နောက်ပိုင်းသင်ခန်းစာများတွင် ထပ်မံရှင်းပြ ပေးဦးမှာ ဖြစ်ပါတယ်။
3. int main(){ သည် program ရဲ့အစ ဖြစ်သည်။ int ကတော့ နောက်က main ဆိုတဲ့ function ရဲ့ type ကို ရည်ညွှန်းထားခြင်းဖြစ်ပါသည်။ int အကြောင်းကို နောက်ထပ် data types သင်ခန်းစာမှ ဆက်လက် ဆွေးနွေးပေးပါမည်။
4. int now=0; // for now time user ဆီမှု data သို့လောင်ရန် now ဆိုသည့် variable ကို ကြော်လှထားခြင်းဖြစ်သည်။ // for now time ကတော့ comment မှတ်သားထားရုံ သက်သက်ပါ။
5. int brith = 0; // for birthday integer data type နဲ့ birth ဆိုတဲ့ variable တစ်ခုကို initial value 0 နဲ့ declare လုပ်ထားပါတယ်။ user ဆီကနေ ရှိက်ထည့်လိုက်တဲ့ data ကိုသို့လောင်ဖို့ အတွက်ပါ။ // for birthday ကတော့ comment မှတ်သားထားရုံ သက်သက်ပါ။
6. int age = 0; // to get age integer data type နဲ့ age ဆိုတဲ့ variable ကို initial value 0 နဲ့ ကြော်လှထားပါတယ်။ now , birth ထို variable နှစ်ခု calculate လုပ်လို့ရတဲ့ value ကို stored လုပ်ရန် အတွက်ဖြစ်ပါသည်။ // to get age ကတော့ ဖတ်ရလွယ်ကူစေရန် comment မှတ်သားထားခြင်းသာ ဖြစ်သည်။ ထည့်မရေးလည်း ရပါသည်။
7. cout << "Please enter time1(now) :"; user ကို data ထည့်ရန် ပြောခြင်းဖြစ်ပါသည်။
8. cin >> now; user က ထည့်ပေးလိုက်တဲ့ data ကို now ဆိုတဲ့ variable ထဲမှာ သွားရောက်ပြီး သို့လောင်ထားသည်ဟု ဆိုလိုခြင်းဖြစ်ပါသည်။
9. cout << "Please enter your birth :"; user ဆီမှု နောက်ထပ် data တစ်ခုထည့်ရန် ပြောခြင်းဖြစ်ပါသည်။
10. cin >> birth; user က ထည့်ပေးလိုက်တဲ့ data ကို birth ဆိုတဲ့ variable ထဲမှာ သွားရောက်ပြီး သို့လောင်ထားသည်ဟု ဆိုလိုခြင်းဖြစ်ပါသည်။

11. `age = now - birth;` now ထဲကနေ birth ကို နှစ်ပြီး age ဆိုတဲ့ variable ထဲမှာ (assign) သိမ်းထားမယ်လို့ ဆိုလိုခြင်းဖြစ်ပါသည်။ - သည် arithmetic operator ထဲတွင် ပါဝင်သည်။ သူ့ကို binary operator ဟူလည်း ခေါ်ပါသည်။ operand နှစ်ခုပါတဲ့ operator တွေကို binary operator ဟိုခေါ်ဆိုပါသည်။ operand နှစ်ခုဆိုသည်မှာ now and birth variable နှစ်ခုကို ဆိုလိုခြင်းဖြစ်သည်။ = သည် assignment ကိုခေါ်ပြီး operator precedence ထဲတွင် နာက်ဆုံးဖြစ်သည်။ operator precedence တွေ အကြောင်းကို နာက်ပိုင်းသင်ခန်းစာများတွင် ဆက်လက် ဆွေးနွေးပေးသွားပါ၍းမည်။

12. `cout << "This is your age :" << age;` အထက်မှာ age ထဲရှိ သို့လျှင်ထားတဲ့ value (တန်ဖိုးတစ်ခု) ကို standard output stream object အနေနဲ့ screen မှာ ပြန်လည်ဖော်ပြပေးရန် ဖြစ်သည်။

13. } main function ပြီးဆုံးကြောင်း ရေးသားထားသည့် curly bracket ဖြစ်သည်။

Good Programming Practice 4

Operator တွေကို operand နှစ်ခုကြားမှာ ရေးတဲ့အခါ space လေးတွေ ပြားပြီး ရေးခြင်းအားဖြင့် အမြင်ရှင်းစေသလို့ program ကိုလည်း (readable) ဖတ်ရ လွယ်ကူးစေပါတယ်။

Memory Concepts Of Variables

အထက်ပါ program မှာပါတဲ့ variable name တွေဖြစ်ကြတဲ့ now, birth and time ဆိုတာတွေက location (တည်နေရာ) ကို ကိုယ်စားပြုထားခြင်းဖြစ်ပါတယ်။ variable တွေရဲ့ value တွေကို computer memory ထဲမှာ သွားရောက် သိမ်းဆည်းထားပါတယ်။ ထို့ကြောင့် variable name တွေရဲ့ထို့ computer memory ထဲမှာ သိမ်းဆည်းထားတဲ့ နေရာတွေကို ကိုယ်စားပြုထားခြင်း ဖြစ်ပါတယ်။

Class and Object in C++

C++ ရဲ့ အဓိကအကြောင်းရင်းများထဲက တစ်ချက်သည် object orientation ကို c programming language ထဲမှာ ပေါင်းထည့်ရန်ဖြစ်သည်။ ထိုကြောင့် c++ သည် OOP (object oriented programming) တစ်ခု ဖြစ်လာပါသည်။ Object Oriented ကို စတင်လေ့လာနိုင်ရန် classes and objects တို့ကို လေ့လာထားရန် လိုအပ်ပါသည်။

Class သည် data များကို စုဝေးထားသည့် user defined data type ဖြစ်သည်။ data များကို စုဝေးထားသည့် ဆိုတဲ့အတိုင်း class တစ်ခုထဲတွင် data များပါဝင်ပါသည်။ class တစ်ခုထဲတွင်ပါဝင်တဲ့ data တွေဆိုတာ variable (member data) and function (member function) များကို ဆိုလိုခြင်း ဖြစ်သည်။ ဥပမာ - dog ဆိုသည့် class တစ်ခုကို တည်ဆောက်လိုက်သည် ဆိုပါစို့။ dog သည် ကိုက်တတ်သည်၊ စားတတ်သည်၊ ဟောင်တတ်သည် စသည့်အချက်တို့သည် dog ဆိုသည့် class ရဲ့ function များဖြစ်ကြပြီး dog ရဲ့ color , age , weight စသည့်တို့သည် dog class ရဲ့ member data များ ဖြစ်ကြပါသည်။ class တစ်ခုအတွင်းမှာ ရေးသားထားသော functions တွေကိုတော့ method လိုခေါ်ပြီး variables တွေကိုတော့ properties လိုခေါ်ပါတယ်။

Syntax : Creating a dog class

```
class Dog{
    int age;
    int weight;
    char color[];
    void eat(){
        cout<<"The dog is eating"<<endl;
    }
    void bike(){
        cout<<"The dog can bike you "<<endl;
    }
}
```

```

}
}
```

အထက်ပါ program သည် properties and methods များပါဝင်သည့် Dog class တစ်ခုကို ဖန်တီးခြင်းဖြစ်သည်။

Dog class ထဲမှာ ရှိသော properties and methods များကို main function ကနေ access လုပ်လိုလျှင် တိုက်ရှိက်မလုပ်နိုင်ပါဘူး။ Dog class ကို ကိုယ်စားပြုမည့် name တစ်ခု သုံးပေးရပါတယ်။

Dog bobo;

Bobo သည် dog class ကို ကိုယ်စားပြုထားသော name တစ်ခုဖြစ်သည်။ object ဆိုသည့်မှာ class ကို ကိုယ်စားပြုထားသော name ကို ဆိုလိုခြင်း ဖြစ်သည်။ ထို့ကြောင့် bobo သည် object ဖြစ်သည်။ Dog class ထဲတွင် ရှိသည့် properties and method တို့ကို bobo ဆိုသည့် object ကိုသုံးပြီး အောက်ပါအတိုင်း access လုပ်နိုင်ပါပြီ။ ထို့ကြောင့် object သည် class တစ်ခုရဲ့ ကိုယ်စား ဖြစ်သည်။ (an object is a instance of class) .

Dog bobo;

bobo.age=5;

bobo.age=5; သည် Dog class ထဲမှာ ရှိသော age ဆိုသည့် properties ကို 5 ဆိုသည့် value တစ်ခု လုမ်းထည့်ပေးလိုက်ခြင်း ဖြစ်သည်။ Class တစ်ခုကို ကိုယ်စားပြုထားသည့် objects များစွာ ရှိနိုင်သည်။ ဆိုလိုသည့်မှာ အထက်တွင် Dog class ကိုသုံးပြီး bobo ဆိုသည့် ခွေးတစ်ကောင် ဖန်တီးလိုက်သလို puppy, marny, hardy ဆိုသည့် ခွေးများကိုလည်း Dog class ကိုသုံးပြီး ဖန်တီးနိုင်ပါသေးသည်။

[Sample Program : Accessing and Creating Properties](#)

```

#include<iostream>
using namespace std;
class Dog{
public :
    string name;
    int age;
    float weight;

};
int main(){
```

```

Dog bobo;
Dog puppy;
bobo.age=5;
bobo.name="bobo";
puppy.age=3;
bobo.weight=5.5;
puppy.weight=3.2;
cout<<"My Dog name is "<<bobo.name<<endl;
cout<<"Age of bobo"<<bobo.age<<endl;
cout<<"Age of puppy"<<puppy.age<<endl;
return 0;
}

```

အထက်ပါ program မဲ့ ငြောင်းမြောက်မှာ public ဆိုတဲ့ စာငြောင်းလေး တစ်ငြောင်း ပိုလာပါတယ်။ အဲဒီ public ဆိုတာဟာ access specifiers ဖြစ်ပါတယ်။ Access Specifiers ဆိုတာ class တစ်ခုအတွင်းမှာ ရှိတဲ့ properties and method တွေကို ပြင်ပကနေ access (ရယူသုံးစွဲခွင့်) ကိုဆိုလိုခြင်း ဖြစ်ပါတယ်။ C++ မှာ Access Specifiers သုံးခုရှိပါတယ် public, protected, private တို့ဖြစ်ပါတယ်။

public : public keyword အောက်မှာ ရေးထားတဲ့ members တွေကို public members တွဲလို့ ခေါ် ပြီး ထို members တွေကို class ရဲ့ အပြင်ဘက် အခြားသော နေရာမှ object and dot operator တို့ကိုသုံးပြီး access လုပ်နိုင်ပါတယ်။

protected: protected members တွေကို အခြားသော နေရာကနေ access မလုပ်နိုင်ပါဘူး။ access လုပ်ချင်တယ်ဆိုရင် မူရင်း class ရဲ့ subclass (derived class) ကနေပဲ access လုပ်နိုင်ပါတယ်။

private: private members တွေကို သက်ဆိုင်ရာ class အတွင်းမှာ ရှိတဲ့ method တွေကနေပဲ access လုပ်နိုင်ပါတယ်။ အခြားသော နေရာတွေကနေ access လုပ်ခွင့် မရပါဘူး။

Sample Program ကို ပြန်သွားရမည်ဆိုလျှင် Dog class ရဲ့ name , age , weight တို့သည် public ဆိုသည့် access specifier အောက်တွင် ရှိသည့်အတွက် main function ကနေ တစ်ဆင့် object ကိုသုံးပြီး access လုပ်နိုင်ခြင်း ဖြစ်ပါတယ်။

Sample Program : Private and Public

```
#include <iostream>
using namespace std;
class Book { //declaration a class
private:
    int year;
    double price;
public:
    string title;
    void printBook();
    void setPrice(double p);
    void setYear(int y);

};

void Book::setPrice(double p) {
    price = p;
}

void Book::setYear(int y) {
    year = y;
}

void Book::printBook() {
    cout << "Title of the book : " << title << endl;
    cout << "Price of the book : " << price << endl;
    cout << "Year : " << year << endl;
}

int main() {
    Book book;
    int bPrice=0;
    cout<<"enter title of the book:";
    cin >> book.title;
    cout<<"enter price of the book:";
    cin>>bPrice;

    book.setPrice(bPrice);
    book.setYear(2013);
```

```

book.printBook();
return 0;
}

```

Program ရှင်းလင်းချက်

```

class Book { //declaration a class
private:
    int year;
    double price;
public:
    String title;
    void printBook();
    void setPrice(double p);
    void setYear(int y);
};

```

အထက်ပါ code သည် Book ဆိုသည့် class တစ်ခုကို ကြော်လှားခြင်းဖြစ်ပါသည်။ ထို class တဲ့တွင် private and public access specifier နှစ်ခုလုံးကို သုံးထားပါသည်။ private တဲ့တွင် ကြော်လှားသော members များကို Book ဆိုသည့် class အတွင်းကသာ ခေါ်သုံးခွင့်ရမှာ ဖြစ်ပါတယ်။ public အောက်တွင် ကြော်လှားသော variable နှင့် methods တွေကိုတော့မည့်သည့်နေရာကမဆို ခေါ်သုံးနိုင်ပါတယ်။ private တဲ့က members တွေကို public ထဲမှ method များကတစ်ဆင့် access လုပ်နိုင်သည်။ ထို့ကြောင့် private variables များကို access and modify လုပ်လိုလျှင် public method များ လိုအပ်ပါသည်။ အထက်တွင် printBook() , setPrice() , setYear() တို့ဖြင့် private ထဲမှ member data များကို access လုပ်နိုင်ပါသည်။

Defining Methods

```

void Book::setPrice(double p) {
    price = p;
}

```

Public အောက်မှာ ရှိသော method တွေကို ယခုကဲ့သို့ (::) double colon / scope resolution operator ကိုသုံးပြီး class အပြင်မှာ လာရေးနိုင်ပါတယ်။ declaration လုပ်ခြင်းကိုသာ class ထဲမှာ

ရေးပြီး ထို method ရဲ့ လုပ်ဆောင်ချက်တွေကို class အပြင်ဘက်မှာ ရေးထားခြင်းဖြင့် ပိုမိုဖတ်ရလွယ်ကူပြီး ရွင်းပါတယ်။ အထက်အတိုင်း (::) scope resolution operator ကိုသုံးပြီး function တစ်ခုကနေ class တစ်ခုကို ချိတ်ဆက်နိုင်ပါတယ်။

setPrice method ကိုရေးရခြင်းမှာ private အောက်တွင် `cout << book` ထားသော double price; ဆိုသည့် variable ကို access လုပ်လိုသောကြောင့်ဖြစ်သည်။ ထိုသို့ class အတွင်းမှာ method ရေးလိုက်မှသာ private အောက်က data များကို access ရယူနိုင်သောကြောင့်ဖြစ်သည်။

```
void Book::setYear(int y) {
```

```
    year = y;
```

```
}
```

```
void Book::printBook() {
```

```
    cout << "Title of the book : " << title <<endl;
```

```
    cout << "Price of the book : " << price <<endl;
```

```
    cout << "Year : " << year << endl;
```

```
}
```

setYear() method နှင့် printBook() method တို့သည်လည်း ထိုနည်းအတိုင်းဖြစ်သည်။

setYear() method သည် private မှ year variable ကို access လုပ်နိုင်မည်ဖြစ်သလို printBook()

method သည်လည်း price ကို access and modify လုပ်နိုင်မည်ဖြစ်သည်။

```
int main() {
```

```
    Book book;
```

```
    int bPrice=0;
```

```
    cout<<"enter title of the book:";
```

```
    cin >> book.title;
```

```
    cout<<"enter price of the book:";
```

```
    cin>>bPrice;
```

```
    book.setPrice(bPrice);
```

```
    book.setYear(2013);
```

```
    book.printBook();
```

```
    return 0;
```

```
}
```

နောက်ဆုံး အနေဖြင့် main function ထဲတွင် class ကို ခေါ်သုံးနိုင်ရန် class ကိုယ်စား book ဆိုသည့် object တစ်ခုကို တည်ဆောက်သည်။ ထို့နောက် user ဆီမှ title ကို တောင်းသည့် ရလာသော title အား public မှ title variable ထဲသို့ ထည့်သည်။ ထိုသို့ထည့်ရန်(.) dot operator ကိုသုံးပြီး book.title ဆိုသည့် code line ကိုရေးထားခြင်း ဖြစ်သည်။ ထို့နောက် user ဆီမှ price ကို ထပ်တောင်းပြီး ထို price အား setPrice method ဂဲ့parameter အနေဖြင့်သုံးပြီး ထိုမှုတစ်ဆင့် book.setPrice(bPrice) ဆိုသည့် code line ကိုရေးပြီး private ထဲမှ price ဆီသို့ ထည့်သွင်းသည်။ book.setYear() method သည်လည်း ထိုနည်းအတိုင်း ဖြစ်ပါသည်။ book.printBook() ဆိုသည့် code ကြောင်းသည် printBook() method ကို လုမ်းခေါ်ခြင်းသာ ဖြစ်ပြီး printBook() method သည် အချက်လက်များကို ပြန်လည်ဖော်ပြပေးရန် ရေးသားထားခြင်း ဖြစ်သည်။

```

class Book { //declaration a class
private:
    int year;
    double price;
public:
    string title;
    void printBook();
    void setPrice(double p);
    void setYear(int y);
};

void Book::setPrice(double p) {
    price = p;
}

void Book::setYear(int y) {
    year = y;
}

void Book::printBook() {
    cout << "Title of the book : " << title
    cout << "Price of the book : " << price
    cout << "Year : " << year << endl;
}

```

"C:\Cpp Programming Code\2class.exe"

enter title of the book:CppProgramming
 enter price of the book:10000
 Title of the book : CppProgramming
 Price of the book : 10000
 Year : 2013

Process returned 0 (0x0) execution time : 21.973 s
 Press any key to continue.

Protected

Sample program : protected

```

#include <iostream>
using namespace std;
class Parent {
protected:
    int data;
};
class Child : public Parent {
public:

```

```

void setData(int id)
{
    data = id;
}
void print()
{
    cout << "Protected data is: "
    << data << endl;
}
};

int main()
{
    Child child;
    child.setData(1001);
    child.print();
    return 0;
}

```

Program ගුණාලන්දායු

```

class Parent {
protected:
    int data;
};

```

അထක් පි code line වෙති protected access specifier පිළිසෙවා Parent ස්ථිවැනු class තමේ ඉග්‍රී තැනු නො ගැනීමෙන් ප්‍රෝටෝලංජ්‍යය ඇති අවබෝධනය වේ

```

class           Child          :           public           Parent          {
public:
    void           setData(int      id)
    {
        data           =           id;
    }
    void           print()
    {
        cout           <<           "Protected           data           is:           "
        <<           data           <<           endl;
    }
}

```

};

အထက်ပါ code line တို့သည် Parent class ကို ထပ်တိုးလိုက်သော(derived class) တစ်ခုကို တည်ဆောက်လိုက်ခြင်း ဖြစ်သည်။ derived class တစ်ခုကို တည်ဆောက်ရာတွင် ပထမဆုံး class keyword ကိုသုံးပြီး class name ရေးပေးရမည် ဖြစ်ပြီး ထိုနောက်တွင် : (colon) တစ်ခုပါဝင်ရမည်။ ပြီးမှ မိမိ derived လုပ်ချင်သော parent class (မူရင်း) ကို ရေးပေးရမည်။ မူရင်း parent class ကို ရေးသားရာတွင်လည်း parent class name ရှုခွဲ့ public ဆိုသည့် keyword ကို ထည့်ပေးရမည်။ ထိုသို့ public ကို ထည့်ပေးမှသာ protected အောက်တွင်ရှိသော data ကို derived လုပ်လိုက်သော class အတွင်းမှ access and modify လုပ်နိုင်မည် ဖြစ်သည်။ အသစ် တည်ဆောက်လိုက်သော child class အောက်တွင် method နှစ်ခုကို public ဖြင့် တည်ဆောက် ထားပါသည်။ ထိုသို့ တည်ဆောက်ထားမှသာ အခြားသောနေရာ (main function) ကနေ access and modify လုပ်နိုင်မည်ဖြစ်သည်။ void setData() method သည် protected အောက်တွင်ရှိသော data ထဲသို့ value လုမ်းထည့်ရန်ဖြစ်ပြီး print() method သည် data ကိုပြန်လည်ဖော်ပြန်ရန်သာ ဖြစ်သည်။

```
Child child;
child.setData(1001);
child.print();
```

Chile child; သည် child object တစ်ခုကို တည်ဆောက်ခြင်းဖြစ်ပြီး child.setData() သည် protected အောက်တွင်ရှိသော data ထဲသို့ 1001 ဆိုသည့် value လုမ်းထည့်ရန်ဖြစ်သည်။ child.print() သည် data ကိုပြန်လည်ဖော်ပြန်ရန်ရေးပေးထားခြင်း ဖြစ်သည်။

Constructor

Constructor ဆိုတာ class တစ်ခုရဲ့ special member function ဖြစ်သလို တစ်ခြားသော member function များနှင့် မတူပါဘူး။ special member function သည် class ရဲ့ object ကို တည်ဆောက်လိုက်သည့်နှင့် တစ်ပြင်နှက် automatically called လိုက်တာ ဖြစ်ပါတယ်။

Sample program : constructor

```
#include<iostream>
using namespace std;
```

```

class Dog{
public :
    int age;
    int weight;

    Dog(){ //creating a constructor
        age=5;
        weight=3;
    }
};

int main(){
    Dog dog;
    cout<<"my dog age is :"<<dog.age<<endl;
    cout<<"my dog weight is :"<<dog.weight<<endl;

    return 0;
}

```

Program ရှင်းလင်းချက်

```

Dog(){ //creating a constructor
age=5;
weight=3;
}

```

အထက်ပါ program တွင် Dog() သည် constructor တစ်ခုဖြစ်သည်။ ထို Dog() constructor ထဲမှ Public အောက်မှ age and weight တို့ကို value များထည့်ပေးမှာ ဖြစ်ပါတယ်။ constructor များသည် automatically called ဟု အထက်တွင် ရှင်းခဲ့သည်။ ဆိုလိုသည်မှာ Class Dog ကို object တည်ဆောက်လိုက်သည်နှင့် တစ်ပြိုင်နက် constructor Dog ဖြစ်သည့် Dog function သည် အလိုလျောက် အလုပ်ထလုပ်ခြင်း ဖြစ်သည်။

```

Dog dog;
cout<<"my dog age is :"<<dog.age<<endl;
cout<<"my dog weight is :"<<dog.weight<<endl;

```

Program တွင် main function မှ Dog dog; ဟု Dog class ကို စသုံးသည့် အခါ constructor Dog() သည်လည်း အလုပ်ချက်ချင်း ထလုပ်ပါသည်။ ထို့ကြောင့် constructor Dog() ထဲမှ instruction များဖြစ်သည့် 5 ကို age ထဲသို့ စတည့်သည်။ 3 ကိုလည်း weight ထဲသို့ စတည့်ပါသည်။ ထို့ကြောင့် public အောက်တွင်ရှိသော age and weight ထဲသို့ data များ ရောက်သွားပါပြီ။ ထို့နောက် main function ထဲတွင် dog.age ဟု ခေါ်လိုက်သောအခါ constructor မှတစ်ဆင့် data ထည့်ခံလိုက်ရသည့် age ရဲ့ value ကိုထုတ်ပြုခြင်း ဖြစ်ပါသည်။ weight သည်လည်း age အတိုင်းပင် ဖြစ်သည်။

Type of Constructors

1. Default Constructor
2. Parameterized Constructor
3. Copy Constructor
4. Default Constructor

Object ကို စတည်ဆောက်လိုက်သည့်နှင့် တစ်ပြိုင်နက် automatically called ခံရသည့် constructor သည် မည့်သည့် argument မျှ ပါဝင်နေခြင်းမရှိတာကို default constructor ဟု ခေါ်ဆိုခြင်းဖြစ်ပါသည်။ အထက်တွင် ရေးသားခဲ့သည့် program သည် default constructor program ဖြစ်သည်။

Parameterized Constructor

Object တစ်ခုကို ပြုလုပ်သည့်နှင့် တစ်ပြိုင်နက် ထို object နောက်တွင် parameters များ ထည့်ပေးလိုက်ခြင်းကို parameterized constructor ဟု ခေါ်ဆိုခြင်း ဖြစ်ပါသည်။ Parameterized Constructor sample program ကို အောက်တွင် ဖော်ပြထားသည်။

Sample Program : Parameterized Constructor

```
#include<iostream>
using namespace std;

class Dog{
public :
    string name;
    int age;
```

```

int weight;

Dog(String dName , int dAge , float dWeight){ //creating a constructor with
parameterized
    name = dName;
    age=dAge;
    weight=dWeight;
}
};

int main(){

String dName="puppy";
int dAge=5;
float dWeight=3.3;
Dog dog(dName,dAge,dWeight);
cout<<"my dog Name is :"<<dog.name<<endl;
cout<<"my dog age is :"<<dog.age<<endl;
cout<<"my dog weight is :"<<dog.weight<<endl;

return 0;
}

```

Program ရှင်းလင်းချက်

အထက်ပါ program တွင် Dog constructor နောက်တွင် Parameters များ ရေးထားပြီး ထိန်ည်းတူ
 Dog dog; dog object နောက်တွင်လည်း arguments များ ထည့်ထားသည့်အတွက် Parameterized
 Constructor ဖြစ်သည်။ အထက်ပါ program ရဲ့ main function ထဲတွင် အောက်ပါ code များကို
 ထပ်ထည့်ပြီး Program ကို upgradeလုပ်နိုင်ပါသေးသည်။

```

cout<<"_____ New Dog _____" << endl;
cout<<"Enter your dog name :"<< endl;
cin>>dName;
cout<<"Enter your dog age:"<< endl;
cin>>dAge;
cout<<"Enter your dog weight:"<< endl;
cin>>dWeight;
Dog newDog(dName,dAge,dWeight);

```

```
cout<<"my New dog Name is :"<<newDog.name<<endl;
cout<<"my New dog age is :"<<newDog.age<<endl;
cout<<"my New dog weight is :"<<newDog.weight<<endl;
```

Copy Constructor

Constructor အမျိုးအစားတွေထဲက နောက်ဆုံးတစ်မျိုးဖြစ်တဲ့ copy constructor သည်
သက်ဆိုင်ရာ class တစ်ခုရဲ့ ရှိပြီးသား object ကို copyလုပ်လိုသောအခါမျိုးမှာ သုံးသည်။
သို့မဟုတ် object တစ်ခု တည်ဆောက်ဖို့အတွက် class မှာရှိပြီးသား object ကို copy ယူပြီး
ပြန်သုံးခြင်း ဖြစ်ပါတယ်။ copy constructor ကို ရေးရာမှာ ပထမဆုံး class name ကို
ရေးပေးရသည်။ ထို့နောက်မှ parentheses ထဲတွင် class နှင့် object တို့ကို ရေးပေးရသည်။

ClassName (const ClassName &obj);

Sample Program : Copy Constructor

```
#include<iostream>
using namespace std;
class CopyConstruct
{
    private:
        int data1, data2;

    public:
        CopyConstruct(int x1, int y1)
        {
            data1 = x1;
            data2 = y1;
        }

        CopyConstruct (const CopyConstruct &sam)
        {
            data1 = sam.data1+1;
            data2 = sam.data2+1;
        }

        void display()
    {
```

```

cout<<data1<<" "<<data2<<endl;
}
};

int main()
{
    CopyConstruct obj1(10, 15); // Normal constructor
    CopyConstruct obj2 = obj1; // Copy constructor
    cout<<"Normal constructor : ";
    obj1.display();
    cout<<"Copy constructor : ";
    obj2.display();
    return 0;
}

```

Program ရှင်းလင်းချက်

```

CopyConstruct(int x1, int y1)
{
    data1 = x1;
    data2 = y1;
}

CopyConstruct (const CopyConstruct &sam)
{
    data1 = sam.data1+1;
    data2 = sam.data2+1;
}

```

အထက်တွင် constructor နှစ်ခု ရေးထားသည်။ ပထမ တစ်ခုသည် object တည်ဆောက်လိုက်သည့်နှင့် အလိုလျောက် အလုပ်လုပ်မည့် constructor ဖြစ်ပြီး ဒုတိယ constructor သည် object ကို copy ကူးပြီး အသစ်ရလာသော object အတွက် ဖြစ်သည်။ ထို့ကြောင့် အရင်းဦးဆုံး object တစ်ခုကို တည်ဆောက်ပြီး ခေါ်သည့်ဆိုပါစို့။ CopyConstruct obj1(10,20) ဟုရေးပြီး parameter ဖြင့် object ကို ထည့်ပေးလိုက်လျှင် ပထမ construct ၏ x1 နေရာသို့ 10 ရောက်သွားပြီး y1 နေရာသို့ 20 ရောက်သွားမည်။ obj1.data1 နှင့် obj1.data2 ဟု Print ထုတ်လျှင် 10 နှင့် 20 ကိုသာ ပြန်ရမည်။ ဒုတိယ တစ်ခုအင့်အနေဖြင့် နောက်ထပ် CopyConstruct class ကို သုံးပြီး object တစ်ခု တည်ဆောက်မည်။ အသစ်

თည်ဆောက်မည့် object သည် အရင်ရှိပြီးသား obj1 ကိုပြန်သုံးမည်။ အသုံးပြုပုံမှာ အောက်ပါ အတိုင်းဖြစ်သည်။

CopyConstruct obj2=obj1;

```
CopyConstruct obj1(10, 15); // Normal constructor
CopyConstruct obj2 = obj1; // Copy constructor
cout<<"Normal constructor : ";
obj1.display();
cout<<"Copy constructor : ";
obj2.display();
```

Obj1 ගි ගොවානු ඇතුළු පයම constructor ගිවාට් ප්‍රි: obj2 ගි ගොවානු ඇතුළු සූතිය
constructor ගිවා ජුෂ: ගොපිවනු. අයග්‍රන්ත රෙඩ්වාහාවා ප්‍රත්වනු object
අවත්තත් ඉභා: මුද්‍රණ: ගීප්‍රි:වා: object ගිව්‍ය්‍යාප්‍රි: තනුසොග්‍රැන්: ප්‍රශ්‍රවනු. ත්‍යිවිෂ්‍ය
තනුසොග්‍රැන්රාතුන් මුද්‍රණ: ගීප්‍රි:වා: constructor ගි ඔහුල්වල් ප්‍රූද්‍රණ:ල්ප්‍රි: අව්‍යාප්‍රැග්‍රැන්:
ප්‍රශ්‍රවනු.

```

private:
int data1, data2;

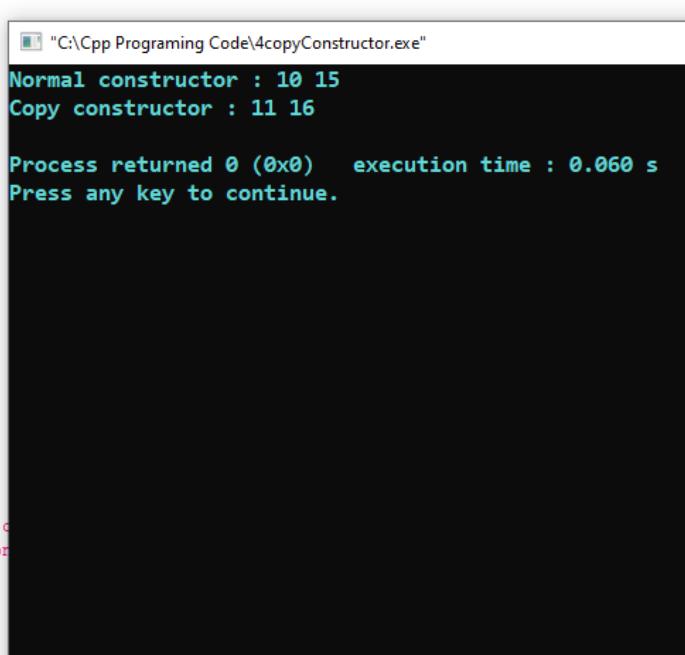
public:
CopyConstruct (int x1, int y1)
{
    data1 = x1;
    data2 = y1;
}

CopyConstruct (const CopyConstruct &sam)
{
    data1 = sam.data1+1;
    data2 = sam.data2+1;
}

void display()
{
    cout<<data1<<" "<<data2<<endl;
}

int main()
{
    CopyConstruct obj1(10, 15);      // Normal c
    CopyConstruct obj2 = obj1;       // Copy co
    cout<<"Normal constructor : ";
    obj1.display();
    cout<<"Copy constructor : ";
    obj2.display();
}

```



Data Structure And Algorithms

What is Data Structure?

Data Structure ဆိတ် data တွေကို နည်းလမ်းမျိုးစုံသုံးပြီး စုပေါင်းထားခြင်း ဖြစ်ပါတယ်။ တူညီတဲ့ data တွေချည်း သီးသန့်စုဆောင်းထားတာ ဖြစ်နိုင်သလို မတူညီတာတွေကို စုဆောင်းထား တာလည်း ဖြစ်နိုင်တယ်။ မိမိတို့ရေးသားသော software application ပေါ်မှုတည်ပြီး အကောင်းဆုံး ဖြစ်မယ့် နည်းလမ်းနဲ့ implement လုပ်ပြီး data structure ကို အသုံးပြုကြပါတယ်။ ထိုသို့ အသုံးပြုခြင်း ဖြင့် data တွေကို ထိတိရောက်ရောက် ကျိုးရှိစွာ ကိုင်တွယ်နိုင်ပါတယ်။

Famous Data Structures(commonly used)

1. **Arrays**
2. **Stacks**
3. **Queues**
4. **Linked Lists**
5. **Trees**
6. **Graphs**
7. **Tries**
8. **Hash Tables**

What is an Algorithm?

Algorithm သည် code မဟုတ်ပါဘူး။ အခက်အခဲ (problem) ကို အဖြော်ရရန် (solution) တစ်ဆင့်ပြီး တစ်ဆင့် (step by step) ဉာဏ်ကြားထားခြင်းကို Algorithm လိုအပ်ပါတယ်။ Algorithm တစ်ခုသည် ပြဿနာကိုဖြေရှင်းနည်းများစွာ ထဲကမှ အကောင်းနည်းလမ်းဖြစ်ရပါမည်။ Algorithm ကောင်းတစ်ခု ဖြစ်ဖို့ အရည်အချင်းတွေ လိုအပ်နေပါသေးတယ်။ ဉာဏ်ကြားချက် တစ်ခုစီတိုင်းသည် အောက်ပါအချက်များနှင့် ပြည့်စုံနေမှုသာ algorithm ကောင်းတစ်ခုလို့ ဆိုနိုင်ပါမယ်။

ဉာဏ်ကြားချက် တစ်ခုစီတိုင်းကို တိတိကျကျ ပြတ်ပြတ်သားသားရေးသား ထားမှသာ algorithm ကောင်းဖြစ်နိုင်မည်။

ဉာဏ်ကြားချက် တစ်ခုစီတိုင်းသည် ထိရောက်မှ ရှိရပါမည်။

သတ်မှတ်ထားတဲ့ အဆင့်တွေ ပြီးသွားတဲ့အခါ algorithm သည် ရပ်သွားရပါမည်။

Algorithm တစ်ခုသည် zero or သတ်မှတ်ထားသည့် inputs များ ရှိရမည်။

Algorithm တစ်ခုသည် 1 or သတ်မှတ်ထားသည့် output များ ထွက်လာရမည်။

ဇွန်ကြားချက်တွေ အားလုံးပြီးမြောက်သွားတဲ့အခါ မျှော်မှန်းထားတဲ့ result ထွက်လာ ရပါမည်။

Algorithm တစ်ခုသည် step by step ဇွန်ကြားချက်များသာ ဖြစ်ရမည် ဖြစ်ပြီး programming code တွေနဲ့ ဖော်ပြလို့ မရပါဘူး။

ဥပမာ အနေဖြင့် လူတစ်ယောက် အသက်သည် ၁၈ နှစ်ထက် ကြီးလား ငယ်လား algorithm တစ်ခုကို ရေးမည်ဆိုလျှင်

အသက်ဘယ်လောက်လဲဆိုတာ တောင်းမည်။

ပေးလာသော အသက်ကို သိမ်းမည်။

သိမ်းထားသော အသက်ကို 18 ထက်ကြီးသလား စစ်မည်။ ကြီးလျှင် ကြီးသည့်အကြောင်း ထုတ်ပြမည်။ ငယ်လျှင် ငယ်သည့် အကြောင်း ထုတ်ပြမည်။

ရပ်မည်။

Abstract Data Types (ADTs)

ကားမောင်းဖို့အတွက် ကားသော့ကို လှည့်မယ်။ ကားစက်နိုးလာပြီဆိုတဲ့နဲ့ သင့်တော်တဲ့ gear ထိုးပြီး ကျွန်တော်တို့ ကားကို မောင်းနိုင်ပါပြီ။ ကားသော့ လှည့်လိုက်တဲ့အချိန်မှာ battery ကနေ တဆင့် starter motor ကို power ဘယ်လို့ရောက်သလဲ။ starter motor ဘယ်လို့ စလည်သလဲ လောင်စာဆီတွေ ဘယ်လို့ပေါက်ကဲ့သလဲဆိုတာ ကျွန်တော်တို့ မသိလည်း ကားမောင်းနိုင်ပါတယ်။ real world မှာ ဆိုရင်တော့ ဒါကို information abstraction လို့ခေါ်ပြီး computer science ရဲ့ Programming နယ်ပယ်မှာ ဆိုရင်တော့ data abstraction လို့ ခေါ်ပါတယ်။

Abstract Data Type (ADT) အကြောင်းကို နားလည်ဖို့ဆိုရင် what is abstract ? and what is data type? ဆိုတာကို အရင်ဆုံးရှင်းလင်းစွာ သိထားရပါမည်။

ပထမဆုံး အနေဖြင့် data type အကြောင်းကို ပြောလိုပါတယ်။ data type တွေဆိုတာ data တွေကို type ခွဲခြားထားတာပါ။ ဥပမာ - number တွေ stored လုပ်ချင်တယ်ဆိုရင် int ကို သုံးပါတယ်။ ထိုနည်းတူ character (စကားလုံး) တွေကို stored လုပ်ချင်ရင် char ကို သုံးပါတယ်။ စာသားတွေကို stored လုပ်ချင်ရင်တော့ string ကို သုံးပါတယ်။

Int data type သည် 4 bytes ရှိသည် ဆိုပါစို့။ (မိမိ compiler အပေါ်မူတည်ပြီး ပြောင်းလဲနိုင်သည်)။ 1 byte သည် 8 bit ရှိတဲ့အတွက် 32 bits ထိ memory ပေါ်မှာ stored လုပ်နိုင်ပါတယ်။ ဥပမာ int a=10 ; and int b=20 ဆိုပါစို့။ int c= a+b (instruction) တွင် c သည် 30 ရသည် ဆိုပါစို့။ ကျွန်ုတ်တို့အနေဖြင့် 10 01 (binary operation) တွေကို သိစရာမလိုပဲ data type အပေါ်မှာ မူတည်ပြီး a and b ကဲ့သို့ data တွေကို arithmetic (+ , - , * , / , %) operation တွေ လုပ်နိုင်ပါတယ်။ ထိုကဲ့သို့ အတွင်းကျကျ သိစရာမလိုပဲ အလွယ်တကူ သုံးလို့ရအောင် ဖုံးအုပ်ထား ခြင်းကို abstract လို့ခေါ်ပါတယ်။

Example with array:

ဥပမာ array သည် abstract data type အမျိုးအစား ဖြစ်ပါတယ်။ array တစ်ခုရဲ့ index 1 မှာ data ထည့်ချင်သည် ဆိုပါစို့။ array[1]=10 ; ထိုကဲ့သို့ ရေးလိုက်ရုံဖြင့် array ရဲ့ index 1 နေရာမှာ 10 ဆိုသည့် data ကို ထည့်နိုင်ပါတယ်။ ထိုနည်းတူ ပြန်ထုတ်ချင်လျှင်လည်း int a = array[i]; ယခု နည်း အတိုင်း ရေးပြီး ပြန်ထုတ်နိုင်ပါတယ်။ array နှစ်ခုကို ပေါင်းချင်လျှင်လည်း int c= array a[1] + array b[1] ; ထိုကဲ့သို့ ရေးပြီး array နှစ်ခုရဲ့ index 1 တို့ကို ပေါင်းနိုင်ပါတယ်။ memory ပေါ်မှာ array နှစ်ခု ဘယ်လို ပေါင်းသွားကြလဲ။ 1010 0011 တွေ ဘယ်လို ပေါင်းသွားကြလဲဆိုတာ မသိလည်း ကျွန်ုတ်တို့ arithmetic operation တွေ လုပ်ဆောင်နိုင်ပါတယ်။ ထိုကဲ့သို့ လုပ်ဆောင်နိုင်တဲ့ data type တွေကို abstract data types တွေလို့ ခေါ်ပါတယ်။ လူသိများတဲ့ stack data structure သည်လည်း abstract data type ပါ။

STL Containers

C++ programming ရဲ့ STL Standard Template library မှာ containers, iterators and algorithms တို့ ပါဝင်ပါတယ်။ Standard Library Container ဆိုတာ object တွေကို စုပေါင်းထားတာကို ဆိုလိုတာပါ။ Container တွေကို အခြေခံအားဖြင့် နှစ်မျိုးခွဲခြားနိုင်ပါတယ်။ sequences and associate container တို့ ဖြစ်ပါတယ်။ sequences container ထဲတွင် array , vector , list , deque , forward_list တို့ ပါဝင်ပါတယ်။ array , vector , list , deque , forward_list တို့သည် sequence containers ဖြစ်သည့်နှင့် အညီ ယင်းတို့ ထဲတွင်ပါဝင်သော သို့မဟုတ် stored လုပ်ထားသော elements များကို (accessed sequentially) အစဉ်လိုက် ရယူခြင်း၊ ထပ်ထည့်ခြင်း၊ လျှော့ချုခြင်းများ လုပ်ဆောင်နိုင်ပါတယ်။ Associate container မှာတော့ set , map , multiset and multimap တို့ ပါဝင်ပါတယ်။ sequence

containers နှင့် associate container ကြားက ခြားနားချက်သည် associate containers များတွင် elements တွေကို key များဖြင့်သာ accessed လုပ်ပါတယ်။ sequences container ထဲမှ vector အကြောင်းကို အောက်တွင် ဆွဲးနွေးသွားပါမည်။

Vector

Vector တွေက array တွေနဲ့ ပုံစံတူပါတယ်။ ဆိုလိုချင်တာက vector တွေသည်လည်း array ကဲ သို့ data တွေကို contiguous memory ပုံစံနဲ့ stored လုပ်ပါတယ်။ Array နဲ့ Vector ကြားမှာ မတူညီတဲ့ အချက်သည် vector တွေဟာ dynamic array တွေဖြစ်ပါတယ်။ ဆိုလိုတာက runtime မှာ size ကို မိမိ လိုသလို ပြောင်းလဲနိုင်ပါတယ်။ ဥပမာ အရင်က ကြော်ပြားသည့် size သည် 5 ဆိုသော်လည်း program run နေရင်း size ထပ်ကြီးလိုတဲ့ အခါ size အသစ်တစ်ခုထပ်ပြီး create လုပ်နိုင်ပါတယ်။

Syntax

```
Vector < object_type > vec;
```

Vector တစ်ခုကို ကြော်ပြာရာတွင် ရှေးဆုံး၍ vector keyword ဆိုသည့် keyword ကို ထည့်ပေးရမည်။ ထိုနောက် bracket ထဲ၌ မိမိ အသုံးပြုလိုသော type ကို ထည့်ပေးရမည်။ နောက်ဆုံးတွင် vector name ကို ထည့်ပေးရမည်။ အထက်တွင် ရေးထားသာ code ၏ vec သည် vector name ဖြစ်သည်။

Example Program: Input Data to Vector and Access element From Vector

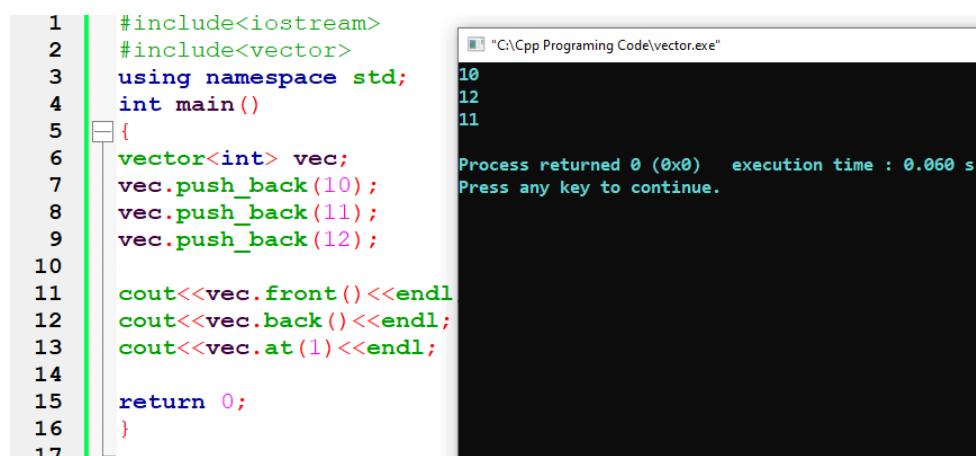
```
#include<iostream>
#include<vector>
using namespace std;
int main()
{
    vector<int> vec;
    vec.push_back(10);
    vec.push_back(11);
    vec.push_back(12);

    cout<<vec.front()<<endl;
    cout<<vec.back()<<endl;

    return 0;
}
```

Vector ကို အသုံးပြုရန် vector ဆိုသည့် header file ကို ရေးပေးထားရန် လိုအပ်ပါသည်။ vector ထဲသို့ data များ ထည့်နိုင်ရန် array ထဲသို့ assignment operator ကို သုံးပြီး data ထည့်သည့် အတိုင်း ထည့်နိုင်သလို push_back function ကိုသုံးပြီးလည်း ထည့်နိုင်သည်။ push_back function သည် မိမိတို့ထည့်လိုက်သော element ကို နောက်ဆုံးမှာ ထည့်ပေးပါသည်။ အထက်ပါ program တွင် vec.push_back(10) ဟု ပထမဆုံး element ကို စထည့်လိုက်သည်။ ထို့နောက်တွင် vec.push_back(11) ဟုရေးကာ 11 ဆိုသည့် element ကို ထည့်လိုက်ပါသည်။ push_back သည် နောက်ဆုံးတွင် element ကို ထည့်သည့်အတွက် ပထမ data ရှိသည့် 10 နောက်တွင် 11 ရောက်လာပါမည်။ ထို့နောက်တွင်မူ 12 ကို ထပ်ထည့်သည့်အတွက် 12 သည် နောက်ဆုံးက ဖြစ်သွားပါမည်။ vec ဆိုသည့် vector ထဲတွင် element တို့သည့် 10 ,11 ,12 ဟု အစဉ်လိုက် ရှိနေပါမည်။ ထို့ကြောင့် ပထမဆုံး element ကို ထုတ်လျှင် 10 ထွက်လာမည်ဖြစ်ပြီး နောက်ဆုံး element ကိုထုတ်လျှင် 12 ထွက်လာပါမည်။ အကယ်၍ 12 နောက်တွင် နောက်ထပ် element တစ်ခု ထပ်ထည့်မည်ဆိုလျှင် နောက်ဆုံး element သည် 12 နောက်က element ဖြစ်ပါမည်။

Vector ထဲမှ elements များကို access ပြန်လုပ်နိုင်ရန် အထွက်တွင် front() function and back() function ကို သုံးထားပါသည်။ front သည် ရွှေ့ဆုံး element ကို access လုပ်လိုတဲ့အခါ အသုံးပြုပြီး back သည် နောက်ဆုံး element ကို access လုပ်လိုတဲ့အခါမှာ အသုံးပြုပါတယ်။ front function သည် return value အနေဖြင့် vector ရဲ့ ပထမဆုံး element ကို return ပြန်ပေးပါတယ်။ front() function ကို အသုံးပြုရတွင် မည်သည့် parameter မှ ထည့်ပေးစရာမလိုပါဘူး။ ထိုနည်းတူ back() function သည်လည်း မည်သည့် parameter မှ ထည့်ပေးစရာ မလိုသလို return value အနေဖြင့် vector ထဲမှ နောက်ဆုံး element ကိုပြန်ပေးပါတယ်။ တကယ်လို့ front and back ကြားထဲမှာ ရှိနေသော elements တွေလို access လုပ်လိုလျှင်တော့ at() function ကို သုံးနိုင်ပါတယ်။ at() function ကို အသုံးပြုဖို့အတွက် မိမိ access လုပ်လိုသော index number ကို ထည့်ပေးရပါတယ်။ ဥပမာအနေဖြင့် အထက်ပါ program တွင် cout<<vec.at(1)<<endl; ယခု code line ကို ထပ်ဖြည့်လိုက်ပါက output အနေဖြင့် 11 ထွက်လာသည်ကို မြင်ရပါမည်။ vector သည်လည်း 0 မှ စတင် ရေတွက်သည့်အတွက် parameter အနေဖြင့် 1 ကို ထည့်လိုက်သည့်အခါ vector ထဲမှ element 11 ထွက်လာခြင်း ဖြစ်ပါသည်။



```

1 #include<iostream>
2 #include<vector>
3 using namespace std;
4 int main()
5 {
6     vector<int> vec;
7     vec.push_back(10);
8     vec.push_back(11);
9     vec.push_back(12);
10
11    cout<<vec.front()<<endl;
12    cout<<vec.back()<<endl;
13    cout<<vec.at(1)<<endl;
14
15    return 0;
16 }
17

```

Vector Functions

Vector တွင် အသုံးပြန်သော functions ပေါင်းများစွာ ရှိသည်။ ထိုတဲ့ကမှ အသုံးများသော vector functions များကို အောက်ပါပုံတွင် ဖော်ပြထားပါသည်။ vector functions တွေကို အခြေခံအားဖြင့် ကောင်းစဉ်လေးခု ခွဲခြားနိုင်ပါသည်။

Iterator	Capacity	Element Access	Modifiers
-begin	- size	- operator[]	- assign
-end	- max_size	- at	- push_back
-rbegin	- resize	- front	- pop_back
-rend	- capacity	- back	- insert
-cbegin	- empty	- data	- erase
-cend	- reserve		- swap
-crbegin	- shrink_to_fit		- clear
-crend			- emplace
			- emplace_back

Sample Program : Begin and End

```

#include<iterator>
#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> myvector;
    for (int i=1; i<=5; i++) {

```

```
myvector.push_back(i);
}
vector<int>::iterator ptr;
cout << "All elements in the container are ";
for ( ptr = myvector.begin(); ptr != myvector.end(); ++ptr){
    cout << *ptr << endl;
}
return 0;
}
```

အထက်ပါ program ကို run ကြည့်မယ်ဆိုရင် output အနေဖြင့် 1,2,3,4,5 တို့ကို ရမှာပါ။ vector ထဲကို elements တွေထည့်ဖို့ push_back() function ကိုသုံးထားပါတယ်။ ထို့ပြင် ယခု program ၏ iterator ကိုလည်း စတင်သုံးထားပါတယ်။ iterator ကိုသုံးဖို့ဆိုရင် iterator header file ကို ကြော်ပြာပေးဖို့လိုသလို iterator ကိုလည်း program ထဲတွင် ထပ်ကြော်ပြာပေးရပါတယ်။ iterator သည် object တစ်ခုဖြစ်သလို pointer နဲ့ ဆင်တူပါတယ်။ သို့သော် memory address ကိုတော့ တိုက်ရှိက် ဖော်ပြလို့ မရပါဘူး။ တိုက်ရှိက်ဖော်ပြမယ်ဆိုရင် compiler ကနေ error ပြမှာပါ။ iterator သည် vector container ထဲမှာ ရှိတဲ့ elements တွေကို point ထောက်ရန်အတွက် အသုံးပြုပါတယ်။ `vector<int> :: iterator ptr; // code line` သည် iterator type နဲ့ ptr ဆိုတဲ့ iterator တစ်ခုကို စတင် ကြော်ပြလိုက်တာပါ။ ထိုကဲ့သို့ ကြော်ပြာလိုက်သော ptr ကို for loop ထဲတွင် `ptr = myvector.begin()` ဆိုပြီး ပြန်သုံးထားပါတယ်။ အလုပ်လုပ်သည့် ပုံစံမှာ begin() function သည် vector container ထဲမှာ ရှိတဲ့ ပထမဆုံး element ကို return ပြန်ပေးပါတယ်။ ထို element ကို ptr ထဲသို့ ထည့်လိုက်ပါတယ်။ ထို့နောက်တွင်မှု *ptr ဟုရေးကာ vector ထဲမှ first element ကို print ထုတ်လိုက်ပါတယ်။ `vector end()` function သည် vector container ထဲမှ နောက်ဆုံး element ကို return ပြန်ပေးပါတယ်။ `ptr != myvector.end()` ဟု ရေးထားသည့်အတွက် vector container ထဲမှာရှိတဲ့ elements အားလုံးကို အဆုံးထိ print ထုတ်ပေးမှာ ဖြစ်ပါတယ်။

Rbegin and Rend Function

5 ကို ရရှိပါမည်။ အဘယ်ကြောင့် မူရင်းရှိသည့် vector elements များ ကို reverse လုပ်လျှင် 5 ,4 ,3,2, 1 တို့ကို ရရှိပါမည်။ ထိုကဲ့သို့ ရရှိလာသော vector တွင် ပထမဆုံး element သည် 5 ဖြစ်သည့်အတွက် rbegin function ကိုသုံးလျှင် 5 ကို ပြန်လည်ရရှိခြင်း ဖြစ်ပါသည်။ ထိုနည်းတူ rend သည်လည်း vector elements များ reverse လုပ်ပြီး နောက်ဆုံး element ကို return value အနေဖြင့် ပြန်ပေးပါသည်။

Sample Program : rbegin() and rend()

```
#include <iostream>
#include<vector>

using namespace std;

int main()
{
    vector<int> myVec;
    myVec.push_back(1);
    myVec.push_back(2);
    myVec.push_back(3);
    myVec.push_back(4);
    myVec.push_back(5);

    cout << "The vector elements in reverse order are:\n";
    for (auto it = myVec.rbegin(); it != myVec.rend(); it++)
        cout << *it << " ";
    return 0;
}
```

အထက်ပါ program အား run ကြည့်လျှင် output အနေဖြင့် 5 ,4 ,3,2 ,1 တို့ကို ရရှိမှာ ဖြစ်ပါတယ်။

cbegin() and cend()

cbeing() သည် container ထဲမှာရှိတဲ့ ပထမဆုံး element ကို ထောက်ထားပြီး const_iterator ကို return ပြန်ပေးပါတယ်။ const ဖြစ်တဲ့အတွက် container ထဲမှာရှိတဲ့ elements တွေကို modify လုပ်လို့ မရပါဘူး။ cend() သည် နောက်ဆုံး element ကို point ထောက်ပြီး return အနေဖြင့် const_iterator ကို ပြန်ပေးပါတယ်။

Sample program : cbegin() and cend()

```

#include <iostream>
#include <string>
#include <vector>
using namespace std;
int main()
{
    vector<string> vec;
    vec.push_back("e_first");
    vec.push_back("e_second");
    vec.push_back("e_third");
    vec.push_back("e_fourth");
    vec.push_back("e_fifth");

    cout << "Contents of the vector:" << endl;
    for (auto itr = vec.cbegin(); itr != vec.end(); ++itr) {
        cout << *itr << endl;
    };
    return 0;
}

```

`cbegin()` နှင့် `begin()` ကြေားက ပြားနားချက်ကို ကွဲပြားစွာသိလိုလှင် အောက်ပါ program ကို ကြည့်ပါ။

```

1  #include<iterator>
2  #include <iostream>
3  #include <vector>
4  using namespace std;
5
6  int main()
7  {
8      vector<int> myvector;
9      for (int i = 1; i <= 5; i++) {
10         myvector.push_back(i);
11     }
12     vector<int>::iterator ptr;
13     cout << "All elements in the container are ";
14     for (ptr = myvector.begin(); ptr != myvector.end(); ++ptr) {
15         *ptr = 11;
16         cout << *ptr << endl;
17     }
18     return 0;
19 }

```

အထက်ပါ ပုံတွေ၏ line number 15 တွင် container တဲ့မှ ရှိသော elements များ ကို modify လုပ်ထားပါသည်။ `*ptr=11;` ဟုရေးကာ elements အားလုံးကို 11 အဖြစ် ပြောင်းပြစ်ပါသည်။

ထိုကြောင့် output များကို ကြည့်လျှင်လည်း 11 များကိုသာ တွေ့ရခြင်း ဖြစ်သည်။ အထက်ပါ program သည် begin ဖြင့်သာ ရေးထားသောကြောင့် modify လုပ်လိုက်ခြင်းဖြစ်ပြီး cbegin() ဖြင့် ရေးမည့်ဆိုလျှင် modify လုပ်၍ မရ နိုင်ပါ။

Vector :: insert()

Insert function သည် container ထဲသို့ elements များ ထပ်ထည့်လိုတဲ့အခါမှာ သုံးပါတယ်။ insert function သည် အနည်းဆုံး arguments နှစ်ခု ယူပါတယ်။ ပထမတစ်ခုသည် မိမိထည့်လိုသော vector ခဲ့ position ဖြစ်ပြီး ဒုတိယတစ်ခုသည် မိမိထည့်လိုသော element ဖြစ်ပါတယ်။ elements တွေကို မိမိစိတ်ကြိုက် ထည့်နိုင်ပါတယ်။ dynamic ဖြစ်တဲ့အတွက် မိမိထည့်ပေးလိုက်သော elements အရေအတွက်အတိုင်း container size က တိုးလာမှာ ဖြစ်ပါတယ်။

Example Program :: Vector::insert()

```
#include <iostream>
#include<vector>
using namespace std;

int main()
{
    vector<int> myVec = { 1,2,3,4,5,6 };
    auto it = myVec.insert(myVec.begin(), 11);
    myVec.insert(it, 22);
    cout << "The vector elements are: ";
    for (auto it = myVec.begin(); it != myVec.end(); ++it)
        cout << *it << " ";

    return 0;
}
```

```

5   int main()
6   {
7
8     vector<int> myVec = { 1,2,3,4,5,6 };
9     auto it = myVec.insert(myVec.begin(), 11);
10    myVec.insert(it, 22);
11    cout << "The vector elements are: ";
12    for (auto it = myVec.begin(); it != myVec.end(); ++it)
13      cout << *it << " ";
14
15    return 0;
16
17 }

```

The vector elements are: 22 11 1 2 3 4 5 6
Process returned 0 (0x0) execution time : 0.055 s
Press any key to continue.

Line number 8 သည် vector များကို ကြော်လှားခြင်းဖြစ်ပြီး line number 9 တွင် insert function ကိုသုံးကာ myVec.begin() ,11 ဟုရေးပြီး container ရဲ့ပထမဆုံးနေရာသို့ 11 ကို ထည့်လိုက်ပါတယ်။ ထိုပြင် line number 10 တွင် container ထဲသို့ 22 ကို ထပ်ထည့်လိုက်ပါတယ်။ နောက်ထပ်ထည့်လိုက်သည့် 22 သည် 11 ရဲ့ရှေ့သူ ထည့်ခြင်းဖြစ်ပါတယ်။ ထို့နောက်တွင် myVec vector ထဲတွင် 22 , 11 ,1 ,2 ,3 ,4,5,6 ဟုအစဉ်လိုက် ရှိနေမည်ဖြစ်ပါသည်။ ထိုပြင် insert function ထဲ၏အောက်ပါအတိုင်း line number 8 နောက်ထပ် argument ထပ်ထည့်နိုင်ပါသေးတယ်။

```

7   vector<int> myVec = { 1,2,3,4,5,6 };
8   auto it = myVec.insert(myVec.begin(), 5,88);
9   myVec.insert(it, 22);
10  cout << "The vector el";
11  for (auto it = myVec.b;
12    cout << *it << " ";

```

The vector elements are: 22 88 88 88 88 88 1 2 3 4 5 6
Process returned 0 (0x0) execution time : 0.073 s
Press any key to continue.

ထိုကဲ့သို့ ထပ်ထည့်လိုက်သောအခါတွင် ဒုတိယတစ်ခု ဖြစ်သည့် 5 သည် element မဟုတ်တော့ပဲ အရေအတွက်သာ ဖြစ်သွားပါမည်။ နောက်ဆုံးဝင်လာသည့် တတိယ argument သည် element ဖြစ်သွားပါမည်။ ဆိုလိုသည့်မှာ container ထဲသို့ 88 ကို 5 ထည့်မည့်ဟု ဆိုလိုခြင်း ဖြစ်ပါသည်။

Vector :: size() and pop_back()

Vector ထဲတွင် ပါဝင်သော size function သည် container ရဲ့ elements အရေအတွက် size ကိုပြန်ပေးပါတယ်။ ဥပမာ vector container ထဲတွင် elements စ ခုရှိလှုပ် return value အနေနဲ့ 8 ကိုပြန်ပေးပါတယ်။ pop_back() သည် vector container ထဲမှာရှိတဲ့ နောက်ဆုံး element တစ်ခုတည်းကိုဖြတ်ထုတ်ပေးပါတယ်။ အောက်ပါ program တွင် size() , push_back() , end() , begin() , pop_back() function တို့ကို တွဲသံးထားပါတယ်။

Sample Program :: size() , push_back() , end() , begin() and pop_back()

```
#include <iostream>
#include <vector>
int main ()
{
    std::vector<int> myVec;
    std::cout << "0. size: " << myVec.size() << '\n';

    for (int i=0; i<10; i++) myVec.push_back(i);
    std::cout << "1. size: " << myVec.size() << '\n';

    myVec.insert (myVec.end(),10,100);
    std::cout << "2. size: " << myVec.size() << '\n';

    myVec.pop_back();
    std::cout << "3. size: " << myVec.size() << '\n';

    std::cout << "All the elements in the container are : >" << std::endl;
    for (auto it=myVec.begin(); it!=myVec.end(); ++it)
        std::cout << *it << " ";
}

return 0;
}
```

```
11
12     myVec.insert (myVec.end(),10,100);
13     std::cout << "2. size: " << myVec.size() << '\n';
14
15     myVec.pop_back();
16     std::cout << "3. size: " << myVec.size() << '\n';
17
18     std::cout << "All the elements in the container are : >" << std::endl;
19     for (auto it=myVec.begin(); it!=myVec.end(); ++it)
20         std::cout << *it << " "
```

Line number 12 တွင် end function သည် vector container ရဲ့ နောက်ဆုံးမှာ 100 ဆိုသည့် elements

10 ခုကို ထည့်ပေးရန် ရေးထားခြင်း ဖြစ်ပါသည်။

`Vector::resize()` and `shrink_to_fit()`

`Vector resize()` function သည် vector container ရဲ့ size ကိုလိုသလို တိုးမြှင့်ခြင်း၊ ငျလျော့ချု ခြင်းများ
ပြလုပ်ရာတွင် အသုံးပြုပါတယ်။ ဥပမာ vector တစ်ခုရဲ့ size သည် 10 ဆိုပါစို့။ `myVector.resize(5)`

လိုပေးလိုက်သည်နှင့် တစ်ပြိုင်နက် vector size သည် 5 ဖြစ်သွားပါသည်။ သို့မဟုတ် ထပ်တိုးလိုတဲ့အခါမှာလည်း တိုးနိုင်ပါတယ်။ သို့ရာတွင် ပြဿနာအနည်းငယ် ရှိပါသည်။ ဥပမာ myVector ဆိုသည့် vector တစ်ခုအတွင်း၌ 1,2,3,4,5,6,7,8,9,10 စသည် elements များ ရှိသည် ဆိုပါစို့။ vector ရဲ့ size သည်လည်း 10 ခု ရှိနေပါမည်။ ထို့နောက် vector size ကို 5 သို့လျော့ချု လိုသောအခါတွင် myVector.resize(5) ဟု ရေးပါမည်။ ပြီးလျှင် vector size ကို ပြန်စစ်ကြည့်သည့် အခါ 5 ဖြစ်နေသည်ကို တွေ့ရပါမည်။ သို့သော် vector ထဲတွင် elements 10 ခုလုံး ရှိနေပါသေးသည်။ vector size ကို resize လုပ်နိုင်သော်လည်း elements များ ပျက်မသွားပါ။ ထိုကဲ့သို့သော အခြေနေမျိုးတွင် shrink_to_fit() function ကို သုံးရပါမည်။ shrink_to_fit() function သည် vector size အတိုင်းသာ elements များကို ထားပေးပါသည်။

Sample Program :: resize() and shrink_to_fit()

```
#include<iostream>
#include<vector>
using namespace std;
#define WINHTUT 10

int main()
{
    vector<int> myVec(WINHTUT);
    for (int i = 0; i < WINHTUT; i++)
        myVec[i] = i;
    cout << "Vector size initially: " << myVec.size();
    cout << "\nVector elements are: ";
    for (int i = 0; i < WINHTUT; i++)
        cout << myVec[i] << " ";
    myVec.resize(5);

    cout << "\n\nVector size after resize(5): "
    << myVec.size();

    cout << "\nVector elements after resize(5) are: ";
    for (int i = 0; i < WINHTUT; i++)
        cout << myVec[i] << " ";
}
```

```

myVec.shrink_to_fit();

cout << "\n\nVector size after shrink_to_fit(): " << myVec.size();

cout << "\nVector elements after shrink_to_fit() are: ";
for (int i = 0; i < WINHTUT; i++)
    cout << myVec[i] << " ";

return 0;
}

8     vector<int> myVec(WINHTUT);
9     for (int i = 0; i < WINHTUT; i++)
10        myVec[i] = i;
11     cout << "Before size of vector container is : " << myVec.size();
12     cout << "\nAll elements are: ";
13     for (int i = 0; i < WINHTUT; i++)
14        cout << myVec[i] << " ";
15     myVec.resize(5);
16
17     cout << "\n\nAfter resize(5): "
18     << myVec.size();
19
20     cout << "\nVector elements after resize(5) are: ";
21     for (int i = 0; i < WINHTUT; i++)
22        cout << myVec[i] << " ";
23     myVec.shrink_to_fit();
24
25     cout << "\n\nVector size after shrink_to_fit(): "
26     << myVec.size();
27
28     cout << "\nVector elements after shrink_to_fit() are: ";
29     for (int i = 0; i < WINHTUT; i++)
30        cout << myVec[i] << " ";
31
32     return 0;
33
34

```

Screenshot of the terminal showing the execution of the program:

```

"C:\Cpp Programming Code\Untitledvect.exe"
Before size of vector container is : 10
All elements are: 0 1 2 3 4 5 6 7 8 9

After resize(5)
Vector elements after resize(5) are: 0 1 2 3 4 5 6 7 8 9

Vector size after shrink_to_fit(): 5
Vector elements after shrink_to_fit() are: 0 1 2 3 4 0 826161658 49364
Process returned 0 (0x0)   execution time : 0.069 s

```

အထက်ပါ program ကို ကြည့်လျှင် line number 4 မှ #define WINHTUT 10 သည် symbolic constant ကို သုံးထားခြင်း ဖြစ်သည်။ ထိုသို့သုံးရခြင်းမှာ အသုံးများသော number များကို name တစ်ခုဖြင့် အသေသတ်မှတ်ထားပေးလိုသောကြောင့် ဖြစ်သည်။ အထက်ပါ code တွင် WINHTUT ဆိုသည့် အတဲတွင် 10 ကို stored လုပ်ထားသည်။ line number 9 တွင် vector တဲ့သို့ elements များ ထည့်သည်။ ထိုနောက်တွင် vector size ကို စစ်ကြည့်သည့်အခါ 10 ဖြစ်နေတာကို မြင်ရပါမည်။ line number 15 တွင် vector size ကို resize(5) ဟုလုပ်ပြီး လျော့ချုလိုက်ပါမည်။ line number 18 တွင် vector size ကို ပြန်စစ်ကြည့်သည့်အခါ 5 ဖြစ်နေသည်ကို တွေ့ရပါမည်။ ထိုနောက် line number 21 တွင် vector

ထဲတွင်ရှိသော elements များကို ပြန်ထုတ်ကြည့်ရာတွင် vector size သာ လျော့ကျ သွားသော်လည်း elements များ ရှိနေသေးသည်ကို မြင်ရပါမည်။ line number 23 တွင် myVec.shrink_to_fit() ဟု ရေးပြနာက် line number 29 တွင် elements များကို ပြန် ထုတ်ကြည့်ရာ၌ element 5 ခု ပြောက်နောက်က အားလုံးကို ဖျက်ပြစ်လိုက်သည်ကို မြင်ရပါမည်။

Vector :: emplace() function

emplace() function သည် element ကို မိမိစိတ်ကြိုက် နေရာတစ်ခုခုမှာ အတိအကျ ထည့်လိုသောအခါ သို့မဟုတ် အသစ်ထပ်ထည့် လိုတဲ့အခါမှာ အသုံးပြုပါတယ်။ emplace function သည် parameters နှစ်ခုယူပါတယ်။ ပထမတစ်ခုသည် position ဖြစ်ပြီး ဒုတိယတစ်ခုသည် element ဖြစ်ပါတယ်။

Sample Program :: emplace() function

```
#include <iostream>
#include<vector>
using namespace std;

int main()
{
    vector<int> myVec = { 10, 20, 30 ,40 ,50 };
    auto it = myVec.emplace(myVec.begin(), 100);

    cout << "All the vector elements are ";
    for (auto it = myVec.begin(); it != myVec.end(); ++it)
        cout << *it << " ";

    return 0;
}
```

အထက်ပါ program တွင်မူ 100 ဆိုသည့် element ကို ပထမ ဆုံးနေရာမှာ သွားထည့်တာပါ။ မူရင်း vector တွင် 10 သည် ပထမဆုံးဖြစ်သော်လည်း myVec.emplace(myVec.begin(), 100) ဆိုသည့် code ကြောင်းကို run ပြီးနောက်တွင် element အသစ်ဖြစ်သည့် 100 သည် ရှေ့ဆုံးသို့ ရောက် ရှိလာပြီး ပထမဆုံး element ဖြစ်သွားပါသည်။ မူလ element ဖြစ်သည့် 10 နေရာတွင် အစားထိုး လိုက်ခြင်းမျိုးမဟုတ်ပဲ အသစ်တစ်ခု ထပ်ထည့်ခြင်းသာ ဖြစ်ပါသည်။ ထိုနည်းတူ vector size လည်း အလိုလျောက် ပြောင်းလဲပေးသွားမည် ဖြစ်ပါသည်။ အောက်ပါပုံတွင် position ကို 2 တိုးပြီး element

ထပ်ထည့်ထားပါသည်။ element အသစ်ဖြစ်သည့် 99 သည် begin() + 2 နေရာသို့ ရောက်ရှိလာမည်ဖြစ်ပါသည်။

```

7   vector<int> myVec = { 10, 20, 30 ,40 ,50 };
8   auto it = myVec.emplace(myVec.begin() + 2, 99);
9
10  cout << "All the vector elements are  ";
11  for (auto it = myVec.begin(); it != myVec.end(); ++it)
12      cout << *it << " ";
13
14  return 0;
15
16 }
```

Vector ::erase

erase() function သည် vector container ထဲမှာရှိတဲ့ elements တွေကို မိမစိတ်ကြိုက် တစ်ခု ခြင်းစီသော်လည်းကောင်း၊ အပိုင်းလိုက်သော်လည်းကောင်း၊ ဖျက်လိုတဲ့အခါမှာ သုံးပါတယ်။ parameter နှစ်ခုထည့်နိုင်သလို တစ်ခုတည်းလည်း ထည့်နိုင်ပါတယ်။ ပထမ parameter သည် position ဖြစ်ပြီး ဒုတိယ parameter သည် range ပါ ဘယ်နေရာကနေ ဘယ်နေရာကို ဖျက်ချင်ပါသည် ဆိုသည့် range ပါ။

Sample Program :: erase()

```

#include <iostream>
#include <vector>
using namespace std;
int main ()
{
    vector<int> myvector;

    for (int i=1; i<=10; i++) myvector.push_back(i);
    myvector.erase (myvector.begin(),myvector.begin()+3);
    myvector.erase (myvector.begin()+5);

    cout << "myvector contains:";
    for (unsigned i=0; i<myvector.size(); ++i)
        cout << ' ' << myvector[i];
    cout << '\n';
```

```

9
10
11
12
13
14
15
16
17
18
19
20
21
    return 0;
}

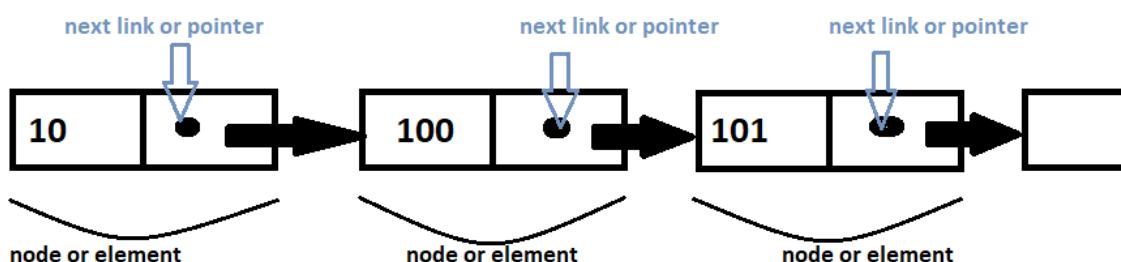
for (int i=1; i<=10; i++) myvector.push_back(i);
myvector.erase (myvector.begin(),myvector.begin()+3);
myvector.erase (myvector.begin()+5);

cout << "myvector contains:";
for (unsigned i=0; i<myvector.size(); ++i)
    cout << ' ' << myvector[i];
cout << '\n';

return 0;
}

```

အထက်ပါ program တွင် line number 11 ၏ erase function ကို parameter နှစ်ခုသုံးပြီးရေးထားပါသည်။ ပထမ parameter သည် vector ရဲ့အစ ဖြစ်ပြီး ဒုတိယ parameter သည် vector ရဲ့ ၃ ခုမြောက် နေရာထိ ဖြစ်ပါသည်။ ထို့ကြောင့် program သည် line number 11 ကို run ပြီးချိန် ၂ ခု မြောက် နေရာထိ elements များကို ဖျက်ထားမှာ ဖြစ်ပါတယ်။ ထို့ပြင် line number 12 ကို run ပြီးချိန် begin() + 5 နေရာကို ဖျက်မှာ ဖြစ်ပါတယ်။ begin သည် 4 ဖြစ်ပြီး + 5 လုပ်လိုက်လျှင် ၆ နေရာကို ဖျက်မှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် နောက်ဆုံး output တွင် 4, 5, 6, 7, 8, 10 ကိုသာ မြင်ရပြီး ၉ ကို မတွေ့ရပါ။



Linked Lists DataStructure

Linked Lists

Array သည် အရမ်းကို အသုံးဝင်သော data structure ဖြစ်သလို သူရဲ့ အသုံးဝင်ပုံတွေကိုလည်း ရှုံးပိုင်းသင်ခန်းစာများမှာ ဖော်ပြုခဲ့ပြီး ဖြစ်ပါတယ်။ Linked list သည် ပုံမှန်အားဖြင့် ကြည့်လျှင် array နှင့် ဆင်တူသော်လည်း array သည် element တစ်ခုစီမှာ data တစ်ခုတည်းသာ ရှိပါတယ်။ linked list

တွင်မှု element တစ်ခုမှာ data တစ်ခုတည်းတင် မဟုတ်ပဲ နောက် data တစ်ခုရဲ့ link ပါပါရှိနေပါသေးတယ်။ ထို့ကြောင့် linked list တွင် element တစ်ခုစီ၌ အပိုင်းနှစ်ပိုင်း ပါဝင်ပါတယ်။ linked list တွင်မှု elements တွေကို nodes တွေဟု ခေါ်ဆိုပြီးနောက် element တစ်ခုရဲ့ link ကိုတော့ next link လို့ခေါ်ဆိုသလို pointer လို့လည်း ခေါ်ဆိုပါတယ်။ next link တွင် နောက်ထပ် node တစ်ခုရဲ့ address ပါဝင်ပါတယ်။ Linked List သုံးမျိုးရှိပါတယ်။

Types of Linked List

1. Singly linked list
2. Doubly linked list
3. Circular linked list

Singly linked list

အထက်ပါပုံတွင် ပြထားသော ပုံသည် singly linked list ဖြစ်ပါတယ်။ ပထမဆုံး node ကိုတော့ Head လို့ ခေါ်ဆိုပြီး နောက်ဆုံး node ကိုတော့ Tail လို့ ခေါ်ဆိုပါတယ်။ Tail ဆိုတဲ့ နောက်ဆုံး node တွင် နောက်ထပ် ထပ်ချိန်ထားစရာ node မရှိတော့တဲ့အတွက် data and null pointer သာ ရှိပါတယ်။ linked list ထဲမှာရှိတဲ့ node တွေကို contiguous location ပုံစံနဲ့ stored လုပ်စရာ မလိုပါဘူး။ ဘာကြောင့် လည်းဆိုတော့ node တစ်ခုသည် နောက်ထပ် node တစ်ခုရဲ့ memory address ကို stored လုပ်ထားသောကြောင့် ဖြစ်သည်။ ထို့ကြောင့် node တွေ ထပ်ထည့်တာမျိုး၊ ဖျက်ထုတ်တာမျိုး စတဲ့ operations များ လုပ်ဆောင်ရာမှာ လွယ်ကူးစေပါတယ်။ array မှာဆိုရင် elements များကြားထဲမှာ နောက်ထပ် element ထပ်ထည့်နိုင်ရန် အခြားသော element တွေကို shifted လုပ်ရပါတယ်။ သို့သော် linked link တွင်မှု node တွေကို link ချိတ်ပေးရုံဖြင့် elements များ ထပ်ထည့်နိုင်ပါတယ်။

Sample Program :: Creating Singly Linked List

```
#include <iostream>
using namespace std;
class Node {
public:
    int data;
    Node* next;
};
void print(Node* n)
{
```

```

while (n != NULL) {
    cout << n->data << " ";
    n = n->next;
}
}

int main()
{
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;

    head = new Node();
    second = new Node();
    third = new Node();

    head->data = 1;
    head->next = second;
    second->data = 2;
    second->next = third;
    third->data = 3;
    third->next = NULL;

    print(head);
    return 0;
}

```

Program ရှင်းလင်းချက်

ပထမဆုံးအနေဖြင့် program ရှင်းလင်းချက်များကို နားလည်စိုင်ရန် ရှေ့ပိုင်းအခန်းများတွင် ဆွဲးနွေးထားသော Structure , class , object , arrow pointer , pointer တို့ကို သေချာ သိထားရပါမည်။

```

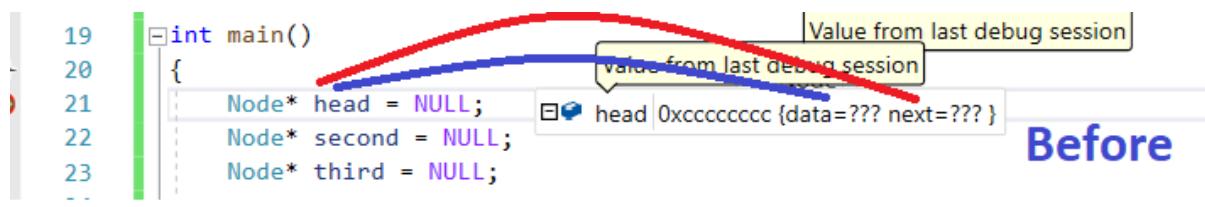
class Node {
public:
    int data;
    Node* next;
};

```

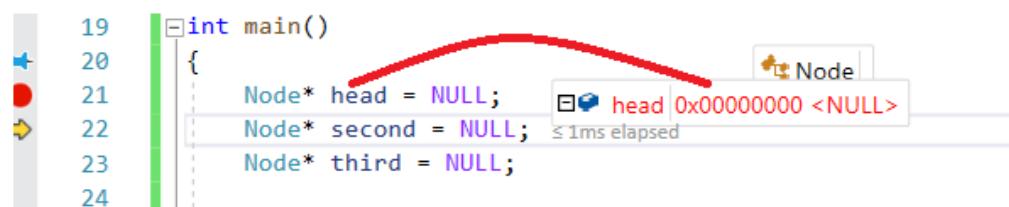
Program တွင် ပထမဆုံးအနေဖြင့် Node ဆိုသည့် class တစ်ခုကို တည်ဆောက်လိုက်ပါသည်။ ထို class ထဲတွင် Public ဖြင့် member data နှစ်ခု ပါဝင်ပါမည်။ ပထမတစ်ခုသည် data store လုပ်ရန်ဖြစ်ပြီး ဒုတိယတစ်ခုသည် address store လုပ်ရန် class type နဲ့ pointer တစ်ခုဖြစ်ပါတယ်။

```
Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;
```

Node* head ဆိုသည့် code သည် Node ဆိုတဲ့ class type နဲ့ head ဆိုတဲ့ node တစ်ခုကို စတင် ကြော်ပြာလိုက်တာပါ။ သတိပြုရန်အချက်မှာ head သည် Node ဆိုသည့် class type ဖြစ်သောကြောင့် ထို class ထဲတွင် ပါဝင်သော properties များအတိုင်း head ဆိုသည့် node တဲ့တွင် ပါဝင်နေမည် ဖြစ်သည်။ Node ဆိုသည့် class ထဲတွင် int data and Node* next ပါဝင်သောကြောင့် head ဆိုသည့် node တဲ့တွင်လည်း int data and Node* next တို့ ပါဝင်နေမည်။ Node* head = NULL; လို့ ရေးလိုက်သောကြောင့် head သည် NULL value သာ ရှိနေပါမည်။ အောက်တွင် ပုံနှစ်ပုံပြထားပါသည်။ ပထမပုံသည် Node* head=NULL ; ဆိုသည့် code ကို မ run ခင်ဖြစ်ပြီး ဒုတိယ သည် run ပြီး ဖြစ်သည်။



Before



After

WinHtut(GreenHackers)

```
head = new Node();
    second = new Node();
    third = new Node();
```

```

head = new Node();
second = new Node();
third = new Node();

```

Node::Node()
+2 overloads
Search Online

head = new Node() သည် Node ဆိတဲ့ class ထဲမှာ ရှိတဲ့ properties များကို head ဆိုသည့် object ထဲကို ထည့်လိုတဲ့အတွက် ရေးခြင်း ဖြစ်ပါသည်။ new ဆိုသည့် keyword သည် heap memory ပေါ်တွင် နေရာအသစ်တည်ဆောက်လိုတဲ့အတွက် အသုံးပြုခြင်း ဖြစ်ပါသည်။ ထို code ကို run ပြီးသောအခါနတွင် head ထဲ၌ data and next တို့ရှိနေမည် ဖြစ်သလို second and third တို့တဲ့တွင်လည်း data and next တို့ရှိနေမည် ဖြစ်ပါသည်။ အောက်ပါပုံတွင် ဖော်ပြထားသည်။

```

head = new Node();
second = new Node();
third = new Node();

```

```

head->data = 1;
head->next = second;
second->data = 2;
second->next = third;
third->data = 3;
third->next = NULL;

```

Head -> data = 1 code line သည် 1 ဆိုသည့် data ကို head ထဲမှာ ရှိသည့် data ထဲကို ထည့်ခြင်း ဖြစ်ပါသည်။ head -> next = second ; သည် second ရဲ့ memory address ကို head ထဲတွင်ရှိသော next ထဲသို့ ထည့်ခြင်းဖြစ်ပါသည်။ second and third တို့သည့်လည်း ထိန်းများ အတိုင်းပင်ဖြစ်ပြီး third တွင်မူ next ထဲသို့ NULL ကိုသာ ထည့်ပါသည်။ အဘယ်ကြောင့်ဆိုသော third မှ တစ်ဆင့်ထပ်ပြီး ချိတ်ထားရန် node မရှိတော့သောကြောင့် ဖြစ်သည်။

```

print(head);
void print(Node* n)
{
    while (n != NULL) {
        cout << n->data << " ";
        n = n->next;
    }
}

```

```

    }
}

```

နောက်ဆုံး node ဖြစ်တဲ့ third မှာ NULL ကို ထည့်ထားခဲ့ပါသည်။ ထို့ကြောင့် node အားလုံးမှာရှိတဲ့ data အားလုံးကို ထုတ်ချင်တဲ့အခါ NULL ကိုသုံးပြီး စစ်နိုင်ပါသည်။ သတိပြုရန် အချက်မှာ print(head) ဟု code ထဲတွင် ရေးထားသောကြောင့် လက်ခံသော Node* n ထဲတွင် head မှ ပါလာသော data အားလုံး ရောက်နေပါမည်။ head ထဲတွင် head တစ်ခုတည်းအတွက်တင် မဟုတ်ပဲ အခြားသော node များရဲ့ address များပါ ပါဝင်နေသည်ကို အောက်ပါပုံတွင် ဖော်ပြထားပါသည်။

```

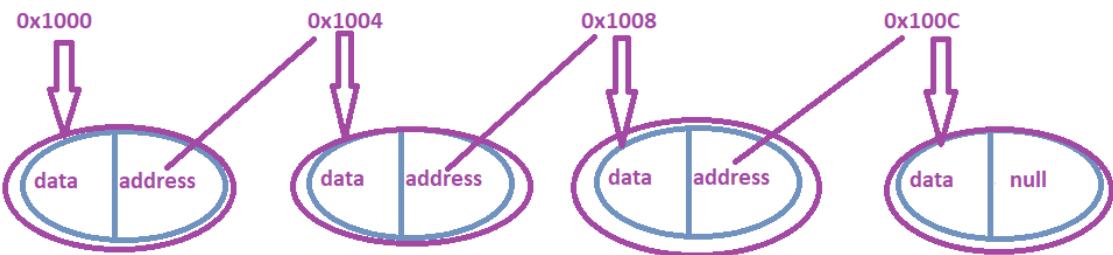
head = new Node();
second = new Node();
third = new Node();

void printList(Node* n)
{
    while (n != NULL) {
        cout << n->data << " ";
        n = n->next;
    }
}

```

The screenshot shows a memory dump with three nodes: head (0x00b2e0e0), second (0x00b2e150), and third (0x00b2e2d8). Each node has data and next pointers. The code below the dump is a printList function that prints the data of each node in a linked list.

အထက်ပါအတိုင်း ရေးပြီးထုတ်မည်ဆိုလျှင် data အားလုံး ထွက်လာမည် ဖြစ်သည်။ **n -> data** သည် ပထမဆုံး node ထဲမှာရှိတဲ့ data ကို print ထုတ်ပါသည်။ ထို့နောက် **n = n->next** ဟုရေးကာ ထို့ node ထဲတွင် ကျန်နေသော next ထဲရှိ memory address ကို n ထဲသို့ ထည့်လိုက်ပါသည်။ ထို့ကြောင့် n သည် နောက် node တစ်ခုရဲ့ memory address ကို stored လုပ်ပြီး အပေါ်သို့ပြန်တက်သွားသည်။ n သည် second node ရဲ့ memory address ရှိနေသေးသောကြောင့် null မဟုတ်သေးသည့်အတွက် အောက်က code များကို ဆက်လက် အလုပ်လုပ်သည်။ ဒုတိယအကြိမ် loop ပတ်သောအခါတွင်မူ့ n သည် second node ရဲ့ memory address ကို stored လုပ်ထားသောကြောင့် ထို့ node ထဲမှာ ရှိသော data ကို print လုပ်ပေးပါသည်။ ပြီးလျှင် ထို့ second node ရဲ့ next ထဲတွင် ရှိသော memory address ကို n ထဲသို့ပြန်ထည့်လိုက်သည်။ ထိုနည်းအတိုင်းပင် third node ထဲတွင် ရှိသော data ကို print လုပ်ပါသည်။ third node တွင် NULL သာ ပေးထားသောကြောင့် အပေါ်သို့ပြန်တက်သောအချိန်တွင် while loop အဆုံးသတ်သွားပါသည်။



Singly Linked List നും sample program നും java and python code പും ഫും ജോന്റും ഫേണ്ടും പിഡാം॥

Java

```

class LinkedList {
    Node head;

    static class Node {
        int data;
        Node next;
        Node(int d) {
            data = d;
            next = null;
        }
    }

    public static void main(String[] args) {
        LinkedList llist = new LinkedList();

        llist.head = new Node(1);
        Node second = new Node(2);
        Node third = new Node(3);

        llist.head.next = second;
    }
}

```

```
    second.next = third;  
}  
}
```

Python

class Node:

```
def __init__(self, data):  
    self.data = data  
    self.next = None
```

class LinkedList:

```
def __init__(self):  
    self.head = None
```

```
if __name__ == '__main__':
```

```
llist = LinkedList()
```

```
list.head = Node(1)  
second = Node(2)  
third = Node(3)
```

```
list.head.next = second;  
second.next = third;
```

Push new data to Singly Linked List

တည်ဆောက်ထားပြီးသား singly linked list တဲ့သို့ data အသစ်များကို ထပ်ထည့်ပါမည်။ ထိုသို့ ထည့်ရန် push ဆိုသည့် function တစ်ခုကို တည်ဆောက်ပါမည်။ ထို function တွင် parameter နှစ်ခု ပါဝင်မည်ဖြစ်ပြီး ပထမတစ်ခုသည် memory address ကိုလက်ခံရန် ဖြစ်ပြီး ဒုတိယတစ်ခုသည် အသစ် ထပ်ထည့်လိုသော data ဖြစ်ပါသည်။

```

void push(Node** head_ref, int new_data)
{
    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

```

```

21 void push(Node** head_ref, int new_data)
22 {
23
24     Node* new_node = new Node();
25
26     new_node->data = new_data;
27
28
29     new_node->next = (*head_ref);
30
31     (*head_ref) = new_node;
32 }
33 ~

```

အထက်ပါ program ၏ line number 21 တွင် Node** head_ref သည် reference(Pointer to Pointer) ဖြစ်သည်။

line number 24 တွင် node အသစ်ဆောက်ရန်အတွက် new_node ဆိုသည့် memory space တစ်ခုကို ဖန်တီးပါသည်။ line number 26 တွင် push ဆိုသည့် function ကို လုမ်းခေါ်ရာတွင် ထည့်ပေးလိုက်သည့် data ကို new_node ထဲမှ data ထဲသို့ ထည့်ပါသည်။ line number 29 တွင် node အသစ်ထပ်မထည့်ခင်ဗျာ ရှိသော push function ကို ခေါ်ရှု၍ ထည့်ပေးလိုက်သည့် head ရဲ့ memory address ကို new_node ရဲ့ next ထဲသို့ ထည့်လိုက်ပါသည်။ ထိုသို့ ထည့်လိုက်သည့်အတွက် အသစ်ဖန်တီးလိုက်သော new_node နှင့် မူရင်း head တို့မှာ ချိတ်ဆက်မိသွားပါပြီ။ သို့သော် new_node ဆိုသို့ point မထောက်ရသေးပါဘူး။ point ထောက်ထားမှုသာ data အားလုံးကို ပြန်လည်ဖော်ပြန်မည် ဖြစ်သည်။ ထို့ကြောင့် line number 31 တွင် new_node ရဲ့ address ကို (*head_ref) ထဲသို့

ပြန်ထည့်ပေးလိုက်ခြင်း ဖြစ်ပါသည်။ အောက်ပါ ပုံ ၂ ပုံ၏ memory address များ ထောက်သွားပုံကို ဖော်ပြထားပါသည်။ main function ထဲတွင် အောက်ပါအတိုင်း function call ကိုပြန်ခေါ် ပြီးနောက် print ထုတ်ကြည့်လျှင် data အသစ် ထပ်ဝင်လာသည်ကို မြင်ရပါမည်။

```
push(&head , 99);
cout<<"\nAfter Inserted A new Data "<<endl;
print(head);
```

```
20 } 21 void push(Node** head_ref, int new_data)
22 {
23     Node* new_node = new Node();
24
25     new_node->data = new_data;
26
27     // new_node->next = (*head_ref);
28
29     (*head_ref) = new_node; ⌈ 1ms elapsed
30 }
31
32 }
```

No issues found

Autos

Name	Type
*head_ref	Node *
new_node	Node *
new_node->next	Node *

Value

0x008fe448 {data=1 next=0x008fe560 {data=2 next=0x008fe3a0 {data=3 next=0x00000000 <NULL>}}}

0x008fe330 {data=99 next=0x008fe448 {data=1 next=0x008fe560 {data=2 next=0x008fe3a0 {data=3 next=0x00000000 <NULL>}}}

0x008fe448 {data=1 next=0x008fe560 {data=2 next=0x008fe3a0 {data=3 next=0x00000000 <NULL>}}}

အထက်ပါပုံတွင် ခရမ်းရောင် လေးထင့်ကွက်ဖြင့် စိုင်းပြထားသော *head_ref ရဲ့ memory address ကိုကြည့်ပါ။ မူရင်းအဟောင်း data structure ကိုသာ ထောက်ထားပြီး အသစ်ဖြစ်သည့် 99 ပါဝင်သည့် data structure ကိုမူ ထောက်ထားခြင်း မရှိသေးပါ။ ထိုပြင် new_node ရဲ့ memory address ကို ကြည့်ပါ။ data 99 ပါဝင်နေသည်ကို မြင်ရပါမည်။ ထို့ကြောင့် new_node ရဲ့ memory address ကို *head_ref ထဲသို့ ထည့်ပေးရန် လိုအပ်ပါသည်။ line number 31 တွင် new_node ရဲ့ memory address ကို *head_ref ထဲသို့ ထည့်ပြီးနောက် အောက်ပါပုံအတိုင်း မြင်ရပါမည်။

```

21 void push(Node** head_ref, int new_data)
22 {
23     Node* new_node = new Node();
24
25     new_node->data = new_data;
26
27     new_node->next = (*head_ref);
28
29     (*head_ref) = new_node;
30 }
31 } ≤ 1ms elapsed
32
33 No issues found

```

After

Name	Type	Value
*head_ref	Node *	0x008fe330 {data=99 next=0x008fe560}
new_node	Node *	0x008fe330 {data=99 next=0x008fe448}
new_node->next	Node *	0x008fe448 {data=1 next=0x008fe560}

Appending new Data To Singly Linked list

အထက်တွင် ရေးသားထားသော singly linked list တဲ့ကို data အသစ်ထပ်ထည့်ရန် ပထမဗီးဆုံး append ဆိုသည့် function တစ်ခု ရေးပါမည်။

ထို function တဲ့တွင် parameter အနေဖြင့် memory address နှင့် အသစ်ထပ်ထည့်မည့် data တို့ ပါဝင်ပါမည်။

ဒုတိယ အနေဖြင့် node အသစ်အတွက် memory block တစ်ခု ဖန်တီးပါမည်။ တတိယအနေဖြင့် ဖန်တီးပြီးသော node ရဲ့ data ထဲသို့ user မှ ထည့်ပေးလိုက်သော data ကို ထည့်ပါမည်။ node အသစ်ရဲ့ next ထဲသို့ null ကို ထည့်ထားပါမည်။

တတိယအဆင့်ကတော့ user မှ ထည့်ပေးလိုက်သော memory address သည် null ဖြစ်နေလျှင် အပေါ် step များတွင် တည်ဆောက်ခဲ့သော node သည် ပထမဆုံးနှင့် တစ်ခုတည်းသော node ဖြစ်ပြီး program ရပ်သွားပါမည်။

Memory address သည် null မဟုတ်ပဲ အခြားသော data များ ပါဝင်နေလျှင် null ရောက်သည် အထိ စစ်ဆေးရပါမည်။ linked list တစ်ခုတွင် နောက်ဆုံး node ရဲ့ next သည် မည်သည့် node ကိုမျှ point ထောက်ထားခြင်း မရှိသည့်အတွက် null ဖြစ်နေပါမည်။ ထို null ဖြစ်နေသော နေရာကို ဒုတိယအဆင့်တွင် တည်ဆောက်ထားသော node အသစ်ရဲ့ memory address ကို ချိတ်ပေးလိုက်ရပါမည်။

Sample Program :: Append

```

void append(Node** head_ref, int new_data)
{
    Node* new_node = new Node();

    Node* last = *head_ref;

    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL)
    {
        *head_ref = new_node;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = new_node;
    return;
}

```

Program ရှင်းလင်းချက်

Node* last = *head_ref;

အထက်ပါ code သည် user မှ ထည့်ပေးလိုက်သည့် memory address ကို last ထဲသို့ ထည့်ထားခြင်းဖြစ်သည်။ ထို last ထဲတွင် ယခင်သင်ခန်းစာများမှ တည်ဆောက်ထားသော linked list ကြီးတစ်ခုလုံးရဲ့ address ပါဝင်နေသည်ကို သတိပြုပါ။ အဘယ်ကြောင့်ဆိုသော append ဆိုသည့် function ကိုခေါ်မည့်အချိန်တွင် append(&head,88); ဟု ယခုအတိုင်း ခေါ်မည့်အတွက်ကြောင့် ဖြစ်သည်။ အောက်ပါပုံတွင် အသေးစိတ်ဖော်ပြထားပြီး အနီရောင် ဂိုင်းထားသည့်နေရာကို ကြည့်ပါ။

```

36     Node* new_node = new Node();
37
38     Node* last = *head_ref;
39
40     last 0x015ce208 {data=99 next=0x015cdfa0 (data=1 next=0x015ce0f0 (data=2 next=0x015ce390 {data=3 next=0x00000000 <NULL
41     new_node->data = new_data; ≤ 1ms elapsed
42     new_node->next = NULL;
43
44     if (*head_ref == NULL)
45     {
46         *head_ref = new_node;
47         return;
48     }

```

```

if (*head_ref == NULL)
{
    *head_ref = new_node;
    return;
}

```

အထက်ပါ code သည် head_ref သည် null ဖြစ်နေသလားကို စစ်တာပါ။ တကယ်လို null ဖြစ်နေခဲ့မယ်ဆိုရင်တော့ မိမိတို့တည်ဆောက်ထားသည့် node ကို point ထောက်လိုက်ရှုဖြင့် node တစ်ခုခု တည်ဆောက်ပြီးစီးသွားမည် ဖြစ်ပါသည်။ ထိုကြောင့် *head_ref = new_node ဟု ရေးထားခြင်း ဖြစ်ပါသည်။ အထက်ပါ code များသည် အလုပ်လုပ်မှာ မဟုတ်ပါဘူး။ အဘယ်ကြောင့်ဆိုသော် *head_ref သည် ပုံတွင် ပြထားသည့်အတိုင်း memory address များ ပါဝင်နေခြင်းကြောင့် ဖြစ်သည်။

```
while (last->next != NULL)
```

```
    last = last->next;
```

```
    last->next = new_node;
```

အထက်ပါ code line များသည် user မှ function call ခေါ်ရာတွင် ထည့်ပေးလိုက်သည့် linked list data structure ကြီး တစ်ခုလုံးတွင် null ရှိသည့်နေရာကို ရှာရန် ဖြစ်ပါသည်။ null ရှိသည့် နေရာသည် သေချာပေါက် tail နောက်ဆုံးမှာ ရှိမှာ ဖြစ်ပါတယ်။ ပထမဆုံးအနေဖြင့် linked list ထဲမှ ပထမဆုံး node ရဲ့ next ကို null ဖြစ်နေသလားလို့ စစ်ပါတယ်။ ပထမဆုံး node ထဲမှ next သည် second ရဲ့ memory address ကို stored လုပ်ထားတာ ဖြစ်တဲ့အတွက် null ဖြစ်နေမှာ မဟုတ်ပါဘူး။ ထိုကြောင့် ဒုတိယ code line ကို အလုပ်ဆက်လုပ်မှာ ဖြစ်ပါတယ်။ အောက်ပါပုံတွင် ကြည့်ပါ။

```

50     while (last->next != NULL)
51         last = last->next;
52     last->next = new_node;
53     return;
54 }
55
56 }
```

အထက်ပါပုံတွင် ပထမဆုံး while loop ပတ်တဲ့အချိန် last သည် 0x015cdfa0 ကို ကိုင်ထားပြီး ထိ memory address ထဲတွင် ရှိသော next သည် null ဖြစ်နေသလားဟု စစ်ခြင်း ဖြစ်ပါတယ်။ တကယ်လို့ null ဖြစ်နေတယ်ဆုံးရင် မိမိတို့ အသစ်တည်ဆောက်လိုက်တဲ့ node ကို last->next = new_node ဟုရေးပြီး ချိတ်ပေးလိုက်မှာ ဖြစ်ပါတယ်။ ထိန်းတူ 100 ဒုတိယအကြိမ် ပတ်သော အခါတွင်လည်း 0xx15ce0f0 ရဲ့ next ကို စစ်ဆေးမှာ ဖြစ်ပါတယ်။ null ကို မတွေ့ မချင်း node အားလုံး ကို အစကနေစပြီး အဆုံးထိ စစ်ဆေးမှာ ဖြစ်ပါတယ်။ null ကိုတွေ့ပြုဆိုလျှင် မိမိတို့ အသစ်တည်ဆောက်လိုက်သော node ရဲ့ link ကို ထို null နေရာတွင် ထည့်ပေးလိုက်ခြင်းဖြင့် singly linked list data structure ထဲသို့ new node ထပ်ထည့်ခြင်း အောင်မြင်သွားပါတယ်။ Program အားလုံး အပြည့်အစုံကို အောက်တွင် ဖော်ပြထားပါသည်။

```

public:
    int data;
    Node* next;
};

void printList(Node* n)
{
    while (n != NULL) {
        cout << n->data << " "
        n = n->next;
    }
}

void push(Node** head_ref, int n)
{
}
```

```
#include <iostream>
using namespace std;
```

```
class Node {
public:
```

```
int data;
Node* next;
};

void printList(Node* n)

{
    while (n != NULL) {
        //cout << n << "" << endl;
        cout << n->data << " " << endl;

        n = n->next;
    }
}

void push(Node** head_ref, int new_data)
{

    Node* new_node = new Node();

    new_node->data = new_data;

    new_node->next = (*head_ref);

    (*head_ref) = new_node;
}

void append(Node** head_ref, int new_data)
{

    Node* new_node = new Node();

    Node* last = *head_ref;

    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL)
```

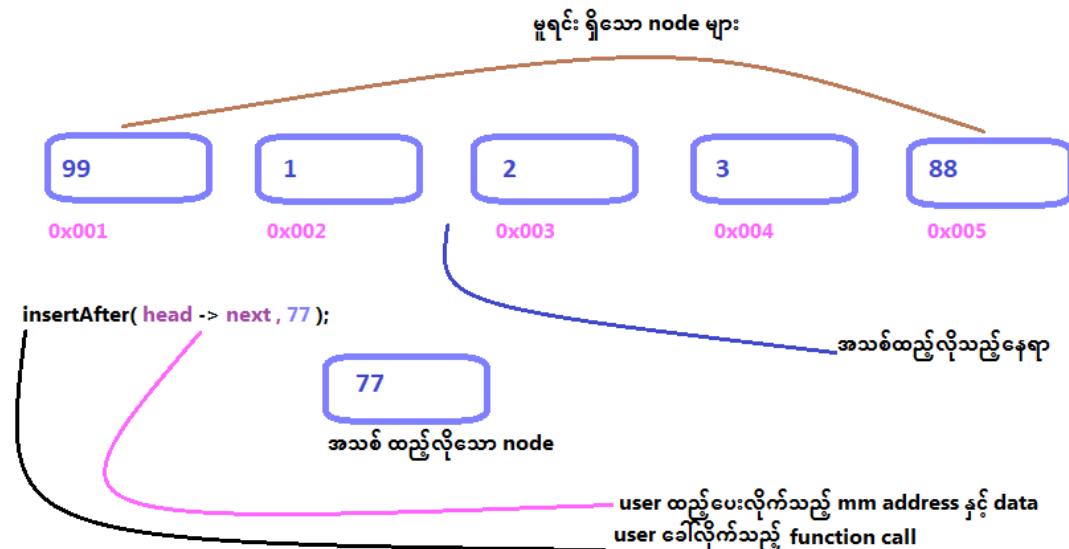
```
{  
    *head_ref = new_node;  
    return;  
}  
  
while (last->next != NULL)  
last = last->next;  
  
last->next = new_node;  
return;  
}  
  
int main()  
{  
    Node* head = NULL;  
    Node* second = NULL;  
    Node* third = NULL;  
  
    head = new Node();  
    second = new Node();  
    third = new Node();  
  
    head->data = 1;  
    head->next = second;  
    second->data = 2;  
    second->next = third;  
  
    third->data = 3;  
    third->next = NULL;  
  
    printList(head);  
    push(&head, 99);  
    cout << "\n After Insert data " << endl;  
    printList(head);  
  
    append(&head, 88);
```

```

cout << "\nAfter Appended " << endl;
printList(head);

return 0;
}

```



Insert After to Singly Linked list

အရင် သင်ခန်းစာများတွင် node အသစ်များကို ရှေ့နှင့် နောက်များတွင်သာ ထည့်ခဲ့ပါသည်။ ယခု သင်ခန်းစာတွင်မူး များတွင် node အသစ်များကို မိမိထည့်လိုသောနေရာမှာ ထည့်မှာ ဖြစ်ပါတယ်။ Append သင်ခန်းစာ ပြီးသွားသောအခါတွင် Data Structure ထဲ့ 99 - 1 - 2 - 3 - 88 စသည့် data များသည် သက်ဆိုင်ရာ node များ၏ ရှိကြေသည်။ 99 သည် push လုပ်ထားသော data ဖြစ်ပြီး 88 သည် append လုပ်ထားသော Data ဖြစ်သည်။ ယခုအခါတွင်မူး 1 ဆိုသည့် node ရဲ့ နောက်၌ နောက်ထပ် node အသစ် တစ်ခု ထပ်ထည့်မည် ဖြစ်ပါသည်။ ထိုကြောင့် ပထမဆုံး insertAfter ဆိုသည့် function တစ်ခုကို တည်ဆောက်ပါမည်။ ထို function ထဲတွင် parameter နှစ်ခု ပါဝင်ပါမည်။ ပထမ တစ်ခုသည် မိမိ ထည့်လိုသော node နေရာ၊ ရှေ့ node ၏ memory address ဖြစ်ပြီး ဒုတိယ parameter သည် မိမိ ထည့်လိုသော data ဖြစ်သည်။

Algorithm

ပထမဆုံး user ဆီက memory address and data ကို တောင်းပါမည်။

ဒုတိယအနေဖြင့် user ထည့်ပေးလိုက်သော memory address သည် null ဖြစ်နေသလား ဆိုတာကိုစစ်ဆေးပါမည်။

Null ဖြစ်နေလျှင် return ကိုရေးပြီး program ကို အဆုံးသတ်ပါမည်။

Null မဟုတ်ဘူးဆိုလျှင် data ထည့်ရန် node အသစ်တစ်ခု တည်ဆောက်မည်။

Node အသစ် တည်ဆောက်ပြီးလျှင် user ထည့်ပေးလိုက်သော data ကို new_node ရဲ့ data ထဲသို့ထည့်မည်။

User ထည့်ပေးလိုက်သော memory address ထဲက next link ကို new node ရဲ့ next link ထဲသို့ထည့်မည်။

နောက်ဆုံးအနေဖြင့် new node ရဲ့ memory address ကို user ထည့်ပေးလိုက်သော memory address ရဲ့ next link ထဲသို့ထည့်ပါမည်။

Sample Program :: insertAfter

```
void insertAfter(Node* prev_node, int new_data)
{
    if (prev_node == NULL)
    {
        cout << "the given previous node cannot be NULL";
        return;
    }
    Node* new_node = new Node();

    new_node->data = new_data;

    new_node->next = prev_node->next;

    prev_node->next = new_node;
}

void insertAfter(Node* prev_node, int new_data)
{
    Node* v_node = 0x00dee7b0 {data=1 next=0x00dee0e8 {data=2 next=0x00dee740 {data=3 next=0x00dee2a8 {data=88 next=0x00000000 <NULL>}}}};
```

အထက်ပါ ပုံသည် insertAfter function ရဲ့ prev_node ထဲသို့ ဝင်လာသည့် memory address များ၏ပုံဖြစ်ပါသည်။ သတိထားရန်အချက်မှာ data 1 သိမ်းထားသော node ရဲ့ memory address မှ စတင်

ဝင်လာခြင်း ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော် main function ထဲမှ function call ခေါ်ရာတွင်လည်း အောက်ပါအတိုင်း ခေါ်သောကြောင့် ဖြစ်သည်။

```
insertAfter(head->next, 8);
```

user ထည့်ပေးလိုက်သော node ရဲ့ memory address ဖြစ်ပါတယ်။

အထက်ပါ code line ကို run ပြီးသွားသည့်အ ချိန်တွင် 0x00dee7b0 ထဲက next ကို new_node ရဲ့ next ထဲသို့ ထည့်လိုက်သည့် အတွက် new_node ထဲမှ next သည် 0x00dee0e8 ဖြစ်သွားပါသည်။

lined list Insert After by Win Htut (Green Hackers)

အထက်ပါ ပုံသည် new_node -> next = prev_node-> next ဆိုသည့် code line ကို မ run ခင်ဗျာင့် run ပြီး memory address များ ပြောင်းလဲသွားသည့်ပုံ ဖြစ်ပါသည်။ user ထည့်ပေးလိုက်သည့် prev_node ရဲ့ memory address သည် 0x00dee7b0 ဖြစ်ပြီး ထို memory address ထဲမှာ ရှိသော next link သည် 0x00dee0e8 ဖြစ်သည်။ code line သည် prev_node ထဲမှာ ရှိတဲ့ memory address ကို new_node ရဲ့ next ထဲကို ထည့်မည်ဟု ရေးထားသောကြောင့် code line ကို run ပြီးသောအခါတွင် 0x00dee0e8 ဆိုသည့် address သည် new_node-> next ထဲသို့ ထည့်လိုက်ပါသည်။

prev_node->next = new_node;
နောက်ဆုံးမှာတော့ အထက်ပါအတိုင်း new_node ရဲ့ memory address ကို user ထည့်ပေးလိုက်သော prev_node ရဲ့ next ထဲသို့ ထည့်ပေးလိုက်ပါသည်။ အထက်ပါ code line သည် နောက်ဆုံး အရေးကြီးဆုံးဖြစ်ပြီး new node ကို data structure ထဲသို့ link ချိတ်ပေးလိုက်ခြင်း ဖြစ်ပါသည်။ အထက်ပါ code ကို run ပြီးသွားသည့် အချိန်တွင် မိမိတို့ မူရင်းရှိသော data structure ထဲ၌ user ထည့်ပေးလိုက်သော data အသစ် တိုးလာသည်ကို မြင်ရပါမည်။ program အပြည့်အစုံနှင့် output ကို အောက်တွင် ဖော်ပြထားသည်။

Sample Program :: Complete Linked list

```
#include <iostream>
using namespace std;
```

```
class Node {  
public:  
    int data;  
    Node* next;  
};  
void printList(Node* n)  
{  
    while (n != NULL) {  
        //cout << n << "" << endl;  
        cout << n->data << " ";  
  
        n = n->next;  
    }  
}  
void push(Node** head_ref, int new_data)  
{  
  
    Node* new_node = new Node();  
  
    new_node->data = new_data;  
  
  
    new_node->next = (*head_ref);  
  
    (*head_ref) = new_node;  
}  
  
void append(Node** head_ref, int new_data)  
{  
  
    Node* new_node = new Node();  
  
    Node* last = *head_ref;  
  
    new_node->data = new_data;  
    new_node->next = NULL;
```

```
if (*head_ref == NULL)
{
    *head_ref = new_node;
    return;
}

while (last->next != NULL)
last = last->next;

last->next = new_node;
return;
}

void insertAfter(Node* prev_node, int new_data)
{
    if (prev_node == NULL)
    {
        cout << "the given previous node cannot be NULL";
        return;
    }

    Node* new_node = new Node();

    new_node->data = new_data;
    new_node->next = prev_node->next;
    prev_node->next = new_node;
}

int main()
{
    Node* head = NULL;
    Node* second = NULL;
    Node* third = NULL;
```

```
head = new Node();
second = new Node();
third = new Node();

head->data = 1;
head->next = second;
second->data = 2;
second->next = third;

third->data = 3;
third->next = NULL;

printList(head);
push(&head, 99);
cout << "\n After Insert data " << endl;
printList(head);

append(&head, 88);
cout << "\nAfter Appended " << endl;
printList(head);

insertAfter(head->next, 8);
cout << "\nAfter insert " << endl;
printList(head);

insertAfter(second->next, 77);
cout << "\nAfter insert " << endl;
printList(head);

return 0;
}
```

```

void insertAfter(Node* prev_node, int new_data)
{
    if (prev_node == NULL)
    {
        cout << "the given previous node cannot be NULL";
        return;
    }

    Node* new_node = new Node();
    new_node->data = new_data;
    new_node->next = prev_node->next;
    prev_node->next = new_node;
}

int main()
{
    Node* head = NULL;
}

```

Microsoft Visual Studio Debug Console

```

1 2 3
After Insert data
99 1 2 3
After Appended
99 1 2 3 88
After insert
99 1 8 2 3 88
After insert
99 1 8 2 3 77 88
C:\Users\winht\source\repos\Project3\Debug\Project3.exe (...
To automatically close the console when debugging stops,
le when debugging stops.
Press any key to close this window . . .

```

Searching in Single Linked List

ယခု သင်ခန်းစာမျာတော့ Singly Linked list ထဲမှာ ရှိတဲ့ မိမိတို့ ရှာချင်တဲ့ data ကို ရှာသွားမည့် function တစ်ခု ရေးသား သွားမှာ ဖြစ်ပါတယ်။

- ပထမဆုံး အနေဖြင့် မိမိတို့ ရှာလိုသည့် linked list ခဲ့ head memory address ကို ယူပါမယ်
- ဒုတိယ အနေဖြင့် ထို head memory address သည် Null ဖြစ်နေသလားကို စစ်ဆေးပါမည်
- Null ဖြစ်နေလျှင် data မရှိကြောင်း ပြန်လည် ဖော်ပြုမည် ဖြစ်ပြီး null ဖြစ်မနေလျှင် ထို memory address တဲ့မှ data နှင့် မိမိတို့ ရှာလိုသည့် data တူနေသလား စစ်ဆေးပါမယ်။
- အကယ်၍ အထက်ပါ အဆင့်ပြီးနောက် မိမိတို့ ရှာလိုသည့် data မတွေ့ သေးဘူးဆိုလျှင် နောက်ထပ် memory address တစ်ခုထဲမှ data ကို ထပ်မံ စစ်ဆေးမှာ ဖြစ်ပါတယ်။
အောက်မှာတော့ search function အတွက် ရေးထားသည့် program အသေးစိတ်ကို ဖော်ပြု ထားပါတယ်။

```

bool search(Node* head , int key){
    Node* current_node = head;
    while (current_node != NULL){
        if( current_node->data == key){
            return true;
        }
        current_node = current_node->next;
    }
    return false;
}

```

<https://github.com/WinHtut/DataStructuresImplementedByWinHtut/commit/68c320b5b13ff12abe4a37d2142fd071edf2a8f0>

Singly Linked List Delete

ယခု သင်ခန်းစာမျာတော့ စာရေးသူတို့ရဲ့ SLL ထဲမှာ ရှိသည့် nodes များထဲက မိမိတို့ ကြိုက်နှစ်သက်တဲ့ node ကို ဖျက်မှာ ဖြစ်ပါတယ်။ တို့သို့ ဖျက်ဖို့ဆိုလျှင် အချက် နှစ်ချက်ကို ထည့်သွင်းစဉ်းစား ရပါမယ် ပထမ အချက်ကတော့ မိမိတို့ ဖျက်လိုသည့် node သည် head ဖြစ်နေလျှင်နှင့် node သည် နောက်မှာ ဖြစ်နေလျှင်ဆိုတဲ့ အချက်ပါ။

မိမိတို့ ဖျက်လိုသည့် node သည် head ဖြစ်နေလျှင်တော့ head ကို delete လုပ်ရုံသာ ဖြစ်သည် ထိုသို့ မဖျက်ခင်မှာတော့ head ထဲမှာ ရှိတဲ့ next ကို head အဖြစ် ချုပ်းပေးရမည်။ ပြီးမှသာ head ကို delete လုပ်နိုင်မည်ဖြစ်သည်။

```

70     void deleteNode(Node** head_ref, int key)
71     {
72         Node* prev = NULL; // for flag where the key was before found
73
74         Node* temp = *head_ref; // we dont touch directly to *head_ref
75         if (temp != NULL && temp->data == key)
76         {
77             *head_ref = temp->next; // changing head if delete key at head
78             delete temp;           // deleting temp we dont need anymore
79             return;               // terminate the function
80         }

```

အထက်ပါ program တွင် line 75 မှာ key သည် head ဖြစ်နေခဲ့လျှင် temp ထဲမှ ရှိတဲ့ next ကို *head_ref အဖြစ် သတ်မှတ်လိုက်ပါတယ်။ ယခု နေရာတွင် temp သည် *head_ref ဖြစ်နေသည် အဘယ်ကြောင့် ဆိုသော မိမိတို့ အနေဖြင့် data များကို ရှာလိုသည့် *head_ref ကို တိုက်ရှိက် မကိုင်တွယ်တော့ပဲ temp ထဲသို့ အော ထည့်ထားပြီး temp ကိုသာ ကိုင်တွယ်မည် ဖြစ်သည်။

Line 72 မှာ prev ဆိုသည့် node ကို NULL အဖြစ်ကြော် ပေးထားပါတယ် ထို prev ရဲ့ အသုံးပြုမည့် နေရာမှာ မိမိတို့ ဖျက်လိုသည့် data သည် head မှာ မဟုတ်တော့ပဲ နောက်ပိုင်းတွင် ရှိသော အခါ့ဗျာ ဖျက်လိုသော node မတိုင်ခင်၍ ရှိသည့် node အား မှတ်ထားရန် ဖြစ်သည်။ ဥပမာ 1 – 2 – 3 ဆိုသည့် Node တွင် 2 ကို ဖျက်မည် ဆိုပါစို့ 1 ဆိုသည့် node အား prev ထို့သို့ ထည့်ထားပြီး မှတ်ထားရန် ဖြစ်သည် ထို့မှာသာ 2 ကို ဖျက်ပြီးသော အခါ့ဗျာ 1 နှင့် 3 တို့ပြန်ဆက်ရန် ဖြစ်သည်။

```

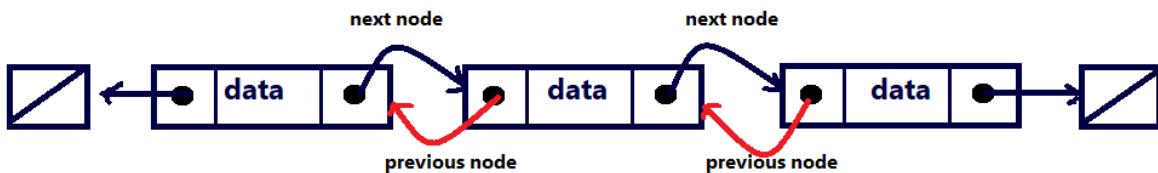
81     else
82     {
83         while (temp != NULL && temp->data != key)
84         {
85             prev = temp; // flag the previous node to be connect after delete
86             temp = temp->next; // changing the node to find key
87         }
88         // If key was not present in linked list
89         if (temp == NULL)
90             return;
91         // if temp not equal to NULL we have to connect 1 and 3 nodes
92         prev->next = temp->next;
93         delete temp;
94     }
95 }

```

Doubly Linked List

Singly Linked list ရဲ့ node တစ်ခုထဲတွင် next link ဆိုသည့် pointer တစ်ခုသာ ပါဝင်ပြီး Doubly Linked List ရဲ့ node ထဲတွင်မူ previous link and next link ဆိုသည့် pointer နှစ်ခု ပါဝင်ပါသည်။ ထို့ကြောင့် doubly linked list (DLL) ထဲတွင် field သုံးခု ပါဝင်ပါသည်။ ပထမဆုံး field သည် previous node ရဲ့ pointer ဖြစ်ပြီး ဒုတိယ field သည် data ဖြစ်သည် တတိယ field သည် next node ရဲ့ pointer ဖြစ်ပါသည်။

Doubly Linked List



WinHtut (Green Hackers Team)

Singly linked list (SLL) တွင် data structure တစ်ခုကို ဖန်တီးရန် class တစ်ခုထဲ၌ data and next pointer တို့ကို ရေးခဲ့ကြသည်။ ရေးသားပုံမှာအောက်ပါအတိုင်း ဖြစ်သည်။

```

class Node {
    public:
        int data;
        Node* next;
};
  
```

Doubly Linked list တွင်မူ previous node ကို ပြန်ထောက်ထားသည့် previous pointer နှင့် next node ကို ထောက်ထားမည့် next pointer တို့ပါရှိရမှာ ဖြစ်သည့်အတွက် program ကို အောက်ပါအတိုင်း ရေးသားနိုင်သည်။

```

class Node {
    public:
        int data;
        Node* next;
};
```

```
Node* prev;
};
```

အထက်ပါအတိုင်း class တစ်ခုရေးသားပြီးလျှင် main function ထဲ၌ head ဆိုသည့် node တစ်ခုကို စတင်တည်ဆောက်ပါမည်။ ပထမဆုံးအနေဖြင့် empty node တစ်ခု တည်ဆောက်မှာဖြစ်သည့်အတွက် NULL ကိုသာ ထို head ထဲသို့ ထည့်ထားပါမည်။ ပြီးလျှင် node များ ထပ်ထည့်ရန် append ဆိုသည့် function တစ်ခု စဆောက်ပါမည်။

Algorithm For Append Function

User ထည့်ပေးလိုက်သော data ကို ထည့်ရန် node အသစ်တစ်ခု ဆောက်ပါမည်။

User ထည့်ပေးလိုက်သော data ကို node အသစ်ရဲ့ data ထဲသို့ ထည့်ပါမည်။

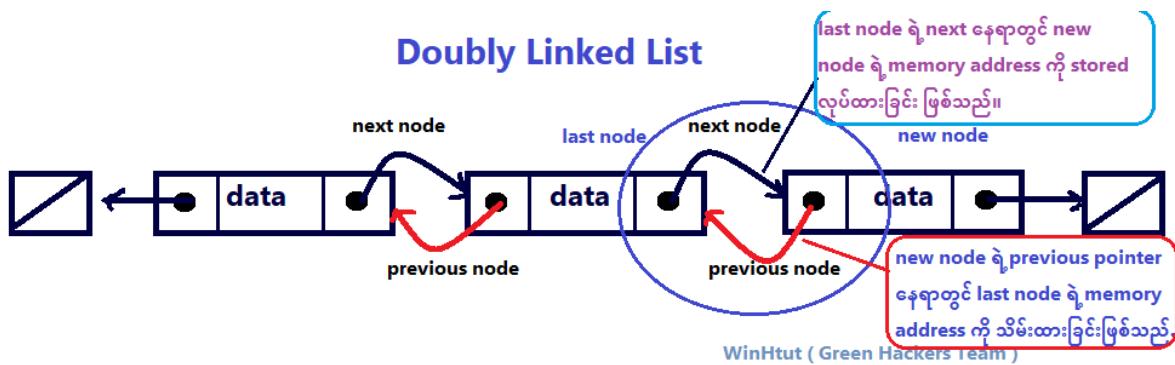
Node အသစ်ရဲ့ next ထဲသို့ null ကို ထည့်ထားပါမည်။ append သည် နောက်ကနေ ထပ်တိုးခြင်း ဖြစ်သည့်အတွက် သူ့နောက်တွင် မည့်သည့်node မျှမရှိတော့ဟု ယူဆပြီး null ကို ထည့်ခြင်း ဖြစ်သည်။

အကယ်၍ user ထည့်ပေးလိုက်သော memory address သည် null ဖြစ်နေလျှင် အထက်ပါ အဆင့်များမှ တည်ဆောက်ခဲ့သော new node ကို head အဖြစ် ပြုလုပ်ပါမည်။

အကယ်၍ user ထည့်ပေးလိုက်သော memory address သည် null မဖြစ်ပဲ node များ ရှိနေလျှင် ထို node အားလုံး ထဲမှ နောက်ဆုံး node ရဲ့ null နေရာတွင် မိမိတို့ new node ကို ချိတ်ဆက်ပေးရမှာ ဖြစ်သည့်အတွက် user ထည့်ပေးလိုက်သော memory address ကို last ဆိုသည့် node တစ်ခု တည်ဆောက်ပြီး ထည့်ပါမည်။

နောက်ဆုံး node ရဲ့ next နေရာတွင် မိမိတို့ တည်ဆောက်လိုက်သော new node ရဲ့ memory address ကို ချိတ်ဆက်ပါမည်။

Doubly linked list ဖြစ်သည့်အတွက် မူရင်းရှိသည့် data structure ထဲမှ last node ရဲ့ memory address ကို new node ရဲ့ previous ထဲသို့ ထည့်ပါမည်။



Sample Program :: Append new node to Doubly Linked list

```
#include <iostream>
using namespace std;
class Node
{
public:
    int data;
    Node* next;
    Node* prev;
};

void append(Node** head_ref, int new_data)
{
    Node* new_node = new Node();

    new_node->data = new_data;

    new_node->next = NULL;

    if (*head_ref == NULL)
    {
        new_node->prev = NULL;
        *head_ref = new_node;
        return;
    }
    Node* last = *head_ref;
    while (last->next != NULL)
        last = last->next;
}
```

```

last->next = new_node;
new_node->prev = last;
return;
}
int main()
{
    Node* head = NULL;
    append(&head, 1);

    return 0;
}

```

Pointer To Pointer

C Programming မှာ Pointer တစ်ခုသည် ပုံမှန်အားဖြင့် variable တစ်ခုရဲ့ memory address တစ်ခုကို store လုပ်ဖို့ အသုံးပြုကြောင်း သိပြီး ဖြစ်ပါတယ်။ Pointer to pointer ကို ကြော်တဲ့ အခါမှာ int **ptr; ဟု ကြော်ပါတယ်။ Pointer to Pointer သည် pointer တစ်ခုကနေပြီး အခြား pointer တစ်ခုသို့ ပြန်ထောက် ထားခြင်း ဖြစ်ပါတယ်။ ထို့ကြောင့် ပထမ pointer နှင့် ဒုတိယ pointer နှစ်ခု ရှိသည်ဟု ဆိုကြပါစို့ ပထမ pointer သည် ပုံမှန် variable တစ်ခုရဲ့ memory address ကို ထောက်ထားပြီး ဒုတိယ pointer သည် ပထမ pointer ရဲ့ memory address ကို ထောက်ထားပါတယ် ထို့ကြောင့် pointer to pointer ဟု ခေါ်ဆိုပြီး double pointer ဟူလည်း ခေါ်ကြပါတယ်။

Pointer To Pointer

C Programming မှာ Pointer တစ်ခုသည် ပုံမှန်အားဖြင့် variable တစ်ခုရဲ့ memory address တစ်ခုကို store လုပ်ဖို့ အသုံးပြုကြောင်း သိပြီး ဖြစ်ပါတယ်။ Pointer to pointer ကို ကြော်တဲ့ အခါမှာ int **ptr; ဟု ကြော်ပါတယ်။ Pointer to Pointer သည် pointer

တစ်ခုကနေပြီး အခြား pointer တစ်ခုသို့ ပြန်ထောက် ထားခြင်း ဖြစ်ပါတယ်။ ထို့ကြောင့် ပထမ pointer နှင့် ဒုတိယ pointer နှစ်ခု ရှိသည်ဟု ဆိုကြပါစို့ ပထမ pointer သည် ပုံမှန် variable တစ်ခုရဲ့ memory address ကို ထောက်ထားပြီး ဒုတိယ pointer သည် ပထမ pointer ရဲ့ memory address ကို ထောက်ထားပါတယ် ထို့ကြောင့် pointer to pointer ဟု ခေါ်ဆိုပြီး double pointer ဟူလည်း ခေါ်ကြပါတယ်။

```
#include <stdio.h>

int main () {

    int normalVariable;
    int *firstPointer;
    int **SecondPointer ; // double pointer

    normalVariable = 95;

    /* getting the address of normalVariable and this is first pointer */
    firstPointer = &normalVariable;

    /* getting the address of first pointer using address of operator & */
    SecondPointer = &firstPointer;

    printf("1:Address and Value %x :-> %d\n", &normalVariable,normalVariable );
    printf("2:Address of First pointer :-> %x\n",&firstPointer);
    printf("3:Second Pointer Pointing To :-> %x\n",SecondPointer);

    printf("5:Address of Second Pointer :-> %x",&SecondPointer);

    return 0;
}
```

```
1:Address and Value 3ebffccc :-> 95
2:Address of First pointer :-> 3ebffcc0
3:Second Pointer Pointing To :-> 3ebffcc0
5:Address of Second Pointer :-> 3ebffcb8
Process finished with exit code 0
```

အထက်ပါ program အား `print` ထုတ်ကြည့်သည့် အခါ ပထမ `output` တွင် `normal variable` နှင့် သူရဲ့ `memory address` ကို မြင်ရပါမည်။ ဒုတိယ `output` ကို ကြည့်သော အခါ First pointer ရဲ့ `memory address` ကို မြင်ရမည် ဖြစ်ပြီး `second pointer` သည် မည်သည့် အရာကို point ထောက်ထားသနည်း ကြည့်သော အခါတွင်လည်း ပထမ `pointer` ရဲ့ `memory address` ကိုသာ `pointer` ထောက် ထားကြောင်း မြင်ရမည် ဖြစ်သည် ထို့ကြောင့် `pointer to pointer` သည် `pointer` တစ်ခုမှ အခြား `pointer` တစ်ခုသို့ ထောက်ထားခြင်းသာ ဖြစ်သည်။ ပိုမို ရှင်းလင်းသွားစေရန်လည်း `second pointer` ရဲ့ `memory address` ကိုပါ ထုတ်ပြထားသည်။

Templates

Template သည် C++ programming တွင် ယနေ့ အချိန်ထိ အစွမ်းထက်သည့် tool တစ်ခုအနေဖြင့် တည်ရှိနေပါသည်။ ပုံမှန်အားဖြင့် မတူညီတဲ့ data types များ function တစ်ခုအတွင်းသို့ ဖြတ်သွားဖို့ အတွက် မတူညီတဲ့ functions များစွာ ရေးရမှာ ဖြစ်ပါတယ်။ ဥပမာ စရေးသူတို့ အနေဖြင့် မတူညီတဲ့ data types များကို sorting စီရမည်ဆိုပါစို့ `sort()` လုပ်မည့် မတူညီသော functions များ များစွာ လိုအပ်နေမှာ ဖြစ်ပါတယ်။ သို့သော C++ တွင်မူ Template ကို support ပေးထားပါသည် ထို template သည် data types မတူညီသည့် parameters များ ဖြတ်သွားဖို့ အတွက် မတူညီသည့် functions များစွာ ရေးစရာ မလိုတော့ပဲ template ကို အသုံးပြုကာ function ကိုရေးခြင်းဖြင့် တစ်ကြိမ် တစ်ခါတည်းသာ ရေးရန် လိုအပ်ပါတော့သည်။

```
#include "iostream"
template <typename T>
T myTem(T x , T y) {
    return (x>y) ? x: y;
}

int main() {

    std::cout<<myTem<int>(3,7)<<std::endl;
    std::cout<<myTem<char>('z','a')<<std::endl;
    return 0;
}
```

Output

7

Z

အထက်ပါ program မှာတော့ x တန်ဘိုးသည် y ထက် ကြီးနေသည် ဆိုလျင် x တန်ဘိုးကို return ပြန်ပေးမှာ ဖြစ်ပြီး x တန်ဘိုးသည် y ထက် cယ်နေသည် ဆိုလျင် y တန်ဘိုးကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။ အထက်ပါ program တွင် myTem ဆိုသည့် template function တဲ့သို့ data များ ဖြတ်ပါသည် ပထမ တစ်ခါ ဖြတ်သော data များသည် int types များဖြစ်ပြီး ဒုတိယ တစ်ခါ ဖြတ်သော data များသည် character types များဖြစ်ပါသည် သို့သော် ဖြစ်လာမည့် data types များ အပေါ် မူတည်ပြီး functions များစွာကို ရေးသားရန် မလိုပဲ template ကိုသုံးကာ တစ်ခုတည်းရေးသား ထားနိုင်သည်။ ထို့ကြောင့် template function ကို သုံးခြင်းဖြင့် parameter များသည် မည်သည့် data types များဖြင့် ဖြတ်လာသည် ဖြစ်စေ template function သို့ ရောက်သည့် အခါ ထို့ဖြတ်လာသော data types များ အတိုင်းသာ လျှင် အစားထိုး ဝင်ရောက်သွားမည် ဖြစ်သည်။

```

1:Address and Value 3ebffccc :-> 95
2:Address of First pointer :-> 3ebffcc0
3:Second Pointer Pointing To :-> 3ebffcc0
5:Address of Second Pointer :-> 3ebffcb8
Process finished with exit code 0

```

အထက်ပါ program အား print ထုတ်ကြည့်သည့် အခါ ပထမ output တွင် normal variable နှင့် သူရဲ့ memory address ကို မြင်ရပါမည်။ ဒုတိယ output ကို ကြည့်သော အခါ First pointer ရဲ့ memory address ကို မြင်ရမည် ဖြစ်ပြီး second pointer သည် မည်သည့် အရာကို point ထောက်ထားသနည်း ကြည့်သော အခါတွင်လည်း ပထမ pointer ရဲ့ memory address ကိုသာ pointer ထောက် ထားကြောင်း မြင်ရမည် ဖြစ်သည် ထို့ကြောင့် pointer to pointer သည် pointer တစ်ခုမှ အခြား pointer တစ်ခုသို့ ထောက်ထားခြင်းသာ ဖြစ်သည်။ ပိုမို ရှင်းလင်းသွားစေရန်လည်း second pointer ရဲ့ memory address ကိုပါ ထုတ်ပြထားသည်။

Template သည် C++ programming တွင် ယနေ့ အချိန်ထိ အစွမ်းထက်သည့် tool တစ်ခုအနေဖြင့် တည်ရှိနေပါသည်။ ပုံမှန်အားဖြင့် မတူညီတဲ့ data types များ function တစ်ခုအတွင်းသို့ဖြတ်သွားဖို့ အတွက် မတူညီတဲ့ functions များစွာ ရေးရမှာ ဖြစ်ပါတယ်။ ဥပမာ စေရေးသူတို့ အနေဖြင့် မတူညီတဲ့ data types များကို sorting စီရမည့်ဆိုပါဖို့ sort() လုပ်မည့် မတူညီသော functions များ များစွာ လိုအပ်နေမှာ ဖြစ်ပါတယ်။ သို့သော် C++ တွင်မူ Template ကို support ပေးထားပါသည် ထို template သည် data types မတူညီသည့် parameters များ ဖြတ်သွားဖို့ အတွက် မတူညီသည့် functions များစွာ ရေးစရာ မလိုတော့ပဲ template ကို အသုံးပြုကာ function ကိုရေးခြင်းဖြင့် တစ်ကြိမ် တစ်ခါတည်းသာ ရေးရန် လိုအပ်ပါတော့သည်။

Pointer To Pointer

C Programming မှာ Pointer တစ်ခုသည် ပုံမှန်အားဖြင့် variable တစ်ခုရဲ့ memory address တစ်ခုကို store လုပ်ဖို့ အသုံးပြုကြောင်း သိပြီး ဖြစ်ပါတယ်။ Pointer to pointer ကို ကြော်တဲ့ အခါမှာ int **ptr; ဟု ကြော်ပါတယ်။ Pointer to Pointer သည် pointer တစ်ခုကနေပြီး အခြား pointer တစ်ခုသို့ ပြန်ထောက် ထားခြင်း ဖြစ်ပါတယ်။ ထို့ကြောင့် ပထမ pointer နှင့် ဒုတိယ pointer နှစ်ခု ရှိသည်ဟု ဆိုကြပါဖို့ ပထမ pointer သည် ပုံမှန် variable တစ်ခုရဲ့ memory address ကို ထောက်ထားပြီး ဒုတိယ pointer သည် ပထမ pointer ရဲ့ memory address ကို ထောက်ထားပါတယ် ထို့ကြောင့် pointer to pointer ဟု ခေါ်ဆိုပြီး double pointer ဟုလည်း ခေါ်ကြပါတယ်။

```
#include <stdio.h>

int main () {

    int normalVariable;
    int *firstPointer;
    int **SecondPointer ; // double pointer

    normalVariable = 95;

    /* getting the address of normalVariable and this is first pointer */
    firstPointer = &normalVariable;
```

```

/* getting the address of first pointer using address of operator & */
SecondPointer = &firstPointer;

printf("1:Address and Value %x :-> %d\n", &normalVariable,normalVariable );
printf("2:Address of First pointer :-> %x\n",&firstPointer);
printf("3:Second Pointer Pointing To :-> %x\n",SecondPointer);

printf("5:Address of Second Pointer :-> %x",&SecondPointer);

return 0;
}

```

```

1:Address and Value 3ebffccc :-> 95
2:Address of First pointer :-> 3ebffcc0
3:Second Pointer Pointing To :-> 3ebffcc0
5:Address of Second Pointer :-> 3ebffcb8
Process finished with exit code 0

```

အထက်ပါ program အား print ထုတ်ကြည့်သည့် အခါ ပထမ output တွင် normal variable နှင့် သူရဲ့ memory address ကို မြင်ရပါမည်။ ဒုတိယ output ကို ကြည့်သော အခါ First pointer ရဲ့ memory address ကို မြင်ရမည် ဖြစ်ပြီး second pointer သည် မည်သည့် အရာကို point ထောက်ထားသနည်း ကြည့်သော အခါတွင်လည်း ပထမ pointer ရဲ့ memory address ကိုသာ pointer ထောက် ထားကြောင်း မြင်ရမည် ဖြစ်သည် ထို့ကြောင့် pointer to pointer သည် pointer တစ်ခုမှ အခြား pointer တစ်ခုသို့ ထောက်ထားခြင်းသာ ဖြစ်သည်။ ပိုမို ရှင်းလင်းသွားစေရန်လည်း second pointer ရဲ့ memory address ကိုပါ ထုတ်ပြထားသည်။

Templates

Template သည် C++ programming တွင် ယနေ့ အချိန်ထိ အစွမ်းထက်သည့် tool တစ်ခုအနေဖြင့် တည်ရှိနေပါသည်။ ပုံမှန်အားဖြင့် မတူညီတဲ့ data types များ function တစ်ခုအတွင်းသို့ ဖြတ်သွားဖို့ အတွက် မတူညီတဲ့ functions များစွာ ရေးရမှာ ဖြစ်ပါတယ်။ ဥပမာ စရေးသူတို့ အနေဖြင့် မတူညီတဲ့ data types များကို sorting စီရမည့်လိုပါစို့ sort() လုပ်မည့် မတူညီသော functions များ များစွာ လိုအပ်နေမှာ ဖြစ်ပါတယ်။ သို့သော် C++ တွင်မူ Template ကို

support ပေးထားပါသည် ထို template သည် data types မတူညီသည့် parameters များဖြစ်သွားဖို့ အတွက် မတူညီသည့် functions များစွာ ရေးစရာ မလိုတော့ပဲ template ကို အသုံးပြုကာ function ကိုရေးခြင်းဖြင့် တစ်ကြိမ် တစ်ခါတည်းသာ ရေးရန် လိုအပ်ပါတော့သည်။

```
#include "iostream"
template <typename T>
T myTem(T x , T y) {
    return (x>y)? x: y;
}

int main() {
    std::cout<<myTem<int>(3,7)<<std::endl;
    std::cout<<myTem<char>('z','a')<<std::endl;
    return 0;
}
```

Output

7

z

အထက်ပါ program မှာတော့ x တန်ဘိုးသည် y ထက် ကြီးနေသည် ဆိုလျှင် x တန်ဘိုးကို return ပြန်ပေးမှာ ဖြစ်ပြီး x တန်ဘိုးသည် y ထက် ငယ်နေသည် ဆိုလျှင် y တန်ဘိုးကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။ အထက်ပါ program တွင် myTem ဆိုသည့် template function ထဲသို့ data များ ဖြတ်ပါသည် ပထမ တစ်ခါ ဖြတ်သော data များသည် int types များဖြစ်ပြီး ဒုတိယ တစ်ခါ ဖြတ်သော data များသည် character types များဖြစ်ပါသည် သို့သော် ဖြစ်လာမည့် data types များအပေါ် မူတည်ပြီး functions များစွာကို ရေးသားရန် မလိုပဲ template ကိုသုံးကာ တစ်ခုတည်းရေးသား ထားနိုင်သည်။ ထို့ကြောင့် template function ကို သုံးခြင်းဖြင့် parameter များသည် မည်သည့် data types များဖြင့် ဖြတ်လာသည် ဖြစ်စေ template function သို့ ရောက်သည့် အခါ ထို ဖြတ်လာသော data types များ အတိုင်းသာ လျှင် အစားထိုး ဝင်ရောက်သွားမည် ဖြစ်သည်။

အထက်ပါ program မှာတော့ x တန်ဘိုးသည် y ထက် ကြီးနေသည် ဆိုလျှင် x တန်ဘိုးကို return ပြန်ပေးမှာ ဖြစ်ပြီး x တန်ဘိုးသည် y ထက် ငယ်နေသည် ဆိုလျှင် y တန်ဘိုးကို return ပြန်

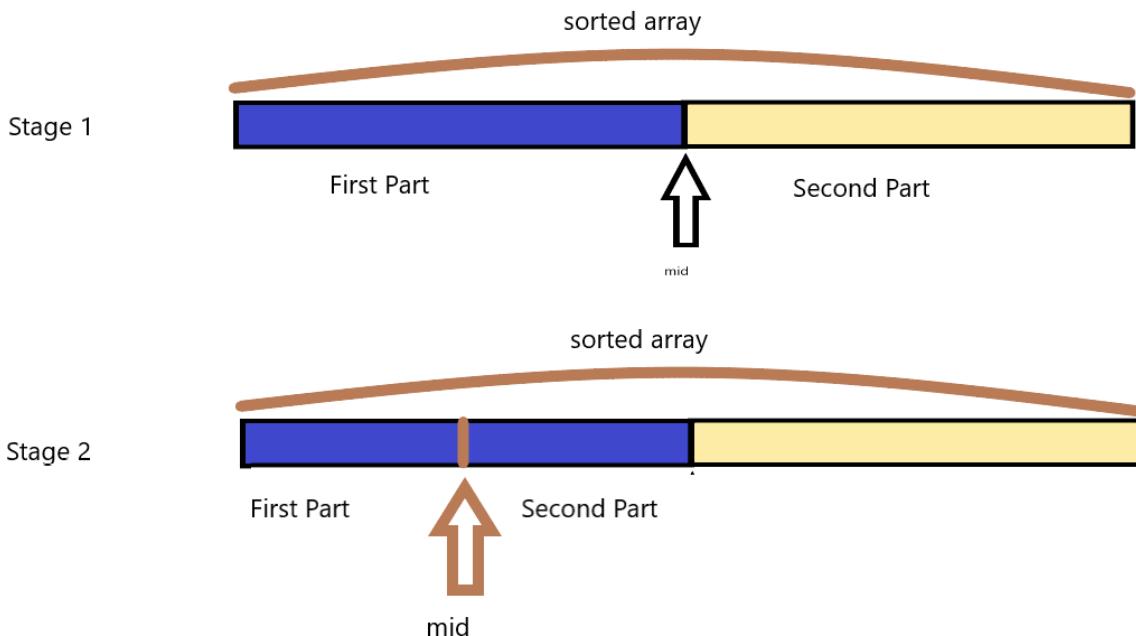
ပေးမှာ ဖြစ်ပါတယ်။ အထက်ပါ program တွင် myTem ဆိုသည့် template function ထဲသို့ data များ ဖြတ်ပါသည် ပထမ တစ်ခါ ဖြတ်သော data များသည် int types များဖြစ်ပြီး ဒုတိယ တစ်ခါ ဖြတ်သော data များသည် character types များဖြစ်ပါသည် သို့သော် ဖြစ်လာမည့် data types များ အပေါ် မူတည်ပြီး functions များစွာကို ရေးသားရန် မလိုပဲ template ကိုသုံးကာ တစ်ခုတည်း ရေးသား ထားနိုင်သည်။ ထို့ကြောင့် template function ကို သုံးခြင်းဖြင့် parameter များသည် မည်သည့် data types များဖြင့် ဖြတ်လာသည် ဖြစ်စေ template function သို့ ရောက်သည့် အခါ ထို့ ဖြတ်လာသော data types များ အတိုင်းသာ လျင် အစားထိုး ဝင်ရောက်သွားမည် ဖြစ်သည်။

Binary Search

Computer Science နယ်ပယ်တွင် Binary search ကို half-interval search ဟုလည်း ခေါ်ဆို ကြပါတယ်။ Binary search algorithm ကိုတော့ sorted လုပ်ထားသည့် array ထဲမှ မိမိတို့ လိုချင်သည့် value တစ်ခုကို ရှာလိုတဲ့ အခါမှာ အသုံးပြုပါတယ်။

1. ပထမဆုံး အနေဖြင့် sorted လုပ်ထားသည့် array ရဲ့ အလယ်ကိန်း(mid) ကိုရှာပါတယ်။
2. ဒုတိယ အနေဖြင့် sorted လုပ်ထားသည့် array တစ်ခုကို အလယ်ကနေ တစ်ဆင့် အပိုင်း နှစ်ပိုင်း ခွဲချု လိုက်ပါတယ်။
3. ထိုကဲ့သို့ ခွဲချုပြီး အလယ်ကိန်း(mid) နှင့် မိမိတို့ ရှာလိုသည့် ကိန်းကို အခြေနေ သုံးခုဖြင့် စစ်ဆေးပါတယ်။
4. ပထမ အဆင့်ကတော့ ထို အလယ်ကိန်း(mid) နှင့် မိမိတို့ ရှာလိုသည့် ကိန်းတို့ တူညီသလား စစ်ဆေးခြင်း ဖြစ်ပါတယ် တူညီနေလျှင် program ရဲ့ ရည်ရွယ်ချက် ပြီးမြောက်သွားပြီး အခြားသော instruction များကို ဆက်လက် လုပ်ဆောင်မှာ ဖြစ်ပါတယ်။
5. ဒုတိယ အဆင့်ကတော့ အလယ်ကိန်း(mid) နှင့် မိမိတို့ ရှာလိုသော ကိန်း(key) အား ကြိုးသလား ငယ်သလား စစ်ဆေးပါတယ်။ (mid < key) or (mid > key)
6. အကယ်၍ ကြိုးနေလျှင် သို့မဟုတ် ငယ်နေလျှင် အထက်ပါ အဆင့် 1 to 5 အတိုင်း ဆက်လက် လုပ်ဆောင်ပါတယ်။

အနှစ်ချုပ် အနေဖြင့် Binary Search သည် sorted လုပ်ထားသည့် data များကို အလယ်ကနေ ထပ်ခါ ထပ်ခါ ပိုင်းကာ မိမိတို့ လိုချင်သည့် data ကို ရှာသည့် နည်းလမ်း ဖြစ်ပါသည်။



Program ရေးသားခြင်း

Binary Search With Iterative Implementation

1. ပထမဆုံးအနေဖြင့် array တစ်ခု တည်ဆောက်ပါမည် ထို array ထဲတွင် ငယ်စဉ်ကြီးလိုက် စီထားသော sorted data များကို ထည့်ထားပါမည်။
2. နောက် တစ်ဆင့် အနေဖြင့် user ဆီမှ ရှာလိုသည့် key value ကို တောင်းပါမည်။
3. တတိယ အဆင့်မှာတော့ sorted လုပ်ထားသော မိမိတို့ array ၏ size ကို ရှာရမှာ ဖြစ်ပါတယ်။

```
int key=0;
int nums[] = { 2, 5, 6, 8, 9, 10 };
printf("Please enter a number to find:");
scanf("%d", &key);

int arrSize = sizeof(nums)/sizeof(nums[0]);
int index = binarySearch(nums, arrSize, key);
```

နေက်တစ်ဆင့် အနေဖြင့် Binary Search ဆိုသည့် function တစ်ခု ရေးပါမည် ထို function သည် parameter 3 ခုကို ရယူမည်ဖြစ်ပြီး ပထမ တစ်ခုသည် array ဖြစ်သည် ဒုတိယ တစ်ခုသည် array ဂဲ size ဖြစ်ပြီး နေက်ဆုံးကတော့ မိမိတို့ ရှာလိုသည့် နံပါတ် ဖြစ်ပါတယ်။

```
#include <stdio.h>

int binarySearch(int nums[], int arrSize, int key)
{
    // array ဂဲ အမြဲ့အမြဲ့ အနိမ့်ဆုံး အမှတ်များကို ရှာဖွေခြင်း
    int low = 0, high = arrSize - 1;
    // return မပြန်မခြင်း အလုပ်လုပ်ရန် while loop ဆုံးထားခြင်း ဖြစ်သည်။
    while (low <= high)
    {
        // အလယ်ကိန်း mid value ကို စွေ့ပြီး မိမိတို့ ရှာလိုသည့် ကိန်းနှင့် တူညီနေသား စစ်ပါတယ်
        int mid = (low + high)/2;

        // အကယ်၍ ပထမဆုံး သတ်မှတ်လိုက်သည့် အလယ်ကိန်းနှင့် မိမိတို့ ရှာလိုတဲ့ key တူညီနေလျှင်
        // program ရပ်သွားမှာ ဖြစ်ပါတယ်။
        if (key == nums[mid]) {
            return mid;
        }

        // key နှင့် အလယ်က ကိန်းကို နှင့်ယူပေါ်သော အခါး key ၏ ကိုယ်နေရာ၏ high ကို mid ဂဲ
        // ရွှေ့က ကိန်းအဖြစ် သတ်မှတ်လိုက်ပါတယ်။
        // ထိုသို့ သတ်မှတ် လိုက်ခြင်းဖြင့် array ကို နစ်ပိုင်း ပိုင်းလိုက်ပြီးသား ဖြစ်သွားပါသည်။
        // ငယ်စဉ် ကြွေးလိုက် စီးထားပြီးသား ဖြစ်သည့် အတွက် key သည် ငယ်နေသောကြောင့် ရွှေ့ပိုင်း
        // အခြမ်းကိုသာ ရှာတော့မှာ ဖြစ်ပါတယ်။

        else if (key < nums[mid]) {
            high = mid - 1;
        }

        // program အစတင်း low သည် 0 မှ စတင်ပြီး အကယ်၍ မိမိတို့ ရှာလိုသော key သည် Mid
        // ထပ်ကြီးနေလျှင်
        // low ကို သော် အသစ် သတ်မှတ် ပေးလိုက်ပါတယ် else if ထဲမှ code သည်
        // ငယ်နေလျှင် အလုပ်လုပ်ရန် ဖြစ်ပြီး အောက်က code သည် ကြီးနေလျှင် အလုပ်လုပ်ရန်
        // ဖြစ်သည်။
        else {
            low = mid + 1;
        }
    }
}
```

```

        }
    }

// അനുഭവിച്ചാണ് ഇത് എന്ന് കൊന്തെന്നു പറയുന്നതുമാണ്
return -1;
}

int main()
{
    int key=0;
    int nums[] = { 2, 5, 6, 8, 9, 10 };
    printf("Please enter a number to find:");
    scanf("%d",&key);

    int arrSize = sizeof(nums)/sizeof(nums[0]);
    int status = binarySearch(nums, arrSize , key);

    if (status != -1) {
        printf("Element found at index %d", status);
    }
    else {
        printf("Element not found in the array");
    }

    return 0;
}

```

അനുഭവിച്ചാണ് binary search മൂലം എന്ന് കൊന്തെന്നു പറയുന്നതുമാണ്
 ഫീൽഡിൽ അനുഭവിച്ചാണ് program മൂലം binary search മൂലം recursive ഫീൽഡിൽ
 സേരിക്കുന്നതുമാണ്.

Binary Search with recursive method

```

#include <iostream>
using namespace std;

int binarySearchRecur(int nums[],int low , int high,int key){
    if(low>high){
        return -1;
    }
    int mid=(low+high)/2;

    if(key==nums[mid]){
        return mid;
    }
    else if(key<nums[mid]){
        return binarySearchRecur(nums,low,mid-1,key);
    }
}

```

```

    }
    else{
        return binarySearchRecur(nums,mid+1,high,key);
    }
    return -1;
}
int main(){
    //           low                  High
    int nums[]={2,5,10,11,15,16,17,19,25,30,40};
    int key=0;
    cout<<"please enter a number to find:";
    cin>>key;

    int arrSize = sizeof(nums)/sizeof(nums[0]);
    int low=0;
    int high=arrSize-1;
    int foundIndex = binarySearchRecur(nums,low,high,key);
    if(foundIndex != -1){
        printf("We found data at index : %d",foundIndex);
    } else{
        printf("We cannot found");
    }
}

```

Linear Search

Linear search သည် algorithms များထဲတွင် အလွယ်ဆုံးဖြစ်ပြီး array တစ်ခုထဲမှာ ရှိတဲ့ data တွေကို ဘယ်ဘက်ဆုံးကနေ တစ်ခုချင်းစီ မိမိတို့ ရှာချင်တဲ့ data မရောက်မချင်းတိုက် စစ်သွားခြင်း ဖြစ်သည်။ အောက်ပါ program မှာတော့ အကယ်၍ မိမိတို့ ရှာချင်တဲ့ data ကိုတွေ့သွားလျှင် index number ကို return ပြန်ပေးလိုက်ပြီး အကယ်၍ မတွေ့လျှင်တော့ -1 ကို return ပြန်ပေးလိုက်ပါတယ်။

Sample program :: Linear Search

```

#include <iostream>
using namespace std;

int linearSearch(int nums[],int arrSize,int key){

    int i=0;
    for(i=0; i<arrSize ; i++){
        if(nums[i]==key){
            return i;
        }
    }
}

```

```
        }
    }
    return -1;
}

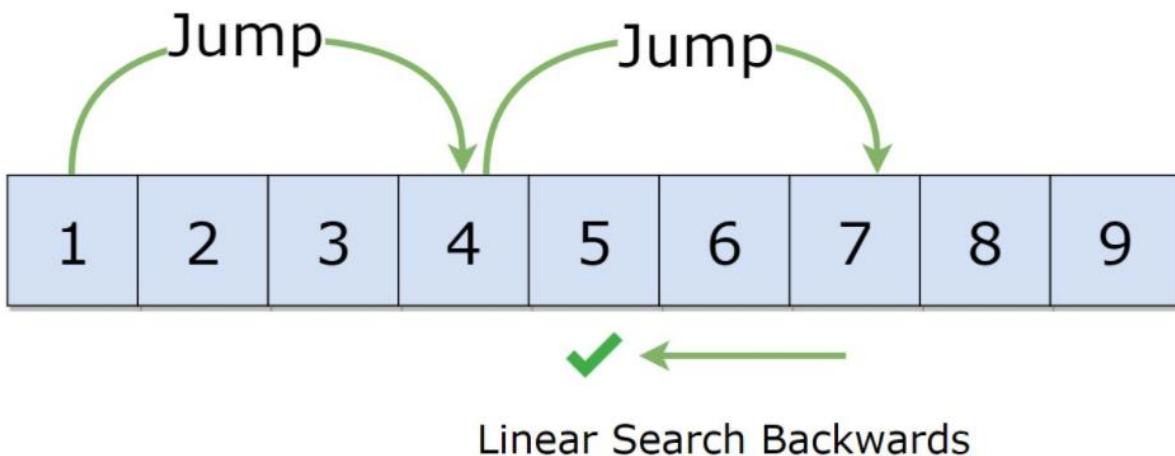
int main(){
    //           low                  High
    int nums[]={2,5,10,11,15,16,17,19,25,30,40};
    int key=0;
    cout<<"please enter a number to find:";
    cin>>key;

    int arrSize = sizeof(nums)/sizeof(nums[0]);

    int foundIndex = linearSearch(nums,arrSize,key);
    if(foundIndex != -1){
        printf("We found data at index : %d \n",foundIndex);
        printf("Data is :%d",nums[foundIndex]);
    } else{
        printf("We cannot found");
    }

    return 0;
}
```

Jump Search



Jump search algorithm သည် sorted လုပ်ထားသည့် data များကို ရှာဖွေရာတွင် အသုံးပြုတဲ့ searching algorithm တစ်မျိုး ဖြစ်ပါတယ်။ algorithm ခဲ့ အမိက ရည်ရွယ် ချက်ကတော့ လိုချင်တဲ့ data ကို မရမချင်း အကြိမ်ရေး နည်းပါးစွာဖြင့် ရှာဖွေ တွေ့ရှိနိုင်ရန် ဖြစ်ပါတယ်။ Jump Search algorithm ခဲ့ ကောင်းတဲ့ အချက်ကတော့ Linear search ထက် ပိုမြန်စွာ ရှာနိုင်ပါတယ်။

ထိုသို့ရှာရန် Jump search သည် တစ်ကြိမ်ပြီး တစ်ကြိမ် data များကို ဘေးတိုက် ရှာ သွားတာ မဟုတ်ပဲ ဘယ်ဘက်ဆုံးကနေ စတင်ပြီး jump ခုန်ကာ ရှာ သွားတဲ့ ဖြစ်ပါတယ်။ တစ်ကြိမ် ခုန်လျှင် မည်မျှ ခုန်ရမည်ကိုတော့ array ထဲမှာ ရှိတဲ့ data အရေအတွက်(n) ကို မူတည်ပြီး square root ရှာကာ တွက်ချက်ခြင်း ဖြစ်ပါတယ်။ ထို့ကြောင့် တစ်ကြိမ် ခုန်လျှင် မည်မျှ ခုန်ရမည်မှာ square of n ဖြစ်ပါတယ်။

Algorithm Steps For Jump Search

1. ပထမဆုံး အနေဖြင့် ခုန်ရမည့် data block size(jumpSize) ကိုရှာပါတယ်။
2. jumpSize ရသွားပြီဆိုလျှင် array[jumpSize] ထက် ငယ်နေသလား သို့မဟုတ် ညီသလား နှင့် jumpSize သည် array size ထက် ငယ်နေသလားဆိုတာကို စစ်ဆေးပါတယ်။
3. အထက်ပါ condition မှန်ကန်နေလျှင် jumpSize ကို variable တစ်ခုထဲမှာ သိမ်းထားပြီး sqrt(array size) ကို ထပ်ရှာပြီး variable ထဲသို့ ထပ်ပေါင်းပါတယ်။ ယခု အဆင့်ပြီးလျှင် jumpSize သည် 3 ဖြစ်နေပါက variable တန်ဘိုးသည် 6 ဖြစ်နေမည် ဖြစ်သည်။

4. အကယ်၍ ခုန်ရင်း အရှိန်လွန်ပြီး jumpSize သည် မူရင်း array size ထက် ကျော်လွန်သွားပါက jumpSize ကို မူရင်း array size ထဲမှ 1 နှုတ်ပြီး သတ်မှတ် ပေးလိုက်ပါမည် ထိုသို့ လုပ်ဆောင်ချက်သည် jumpSize အား array ရဲ့ နောက်ဆုံး Index အဖြစ် သတ်မှတ် ပေးလိုက်ခြင်း ဖြစ်သည်။
5. မိမိတို့ ရှာလိုသည့် data သည် မူရင်း array ရဲ့ index number 8 တွင် ရှိသည် ဆိုပါစို့ ပထမ တစ်ခါ JumpSize (index 3) , ဒုတိယ တစ်ခါ JumpSize(index 6) , တတိယ တစ်ခါ ထပ် Jump သော အခါတွင် JumpSize(index 9) ဖြစ်နေမည် ထိုသို့ သော အခြေနှင့် မျိုးတွင် JumpSize ရဲ့ ဒုတိယ အကြိမ် Index 6 အား မှတ်ထားပြီး ထို Index 6 နေရာမှ Linear Search ကို သုံးကာ မိမိတို့ လိုချင်သည့် data ရောက်သည် အထိ ရှာသွားခြင်း ဖြစ်ပါသည်။

Sample Program :: Jump Search

```
#include "iostream"
#include "cmath"

using namespace std;

int jumpSearch(int array[], int size, int key){
    int start=0;
    int jumpSize = sqrt(size); // how we should jump
    while(array[jumpSize] <= key && jumpSize<size){
        start=jumpSize;//for linear search
        jumpSize += sqrt(size);
        if(jumpSize > size-1){ // index number
            jumpSize = size;// array size
        }
    }
    for( int i = start ; i< jumpSize ; i++){
        if(array[i]==key){
            return i;
        }
    }
    return -1;
}

int main() {
    int nums[] = {2, 5, 10, 11, 15, 16, 17, 19, 25, 30, 40};
```

```

int key = 0;
cout << "please enter a number to find:";
cin >> key;

int arrSize = sizeof(nums) / sizeof(nums[0]);
int foundIndex = jumpSearch(nums, arrSize, key);
if (foundIndex != -1) {
    printf("We found data at index : %d", foundIndex);
} else {
    printf("We cannot found");
}
}

```

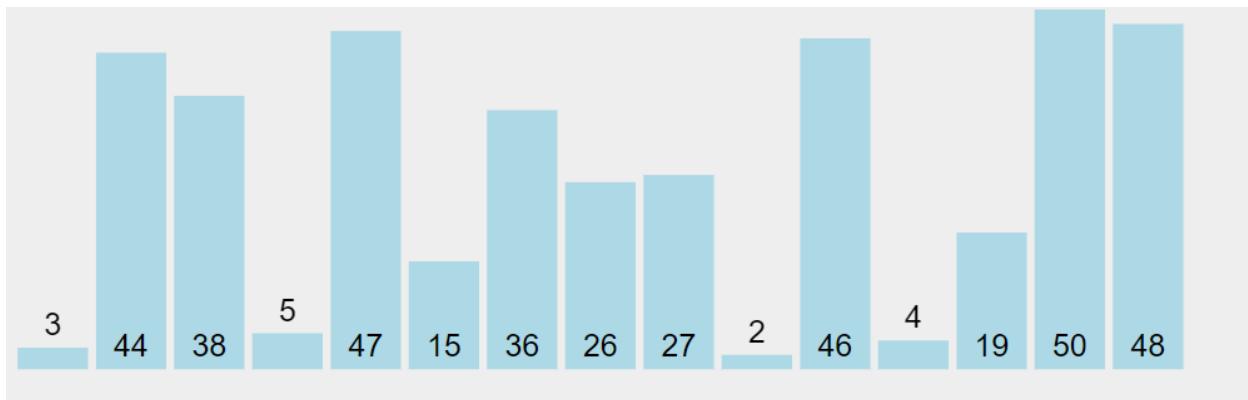
Sorting Algorithms

Bubble Sort

Bubble sort သည် data များကို ငယ်စဉ်ကြီးလိုက် သို့မဟုတ် ကြီးစဉ်ငယ်လိုက် အစီအစဉ် တကျ စီရန် အသုံးပြုသည့် sorting algorithm တစ်ခု ဖြစ်ပါတယ်။ Bubble sort သည် sorting စီရန် elements တွေ အားလုံးကို တစ်ခုချင်းစီ နှင့်ယူဉ်ပြီး ငယ်စဉ်ကြီးလိုက် သို့မဟုတ် ကြီးစဉ် ငယ်လိုက် ဖြစ်အောင် နေရာ ပြောင်းလဲ ပေးခြင်း ဖြစ်ပါတယ်။ ဥပမာ array ထဲမှာ ရှိတဲ့ ပထမ element တစ်ခုသည် ဒုတိယ element တစ်ခုထက် ကြီးနေသည် ဆိုလျှင် ထိန္တစ်ခု ကို နေရာ ချိန်းပေးလိုက်ပါတယ်။ ထိုနောက် ဒုတိယ element နှင့် တတိယ element ကို ထပ်နှင့်ယူဉ်ကာ ဒုတိယ element သည် ပထမ element ထပ်ကြီးနေပါက နေရာ ထပ်ချိန်းပါတယ်။ ထိုကဲ့သို့ element များကို သတ်မှတ်ထားတဲ့ အကြိမ်အရေ အတွက်ထိ ထပ်ကာထပ်ကာ နှင့်ယူဉ်ပြီး နေရာ ချိန်းပေးသွားခြင်းကို bubble sort လို့ ခေါပါတယ်။

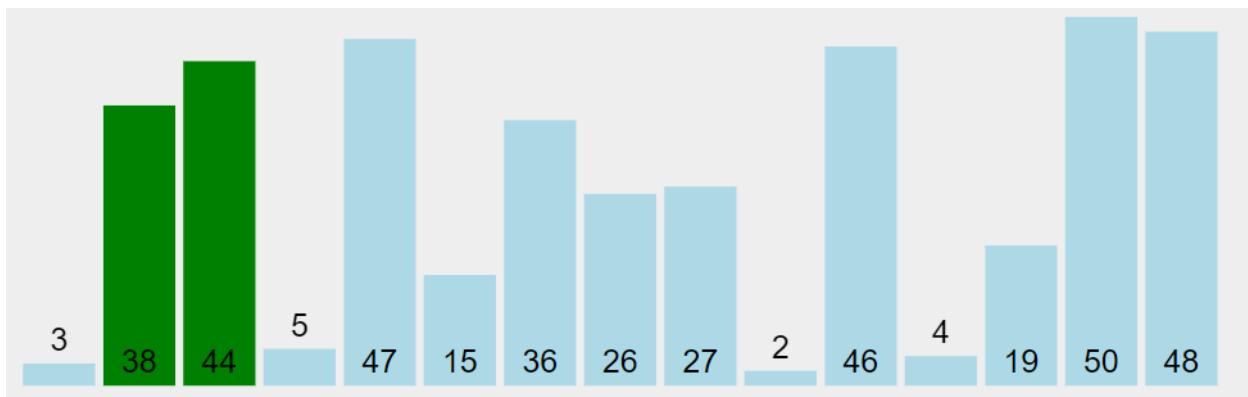
Bubble sort သည် sorting algorithms များထဲတွင် အနေးဆုံး algorithm ဖြစ်သော်လည်း implement လုပ်ရတာ လွယ်ကူခြင်းကြောင့် အသုံးများပါတယ်။

Stage 1:



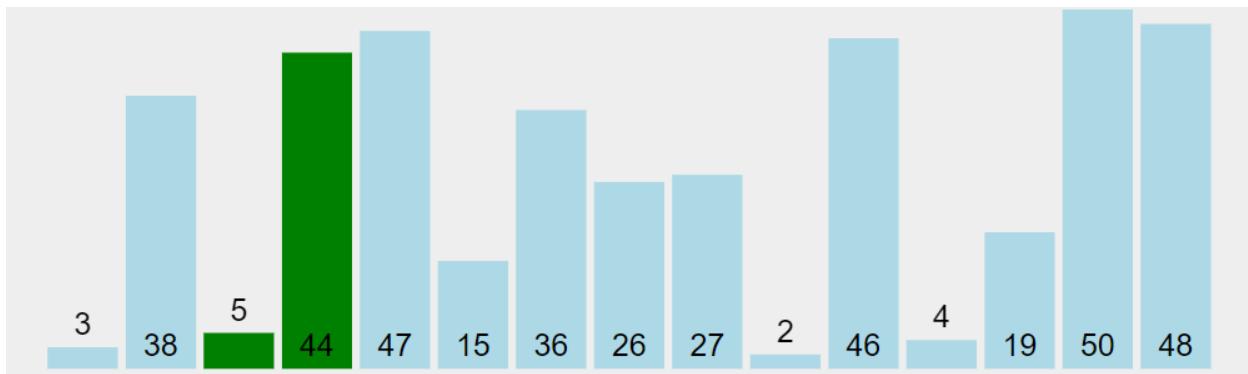
အထက်ပါ ပုံသည် bubble sort ခဲ့ stage 1 ပုံဖြစ်ပါတယ်။ ပထမဆုံး အနေဖြင့် ထိပ်ဆုံးမှာ ရှိတဲ့ 3 နဲ့ 44 ကို compare လုပ်ပါတယ် အကယ်၍ 3 က ကြီးနေလျှင် 3 ကို နောက်ပို့မှာ ဖြစ်ပါတယ်။ သို့သော် 3 သည် 44 ထက် မကြီးသည့် အတွက် နောက်ထပ် elements များကို ထပ်စစ်ပါတယ်။

Stage 2:



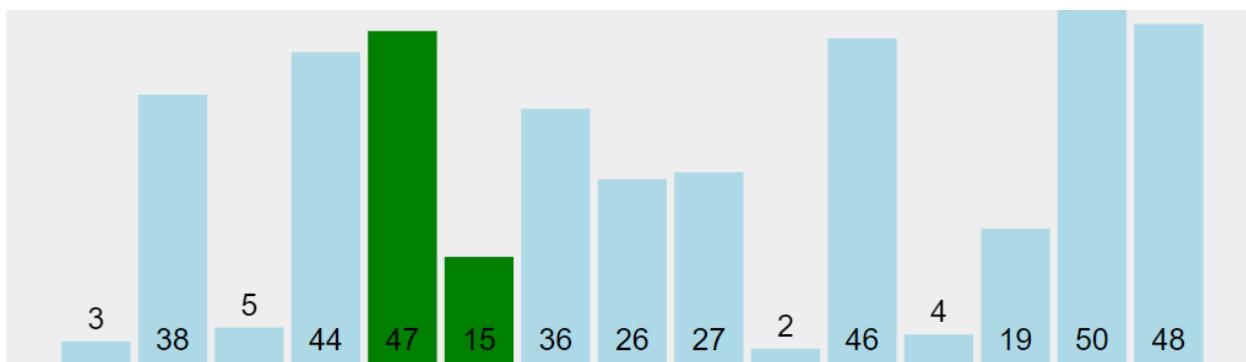
အထက်ပါ ပုံအတိုင်း stage 2 မှာတော့ index 1 မှာ ရှိတဲ့ 44 သည် index 2 မှာ ရှိနေသည့် 38 ထက် ကြီးနေသည့် အတွက် 38 ကို ရှုံးပို့ လိုက်ပါတယ်။

Stage 3:

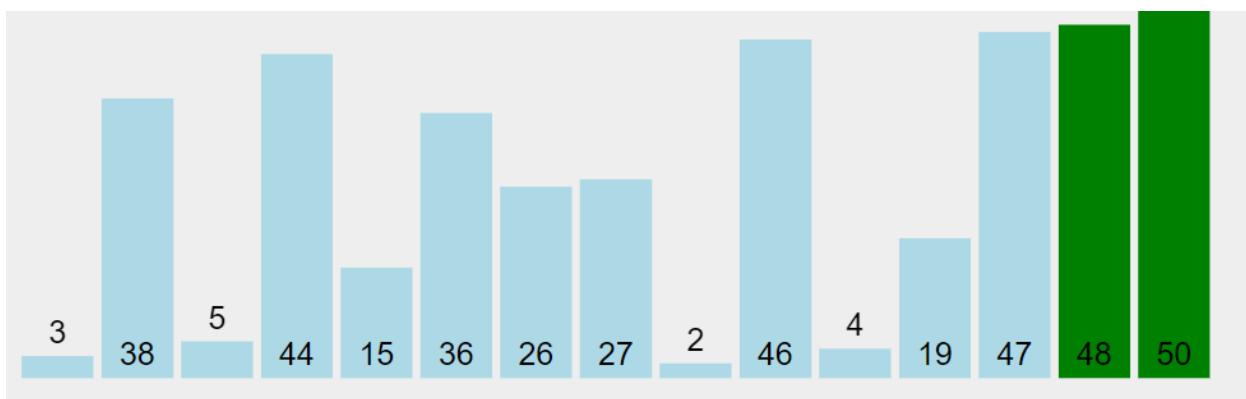


Stage 3 မှတော့ index 2 တွင် ရှိနေသည့် 44 သည် index 3 မှ ရှိနေတဲ့ 5 ထက်
ကြီးနေသည့် အတွက် 5 ကို ရွှေ့ပို့လိုက်ပါတယ်။

Stage 4:



ပထမ တစ်ခါ အဆုံးထိ looping ပတ်ပြီးသွားတဲ့ အခါတွင် elements များသည် အောက်ပါ
ပုံအတိုင်း sorted ဖြစ်မနေသေးပါဘူး ထို့ကြောင့် elements အရေအတွက်အတိုင်း အပြင်ဆုံး loop
က ပတ်ပြီးမှ ငယ်စဉ်ကြီးလိုက် အစဉ်လိုက် ဖြစ်သွားမှာဖြစ်ပါတယ်။



Sample Program:: Bubble Sort

```
#include "stdio.h"
void bubbleSort(int arr[],int arrSize){
    int i;
    int j;// for index Number
    int temp;//temp is for swap data
    for(i =0 ; i<arrSize ; i++){ // outer loop
        int data = arrSize-i-1;
        for(j=0; j< data ; j++){ // for swap data and for less looping
```

```
        if(arr[j] > arr[j+1]){
            temp = arr[j];
            arr[j]=arr[j+1];
            arr[j+1]=temp;
        }
    }
printf("Sorted data:");
for(i=0; i<arrSize ; i++){
    printf("%d ",arr[i]);
}
int main(){

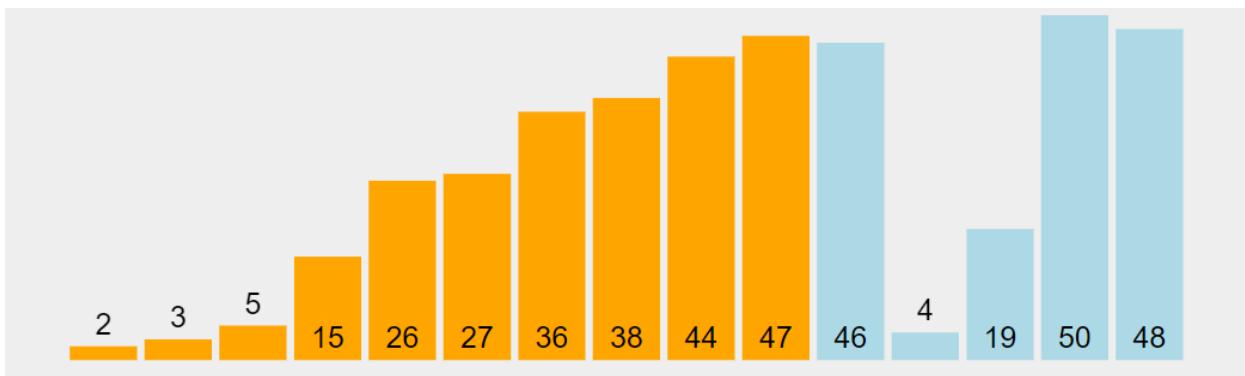
    int arr[]={9,7,8,5,6,3,2,1};
    int arrSize = sizeof(arr)/sizeof(arr[0]);

    bubbleSort(arr,arrSize);

    return 0;
}
```

Insertion Sort

Bubble sort သည် အထက်မှာ ဖော်ပြုခဲ့အတိုင်း elements အားလုံး ဆုံးသည် အထိန္ဒိုင်းယူဉ်ကာ နေရာ ချိန်းသွားသော်လည်း ပထမ တစ်ခါ အဆုံးထိ looping ပတ်ပြီးသော အခါ elements များသည် ငယ်စဉ် ကြီးလိုက် အစီအစဉ် မကျသေးပါဘူး။ Insertion Sort ကတော်တိနည်း အတိုင်း မဟုတ်ပဲ index 3 ခုထိ စစ်ပြီးပြီဆိုလျှင် index 0,1,2 တို့သည် index 3 ထက်ကြီးနေသလားဆိုပြီး ပြန်စစ်ဆေးပါသေးတယ်။



အထက်ပါ ပုံစွင် insertion sort သည် element 47 ထိ sort လုပ်ပြီးလျှင် ရွှေ့မှာ ရှိသော ကျိုး elements များသည် 47 ထက် ငယ်နေပြီး အစီစဉ်တိုင်း ဖြစ်နေသလား စစ်ပြီးမှ element 46 ကို ဆက်ပြီး အလုပ်လုပ်ပါတယ်။

Sample Program : Insertion Sort

```
#include "stdio.h"

void insertionSort(int arr[],int arrSize){
    int i; //For outer loop
    int j; // for swapping data
    int temp; // to stored data temporary

    for(i=1; i<arrSize ; i++){ //outer loop
        j=i;
        while ( j > 0 && arr[j-1]> arr[j]){ // inner loop
            temp = arr[j];
            arr[j] = arr[j-1];
            arr[j-1] = temp;
            j--;
        }
    }
    printf("Sorted Data :");
    for(i=0; i < arrSize ; i++){
        printf("%d ",arr[i]);
    }
}

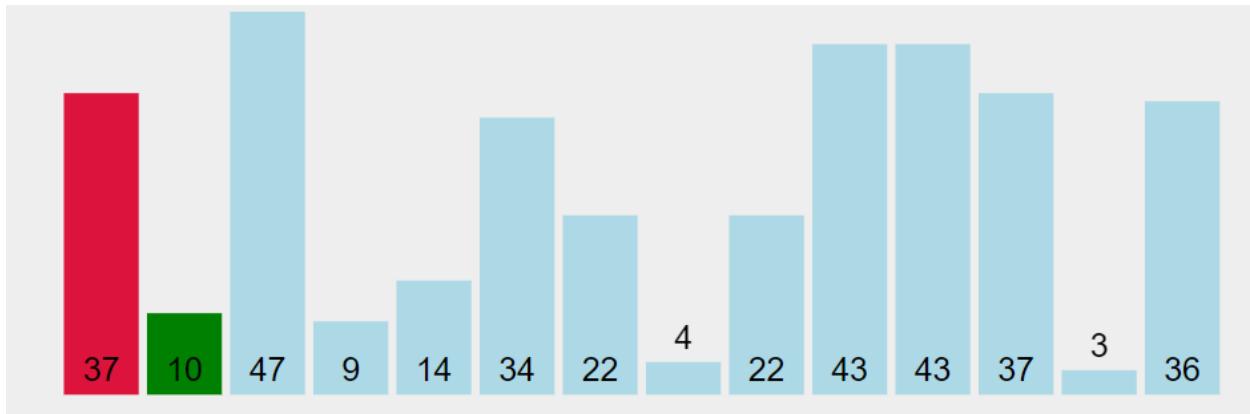
int main(){

    int arr[]={9,7,8,5,6,3,2,1};
    int arrSize = sizeof(arr)/sizeof(arr[0]);
    insertionSort(arr,arrSize);
    return 0;
}
```

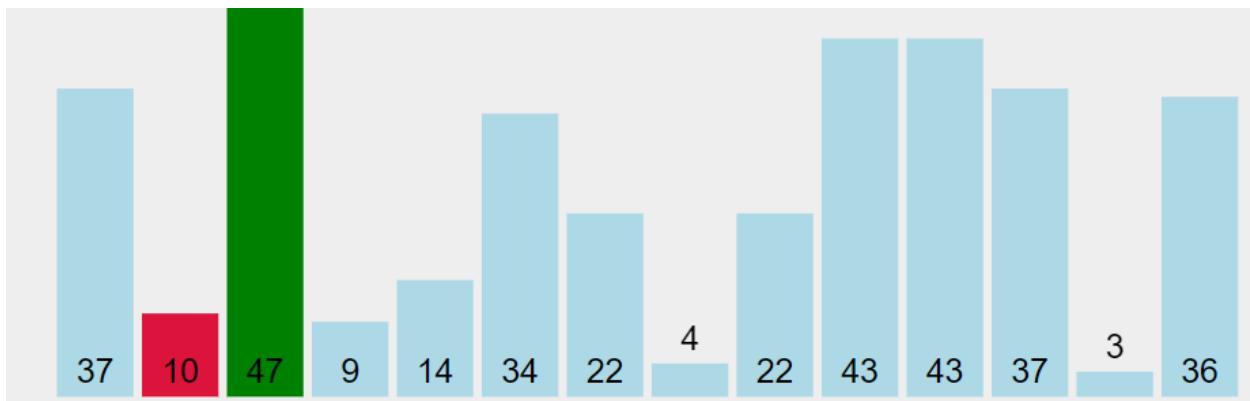
Selection Sort

Selection sort သည် ပထမဆုံး အနေဖြင့် ပေးထားသော array တစ်ခုထဲမှ ရွှေ့ဆုံး data index 0 ကို အငယ်ဆုံး ကိန်း အဖြစ် မှတ်ယူ ပါတယ်။ ထို့နောက် index 0 ဖြင့် index 1 ကို

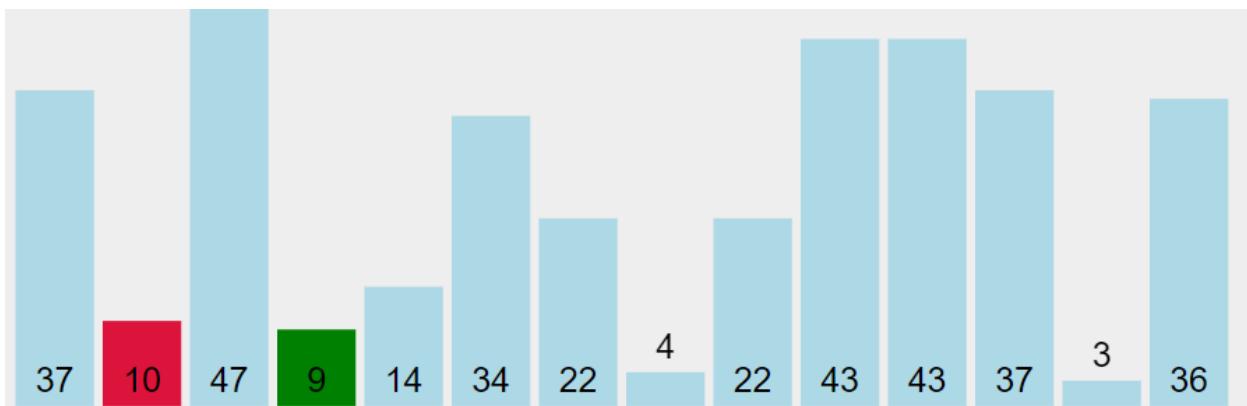
နိုင်းယှဉ်ပါတယ် အကယ်၍ index 1 သည် index 0 ထက် ငယ်နေခဲ့လျှင် index 1 အား အင်ယံးကိန်း အဖြစ်မှတ်ယူပြီး ထို index 1 နှင့် index 0 နေရာအား ချိန်းပေး လိုက်ပါတယ်။



အထက်ပါပုံတွင် အနီရောင်ဖြင့် ပြထားသော ပထမဆုံး index 0 (37) နေရာအား အင်ယံးနံပါတ် အဖြစ် သတ်မှတ် ထားခြင်း ဖြစ်ပါတယ်။ ထို့နောက် index 1 မှ 10 ဖြင့် နိုင်ယှဉ်ပါတယ် အကယ်၍ index 1 ကင်းယောက်နေလျှင် index 1 ကို အနီရောင် အဖြစ် ပြောင်းလိုက်ပြီး အင်ယံးကိန်းအဖြစ် သတ်မှတ်လိုက်မှာ ဖြစ်ပါတယ်။ ယခု နိုင်ယှဉ်မှုများလည်း index 1 သည် ငယ်နေသည့် အတွက် index 1 အား အနီရောင် ပြောင်းပေးမှာ ဖြစ်ပါတယ်။ အောက်ပါ ပုံတွင်ကြည့်ပါ။



နေရာကိုတစ်ဆင့် အနေဖြင့် index 2 မှ 47 ဖြင့် နိုင်းယှဉ်ပါမည် အကယ်၍ index 2 သည် index 1 မှ 10 ထက်ငယ်နေလျှင် 47 အားအနီရောင် ချိန်းမည် ဖြစ်ပြီး ယခု နိုင်းယှဉ်ချက်မှာတော့ index 2 သည် ကြီးနေသည့်အတွက် အနီရောင် ချိန်းမည် မဟုတ်ပဲ index 4 ဖြင့် ဆက်လက်စစ်ဆေးသွားမှာ ဖြစ်ပါတယ်။ အောက်ပါ ပုံတွင်ကြည့်ပါ။



Data များ အားလုံးအား ယခု ဖော်ပြသည့် နည်းလမ်း အတိုင်း စစ်ဆေး သွားပြီး အင်ယ်ဆုံး ကိန်းအား ရှေ့ဆုံးမှ index 0 ဖြင့် swap လိုက်မှာ ဖြစ်ပါတယ်။ အထက်ပါ data များတွင်မူ 3 ဖြင့် 37 ကို ချိန်းမှာ ဖြစ်ပါတယ်။ ယခု လုပ်ဆောင် ချက်များသည် ပထမ Looping တစ်ကြိမ် အတွက်သာ ဖြစ်ပြီး ဒုတိယ တစ်ကြိမ်ရောက်သော အခါမှာတော့ index 1 ကို စတင် စစ်ဆေး သွားမှာ ဖြစ်ပြီး တတိယ အကြိမ်မှာတော့ index 2 ကနေ စတင် စစ်ဆေးသွားမှာ ဖြစ်ပါတယ်။ ထိုကဲ့သို့ အဆင်ဆင့် စစ်ဆေးပြီး နေရာချိန်းကာ ငယ်စဉ်ကြီးလိုက် စီသွားခြင်းကို selection sort ဟု ခေါ်ဆို ခြင်း ဖြစ်ပါတယ်။

SampleProgram : Selection Sort

```
#include <stdio.h>

void swap(int arr[], int i, int j)
{
    int temp = arr[i];
    arr[i] = arr[j];
    arr[j] = temp;
}

void selectionSort(int arr[], int n)
{
    for (int i = 0; i < n - 1; i++)
    {
        int min = i;

        for (int j = i + 1; j < n; j++)
        {

            if (arr[j] < arr[min]) {
                min = j;
            }
        }

        swap(arr, i, min);
    }
}
```

```

        }
    }

    swap(arr, min, i);
}
}

void printArray(int arr[], int n)
{
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
}

int main(void)
{
    int arr[] = { 3, 5, 8, 4, 1, 9, -2 };
    int n = sizeof(arr) / sizeof(arr[0]);

    selectionSort(arr, n);
    printArray(arr, n);

    return 0;
}

```

Merge Sort

C-typedef

Typedef သည် c programming ရဲ့ keyword တစ်ခု ဖြစ်ပြီး မိမိတို့ ဖန်တီးလိုက်တဲ့ data structure သို့မဟုတ် data type တွေကို name အသစ်တွေ ပေးပြီး အသုံးပြုလိုတဲ့ အချင်မှာ အသုံးပြုပါတယ်။

Example: `typedef data_type new_name;`

typedef: သည် keyword ဖြစ်သည်။

`data_type:`သည် ရှိထားပြီးသား data types များလည်း ဖြစ်နိုင်သလို structure/union တို့ကို သုံးပြီး ဖန်တီးထားတဲ့ types များလည်း ဖြစ်နိုင်ပါတယ်။

new_name: သည် **data_types** ကိုယ်စား အသုံးပြုလိုသည့် **name** သို့မဟုတ် **name** အသစ် ပေးရန် ဖြစ်သည်။

ဥပမာ **typedef int ncc;**

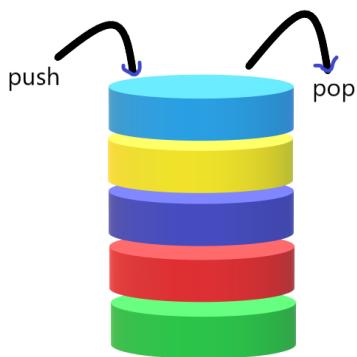
Ncc i=0; // ယခု အတိုင်း ရေးမည်ဆိုလျှင် **int i=0** နှင့် တူညီပါသည်။

Typedef ကို ကြော်ကြော် အခိုင်းမှာ **name** တစ်ခုတည်း မဟုတ်ပဲ **name** နှစ်ခုဖြင့်လည်း ကြော် နိုင်ပါတယ်။

Example: typedef int ncc, gh ;

အထက်ပါ အတိုင်းဆိုလျှင် **int** ကိုယ်စား အသုံးပြုမည့် **ncc** နှင့် **gh** ဆိုသည့် **name** နှစ်ခုကို ကြော် လိုက်တာ ဖြစ်ပါတယ်။

Stack Data Structure



Stack သည် **linear data structure** ဖြစ်ပါတယ် **linear** ဆိုသည်မှာ **data** များကို အစီအစဉ်တကျ စီစဉ်ထားရာတွင် ရှေ့က **data** နှင့် နောက်က **data** တို့ကို ချိတ်ဆက်ထားသည့် ပုံစံ ဖြစ်သည်။ ထို့ကြောင့် **stack data structure** ကို **LIFO** (Last in First Out) သို့မဟုတ် **FILO** (First in Last Out) ဟုလည်း ခေါ်သည်။ **Last in First Out** ဆိုသည်မှာ နောက်ဆုံး ဝင်လာသည့် **data** သည်

ပထမဆုံးပြန်ထွက်ရမည် ဖြစ်ပြီး FILO ပထမဆုံး ဝင်လာသည့် data သည်လည်း နောက်ဆုံးမှ ထွက်လာရမည် ဖြစ်သည်။

ယခု Stack idea သည် စားပွဲပေါ်မှာ ရှိတဲ့ “stack of plates” ဆိုတဲ့ အရာမှာ လာခြင်း ဖြစ်ပါတယ် ဆိုလိုသည်မှာ စားပွဲပေါ်မှာ တင်ထားတဲ့ ပန်းကန် ပြားတွေ လိုပါပဲ ပန်းကန်ပြားတွေ ထပ်ထည့်လိုလျှင်လည်း ထိပ်မှာ ထပ်ထည့်ရမှာ ဖြစ်သလို ထုတ်ရင်လည်း ထို ထိပ်မှာ ပြန်ထုတ်ရမှာ ဖြစ်ပါတယ်။

Stack data structure ကို Implement လုပ်ရာတွင် array ကိုလည်း အသုံးပြုလို့ ရသလို ရွှေ့ပိုင်း သင်ခန်းစာများတွင် ဖော်ပြထားဖူးသည့် linked list ကိုလည်း အသုံးပြုကာ implement လုပ်နိုင်ပါတယ်။ ယခု သင်ခန်းစာများတော့ array ကိုသုံးကာ implement လုပ်သွားမှာ ဖြစ်ပါတယ်။ ဤ သင်ခန်းစာအား မလေ့ လာခင် ရွှေ့ပိုင်းတွင် ဖော်ပြခဲ့ပြီး ဖြစ်သော array , structure , pointer , arrow pointer နှင့် memory allocation တို့ကို သိထားပြီးသား ဖြစ်ရမှာ ဖြစ်ပါတယ်။ အောက်တွင် example program ကို ဖော်ပြထားပါသည်။

ပထမဆုံး အနေဖြင့် structure ကို သုံးပြီး stack data structure ရှိရမည့် ပုံစံကို စတင်ရေး သားပါမည်။

```
typedef struct {
    int top;
    int ST[stackSize];
} StackType, *Stack;
```

အထက်ပါ program တွင် struct ထဲ၌ top နှင့် ST ဆိုသည့် array တစ်ခုကို ကြော်ကြာ ထားပါသည်။ top သည် stack data structure ထဲတွင် ရှိမည့် data အရေအတွက်ကို counting လုပ်ရန် ဖြစ်သည် ထိုသို့ counting လုပ်ထားမှသာ stack overflow ဖြစ်သလား သို့မဟုတ် data တစ်ခုမှ မရှိပဲ empty

ဖြစ်နေသလား	ဆိတာကို	သိရှိ	နှင့်မှာ	ဖြစ်ပါတယ်။
-------------------	----------------	--------------	-----------------	-------------------

```


11 int main() {
12     Stack initStack();
13     int empty(Stack);
14     void push(Stack, int);
15     int pop(Stack);
16     int n;
17
18     Stack S = initStack();
19     printf("Press 0 if you want to quit or push some data like integers:\n");
20     scanf("%d", &n);
21     while (n != 0) {
22         push(S, n);
23         scanf("%d", &n);
24     }
25     printf("Numbers in reverse order\n");
26     while (!empty(S))
27         printf("%d ", pop(S));
28     printf("\n");
29 }

```

အထက်ပါ program တွင် Line 12 ၏ initStack() ဆိတဲ့ function ကို Stack type ဖြင့် ကြော်လော်တော်မူပါတယ်။ ထို Stack သည် ယခင် ကြော်လော်တော်မူသော struct မှ ဖြစ်ပြီး Stack သည် pointer type ဖြင့် ကြော်လော်တော်မူပါတယ်။ ထိုကဲ့သို့ pointer type ဖြင့် ကြော်လော်တော်မူခဲင်းဖြင့် data structure တစ်ခုလုံးကို ကိုင်တွယ်ရာမှာ ပိုမို အဆင်ပြေဆောင်ပါတယ်။

Line number 12,13,14,15 တို့သည် သက်ဆိုင်ရာ Function များကို declare လုပ်တော်မူပါတယ်။ Line number 18 တွင် initStack() ဆိတဲ့ function ကို စခေါပါတယ် ထိုသို့ခေါ်ပြီး ပြန်လာတဲ့ return value ကိုတော့ S ဆိတဲ့ Stack type တဲ့ကို ထည့်သော်မူပါတယ်။

```

Stack initStack() {
    Stack sp = (Stack) malloc(sizeof(StackType));
    sp -> top = -1;
    return sp;
}

```

initStack function သည် malloc ကိုသုံးပြီး stack data structure တစ်ခုကို စောက်ပါတယ် အထက်ပါ program တွင် ကြည့်ပါ ထို stack data structure ရဲ့ size ကိုတော့ StackType ဟုရေးပြီး မိမိတို့ struct ဖြင့် တည်ဆောက်ခဲ့သော size ကို ပြန်ယူ ထော်ပါတယ်။ နောက်ထပ်

သတိပြုရန် အချက်မှာ malloc function ရှေ့မှ Stack ဖြစ်ပါတယ် သူသည် stack data structure ကို မည်သည့် type ဖြစ် တည်ဆောက် မည် ဖြစ်ကြောင်း ထည့်ပေးရတာ ဖြစ်ပါတယ် မိမိတို့ Stack type ဖြစ်သာ လိုချင်သည့် အတွက် Stack ဟု ရေးပေးထားခြင်း ဖြစ်ပါတယ်။ ယခု သင်ခန်းစာ အကြောင်းကို ပိုမို နားလည်စေရန် ရှေ့ပိုင်းတွင် ဖော်ပြုခဲ့ပြီး ဖြစ်သော dynamic memory allocation သင်ခန်းစာ ကို ကြိုးကြောင်း လေ့လာတော်းရမှာ ဖြစ်ပါတယ်။

Memory allocation လုပ်သော အခါ Stack type ဖြင့် တည်ဆောက် တာ ဖြစ်တဲ့ အတွက် ထို

Memory allocation ယူလိုက်သော sp ဆိုတဲ့ type ထဲတွင်

```
int top;
```

```
int ST[stackSize];
```

ယခု အတိုင်း ပါနေမှာ ဖြစ်ပါတယ်။ အောက်ပါ ပုံတွင် ကြည့်ပါ။

```

sp = {Stack} 0x14328dc16e0
  01 top = {int} -1163005939
  ✓  ST = {int [10]}
    01 [0] = {int} -1163005939
    01 [1] = {int} -1163005939
    01 [2] = {int} -1163005939
    01 [3] = {int} -1163005939
    01 [4] = {int} -1163005939
    01 [5] = {int} -1163005939
    01 [6] = {int} -1163005939
    01 [7] = {int} -1163005939
    01 [8] = {int} -1163005939
    01 [9] = {int} -1163005939

```

Memory allocation တည်ဆောက် ပြီးသော အခါ top သည် အထက်ပါ ပုံတဲ့မှ အတိုင်း random value များ ဝင်နေသည်ကို မြင်ရမည်။ ဒုတိယ တစ်ဆင့် အနေဖြင့် top ထဲသို့ sp -> top = -1; ကိုထည့်လိုက်ပါသည်။ ထို့နောက် တည်ဆောက် ပြီးသော sp ဆိုသည့် data structure တစ်ခုလုံးကို return အနေဖြင့် ပြန်ပေးလိုက်ပါတယ်။

```

18     Stack S = initStack();
19     printf("Press 0 if you want to quit or push some data like integers:\n");
20     scanf("%d", &n);
21     while (n != 0) {
22         push(S, n);
23         scanf("%d", &n);
24     }
25     printf("Numbers in reverse order\n");
26     while (!empty(S))
27         printf("%d ", pop(S));
28     printf("\n");
29 }
```

အထက်ပါ ပုံတဲ့မှ အတိုင်း line number 18 တွင် S ဆိုသည့် stack data structure တည်ဆောက် ပြီးသည့် အခါ line 19 အတိုင်း data အချို့ကို တောင်းပါတယ်။ ထည့်ပေးလိုက်တဲ့ data တွေသည် zero(0) မဖြစ်လျှင် line 22 ကို အလုပ် ဆက်လုပ်မှာ ဖြစ်ပါတယ်။ Line 22 တွင် push ဆိုတဲ့ function ကို ဘွားခေါ်ပါတယ် push function သည် data များ ထည့်ရန် တည်ဆောက် ထားခြင်း ဖြစ်သည်။

```

void push(Stack S, int n) {
    if (S -> top == stackSize - 1) {
        printf("\nStack Overflow\n");
        exit(1);
    }
    ++(S -> top);
    S -> ST[S -> top] = n;
}
```

အထက်ပါ push function သည် parameter နှစ်ခု ရယူပါသည်။ ပထမ တစ်ခုသည် stack type ဖြင့် လက်ခံမည့် S ဆိုသည့် Data Structure တစ်ခုလုံး ဖြစ်ပြီး ဒုတိယ တစ်ခုသည် DS ထဲသို့ ထည့်မည့် data ဖြစ်သည်။

အကယ်၍ S ဆိုတဲ့ stack DS ထဲမှာ ရှိတဲ့ top သည် stackSize - 1 နှင့် တူညီ နေခဲ့လျှင် stack Overflow ဖြစ်သွားမည် ဖြစ်သည်။ အဘယ်ကြောင့်ဆိုသော stackSize ထပ် data ပိုထည့်ရန် ကြိုးစား မိသော ကြောင့် ဖြစ်သည်။ S stack DS ထဲမှာ ရှိတဲ့ top သည် S stack DS ထဲတွင် data မည့်မျှ ရှိသည်ကို မှတ်တမ်း တင်ထားရန် ဖြစ်သည်။

```
++(S -> top);
```

အကယ်၍ stack overflow မဖြစ်သေးလျှင် S stack DS ထဲမှာရှိတဲ့ top ကို တစ်ပေါင်းထည့်ပါတယ် သို့မှသာ S stack DS ထဲတွင် data မည်မျှ ရှိသည်ကို မှတ်တမ်း တင်ထားနိုင်မှာ ဖြစ်ပါတယ်။

S -> ST[S -> top] = n;

ထို့နောက် n ဆိုသည့် data ကို S -> top ဆိုတဲ့ နေရာ သို့ ထည့်မှာ ဖြစ်ပါတယ် ဥပမာ S -> top သည် 5 ဖြစ်နေလျှင် ST ဆိုတဲ့ array ရဲ့ ST[5] နေရာသို့ n ဆိုတဲ့ data ကို ထည့်မှာ ဖြစ်ပါတယ်။

```

19     printf("Press 0 if you want to quit or push some data like integers:\n");
20     scanf("%d", &n);
21     while (n != 0) {
22         push(S, n);
23         scanf("%d", &n);
24     }
25     printf("Numbers in reverse order\n");
26     while (!empty(S))
27         printf("%d ", pop(S));
28     printf("\n");
29 }
```

အကယ်၍ user မှ data ထပ်ထည့်လျှင် zero မဟုတ်ပါက push function ကို အထက်တွင် ဖော်ပြု ထားသည့် အတိုင်း ပြန်ခေါ်မှာ ဖြစ်ပါတယ်။

Line number 26 မှာတော့ push လုပ်ထားသည့် data များကို ပြန်ထုတ်ပြုမှာ ဖြစ်ပါတယ်။ ထို သို့ ပြန်ထုတ်ပြု ရာတွင် S stack DS ထဲတွင် data တွေ ရှိသလား မရှိသလား ဆိုတာကို အရင် စစ်ဆေးပါတယ်။ မရှိလျှင် data များကို ထုတ်ပြစ်ရာ မလိုသည့် အတွက် while ဖြင့် စစ်ဆေးထားပါတယ်။ Line 26 တွင် empty() ဆိုတဲ့ function ကို စခေါ်ထားပါတယ်။

```
int empty(Stack S) {
    return (S -> top == -1);
}
```

Empty function သည် Stack DS type ကို parameter အဖြစ် လက်ခံ သည့် အတွက် Stack S ဟု ကြော်ထားပါတယ်။ (S -> top == -1) အကယ်၍ S -> top ထဲမှ value သည် -1 ဖြစ်နေလျှင် မည်သည့် data မျှ မရှိသည့် အတွက် ထို condition မှန်နေသော ကြောင့် 1 ကို return ပြန် ပေးလိုက်မှာ ဖြစ်ပါတယ် သို့မဟုတ် မတူပဲ data ရှိနေပါက 0 ကို return ပြန် ပေးမှာ ဖြစ်ပါတယ်။

```

26     while (!empty(S))
27         printf("%d ", pop(S));
28     printf("\n");

```

အကယ်၍ empty function သည် 1 ကို return ပြန်ပေးလျှင် while (!1) ဖြစ် မည်
ဆိုလိုသည်မှာ not True (မ မှန်ဘူး) ဟု ဆိုလိုခြင်း ဖြစ်သောကြောင့် ထို while loop ကို အလုပ်
လုပ်တော့ မည် မဟုတ်ပါ အကယ်၍ empty function သည် 0 ကို return ပြန် ပေးလျှင် while (!0) ဖြစ်မည်
ဆိုလိုသည်မှာ while (not False) မ မှားဘူး ဟု ဆိုလိုသောကြောင့် မှန်သည်
မှန်သောကြောင့် while loop ထဲမှ Program များကို ဆက်လက် အလုပ် လုပ်သွားမှာ ဖြစ်ပါတယ်။

ဆက်လက် အလုပ်လုပ်သော အခါတွင် line 27 မှ pop function ကို သွားခေါ်မှာ ဖြစ်ပါတယ်
pop function သို့ ခေါ်ရာတွင် S stack DS တစ်ခုလုံးကိုပါ ထည့်ပေးလိုက်ပါတယ်။

```

int pop(Stack S) {
    if (empty(S)) return RogueValue;
    int hold = S -> ST[S -> top];
    --(S -> top);
    return hold;
}

```

pop function သည် Stack type ဖြင့် parameter ကို လက်ခံ လိုက်ပြီး S သည် empty
ဖြစ်နေပါက ဆိုလိုသည်မှာ stack data structure ထဲတွင် မည်သည့် data မှု ရှိမနေပါက
RogueValue ကို return ပြန် ပေးလိုက်ပါတယ်။ RogueValue ဆိုတာဟာ symbolic constant
ကိုသုံးပြီး ကြိုတင် သတ်မှတ် ပေးထားသည့် value ဖြစ်ပြီး မိမိတို့ အသုံးလိုတဲ့ အခါ ပြန်သုံးခြင်း
ဖြစ်သည် rogue value သည် flag value , trip value , signal value နှင့် dummy data
စသည်တိနှင့် လည်း အတူတူပင် ဖြစ်သည်။

```
int hold = S -> ST[S -> top];
```

အထက်ပါ code ကတော့ S dS ထဲမှာ ရှိတဲ့ ST ဆိုတဲ့ array ထဲက top ကို မှတ်ထားတဲ့
number ကနေ စတုတ်မှာ ဖြစ်ပါတယ် ဥပမာ data 9 လုံးထည့်ထားရင် top သည် 9 ဖြစ်နေမှာ
ဖြစ်ပါတယ် ထို့ကြောင့် ပထမဆုံး အနေဖြင့် S->ST[9] ပုံစံဖြင့် array ရဲ့ index 9 နေရာမှာ ရှိတဲ့ data
ကို စတုတ်မှာ ဖြစ်ပါတယ်။

```
#include <stdio.h>
```

```
#include <stdlib.h>
#define RogueValue -9595
#define stackSize 10
typedef struct {
    int top;
    int ST[stackSize];
} StackType, *Stack;

int main() {
    Stack initStack();
    int empty(Stack);
    void push(Stack, int);
    int pop(Stack);
    int n;

    Stack S = initStack();
    printf("Press 0 if you want to quit or push some data like integers:\n");
    scanf("%d", &n);
    while (n != 0) {
        push(S, n);
        scanf("%d", &n);
    }
    printf("Numbers in reverse order\n");
    while (!empty(S))
        printf("%d ", pop(S));
    printf("\n");
}

Stack initStack() {
    Stack sp = (Stack) malloc(sizeof(StackType));
    sp -> top = -1;
    return sp;
} //end initStack
int empty(Stack S) {
    return (S -> top == -1);
} //end empty

void push(Stack S, int n) {
    if (S -> top == stackSize - 1) {
        printf("\nStack Overflow\n");
        exit(1);
    }
    ++(S -> top);
```

```

S -> ST[S -> top]= n;
} //end push
int pop(Stack S) {
    if (empty(S)) return RogueValue;
    int hold = S -> ST[S -> top];
    --(S -> top);
    return hold;
} //end pop

```

အောက်ပါ Github link တွင်လည်း source code ကို ရယူ နိုင်ပါတယ်။

<https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/stackWithArray.c>

Stack Data Structure with LinkedList

ယခု သင်ခန်းစာမျာတော့ Stack DS ကို LinkedList သုံးပြီး ရေးပြသွားမှာ ဖြစ်ပါတယ်။
ထိုကြောင့် ယခု သင်ခန်းစာအား လေ့လာ မည်ဆိုလျှင် ဖော်ပြခဲ့ပြီး ဖြစ်သော LinkedList
သင်ခန်းစာအား လေ့လာ ထားပြီးမှသာလျှင် အဆင်ပြေမှာဖြစ်ပါတယ်။

```

typedef struct node {
    int num;
    struct node *next;
} Node, *NodePtr;

typedef struct stackType {
    NodePtr top;
} StackType, *Stack;

```

ပထမဆုံး အနေဖြင့် struct ကိုသုံးပြီး node တစ်ခုရဲ့ ပုံစံအား တည်ဆောက်ပါသည်
LinkedList ရဲ့သဘော သဘာဝအတိုင်း data တစ်ခုနှင့် နောက်ထပ် node တစ်ခုအား ချိတ်ဆက်မည့်
pointer တစ်ခုပါဝင်ပါသည်။

```

14 int main() {
15     Stack initStack();
16     int empty(Stack);
17     void push(Stack, int);
18     int pop(Stack);
19     int n;
20     Stack S = initStack();
21     printf(" Press 0 if you want to quit or push some data like integers:\n");
22     scanf("%d", &n);
23     while (n != 0) {
24         push(S, n);
25         scanf("%d", &n);
26     }
27     printf("Numbers in reverse order\n");
28     while (!empty(S))
29     {
30         printf("%d ", pop(S));
31     }
32 } //end main

```

နောက်တစ်ဆင့် အနေဖြင့် line 20 တွင် initStack() ဆိုသည့် function ကို စခေါပါတယ်။

```

32 Stack initStack() {
33     Stack sp = (Stack) malloc(sizeof(StackType));
34     sp -> top = NULL;
35     return sp;
36 }

```

initStack() function ထဲတွင် memory ပေါ်၍ နေရာ တစ်ခု စယူ ရန် malloc function ကိုသုံးပြီး StackType အတိုင်း size တစ်ခုကို တည်ဆောက်ပါတယ် type ကိုတော့ Stack ဆိုတဲ့ type ကိုသာ အသုံးပြုပါတယ်။ လုပ်ဆောင်ပုံမှာ size ကိုတော့ StackType ရဲ့ size ကိုသုံးပြီး type ကိုတော့ Stack ရဲ့ type ကို သုံးထားခြင်း ဖြစ်သည်။ အသေးစိတ် ပိုမို နားလည် စေရန် အောက်ပါ ပုံတွင် ကြည့်ပါ။

```

    < sp = {Stack} 0x1fe957216e0
      < top = {NodePtr} 0xbaadf00dbaadf00d
        01 num = {int}
        < next = {node *}
          01 num = {int}
        > next = {node *}
  
```

Sp ဆိုသည့် Stack တဲ့တွင် ကြည့်သော အခါ top ဆိုသည့် NodePtr type ဖြင့် node တစ်ခု ကိုတွေ့ရမည် အဘယ်ကြောင့် ဆိုသော် top ကို stackType ထဲတွင်ကြပြေသော အခါ NodePtr type ဖြင့် ကြပြေ ထားသော ကြောင့် ဖြစ်သည်။ NodePtr type ဖြင့် top အား ကြပြေထားသော ကြောင့် top ထဲတွင် num နှင့် next တို့ ပါဝင် နေသည်ကိုလည်း သတိပြုရမည်။

ထိုကဲ့သို့ sp ဆိုသည့် memory allocation ယူပြီးသော အခါ ထို sp ထဲမှ top ထဲသို့ NULL ဆိုသည့် data တစ်ခုသာ ထည့်ထားပြီး sp တစ်ခုလုံးကို return ပြန် ပေးလိုက်ပါတယ်။

Line number 20 အား ပြန်ကြည့်မည် ဆိုလျှင် S ဆိုသည့် နေရာ၏ initStack() မှ return ပြန် ပေးလိုက်သော sp ရောက်ရှိနေသည်ကို တွေ့ရမည်။ Line 24 တွင် push() ဆိုသည့် function ကို စခေါပါသည် ထိုသို့ ခေါ်ရောတွင်လည်း S နှင့် user မှ ထည့်ပေးလိုက်သော n ဆိုသည့် data တို့ကိုပါ တစ်ခါတည်း ထည့်ပေးလိုက်ပါသည်။

Push function မှ ပထမဆုံး parameter အား လက်ခံမည်ဆိုလျှင် Stack type ဖြင့် လက်ခံ ရမည်ကို သတိပြုရမည် အဘယ်ကြောင့် ဆိုသော် ဝင်လာမည် parameter သည် Stack type ဖြစ်သောကြောင့် ဖြစ်သည်။

```

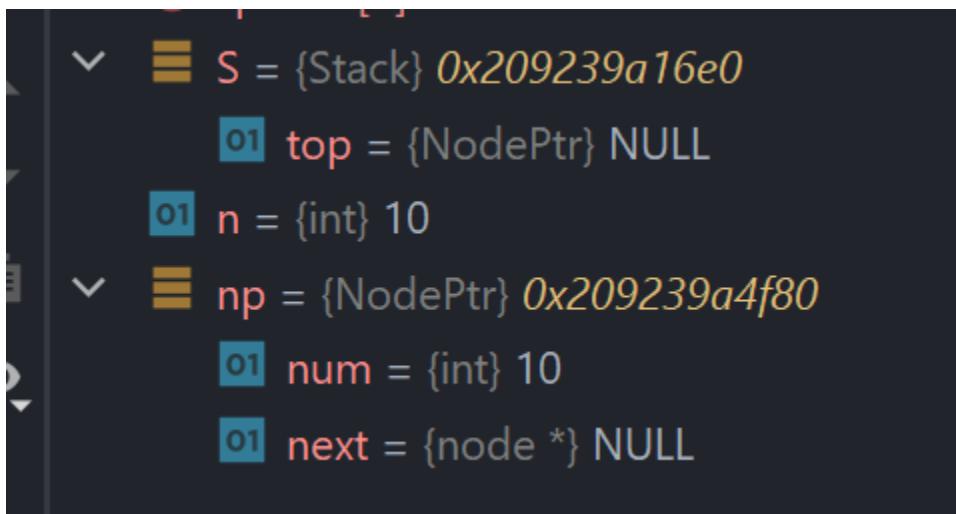
41     void push(Stack S, int n) {
42         NodePtr np = (NodePtr) malloc(sizeof(Node));
43         np -> num = n;
44         np -> next = S -> top;
45         S -> top = np;
46     }

```

Line 42 တွင် `malloc` function ကိုသုံးပြီး `node` တည်ဆောက်ရန် စတင်သည့် `size` ကို `Node` ရဲ့ `size` အား သုံးပြီး `type` ကိုတော့ `NodePtr` `type` ကို သုံးသည့် ထိုကြောင့် ယူ ဆောက်လိုက်သော `node` သည် `NodePtr` အတိုင်း ပြုမှုမည် ဖြစ်ပြီး `NodePtr` ထဲတွင်လည်း `num` အား ပါဝင်နေသည်။ `node` တစ်ခုတည်ဆောက်ပြီးသည့် အခါ `np` ဟု နံမည် ပေးလိုက်ပြီး ထို `np` ထဲမှ `num` ထဲကိုတော့ `user` ထည့်ပေးလိုက်သည့် `number` ထည့်ပေးလိုက်ပါသည်။

ထိုနောက် `S-> top` ကို `np-> next` ထဲသို့ ထည့်ပေးလိုက်ပါသည်။ ထိုကဲ့သို့ ရေးရခြင်း အကြောင်းကို အောက်တွင် အသေးစိတ် ကျကျ ရှင်းလင်း သွားပါမည်။ စထမဆုံး အနေဖြင့် ထည့်လိုက်သော `data` များအား ရှေ့သို့ ပိုပြီး အစဉ်လိုက် စိတ် ထားသွားမှာ ဖြစ်ပါတယ်။ ဥပမာ စထမဆုံး ထည့်သည့် `data` သည် 10 ဆိုပါစို့ ဒုတိယ ထည့်သည့် `data` သည် 20 ဆိုပါစို့ 20->10 ဖြင့် ယူ ပုံစံ အတိုင်း ထားရှိသွားမှာ ဖြစ်ပါတယ်။ 20 ဆိုတဲ့ `node` ရဲ့ `next` ထဲတွင် 10 ကို ထည့်သွားဖြစ်ပါတယ်။ အထက်အောက် ကြည့်မည် ဆိုလျှင် 20 သည် `top` ဖြစ်ပြီး 10 သည် အောက်ဆုံး ဖြစ်သည်။

အကယ်၍ 30 ဆိုသည့် `data` ထပ်ထည့်မည် ဆိုပါက LIFO အတိုင်း 30 -> 20 -> 10 စသည် ဖြင့် အစဉ်လိုက်စိသွားမှာ ဖြစ်ပါတယ်။ 30 ရဲ့ `next` ထဲတွင် 20 ရှိမည် ဖြစ်ပြီး 20 ရဲ့ `next` ထဲတွင် 10 ရှိမှာ ဖြစ်ပါတယ် ထိုကြောင့် 30 သည် `top` ဖြစ်သွားမှာ ဖြစ်ပါတယ်။



```

void push(Stack S, int n) {
    NodePtr np = (NodePtr) malloc(sizeof(Node));
    np->num = n;
    np->next = S->top;
    S->top = np;
}

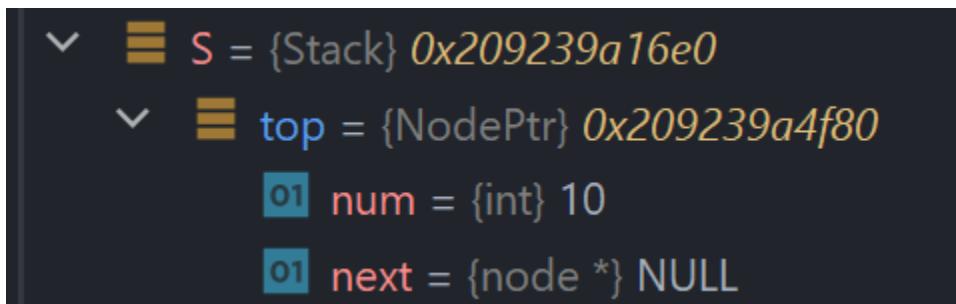
```

The screenshot shows the state of variables in a debugger. The stack variable `S` is at address `0x209239a16e0`. It contains a `top` pointer to a node. The node `np` is at address `0x209239a4f80`. It contains a `num` value of 10 and a `next` pointer to `NULL`.

အထက်ပါ ပုံအား ကြည့်ပါ `S` ဆိုသည့် `Stack` ထဲတွင် `top` ဆိုသည့် နေရာ တစ်ခု သာ ရှိနေပြီး ထို နေရာသည်လည်း `NULL` ဖြစ်နေပါသည်။ ထိုနောက် တရေးသူ အနေဖြင့် 10 ဆိုသည့် `data` အား ထည့်လိုက်ပါသည်။ ထိုကြောင့် `np` ဆိုသည့် `node` ထဲမှ `num` ထဲသို့ 10 ဆိုသည့် `data` ကို ထည့်ထား ပါသည်။ နောက်တစ်ခုတွင် `S` ထဲမှ `top` ထဲသို့ `np` ကို ထည့်ပေးလိုက်ပါသည်။

ဆိုလိုသည့်မှာ 10 ပါသော `node` အား `Stack` ထဲတွင် `top` `node` အဖြစ် သတ်မှတ် ပေးလိုက် ခြင်း ဖြစ်ပါတယ်။

အထက်ပါ အတိုင်း `push function` ထဲမှ `code` အားလုံး အလုပ် လုပ်ပြီးသွားသော အခါတွင် အောက်ပါ ပုံထဲက အတိုင်း တည်ရှိနေမှာ ဖြစ်ပါတယ်။



```

void push(Stack S, int n) {
    NodePtr np = (NodePtr) malloc(sizeof(Node));
    np->num = n;
    np->next = S->top;
    S->top = np;
}

```

ပိုမို နားလည် စေရန် 20 ဆိုသည့် data တစ်ခု ထပ်ထည့် ပါမည်။ ထို့သို့ ထပ်ထည့်ရာတွင် 20 သည် top အဖြစ် ထားရှိ ခံရမှာဖြစ်ပါတယ်။ ထို့နေက် 20 ရဲ့ next ထဲမှာမှ 10 ကို ထားရှိလိုက်မှာ ဖြစ်ပါတယ်။ အသေးစိတ် အား အောက်ပါ ပုံတွင် ကြည့်ပါ။

```

    S = {Stack} 0x209239a16e0
      top = {NodePtr} 0x209239a1720
        num = {int} 20
      next = {node *} 0x209239a4f80
        num = {int} 10
        next = {node *} NULL
  
```

နေက် တစ်င့် အနေဖြင့် 30 ဆိုသည့် data အား ထပ်ထည့် ကြည့်ပါမည် ထိုအခါ 30 သည် top အဖြစ် ထားရှိ ခြင်း ခံရမှာ ဖြစ်သလို ပြန်ထုတ် သော အခါတွင်လည်း 30 သည် ထိပ်ဆုံးမှ ထွက်လာရမှာ ဖြစ်ပါသည်။ အသေးစိတ်စား အောက်ပါ Stack DS ထဲတွင် ကြည့်ပါ။

```

    S = {Stack} 0x209239a16e0
      top = {NodePtr} 0x209239a1760
        num = {int} 30
      next = {node *} 0x209239a1720
        num = {int} 20
      next = {node *} 0x209239a4f80
        num = {int} 10
        next = {node *} NULL
  
```

Line number 29 တွင် pop function အား စခေါပါသည်။ ထို့သို့ မခေါ်ခဲ့ line 28 ၏ empty() function ကို ခေါပါသေးသည်။ empty function သည် ယခင် array ဖုန်း implement

လုပ်သည့် သင်ခန်းစာတွင် အသေးစိတ် ဖော်ပြခဲ့ပြီး ဖြစ်တဲ့ အတွက် ယခု သင်ခန်းစာ တွင် မဖော်ပြလိုကောင် empty function အား နားမလည်လျှင် ယခင် သင်ခန်းစာအား ပြန်ကြည့်ရန် လိုအပ်ပါသည်။

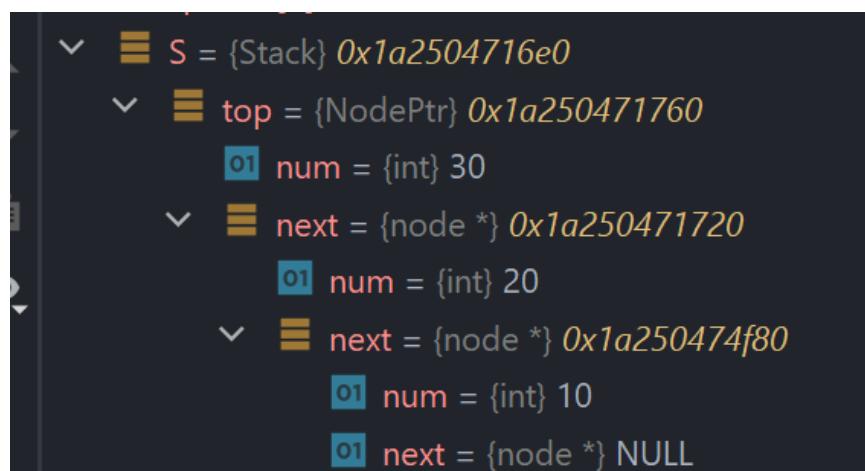
```

47  int pop(Stack S) {
48      if (empty(S)) return RogueValue;
49      int hold = S -> top -> num;
50      NodePtr temp = S -> top;
51      S -> top = S -> top -> next;
52      free(temp);
53      return hold;
54 }
```

`pop()` function မှ line 49 တွင် `S` ဆိုသည့် Stack ထဲမှ `top` node ထဲတွင် ရှိသော `num` ကို ထုတ်ပြီး `hold` ဆိုသည့် variable ထဲသို့ ထည့်ထားပါသည်။

နောက် တဆင့် အနေဖြင့် ဖြင့် `S` ထဲမှ `top` အား `temp` ဆိုသည့် `NodePtr` type ထဲသို့ ထည့်ထားသည်။

ထို့နောက် `top` အား နေရာ ချိန်းပေးရန် လိုအပ်သည် အဘယ်ကြောင့် ပထမဆုံး ဖြစ်သည့် `top` node အား ထုတ်ပြီးလျှင် ဒုတိယ `Node` အား `top` node အဖြစ် သတ်မှတ် ပေးရန် လိုအပ်သည်။ ထို့ကြောင့် Line 51 တွင် `S` stack DS ထဲတွင်ရှိသော `top` ထဲမှ `next` အား Stack ခဲ့`top` အဖြစ် သတ်မှတ် ပေးထားခဲ့ပါသည်။



Line 50 နဲ့ 51 တို့အား မ run ခင် Stack DS ထဲတွင် ရှိသော node များပုံဖြစ်ပါတယ်။

```

S = {Stack} 0x1a2504716e0
  top = {NodePtr} 0x1a250471760
    num = {int} 30
    next = {node *} 0x1a250471720
      num = {int} 20
      next = {node *} 0x1a250474f80
        num = {int} 10
        next = {node *} NULL
    hold = {int} 30
  temp = {NodePtr} 0x1a250471760
    num = {int} 30
    next = {node *} 0x1a250471720
      num = {int} 20
      next = {node *} 0x1a250474f80
        num = {int} 10
        next = {node *} NULL

```

အထက်ပါ ပုံသည် Line 50 အား run ပြီးသော အခါ Stack DS ထဲတွင် ရှိသည့် ပုံဖြစ်ပါတယ်။

temp တစ်ခု တည်ဆောက်ပြီး copy ကူးလိုက်တာက လွှဲ၍ ပြောင်းလဲမှု မရှိသေးပါ။

```

S = {Stack} 0x1a2504716e0
  top = {NodePtr} 0x1a250471720
    num = {int} 20
    next = {node *} 0x1a250474f80
      num = {int} 10
      next = {node *} NULL

```

Line 51 အား run ပြီးသော အခိုန်တွင် အထက်ပါ အတိုင်း ဖြစ်သွား ပြီး top ဖြစ်သည့် 30 အား ထုတ်လိုက်ပါပြီ။ ထို့ကြောင့် 20 သည် top ဖြစ်သွား ပါပြီ။ ထို့နောက် temp အား free လုပ်

ပစ်လိုက်ပါသည် ဤ နည်းဖြင့် Stack DS ထဲတွင်ရှိသော data များအား တစ်ခုခြင်းစီ ပြန်လည် ထုတ်ယူခြင်းကို pop လုပ်သည်ဟု ခေါ်ဆိုခြင်း ဖြစ်ပါတယ်။

```
#include <stdio.h>
#include <stdlib.h>
#define RogueValue -9595
typedef struct node {
    int num;
    struct node *next;
} Node, *NodePtr;

typedef struct stackType {
    NodePtr top;
} StackType, *Stack;

int main() {
    Stack initStack();
    int empty(Stack);
    void push(Stack, int);
    int pop(Stack);
    int n;
    Stack S = initStack();
    printf(" Press 0 if you want to quit or push some data like integers:\n");
    scanf("%d", &n);
    while (n != 0) {
        push(S, n);
        scanf("%d", &n);
    }
    printf("Numbers in reverse order\n");
    while (!empty(S))
        printf("%d ", pop(S));
    printf("\n");
} //end main
Stack initStack() {
    Stack sp = (Stack) malloc(sizeof(StackType));
    sp -> top = NULL;
    return sp;
}
int empty(Stack s) {
    return (s -> top == NULL);
}

void push(Stack s, int n) {
```

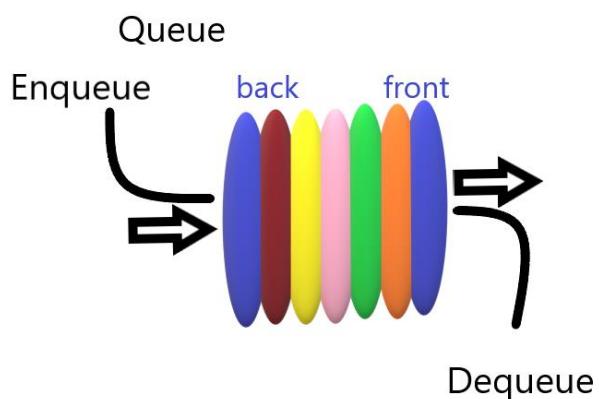
```

NodePtr np = (NodePtr) malloc(sizeof(Node));
np -> num = n;
np -> next = S -> top;
S -> top = np;
}
int pop(Stack S) {
    if (empty(S)) return RogueValue;
    int hold = S -> top -> num;
    NodePtr temp = S -> top;
    S -> top = S -> top -> next;
    free(temp);
    return hold;
}

```

<https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/stackWithLinkedList.c>

Queue



Queue သည်လည်း stack ကဲသို့ data တွေကို စနစ် တကျ store လုပ်ရန် အသုံးပြုသည့် data structure တစ်ခု ဖြစ်ပါတယ်။ Stack DS တွင် top and bottom ရှိပြီး queue မှာတော့ back front ဆိုပြီး ရှိပါတယ်။ Back ကနေ ပြီး data များထည့်တဲ့ အလုပ်ကို လုပ်ပြီး ထို back ကို rear

လိုလည်း ခေါ်ပါတယ်။ ထိုကဲ့သို့ data ထည့်ခြင်း process ကို enqueue လုပ်တယ်လို့ ခေါ်ဆိုပြီး data တွေ ပြန်ဖျက် လိုတဲ့ အခါ မှာတော့ front ကနေ ပြန်ဖျက်ပါတယ် ထိုကဲ့သို့ ပြန်ဖျက်သည့် process ကို dequeue လုပ်တယ်လို့ ခေါ်ပါတယ်။

Queue DS သည် stack နှင့် မတူဘဲ FIFO(First In First Out) ဖြစ်ပါတယ် ဥပမာ ဆိုကြပါစ္စာ bank တစ်ခုမှာ ငွေနဲ့ ပတ်သက်တဲ့ လုပ်ငန်းတွေ လုပ်ဆောင် ဖို့ လူတွေ တန်းစီ နေသလိုပါပဲ။ ပထမဆုံး တန်းစီတဲ့ တစ်ယောက်သည် ပထမဆုံး လုပ်ငန်း process တွေကို လုပ်ဆောင်နိုင်ပြီး ပထမဆုံး ထွက်သွား နိုင်မှာ ဖြစ်ပါတယ် နောက်ထပ် customer အသစ်တွေ လာတိုင်းလည်း ထိုနည်း အတိုင်း rear တွင် တန်းစီရှုမှာ ဖြစ်ပါတယ်။

Stack မှာကဲ့သို့ပင် middle မှာရှိတဲ့ data တွေကို access လုပ်ခွင့် မရှိသလို အကယ်၍ array ကိုသုံးပြီး queue ds ကို ရေးထားမယ်ဆိုလျှင်တောင် မရပါဘူး။ Stack မှာကဲ့သို့ပင် queue ကိုလည်း array သာမက linked list နှင့်လည်း implement ပြုလုပ်နိုင်ပါတယ်။

Queue DS implementation with array

```

4   #define MaxQ 10
5   typedef struct {
6   |   int head, tail;
7   |   int QA[MaxQ];
8   } QType, *Queue;
9

```

ပထမဆုံး အနေဖြင့် Qtype နှင့် *Queue ဆိုတဲ့ structure တစ်ခုကို တည်ဆောက်ပါတယ်။ ထို structure ထဲတွင် head and tail ဆိုတဲ့ data နှစ်ခု ပါဝင်သလို array ကိုသုံးပြီး ရေးသားမှာ ဖြစ်တဲ့ အတွက် QA ဆိုတဲ့ array တစ်ခု ပါဝင်ပါတယ်။ array size ကိုတော့ symbolic constant ဖြင့် သုံးထားပြီး မိမိတို့ လိုသလို ပြုပြင်နိုင်ပါတယ်။

```

43 int main() {
44     int n;
45     Queue Q = initQueue();
46     printf("Enter a positive integer: ");
47     scanf("%d", &n);
48     while (n > 0) {
49         enqueue(Q, n % 10);
50         n = n / 10;
51     }
52     printf("\nDigits in reverse order: ");
53     while (!empty(Q))
54     {
55         printf("%d", dequeue(Q));
56     }
57 }
```

ပထမဆုံး အနေဖြင့် line 45 တွင် initQueue ဆိုသည့် function ကို စခေါပါတယ်။

```

10 Queue initQueue() {
11     Queue qp = (Queue) malloc(sizeof(QType));
12     qp -> head = qp -> tail = 0;
13     return qp;
14 }
```

initQueue function သည် Queue type ကိုသုံးထားတာ ဖြစ်သည့် အတွက် return ပြန်ရောက်လည်း Queue type ဖြင့် သာ return ပြန်မည် ဖြစ်သည်။ initQueue function ထဲတွင် Queue ရဲ့ type ကိုသုံးပြီး QType ရဲ့ size ကိုသုံးကာ memory တွင် နေရာ တစ်ခု စယူပါတယ် ထိုနေရာအား qp ဟု နံမည် ပေးလိုက်ပြီး ထို qp DS ထဲမှ head and tail အား zero value များ အဖြစ် သတ်မှတ် ပေးထားလိုက်ပါတယ်။

```

    qp = {Queue} 0x251bffc16e0
        head = {int} 0
        tail = {int} 0
    QA = {int [10]}
        [0] = {int} -1163005939
        [1] = {int} -1163005939
        [2] = {int} -1163005939
        [3] = {int} -1163005939
        [4] = {int} -1163005939
        [5] = {int} -1163005939
        [6] = {int} -1163005939
        [7] = {int} -1163005939
        [8] = {int} -1163005939
        [9] = {int} -1163005939

```

Line 12 ထိ run ပြီး သွားသော အခါတွင် qp DS ထဲ၌ အထက်ပါ ပုံအတိုင်း တည်ရှိနေမှာ ဖြစ်ပါတယ်။ head and tail ရဲ့ အသုံးပြုပုံကိုတော့ enqueue function တွင် data များ ထည့်သော အခါ ရှင်းလင်းမည် ဖြစ်သည်။

Main function ထဲမှ line 49 တွင် enqueue function ကို စခေါပါတယ်။ ထိုသို့ ခေါ်ပေါ်တွင် user ထည့်ပေးလိုက်သော data အား remainder ရှာပြီးမှ ခေါ်ဆိုခြင်း ဖြစ်ပါတယ်။ ဥပမာ user မှ 123 ဆိုသည့် data ကို ထည့်ပေးလိုက်မည်ဆိုလျှင် ၁၀ နဲ့ စားပြီး အကြွင်း ဖြစ်သည့် ၃ ကိုသာ ထည့်ပေးလိုက်မှာ ဖြစ်ပါတယ်။

```

18 void enqueue(Queue Q, int n) {
19     if (Q -> tail == MaxQ - 1) {
20         Q -> tail = 0;
21     }
22     else {
23         ++(Q->tail);
24     }
25     if (Q -> tail == Q -> head) {
26         printf("\nQueue is full\n");
27         exit(1);
28     }
29     Q -> QA[Q -> tail] = n;
30 }
```

.Enqueue function ထဲသို့ ရောက်သော အခါ Queue DS ထဲမှာ ရှိတဲ့ data တွေ ပြည့်နေသလား ဆိတာကို အရင် စစ်ဆေးပါတယ်။ ထိုသို့ စစ်ဆေးရာတွင် $Q \rightarrow tail$ ဆိတဲ့ data နဲ့ $MaxQ - 1$ တို့အား တူသလား ဆိုပြီး စစ်ဆေးတာ ဖြစ်ပါတယ် ထိုသို့ စစ်ဆေး ရခြင်းမှာ Q ထဲမှာရှိတဲ့ tail အား တစ်ကြိမ် data ထည့်တိုင်း 1 ပေါင်းထားသော ကြောင့် ဖြစ်ပါတယ်။ data တစ်ခါ ထည့်တိုင်း 1 တစ်ခါ ပေါင်းသွား သောကြောင့် $MaxQ - 1$ နှင့် တူလာလျှင် Q ထဲက tail ကို 0 ထည့်ပေးလိုက်တော့မှာ ဖြစ်ပါတယ်။ ထိုသို့ ထည့်ပေးလိုက်သော အခါတွင် $head$ သည်လည်း 0 ဖြစ်သွားသလို tail သည်လည်း 0 ဖြစ်သွားသည့် အတွက် line 25 and 26 တွင် Queue DS အား data ပြည့်သွား သည်ဟု မှတ်ယူသွားပြီး program မှ ထွက်သွားမှာ ဖြစ်ပါတယ်။

```

    void enqueue(Queue Q, int n) {
        if (Q->tail == MaxQ - 1) {
            Q->tail = 0;
        } else {
            ++(Q->tail);
        }
        if (Q->tail == Q->head) {
            printf(format: "\nQueue is full\n");
            exit(Code: 1);
        }
        Q->QA[Q->tail] = n;
    }
}

```

အထက်ပါ ပုံတွင် ကြည့်ပါ $Q \rightarrow tail$ နှင့် $MaxQ-1$ တို့ မတူညီ နေသေးသည့် အတွက် line 23 ကို အလုပ် လုပ်သွားပြီး $tail$ ထဲမှာ 1 ပေါင်းသွားကြောင်းကို ပုံရဲ့ ညာဘက် ချမ်းမှာ တွေ့မြင်နိုင်ပါတယ်။

Line 29 တွင် n ဆိုသည့် data အား $Q \rightarrow QA[1]$ ဆိုသည့် နေရာသို့ထည့်ပေးလိုက်ပါတယ်။ ယခု သင်ခန်းစာတွင် n သည် 123 မှ remainder 3 ဖြစ်သည့် အတွက် Line 29 အား run ပြီးသွားလျှင် အောက်ပါ ပုံအတိုင်း အသေးစိတ် မြင်တွေ့နိုင်ပါတယ်။

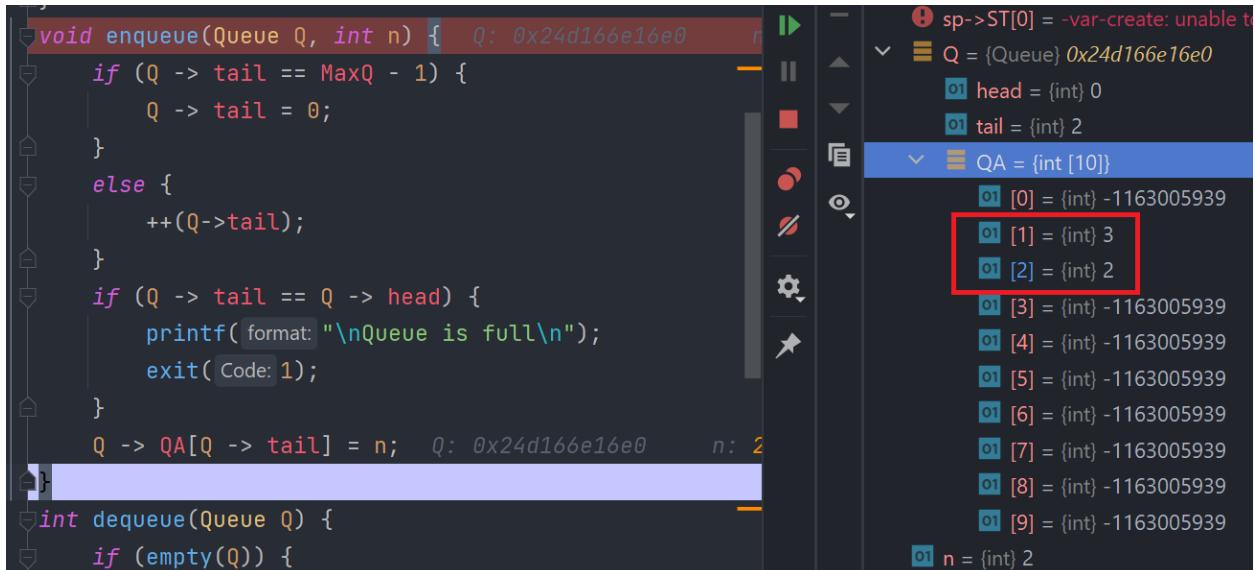
```

    Q = {Queue} 0x24d166e16e0
        head = {int} 0
        tail = {int} 1
    QA = {int [10]}
        [0] = {int} -1163005939
        [1] = {int} 3
        [2] = {int} -1163005939
        [3] = {int} -1163005939
        [4] = {int} -1163005939
        [5] = {int} -1163005939
        [6] = {int} -1163005939
        [7] = {int} -1163005939
        [8] = {int} -1163005939
        [9] = {int} -1163005939
    n = {int} 3

```

QA ဆိတဲ့ array ရဲ့ index 1 နေရာတွင် 3 ဆိုသည့် data ကို ထည့်ထားလိုက်တာ ဖြစ်ပါတယ်။ line number 48,49,50 တို့တွင် n တန်ဖိုးသည် zero ထက် ကြီးတဲ့ အချိန်ထိ အလုပ်လုပ်မည်ဟု ဆိုထားသောကြောင့် enqueue function ကို ဆက်လက် အလုပ်လုပ်ရန် ကျွန်ုန်းဖော်ပါသေးသည်။ line 50 တွင် 123 အား 10 ဖြင့် စားထားသောကြောင့် float တန်ဘိုး မယူလျှင် n တန်ဖိုးသည် 12 အဖြင့် ကျွန်ုန်းဖော်မည်။ ယခု အခါတွင် 12 အား 10 ဖြင့် စားကာ အကြွင်း ပြန်ရှာ ဦးမည် ဖြစ်သောကြောင့် enqueue ကို ပြန်ခေါ်သော အခါ 2 ဖြင့် ပြန်ခေါ်မည် ဖြစ်သည်။ Index ကို 1 နေရာကနေ စတည့်ပါတယ် ထို့သို့ ထည့်ရခြင်း ရည်ရွယ်ချက်မှာ index 0 နေရာကိုလွှတ်ထားပြီး data တွေ ပြည့်နေလား သို့မဟုတ် လွှတ်နေလား ဆိုတာကို စစ်ဆေးရန် ဖြစ်ပါတယ်။

ဒုတိယ တစ်ကြိမ် အလုပ် ပြန်လုပ်သော အခါတွင်လည်း Q-> tail နဲ့ MaxQ-1 တို့ မတူ
သေးသောကြောင့် Q-> tail အား တစ်ထပ် ပေါင်းပါသည်။ ထို့နောက် line 29 တွင် Q-> QA[2]
နေရာအား 2 ကို ထည့်လိုက်မည်ဖြစ်သည်။



```
void enqueue(Queue Q, int n) { Q: 0x24d166e16e0
    if (Q->tail == MaxQ - 1) {
        Q->tail = 0;
    }
    else {
        ++(Q->tail);
    }
    if (Q->tail == Q->head) {
        printf(format: "\nQueue is full\n");
        exit(Code: 1);
    }
    Q->QA[Q->tail] = n; Q: 0x24d166e16e0 n: 2
}
int dequeue(Queue Q) {
    if (empty(Q)) {

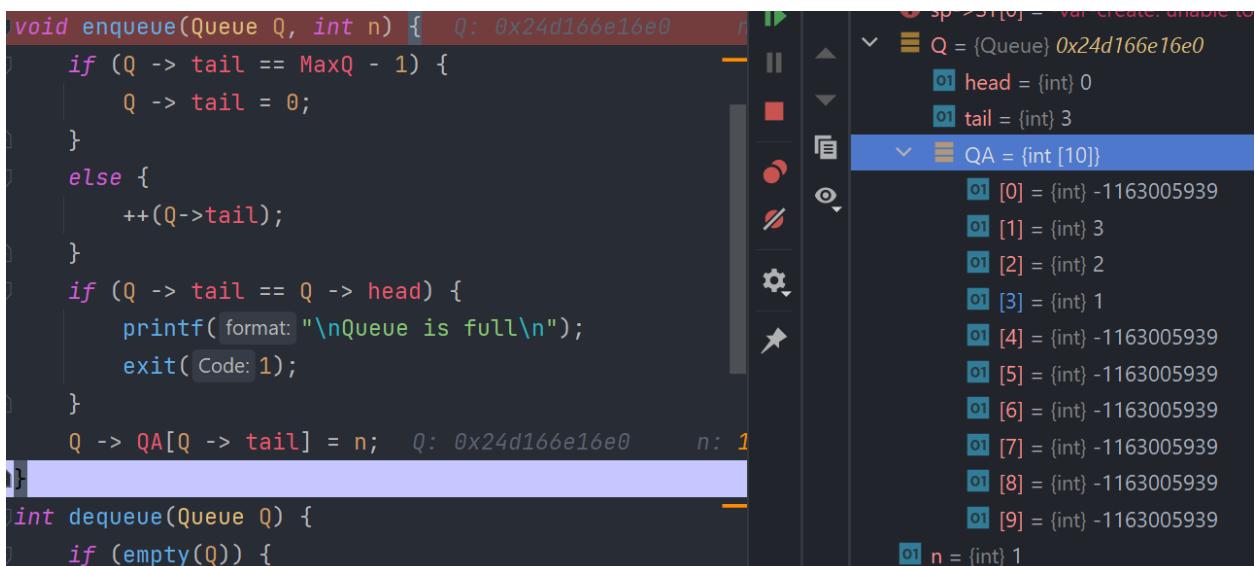
```

QA = {int [10]}

- [0] = {int} -1163005939
- [1] = {int} 3
- [2]** = {int} 2
- [3] = {int} -1163005939
- [4] = {int} -1163005939
- [5] = {int} -1163005939
- [6] = {int} -1163005939
- [7] = {int} -1163005939
- [8] = {int} -1163005939
- [9] = {int} -1163005939

n = {int} 2

နောက်တစ်ဆင့် တွင်မှ 12 ကို remainder ရှာပြီး 1 ဆိုသည့် data အား ထပ်မံ ထည့်လိုက်မှာ
ဖြစ်ပါတယ်။



```
void enqueue(Queue Q, int n) { Q: 0x24d166e16e0
    if (Q->tail == MaxQ - 1) {
        Q->tail = 0;
    }
    else {
        ++(Q->tail);
    }
    if (Q->tail == Q->head) {
        printf(format: "\nQueue is full\n");
        exit(Code: 1);
    }
    Q->QA[Q->tail] = n; Q: 0x24d166e16e0 n: 1
}
int dequeue(Queue Q) {
    if (empty(Q)) {

```

QA = {int [10]}

- [0] = {int} -1163005939
- [1] = {int} 3
- [2] = {int} 2
- [3]** = {int} 1
- [4] = {int} -1163005939
- [5] = {int} -1163005939
- [6] = {int} -1163005939
- [7] = {int} -1163005939
- [8] = {int} -1163005939
- [9] = {int} -1163005939

n = {int} 1

အထက်ပါ အဆင့်များ ပြီးသွား သော အခါ main function ထဲမှ Line 55 တွင် dequeue()
function ကို ပြန်ခေါ်ပေးသည်။ Dequeue function သည် queue DS ထဲတွင်ရှိတဲ့ data တွက်

အစဉ်လိုက် ပြန်ထဲတ် ပေးရန် ဖြစ်သည် အစဉ်လိုက်ဆိုသော နေရာတွင် ပထမဆုံး ဝင်သည့် data အား ပထမဆုံး ထဲတ်ပြီး FIFO(First In First Out) စနစ်ဖြင့် ထဲတ်သွားမှာ ဖြစ်ပါတယ်။

```

31     int dequeue(Queue Q) {
32         if (empty(Q)) {
33             printf("\nAttempt to remove from an empty queue\n");
34             exit(1);
35         }
36         if (Q -> head == MaxQ - 1) {
37             Q -> head = 0;
38         }
39         else ++(Q -> head);
40         return Q -> QA[Q -> head];
41     }

```

dequeue function သည် Queue type ဖြင့် queue DS တစ်ခုကို ယူပြီး data များ အစဉ်လိုက် ထဲတ်ပေးသည့် လုပ်ငန်းကို လုပ်ဆောင်မှာ ဖြစ်ပါတယ်။ ပထမဆုံး အနေဖြင့် stack DS အတိုင်း empty ဖြစ်နေသလား စစ်ဆေးပါတယ် empty ဖြစ်မနေဘဲ data များ ရှိနေမှုသာလျှင် ကျွန်ုတ် လုပ်ငန်းများ ကို ဆက်လက် လုပ်ဆောင်မှာ ဖြစ်ပါတယ်။

line 36 တွင် Q ထဲမှာ ရှိတဲ့ head နဲ့ MaxQ-1 မှာ ရှိတဲ့ data များအား တူညီနေသလား စစ်ဆေးပါတယ် အကယ်၍ တူညီ နေခဲ့လျှင် Q -> head ထဲသို့ zero ကို ထည့်ပေးလိုက်ပါတယ်။ အထက်ပါ စစ်ဆေးချက် အတိုင်း မတူနေလျှင် Q ထဲမှ head အား 1 ပေါင်းထည့်ပါတယ်။ ထိုသို့ ထည့်ပေးရခြင်းမှာ ပေါင်းထည့်၍ ရလာသော number အား index number အဖြစ် QA array ထဲမှ data များကို ပြန်လည် ထဲတ်ယူသော အခါ အသုံးပြုရန် ဖြစ်သည်။

```

int dequeue(Queue Q) {    Q: 0x25d869716e0
    if (empty(Q)) {
        printf(format: "\nAttempt to remove from an empty queue");
        exit(Code: 1);
    }
    if (Q -> head == MaxQ - 1) {
        Q -> head = 0;
    }
    else ++(Q -> head);
    return Q -> QA[Q -> head];    Q: 0x25d869716e0
}

int main() {
    int n;
    Queue Q = initQueue();
}

```

အထက်ပါ အဆင့်သည် ပထမဆုံး ထည့်လိုက်သော data ဖြစ်သည့် 3 အား ပြန်လည်
ထုတ်ယူသည့် ပုံဖြစ်သည်။

```

int dequeue(Queue Q) {    Q: 0x25d869716e0
    if (empty(Q)) {
        printf(format: "\nAttempt to remove from an empty queue");
        exit(Code: 1);
    }
    if (Q -> head == MaxQ - 1) {
        Q -> head = 0;
    }
    else ++(Q -> head);
    return Q -> QA[Q -> head];    Q: 0x25d869716e0
}

int main() {
    int n;
}

```

အထက်ပါ အဆင့်သည်လည်း ဒုတိယ ထည့်လိုက်သော data ဖြစ်သည့် 2 အား
ပြန်ထုတ်ယူသော အဆင့်ဖြစ်သည် 1 အားလည်း ထိန်ည်းတူ ပြန်လည် ထုတ်ယူမှာ ဖြစ်ပါတယ်။

Sample program:

```

#include <stdlib.h>
#include <stdio.h>
#define MaxQ 10

typedef struct {
    int head, tail;
}

```

```
int QA[MaxQ];
} QType, *Queue;// int a , *b;

Queue initQueue() {
    Queue qp = (Queue) malloc(sizeof(QType));
    qp -> head = qp -> tail = 0;
    return qp;
}

int empty(Queue Q) {
    return (Q -> head == Q -> tail);
}
void enqueue(Queue Q, int n) {
    if (Q -> tail == MaxQ - 1) Q -> tail = 0;
    else ++(Q -> tail);
    if (Q -> tail == Q -> head) {
        printf("\nQueue is full\n");
        exit(1);
    }
    Q -> QA[Q -> tail] = n;
}
int dequeue(Queue Q) {
    if (empty(Q)) {
        printf("\nAttempt to remove from an empty queue\n");
        exit(1);
    }
    if (Q -> head == MaxQ - 1) Q -> head = 0;
    else ++(Q -> head);
    return Q -> QA[Q -> head];
}

int main() {
    int n;
    Queue Q = initQueue();
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    while (n > 0) {
        enqueue(Q, n % 10);
        n = n / 10;
    }
    printf("\nDigits in reverse order: ");
    while (!empty(Q))
        printf("%d", dequeue(Q));
    printf("\n");
}
```

Queue implementation with Linked List

ယခု သင်ခန်းစာမျာတော့ Queue DS ကိုပဲ linked list ကိုသုံးပြီး implement လုပ်ပြသွားမှာဖြစ်ပါတယ်။ ပထားး အနေဖြင့် queue ထဲတွင် ထည့်မည့် data အတွက် structure တစ်ခုတည်ဆောက်ပါမည်။ ထို့နောက် node ကို တည်ဆောက်ပါမည် နောက်တစ်ဆင့် အနေဖြင့် queueType တို့ကို တည်ဆောက်မှာ ဖြစ်ပါတယ်။

```

3   typedef struct {
4       int num;
5   } QueueData;
6
7   typedef struct node {
8       QueueData data;
9       struct node *next;
10  } Node, *NodePtr;
11
12  typedef struct queueType {
13      NodePtr head, tail;
14  } QueueType, *Queue;
```

အထက်ပါ အဆင့်များ ပြီးပါက main function ကို စတင်ရေးသားမှာ ဖြစ်ပါတယ်။

```

53 int main() {
54     int n;
55     QueueData temp;
56     Queue Q = initQueue();
57     printf("Enter a positive integer: ");
58     scanf("%d", &n);
59     while (n > 0) {
60         temp.num = n % 10;
61         enqueue(Q, temp);
62         n = n / 10;
63     }
64     printf("\nDigits in reverse order: ");
65     while (!empty(Q))
66         printf("%d", dequeue(Q).num);
67     printf("\n");
68 } //end main

```

ပထမဆုံး အနေဖြင့် line 56 တွင် initQueue() ဆိုသည့် function ကို စခေါ်ပါတယ်။

```

17 Queue initQueue() {
18     Queue qp = (Queue) malloc(sizeof(QueueType));
19     qp -> head = NULL;
20     qp -> tail = NULL;
21     return qp;
22 }

```

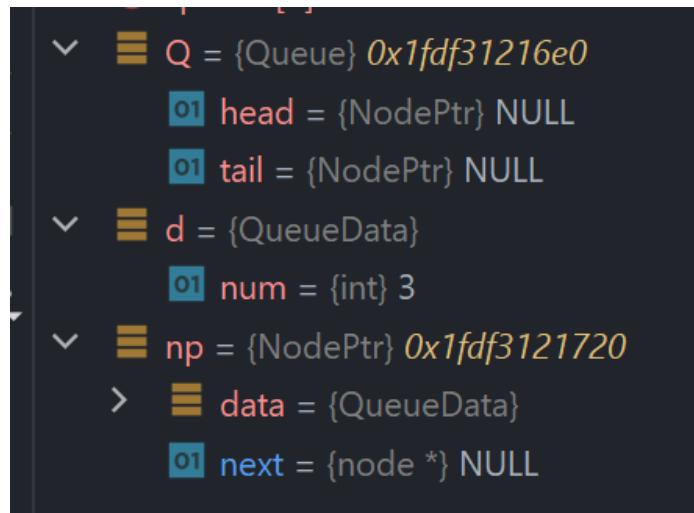
initQueue() function တဲ့တွင် ပထမဆုံး အနေဖြင့် array lesson သင်ခန်းစာများ ကဲ့သို့ QueueType ကိုသုံးပြီး memory ပေါ်မှာ နေရာ တစ်ခု စယူပါတယ်။ ထို့နောက် ငှါး qp DS တဲ့ head and tail ထဲသို့ NULL များကို ထည့်ထားပါတယ်။

ထို့နောက် main function တဲ့မှ line 61 တွင် enqueue function ကို ခေါ်ထားပါတယ်။ စာရေးသူ အနေဖြင့် 123 ဆိုတဲ့ data ကို ထည့်မှာ ဖြစ်တဲ့ အတွက် 321 ဆိုပြီး data များ ဝင်သွားမှာ ဖြစ်ပါတယ်။

```

27 void enqueue(Queue Q, QueueData d) {
28     NodePtr np = (NodePtr) malloc(sizeof(Node)
29     np -> data = d;
30     np -> next = NULL;
31     bool data = empty(Q);
32     if (data) {
33         Q -> head = np;
34         Q -> tail = np;
35     }
36     else {
37         Q -> tail -> next = np;
38         Q -> tail = np;
39     }
40 }
```

enqueue function ထဲတွင်လည်း np ဆိုသည့် node pointer တစ်ခု တည်ဆောက် ပါတယ်
ပြီးလျှင် np ထဲမှ data ထဲသို့ d ကို ထည့်ပြီး next ထဲသို့ NULL ကို ထည့်ထားပါတယ်။ အထက်ပါ
ပုံတွင် line number 30 ထိ ပြီးသော အခါ အောက်ပါ ပုံစံ အတိုင်းသာ Q DS ထဲတွင် တည်ရှိနေမှာ
ဖြစ်ပါတယ်။



ထို့နောက် Line number 31 တွင် empty ဖြစ်နေသလား ဆိုတာကို စတင် စစ်ဆေးပါတယ်။
ထို့သို့ စစ်ဆေးဖို့ အတွက် empty function ကို ခေါ်ပါတယ် empty function သည် သူကို ခေါ်သော
အခါတွင် head အတွင်း၌ NULL ဖြစ်နေသလားကို စစ်ဆေးပါတယ်။ ထိုကဲ့သို့ စစ်ဆေး သော အခါ Q-

>head သည် NULL ဖြစ်နေလျှင် true ကို ပြန်ပေးမှာ ဖြစ်ပါတယ် ဆိုလိုသည်မှာ Q DS ထဲက head သည် NULL ဖြစ်နေလျှင် Q DS သည် empty ဖြစ်နေမှာ ပါ ထိုကြောင့် line 31 တွင် true ကို ပြန်ရမှာ ဖြစ်ပါတယ်။ line 32 တွင် if condition ထဲမှာ true ဖြစ်နေသည့် အတွက် ကျွန်ုတ် Line များကို အလုပ် ဆက်လုပ်သွားမှာဖြစ်ပါတယ်။

အချို့ကြီး သတိပြုရမည့် အချက် ဖော်ပြန်ခြင်းကို ဖြစ်နေသော အချိန်တွင် head and tail နှစ်ခုလုံးကို np ဆိုသည့် အသစ်ဆောက်လိုက်သော node pointer ထည့်ပေးထားပါတယ်။ ဆိုသော် ဒုတိယ အကြိမ် data ထည့်သော အခါ Q DS သည် empty ဖြစ်မင်္ဂလာ တော့သည့် အတွက် Q->tail ->next ထဲသို့ node pointer ကို ထည့်ပေးလိုက်ပါတယ်။ ထိုနောက် Q->tail ကိုလည်း node pointer ထပ်ထည့် ပေးပါသေးတယ်။ သတိပြုရန်မှာ Q DS ထဲတွင်ရှိသော head and tail တို့သည် node pointer တစ်ခုတည်းကိုသာ ဖော်ပြန်ခြင်းကိုသာ ပေးပါတယ်။ data ထည့်သော အခါတွင် ထည့်ခံ ခဲ့ရသည့် အတွက် memory address များ တူညီနေမှာ ဖြစ်ပါတယ်။ အသေးစိတ်ကို အောက်ပါပုံတွင် ကြည့်ပါ Q DS ထဲမှ head and tail နှစ်ခုလုံးသည် memory address တူညီနေသည်ကို တွေ့ရမည်။

```

>   Q->tail->data = {QueueData}
<
>     Q = {Queue} 0x217f08516e0
>       head = {NodePtr} 0x217f0851720
>         data = {QueueData}
>           num = {int} 3
>         next = {node *} 0x217f0851760
>           data = {QueueData}
>             num = {int} 2
>             next = {node *} NULL
>       tail = {NodePtr} 0x217f0851720
>         data = {QueueData}
>           num = {int} 3
>         next = {node *} 0x217f0851760
>           data = {QueueData}
>             num = {int} 2
>             next = {node *} NULL
>   d = {QueueData}
>   np = {NodePtr} 0x217f0851760
>     data = {bool} false

```

အထက်ပါ ပုံသည် Line number 37 ကို run ပြီးသွားသော အခါ Q DS တဲတွင် ရှိသည့် node များပုံ ဖြစ်ပါတယ်။

သို့သော် line number 38 ကို run ပြီးသွားသော အခါတွင် tail အပိုင်းတွင်ရှိသော nodes များ မှာ ပြောင်းလဲသွား မှာ ဖြစ်ပါတယ် အဘယ်ကြောင့် ဆိုသော် Linked list တွင် node အားလုံးကို head ကနေ တဆင့် အစဉ်တိုင်း ချိတ်ဆက်ထားသောကြောင့်ဖြစ်သည်။ အောက်ပါပုံသည် Line 38 ကို run ပြီးသွားသော အခါ ပြောင်းလဲ သွားသည့် ပုံ ဖြစ်ပါတယ်။

```

>   └─ Q->tail->data = {QueueData}
└─ Q = {Queue} 0x217f08516e0
  └─ head = {NodePtr} 0x217f0851720
    └─ data = {QueueData}
      └─ num = {int} 3
  └─ next = {node *} 0x217f0851760
    └─ data = {QueueData}
      └─ num = {int} 2
      └─ next = {node *} NULL
  └─ tail = {NodePtr} 0x217f0851760
    └─ data = {QueueData}
      └─ num = {int} 2
      └─ next = {node *} NULL

```

321 ဆိုသည့် data အားလုံး ထည့်ပြီးသည့် အခါ Q DS ထဲတွင် အထက်ပါ ပုံစံက အတိုင်း nodes များ တည်ရှိနေမှာ ဖြစ်ပါတယ်။

[Dequeue with Linked list](#)

Main function ထဲမှ Line number 66 တွင် dequeue function ကို စခေါပါတယ်။

```

41 QueueData dequeue(Queue Q) {
42     if (empty(Q)) {
43         printf("\nAttempt to remove from an empty queue\n");
44         exit(1);
45     }
46     QueueData hold = Q -> head -> data;
47     NodePtr temp = Q -> head;
48     Q -> head = Q -> head -> next;
49     if (Q -> head == NULL) Q -> tail = NULL;
50     free(temp);
51     return hold;
52 } //end dequeue

```

Queue function ထဲတွင် ပထမဆုံး အနေဖြင့် Q DS သည် empty ဖြစ်နေသလား ဆိုတာကို စတင် စစ်ဆေးပါတယ်။ အကယ်၍ Q DS ထဲတွင် မည့်သည့် nodes မှာ ရှိမနေဘူးဆိုလျှင်တော့ program ကို ထွက်လိုက်ပါတယ်။ (line 44)

Dequeue လုပ်မည့် algorithms သည် ပထမဆုံး Q DS ထဲမှ ပထမဆုံး data ကို ထုတ်ပြီး သိမ်းထားမှာ ဖြစ်ပါတယ်။ (line 46)

နောက်တဆင့်မှာတော့ head ကို နေရာ ချိန်းဖို့ အတွက် Q DS ထဲမှ head ကို temp ထဲသို့ ထည့်ထားမှာ ဖြစ်ပါတယ်။ (line 47)

ယခု အဆင့်မှာတော့ နေရာ စချိန်းပါပြီ `Q->head->next` ကို `Q -> head` ထဲသို့ စထည့်မှာ ဖြစ်ပါတယ်။ (line 48)

ထို ကဲ့သို့ အဆင့်ဆင့် ပြုလုပ် လာရင်းဖြင့် `Q->head` သည် `NULL` ဖြစ်သွားသော အခါတွင် `Q->tail` ကိုလည်း `NULL` အဖြစ်သတ်မှတ် ပေးလိုက်မှာ ဖြစ်ပါတယ်။ (line 49)

ထို့နောက် `temp` ကို free function သုံးပြီး ဖျက်လိုက်ပါတယ်။ နောက်ဆုံးမှာတော့ `node` ထဲက ထုတ်ထားတဲ့ `data` ကို return ပြန် ပေးလိုက်ပါတယ်။ (line 50 and 51)

```

>   temp = {QueueData}
< 
  v   Q = {Queue} 0x1a556e216e0
    < 
      head = {NodePtr} 0x1a556e21760
        < 
          data = {QueueData}
            01 num = {int} 2
        < 
          next = {node *} 0x1a556e217a0
            < 
              data = {QueueData}
                01 num = {int} 1
                01 next = {node *} NULL
            < 
              tail = {NodePtr} 0x1a556e217a0
                < 
                  data = {QueueData}
                    01 num = {int} 1
                    01 next = {node *} NULL

```

အထက်ပါ ပုံသည် 3 ဆိုသည့် `data` အား ပြန်လည် ထုတ်ယူထားသည့် ပုံ ဖြစ်ပါတယ်။

```

        exit( Code: 1);
    }
    QueueData hold = Q -> head -> data;  hold: QueueData
    NodePtr temp = Q -> head;  temp: 0x1a556e21760
    Q -> head = Q -> head -> next;  Q: 0x1a556e216e0
    if (Q -> head == NULL) Q -> tail = NULL;
    free( Memory: temp);
    return hold;
} //end dequeue

```

ယခု ပုံသည် ဒုတိယ အကြမ် data ထုတ်ပြီး head မချိန်းခင် ပုံဖြစ်ပါတယ်။ လက်ရှုမှု head ရဲ့ memory address သည် 21760 ဖြစ်ပြီး ချိန်ပြီးသွားသော အခါ 217a0 ဖြစ်သွားမှာပါ။

```

        exit( Code: 1);
    }
    QueueData hold = Q -> head -> data;  hold: QueueData
    NodePtr temp = Q -> head;  temp: 0x1a556e217a0
    Q -> head = Q -> head -> next;
    if (Q -> head == NULL) Q -> tail = NULL;
    free( Memory: temp);
    return hold;
} //end dequeue

```

အထက်ပါ နည်းအတိုင်း head များကို နေရာ ခိုင်းသွားကာ အကယ်၍ head ထဲတွင် မည့်သည့် data မှု မရှိသော အခိုင်းသော ရောက်သော အခါတွင်မူ tail ထဲသို့ NULL ကို ပါ ထည့်ပေးလိုက်မှာ ဖြစ်ပါတယ်။

```

QueueData hold = Q -> head -> data;  hold: QueueData
NodePtr temp = Q -> head;  temp: 0x1a556e217a0
Q -> head = Q -> head -> next;
if (Q -> head == NULL) Q -> tail = NULL;  Q: 0x1a556e216e0
free( Memory: temp);  temp: 0x1a556e217a0
return hold;
} //end dequeue

```

အထက်ပါ ပုံသည် Q ထဲမှ tail NULL ဖြစ်သွားသည့် ပုံဖြစ်ပါတယ်။

Sample Program:

```
#include <stdlib.h>
#include <stdio.h>
typedef struct {
    int num;
} QueueData;
```

```
typedef struct node {
    QueueData data;
    struct node *next;
} Node, *NodePtr;

typedef struct queueType {
    NodePtr head, tail;
} QueueType, *Queue;

Queue initQueue() {
    Queue qp = (Queue) malloc(sizeof(QueueType));
    qp -> head = NULL;
    qp -> tail = NULL;
    return qp;
}

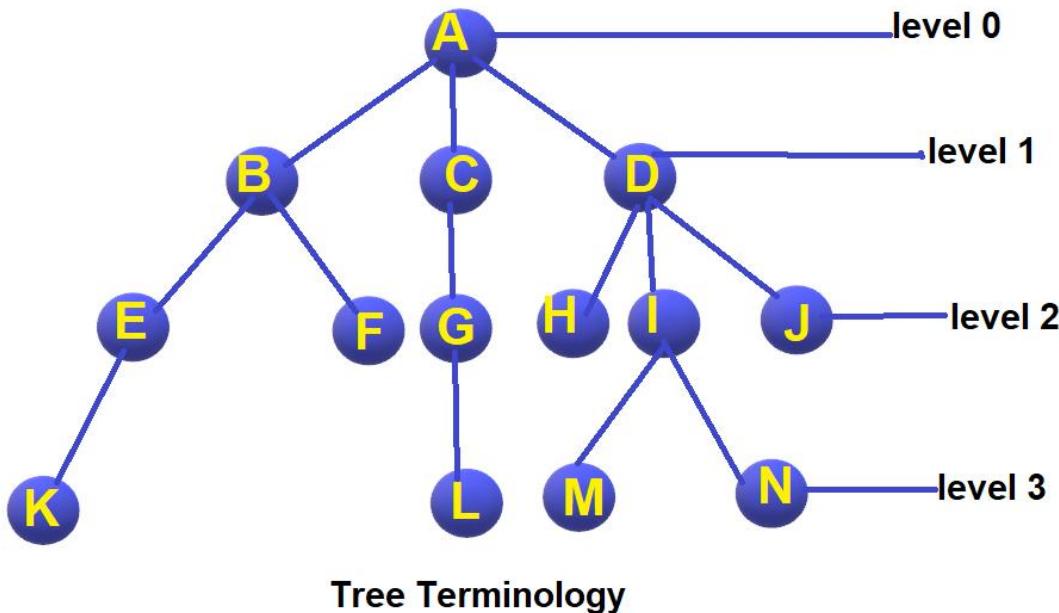
int empty(Queue Q) {
    return (Q -> head == NULL);
} //end empty
void enqueue(Queue Q, QueueData d) {
    NodePtr np = (NodePtr) malloc(sizeof(Node));
    np -> data = d;
    np -> next = NULL;
    bool data = empty(Q);
    if (data) {
        Q -> head = np;
        Q -> tail = np;
    }
    else {
        Q -> tail -> next = np;
        Q -> tail = np;
    }
}
QueueData dequeue(Queue Q) {
    if (empty(Q)) {
        printf("\nAttempt to remove from an empty queue\n");
        exit(1);
    }
    QueueData hold = Q -> head -> data;
    NodePtr temp = Q -> head;
    Q -> head = Q -> head -> next;
    if (Q -> head == NULL) Q -> tail = NULL;
    free(temp);
}
```

```
    return hold;
} //end dequeue
int main() {
    int n;
    QueueData temp;
    Queue Q = initQueue();
    printf("Enter a positive integer: ");
    scanf("%d", &n);
    while (n > 0) {
        temp.num = n % 10;
        enqueue(Q, temp);
        n = n / 10;
    }
    printf("\nDigits in reverse order: ");
    while (!empty(Q))
        printf("%d", dequeue(Q).num);
    printf("\n");
} //end main
```

Github:

<https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/QueueWithLinkedList.c>

TREE

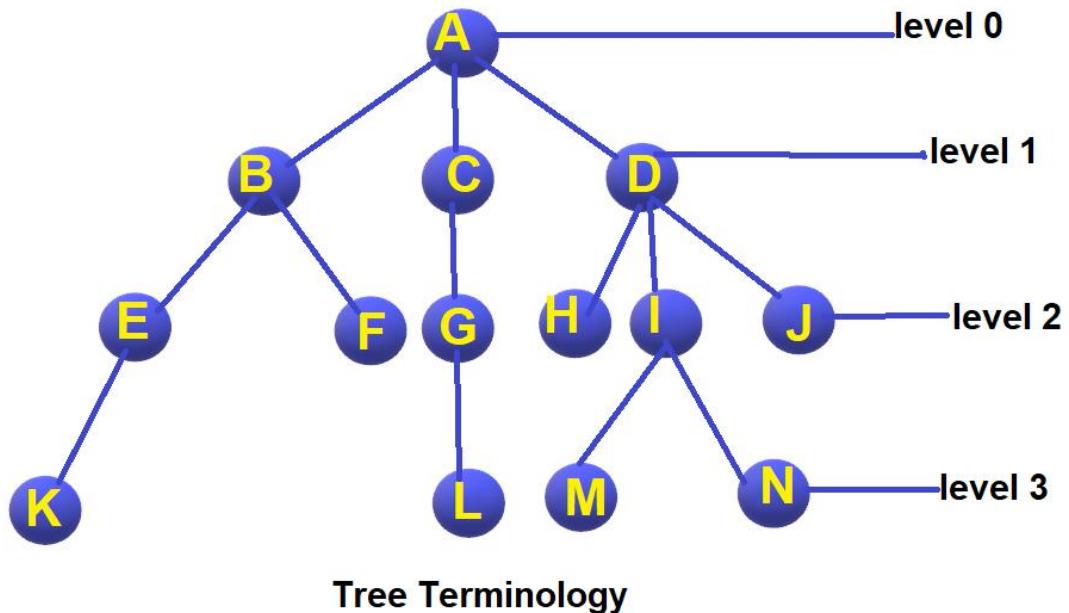


Tree Terminology

Tree သည် elements များကို parent and child ချိတ်ဆက်ပုံဖြင့် တည်ဆောက်ထားသော nonlinear data structure ဖြစ်သလို dynamic data structure လည်း ဖြစ်ပါတယ်။

- **Node :** Tree DS မှာ elements တွေကို nodes တွေလို့ခေါ်ပါတယ်။
- **Root :** Tree DS ရဲ့အစ ထိပ်ဆုံးမှာ ရှိနေတဲ့ node ကိုတော့ root node လို့ ခေါ်ဆိုပါတယ် အထက်ပါ ပုံမှာတော့ A သည် root node ဖြစ်ပါတယ်။
- **Parent :** အထက်ပါ ပုံတွင် H , I , J စသည့် nodes များအတွက် parent node သည် D ဖြစ်ပါတယ်။ တစ်နည်းအားဖြင့် parent node ကို predecessor လို့လည်း ခေါ်ပါတယ်။ E နှင့် F တို့ရဲ့ parent node သည် B ဖြစ်ပါတယ်။
- **Edge :** Nodes တစ်ခုနဲ့ တစ်ခုကြားမှာ ရှိတဲ့ ဆက်သွယ်ချက်ကို Edges လို့ ခေါ်ပါတယ်။ အထက်ပါ ပုံတွင် A နှင့် D ကြားမှာ ရှိတဲ့ ဆက်သွယ်ချက် ချက်ကို edge လို့ ခေါ်ပါတယ်။
- **Child :** H ,I , J တို့အား D ဆိုသည့် node ရဲ့ child node လို့လည်း ခေါ်သလို တစ်နည်းအားဖြင့် successor လို့လည်း ခေါ်ပါတယ်။

- **Siblings** : Parent node တစ်ခုတည်းကနေလာတဲ့ child nodes တွေကို siblings လို့ ခေါပါတယ် အထက်ပါ ပုံမှာ ဆိုလျှင် H , I , J တို့သည် siblings များ ဖြစ်ကြပါတယ်။ ထိုနည်းတူ E and F တို့သည်လည်း siblings များဖြစ်ကြပါတယ်။
- **Internal Node**: Node တစ်ခုတွင် အနဲ့ဆုံး child node တစ်ခုရှိလျှင် ငါး node အား internal node လို့ ခေါပါတယ်။ အထက်ပါ ပုံမှာ ဆိုလျှင် A , B , C , D , E , G , I တို့သည် internal node များ ဖြစ်ကြပါတယ်။
- **Leaf Node** : Node တစ်ခုတွင် မည်သည့် child nodes မူး ရှိမနေခြင်းကို leaf node လို့ ခေါပါတယ် အထက်ပါ ပုံတွင် K , F , L , H , M , N , J တို့သည့် leaf node များ ဖြစ်ကြပါတယ်။
- **Level** : Tree structure တစ်ခုမှာ ဆိုလျှင် ထိပ်ဆုံး တစ်ခုကနေစပီးအောက်ဆုံး ထိအတိုင်းအတာ ကို level ဟု ခေါပါတယ် ဥပမာ အထက်ပါ ပုံတွင် A သည် level 0 ဖြစ်ပြီး B,C,D သည် level 1 ဖြစ်ပါတယ်။ E,F,G,H,I,J တို့သည် level 2 ဖြစ်ကြပြီး K , L , M , N တို့သည့် level 3 ဖြစ်ကြပါတယ်။
- **Height of the node**: edge စုစုပေါင်း အားလုံးကို node တစ်ခုရဲ့ height ဟု ခေါ်ဆိုပါတယ် အထက်ပါ ပုံတွင် B မှ E သည် edge တစ်ခု , E မှ K သည် edge တစ်ခု ဖြစ်ပါတယ်။ K သည် leaf node ဖြစ်သွားသည့် အတွက် အဆုံး သတ်သွားပါပြီ ထို့ကြောင့် B ရဲ့ height သည် 2 ဖြစ်ပါတယ်။ K , F , L , H , M , N , J တို့ရဲ့ height ကတော့ zero ဖြစ်ပါတယ် အဘယ်ကြောင့် ဆိုသော် သူတို့သည် leaf nodes များ ဖြစ်တဲ့ အတွက်ကြောင့်ပါ။



- **Depth:** Node တစ်ခုရဲ့ depth ကို သိလိုတဲ့ အခါ root node ကနေ edge ဘယ်လေက ရှိသလဲ ဆိုတာကို တွက်ပါတယ် edge အရေး အတွက် သည် ထို node တစ်ခုရဲ့ depth ဖြစ်ပါတယ်။ အထက်ပါ ပုံတွင် B ရဲ့ depth သည် 1 ဖြစ်ပြီး K ရဲ့ depth သည် 3 ဖြစ်ပါတယ်။ Root node ဖြစ်တဲ့ A ရဲ့ depth ကတေသာ zero ဖြစ်ပါတယ်။
- **Subtree :** မူရင်း tree မှ တစ်ဆင့် နောက်ထပ် ဖြာထွက်သော trees များကို sub trees လို့ ခေါ်ပါတယ် အထက်ပါ ပုံတွင် B , C , D တို့သည် sub tree များ ဖြစ်ကြပါတယ်။
 - **Forest :** မ ဆက်ထားတဲ့ trees တွေကို forest လို့ ခေါ်ပါတယ် ဥပမာ B , C ,D တို့ ဖြစ်သည်။
- **Degree:** Node တစ်ခုရဲ့ degree နှင့် Tree တစ်ခုရဲ့ degree ဟူ၍ နှစ်မျိုး ရှိပါတယ် Node တစ်ခု ရဲ့ Degree ဆိုတာ ကတေသာ ထို node တစ်ခု မှာ ရှိတဲ့ child nodes အရောအတွက်ကို ပြောတာပါ။ ဥပမာ B ရဲ့ degree သည် 2 ဖြစ်ပြီး D ရဲ့ degree သည် 3 ဖြစ်ပါတယ် အဘယ်ကြောင့်ဆိုသော D အောက်တွင် H , I , J ဆိုသည့် child သုံးခု ရှိနေပြီး B အောက်တွင်မူ E and F ဆိုသည့် child နှစ်ခုသာ ရှိနေသောကြောင့် ဖြစ်သည်။

Tree တစ်ခုရဲ့ Degree ဆိုတာကတော့ ထို tree မှာရှိတဲ့ node တွေထဲမှာ အမြင့်ဆုံး degree ရှိတဲ့ node ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။

Inorder Traversal , Preorder Traversal , Postorder Traversal

Inorder Traversal ဆိုတာက ပထမဆုံး tree တစ်ခုရဲ့ left ဘက်ခြမ်းမှ nodes များဆီသို့ အရင်သွားပြီး ထို့နောက် root node ဆီသို့ သွားပါတယ် နောက်ဆုံးမှသာလျှင် tree ရဲ့ right ဘက်ခြမ်းမှ nodes များဆီသို့ သွားပါတယ်။

Preorder Traversal ဆိုတာကတော့ အရင်ဆုံး root node ကို သွားပါတယ် ထို့နောက် Tree ရဲ့ left ဘက်ခြမ်းကို သွားပြီး နောက်ဆုံးမှ tree ရဲ့ညာဘက်ခြမ်းကို သွားပါတယ်။

Postorder Traversal ကတော့ tree တစ်ခုမှာ ရှိတဲ့ left ဘက်က nodes တွေဆီကို အရင်သွားပါတယ် ထို့နောက် right မှ nodes တွေဆီကို အရင်သွားပါတယ် ပြီးမှ သာလျှင် root node ဆီသို့ သွားပါတယ်။

```

37 int main() {
38     struct Node *root = new Node(8);
39     root->left = new Node(10);
40     root->right = new Node(3);
41     root->right->left = new Node(17);
42     root->left->left = new Node(4);
43     root->left->right = new Node(2);
44
45     cout << "Inorder traversal ";
46     inorder(root);
47
48     cout << "\nPreorder traversal ";
49     preorder(root);
50
51     cout << "\nPostorder traversal ";
52     postorder(root);
53
54 }
```

ပထမဆုံး အနေဖြင့် အထက်ပါ ပုံအတိုင်း line 38 တွင် node အသစ် တစ်ခု စဆောက်ထား ပါတယ်။ Node ရဲ့တည်ဆောက်ပုံမှာ အောက်ပါ အတိုင်း ဖြစ်ပါတယ်။

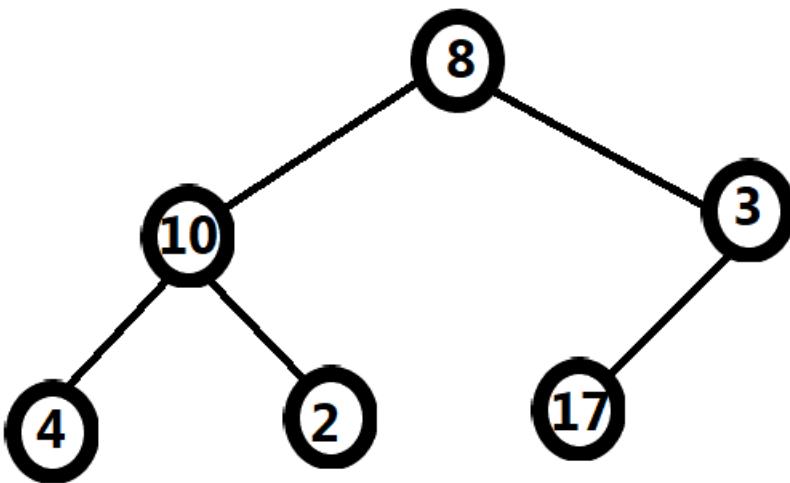
```

4   struct Node {
5       int data;
6       struct Node *left, *right;
7       Node(int data) {
8           this->data = data;
9           left = right = NULL;
10      }
11  };

```

Node တည်ဆောက်ရန် structure ကိုသုံးထားပြီး data တစ်ခု ပါဝင်ပါတယ်။ ထို့နောက် Node ထဲတွင် Left and Right ဆိုသည့် Node type ဖြင့် နှစ်ခု ထပ်ပါ ပါတယ်။ Line 8 မှ this သည် Node ကိုပဲ ပြန်ပြီး ရည်ညွှန်းချင်တာ ဖြစ်ပါတယ်။ ပထမဆုံး စဆောက် သည့် အချိန်များတွင် left and right ထဲသို့ NULL များသာ ထည့်ထားပါတယ်။

Main function ရဲ line 39 မှ 43 ထိုသည် data များ ထည့်ပေးထားခြင်း ဖြစ်ပါတယ်။ data များ ထည့်ထားပုံမှာ အောက်ပါ Tree Structure အတိုင်း ဖြစ်ပါတယ်။



```

    root = {Node *} 0x1d3b9951eb0
        data = {int} 8
    left = {Node *} 0x1d3b9951c90
        data = {int} 10
    left = {Node *} 0x1d3b99541f0
        data = {int} 4
        left = {Node *} NULL
        right = {Node *} NULL
    right = {Node *} 0x1d3b9954240
        data = {int} 2
        left = {Node *} NULL
        right = {Node *} NULL
    right = {Node *} 0x1d3b9951ce0
        data = {int} 3
    left = {Node *} 0x1d3b99541a0
        data = {int} 17
        left = {Node *} NULL
        right = {Node *} NULL
        right = {Node *} NULL

```

ထို့နောက် main function တဲ့မှ line 46 တွင် inorder function ကို စခေါပါတယ်။ inorder function ကို စခေါ်သော အခါ အတွင်း root တစ်ခုလုံးကို parameter အနေဖြင့် ထည့်ပေးလိုက်ပါတယ်။

```

21 void inorder(struct Node* node) {
22     if (node == NULL)
23         return;
24
25     inorder(node->left);
26     cout << node->data << "->";
27     inorder(node->right);
28 }

```

Inorder function ထဲတွင် ပထမဆုံး အနေဖြင့် ထည့်ပေးလိုက်သော root သည် NULL ဖြစ်နေသလား မဖြစ်ဘူးလား ဆိုတာကို စစ်ဆေးပါတယ်။ ထို့နောက် inorder function ကို တစ်ကြိမ်ပြန်စခေါပါတယ်။ ယခု နေရာတွင် recursive function ကို သုံးထားသည့် အတွက် stack data structure သင်ခန်းစာကို နားလည်ထားပြီးမှသာ ယခု သင်ခန်းစာကို ပိုမိုနားလည်မှာ ဖြစ်ပါတယ်။

ယခု တစ်ခါ inorder function ကို ပြန်ခေါ်သော အခါတွင် node-> left မှာ ရှိတဲ့ node ကို parameter အနေဖြင့် ပြန်ထည့် ပေးလိုက်ပါတယ်။ ထို့ left ထဲတွင် ရှိသော data သည် 10 ဖြစ်ပါတယ်။ ယခင်က node သည် root node ဖြစ်သည့် data 8 ရှိသော eb0 ဖြစ်သော်လည်း ယခု အခါတွင်မူ c90 ဖြစ်နေသည်ကိုတွေ့ရပါမည်။ အသေးစိတ် ပိုမို နားလည်စေရန် စာရေးသူ ဖော်ပြထားသော ပုံနှင့် program ကို ယူဉ်တဲ့ ကြည့်ရန် လိုအပ်ပါသည်။

```
▼ └─ node = {Node *} 0x242a8231c90
    01 data = {int} 10
    > └─ left = {Node *} 0x242a82341f0
    > └─ right = {Node *} 0x242a8234240
```

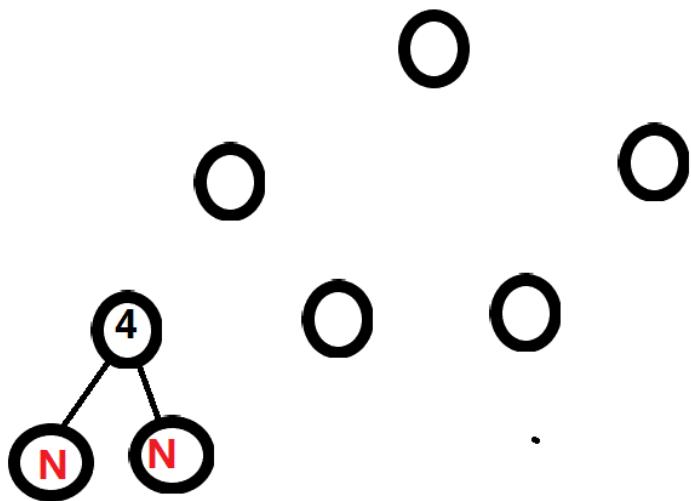
ထို့နောက် c90 အား NULL ဖြင့်ညီသေား ပြန်စစ်ပါတယ် null ဖြင့် မညီသည့် အတွက် line 25 ကို အလုပ် ဆက်လုပ်ပါတယ်။ ယခု အခါတွင်လည်း node->left ဖြင့် ပြန်ခေါ်တာ ဖြစ်သည့် အတွက် c90 မဟုတ်တော့ပဲ 41f0 ဖြင့် ပြန်ခေါ်တာ ဖြစ်ပါတယ်။

```
▼ └─ node = {Node *} 0x242a82341f0
    01 data = {int} 4
    01 left = {Node *} NULL
    01 right = {Node *} NULL
```

ယခု တစ်ကြိမ်တွင်လည်း NULL ဖြစ်လား ထပ်စစ်ပါတယ်။ NULL ဖြစ်မနေသည့်အတွက် line 25 အား အလုပ် ဆက်လုပ်ပါတယ်။ ယခု တစ်ခါတွင် လည်း node-> left ဖြင့် ပြန်ခေါ်ပါတယ် node-> left သည် NULL ဖြစ်နေသည့် အတွက် NULL ပြန်ခေါ်မှာ ဖြစ်ပါတယ်။ NULL ဖြစ်နေသော ကြောင့် return ပြန်ပြီး function တစ်ခု အဆုံး သတ်ပါမယ်။

ထို့နောက် line 26 ကိုသွားပါမယ် ပြီးလျှင် line 27 ကို ဆက်လုပ်မှာ ဖြစ်ပါတယ်။ Line 26 ကို ရောက်သော အခါတွင် 4 ဆိုသည့် data ကို ကန်းပြား ထုတ်ပေးခဲ့ပါသည်။ ထို့ကြောင့် 4 ဆိုသည့် data သည် ပထမဆုံး ထွက်လာသော data ဖြစ်သည်ကို သတိပြုပါ။

Line 27 သို့ရောက်သော အခါတွင် 4 ဆိုသည့် node ရဲ့ ညာဘက်က node ကို အလုပ် ဆက်လုပ်ပါသည် ညာဘက်က node သည် NULL ဖြစ်နေသည့် အတွက် နောက်တစ်ကြိမ် inorder function ကို NULL ဖြင့် ထပ်ခေါ်ပါသည်။ NULL ဖြစ်နေသည့် line 22 ထဲသို့ ဝင်သွားပြီး line 23 တွင် အဆုံး သတ်ပါသည်။ အောက်ပါ ပုံတွင် ကြည့်ပါ 4 Node ရဲ့ Left and right တွင် NULL များ ရှိနေသည်ကို ဖော်ပြထားပါသည်။



4 Node ရဲ left and right များကို စစ်ဆေးပြီး 4 ရဲ data ကို ထုတ်ပြီးသော အခါတွင်မူ သူ့အပေါ်
node ဖြစ်သည် 10 ဆိုသို့ ပြန်တက်ပါသည် အဘယ်ကြောင့် ဆိုသော Last in First out
ဖြစ်သောကြောင့် 4 သည် နောက်ဆုံးမှ ဝင်လာပြီး 4 အရင် ဝင်လာသော 10 ကို ထုတ်ရန် လိုအပ်
သေးသည် ထို့ကြောင့် ယခု တကြိမ်တွင် node သည် c90 ဆိုသည့် 10 node ရှိရာသို့ ပြန်ထောက်
ထားမည် ဖြစ်ပြီး ဦးစွာ 10 data ကို ထုတ်ပေးသည် ထို့နောက် 10 node ရဲ right ဘက်ခြမ်းသို့
သွားပါသည်။

```

inorder( node->left );
cout << node->data << "->";    node: 0x2ab69ac1c90
inorder( node->right );
}

void preorder(struct Node* node) {
    if (node == NULL)
        return;
}

```

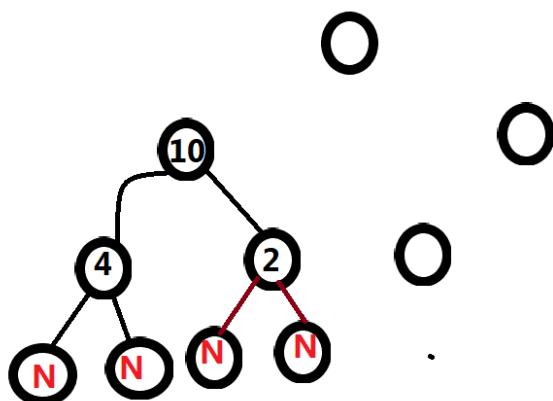
root->right->right = -var-create: unable to create variable

node = {Node *} 0x2ab69ac1c90

data = {int} 10

left = {Node *} 0x2ab69ac41f0

right = {Node *} 0x2ab69ac4240



10 Node ရဲ့ right ဘက်ခြမ်းတွင် 2 ရှိနေသော ကြောင့် 2 ဆီသို့ ဆက်သွားပါမည် ထိုသို့ ဆက်သွားရန် line 27 တွင် node-> right ဟု ရေးထားပါသည်။

```
void inorder(struct Node* node) {    node: 0x2ab69ac4240
    if (node == NULL)
        return;

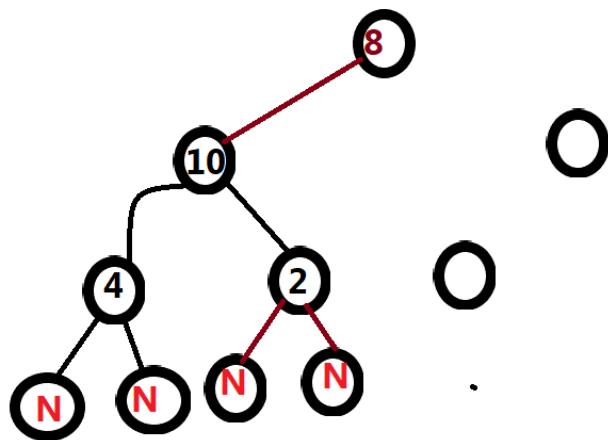
    inorder( node: node->left);
    cout << node->data << "->";    node: 0x2ab69ac4240
    inorder( node: node->right);
}
```

2 ဆီသို့ ရောက်သော အခါတွင် 2 ရဲ့ left ကို ထပ်မံ စစ်ဆေးပါသည် 2->left သည် NULL ဖြစ်နေသည် အတွက် ထို line 25 အတွက် function တစ်ကိမ် ပြန်ခေါ်ခြင်းသည် Line 23 တွင် အဆုံး သတ်သွားပါသည်။ ထိုနောက် line 26 တွင် 2 ဆိုသည့် data ကို ထပ်ထုတ်ပါသည်။ ထိုကြောင့် စုစုပေါင်း data သုံးခု တွက်ထားပြီး 4-> 10 ->2 တို့ဖြစ်သည်။
ထိုနောက် ယခင် 4 Node အတိုင်းပင် left and right တို့ကို ဆက်လက် စစ်ဆေးပြီး NULL များ ဖြစ်နေသောကြောင့် line 23 တွင် return ဖြင့် ရပ်ပြီး 10 ဆီသို့ အောက်ပါ ပုံစံက အတိုင်း ပြန်သွားပါတယ်။

```
void inorder(struct Node* node) {    node: 0x2ab69ac1c90
    if (node == NULL)
        return;

    inorder( node: node->left);
    cout << node->data << "->";
    inorder( node: node->right);    node: 0x2ab69ac1c90
}
```

ထိုနောက် 10 အတွက်လည်း left and right များ စစ်ဆေးပြီးသည့် အတွက် root node ဖြစ်သော 8 ဆီသို့ ပြန်ရောက်သွားပါသည် ပြီးလျှင် left ဘက်ခြမ်းများပြီးသည့် အတွက် right ဘက်ခြမ်းများကို စစ်ဆေးရန်သာ ကျွန်ုပါတော့သည်။



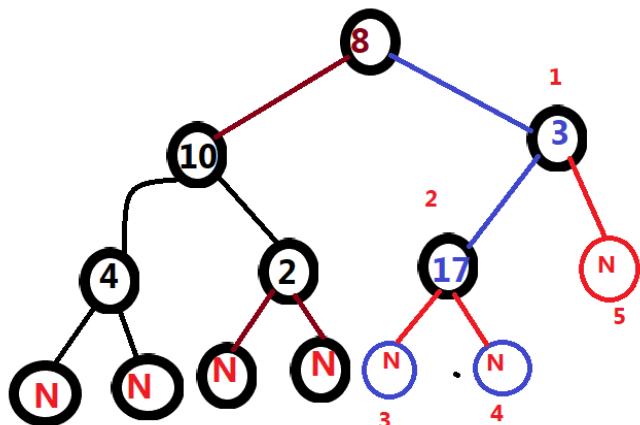
Right ဘက်ခြမ်းကို စစ်ဆေးရန် ဦးစွာ root node ရဲ့ right ကို သွားပါသည် root node ရဲ့ right သည် ၃ ဖြစ်သောကြောင့် ၃ ဆီသို့ သွားပါသည်။

```

inorder( node: node->left);
cout << node->data << "->";
inorder( node: node->right);    node: 0x225939e1eb0
}
void preorder(struct Node* node) {
    if (node == NULL)
        return;

```

အထက်ပါ ပုံသည် root node ရဲ့ right ဆီသို့ မသွားခင်ပုံ ဖြစ်ပါသည်။ နောက်တဆင့် အနေဖြင့် root node ရဲ့ right ဆီသို့ သွားပြီးလျှင် ထို right ရဲ့ left node ဆီသို့ ထပ်သွားမှာ ဖြစ်ပါတယ်။



ပိုမို အသေးစိတ် နားလည်ရန် အထက်ပါ ပုံကို ကြည့်ပါ ၃ ရှိသည့် နေရာသို့ သွားပြီးလျှင် 17 ဆီသို့ သွားမှာ ဖြစ်ပါတယ်။

```

    void inorder(struct Node* node) {
        if (node == NULL)
            return;

        inorder(node->left);
        cout << node->data << "->";
        inorder(node->right);
    }
}

```

ထို့နောက် 17 node ဆီသို့ ရောက်သွားပြီး 17 node ခဲ့ Left ကို ထပ်စစ်ပါတယ် NULL ဖြစ်နေသောကြောင့် terminate ဖြစ်ပြီး 17 ရှိသည့် data ကို ထုတ်ပါတယ်။ ယခု အခါတွင် 4->10->2->8->17 ရှိသည့် data များ ထုတ်ပြီး ပြီ ဖြစ်သည်။ နောက်ဆုံး တစ်လုံး အနေဖြင့် 3 သာ ကျွန်တော့သည်။ 17 ခဲ့ right ကို ထပ်စစ်ပါသည် right သည် NULL ဖြစ်နေသောကြောင့် terminate ဖြစ်ပြီး 3 node ရှိရာ နေရာသို့ ပြန်သွားပါသည်။ ထို 3 node နေရာတွင် left ဘက် ခြမ်းသည် စစ်ဆေးပြီးသား ဖြစ်သောကြောင့် data ကို ထုတ်ပြီး right ဘက်ခြမ်းကို ဆက်လက် စစ်ဆေးပါသည်။ ယခု အခါတွင် data အားလုံး complete ထုတ်ပြီး မည် ဖြစ်သည်။ right သည်လည်း NULL ဖြစ်နေသောကြောင့် ထို function call သည် terminate ဖြစ်ပြီး program သည် root node ဆီသို့ ပြန်သွားပါမည် ထို့နောက် inorder function သည် အပြီးပိုင် terminate လုပ်သွားပါသည်။ Output အနေဖြင့် 4->10->2->8->17->3 တို့ကို ရရှိပါသည်။

အထက်ပါ ဖော်ပြချက်များ အားလုံးသည် inorder function တစ်ခု တည်း အတွက်သာ ဖြစ်ပြီး Preorder and Postorder Traversal လုပ်ခြင်းများသည်လည်း left , right , root သာ ကွဲသွားပြီး ကျွန်တာ အားလုံး အတူတူသာပင် ဖြစ်သည်။

Exercise

ယခု သင်ခန်းစာအား ပိုမိုနားလည် စေရန် preorder traversal နှင့် postorder traversal တို့ကို မိမိတို့ဘာသာ အသေးစိတ် debugging လုပ်ရပါမည်။

သတိပြုရန် အချက်

Preorder Traversal ဆိတာကေတဲ့ အရင်ဆုံး root node ကို သွားပါတယ် ထိုနောက် Tree ရဲ့ left ဘက်ခြမ်းကို သွားပြီး နောက်ဆုံးမှ tree ရဲ့ညာဘက်ခြမ်းကို သွားပါတယ်။

Postorder Traversal ကေတဲ့ tree တစ်ခုမှာ ရှိတဲ့ left ဘက်က nodes တွေဆီကို အရင်သွားပါတယ် ထိုနောက် right မှ nodes တွေဆီကို အရင်သွားပါတယ်ပြီးမှ သလ္ဂုတ် root node ဆီသို့ သွားပါတယ်။

Program on github :

<https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/Traversal.cpp>

Full Binary Tree

Internal Node သို့မဟုတ် Parent Nodes တွေမှာ two childrens သို့မဟုတ် children node တစ်ခုမှ မရှိလျှင် ထို Tree ကို full binary tree ဟု ခေါ်ဆိုပါတယ်။ Tree တစ်ခုသည် Full ဖြစ်သလာ မဖြစ်ဘူးလားဆိုတာကို အောက်ပါ program အတိုင်း စစ်ဆေးနိုင်ပါတယ်။ ပထမဆုံး အနေဖြင့် main function တစ်ခု ရေးထားပြီး ငါး main function ထဲတွင် အောက်ပါ အတိုင်း ရေးသားထားပါတယ်။

```
--  
29 int main() {  
30     struct Node *root = NULL;  
31     root = newNode(8);  
32     root->left = newNode(10);  
33     root->right = newNode(3);  
34     root->left->left = newNode(4);  
35     root->left->right = newNode(2);  
36     root->left->right->left = newNode(17);  
37     root->left->right->right = newNode(7);  
38  
39     if (isFullBinaryTree(root))  
40         cout << "Full Binary Tree\n";  
41     else  
42         cout << "Is not Full Binary Tree\n";  
43 }
```

ပထမဆုံး အနေဖြင့် line 30 တွင် root ဆိုသည့် node တစ်ခု တည်ဆောက်ထားပြီး NULL data ကို ထည့်ထားပါတယ်။ ထိုနောက် Line 31 တွင် newNode function ကိုခေါ်ပြီး 8 ဆိုသည့် data ကို ထည့်ကာ root အဖြစ် သတ်မှတ်ထားပါတယ်။

```

4   struct Node {
5     int key;
6     struct Node *left, *right;
7   };
8   struct Node *newNode(char data) {
9     struct Node *node = (struct Node *)malloc(sizeof(struct Node));
10    node->key = data;
11    node->right = node->left = NULL;
12    return node;
13 }

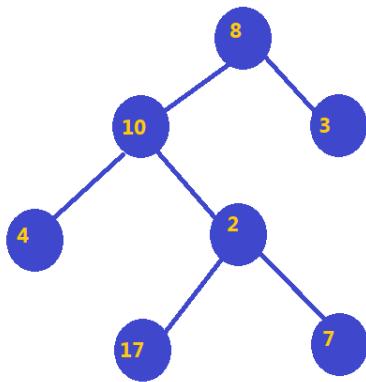
```

Node တစ်ခုတွင် key နှင့် left and right တို့ပါဝင်ပြီး newNode function တွင်မူ malloc ကို
သုံးကာ memory ပေါ်တွင် နေရာယူထားပြီး user ထည့်ပေးလိုက်သည့် data ကို node ရဲ့ data
အဖြစ် ထည့်ပေးပါတယ်။ ထို့နောက် left and right တို့ကို NULL သတ်မှတ်ပေးပြီး node တစ်ခု
တည်ဆောက်ကာ return ပြန်ပေးလိုက်ပါတယ် ယခု လုပ်ဆောင်ချက်များသည် ရှေ့ပိုင်း
သင်ခန်းစာများတွင်လည်း အသေးစိတ် ဖော်ပြခဲ့ ပြီးပြီ့မူ အသေးစိတ် မသိသေးပါက ရှေ့ပိုင်း
သင်ခန်းစာများကို ပြန်လည် လေ့လာရန် အကြံပြုပါတယ်။

Main function ရဲ့ line 39 တွင် isFullBinaryTree ဆိုတဲ့ function ကို လုမ်းခေါပါတယ် ထို
function အမိက အလုပ်လုပ်မည့် function ဖြစ်ပြီး အသေးစိတ်ရှင်းလင်းချက်ကို အောက်တွင်
ရှင်းလင်းသင်ကြား ပေးပါမယ်။

အထက်ပါ main function ရဲ့ program များအရ စု run လိုက်သော အခါတွင် data များအားလုံး
ထည့်ပြီးချိန့်အောက်ပါ ပုံအတိုင်း Tree structure တစ်ခုကို ရရှိမှာ ဖြစ်ပါတယ်။

Algorithm မှာ ပထမဆုံး အနေဖြင့် root သည် NULL ဖြစ်သလား ဆိုတာကို စစ်ဆေးပါတယ်
အကယ်၍ root သည် NULL ဖြစ်နေခဲ့လျှင် true ကို return ပြန်ပေးပြီး full binary tree
ဖြစ်ကြောင်းကို output ပြန်ပြ ပေးမှာပါ။ ထိုသို့မဟုတ် root သည် NULL မဖြစ်သော်လည်း root ရဲ့
left and right တို့သည့် NULL ဖြစ်နေလျှင် full binary tree ဟု ယူဆပြီး true ကိုသာ return ပြန်
ပေးမှာ ဖြစ်ပါတယ်။



bool isFullBinaryTree(struct Node *root) {

```

if (root == NULL)
    return true;

if (root->left == NULL && root->right == NULL)
    return true;

if ((root->left) && (root->right))
    return (isFullBinaryTree(root->left)
        && isFullBinaryTree(root->right));

return false;
}

int main() {
    struct Node *root = NULL;    root: 0x212849b1eb0
    root = newNode( data: 8);
    root->left = newNode( data: 10);
    root->right = newNode( data: 3);
    root->left->left = newNode( data: 4);
    root->left->right = newNode( data: 2);
    root->left->right->left = newNode( data: 17);
    root->left->right->right = newNode( data: 7);
}
  
```

root = {Node *} 0x212849b1eb0
 key = {int} 8
 left = {Node *} 0x212849b1c90
 key = {int} 10
 left = {Node *} 0x212849b41a0
 key = {int} 4
 left = {Node *} NULL
 right = {Node *} NULL
 right = {Node *} 0x212849b41f0
 key = {int} 2
 left = {Node *} 0x212849b4240
 key = {int} 17
 left = {Node *} NULL
 right = {Node *} NULL
 right = {Node *} 0x212849b4290
 key = {int} 7
 left = {Node *} NULL
 right = {Node *} NULL
 right = {Node *} 0x212849b1ce0
 key = {int} 3
 left = {Node *} NULL
 right = {Node *} NULL

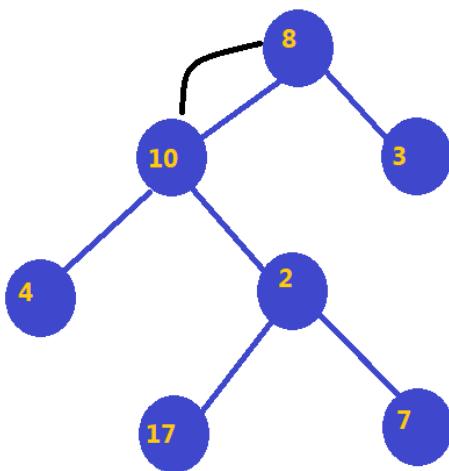
```

23      if ((root->left) && (root->right))
24          return (isFullBinaryTree(root->left)
25              && isFullBinaryTree(root->right));
26
  
```

የወጪ አዎንታዎች የroot ቀጥል ተችሉ ነው እና የroot ቀጥል ተችሉ ነው ይላል
 እና የroot ቀጥል ተችሉ ነው ይላል እና የroot ቀጥል ተችሉ ነው ይላል

ဖြစ်နေလျှင် full binary tree မဖြစ်သလို right သာ ရှိပြီး left က NULL ဖြစ်နေလျှင်လည်း full binary tree ဖြစ်မည် မဟုတ်ပါ။

ယခု သင်ခန်းစာတွင်မူ ပထာဆုံး root ဖြစ်သည့် 8 ဆိုသည့် Node တွင် left and right နှစ်ခုလုံးရှိနေပါသည် ထို့ကြောင့် root->left ကို parameter အနေဖြင့် သုံးကာ isFullBinaryTree function ကို recursive ပုံစံဖြင့်ပြနေခေါ်မှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် ဒုတိယ အကြိမ် function ထဲသို့ပြန်ဝင်လာသော အခါတွင် root ရဲ့ memory address သည် 1c90 ဖြစ်နေပြီး data မှာ 10 ဖြစ်နေမှာပါ။



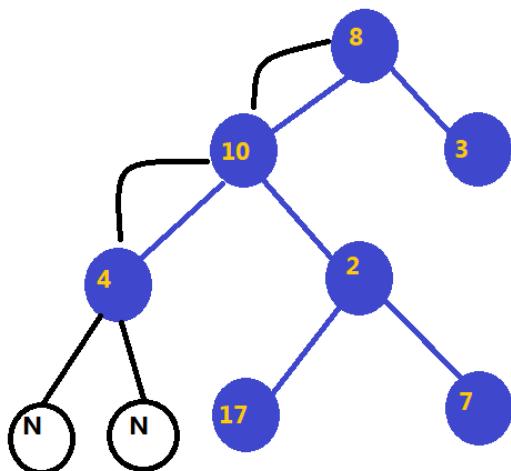
ထို့နောက် root သည် NULL ဖြစ်သလား နှင့် left and right တို့သည် NULL ဖြစ်နေသလား တို့ကိုစစ်ဆေးပြီး line 17 မှ line 21 အထိပြီးလျှင် line 23 ကို ဆက်လက် စစ်ဆေးပါတယ်။ line 23 condition မှန်နေလျှင် line 24 မှ တဆင့် root->left ကိုသုံးပြီး function တစ်ကြိမ် ပြနေခေါ်ပါတယ်။ ယခု အခါ function ထဲသို့ပြန်ဝင်လာလျှင် root node သည် 41a0 ဖြစ်နေမည် ဖြစ်ပြီး data သည် 4 ဖြစ်နေမှာပါ။

```

16
17 → if (root == NULL)    root: 0x212849b41a0
18     return true;
19
20     if (root->left == NULL && root->right == NULL)
21         return true;
22
23     if ((root->left) && (root->right))
24         return (isFullBinaryTree( root: root->left)
25             && isFullBinaryTree( root: root->right));
26
27     return false;

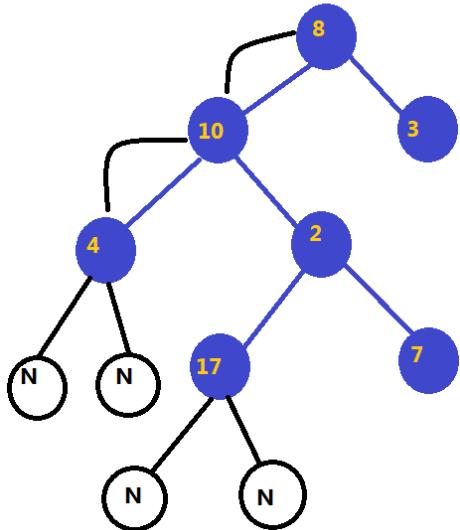
```

ယခု အဆင့်တွင်မှ line 20 သို့ ရောက်သော အခါ 4 node ရဲ့ left and right တို့သည် NULL ဖြစ်နေသည့် အတွက် true ကို return ပြန်ပေးပြီး function call တစ်ခု terminate လုပ်ပါမည်။

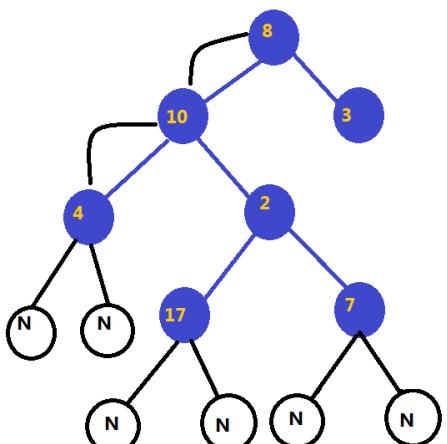


အထက်ပါ အဆင့်ပြီးပါက 4 node ရဲ့ parent node ဖြစ်သော 10 ကို root အဖြစ် ပြန်ယူပါမည် ထို့နောက် ထို့နောက် root->right ကို Parameter အဖြစ်သုံးကာ 2 node ဆီသို့ သွားပါမည်။ 2 node သို့ ရောက်သော အခါတွင် ယခင် အဆင့်များ အတိုင်း စစ်ဆေးပြီး 2 node ရဲ့ left ဆီသို့ ဆက်သွားပါမည်။ 2 node ရဲ့ left တွင် 17 node ရှိသည့် အတွက် 17 node ကို ထပ်မံ ခေါ်ယူပါမည်။ 17 node ကို left and right စစ်ဆေးသော အခါတွင် နှစ်ခု လုံးသည် NULL ဖြစ်နေသည့်အတွက် line 20 condition ဖြင့် ကိုက်ညီ သွားပြီး true ကို return ပြန်ပါမည်။ ပထမ leaf node ဖြစ်သော 4 ကို

စစ်ဆေးရာတွင်လည်း true ရသည် second leaf node ဖြစ်သည့် 17 ကို စစ်ဆေးရာတွင်လည်း true ရပါသည်။

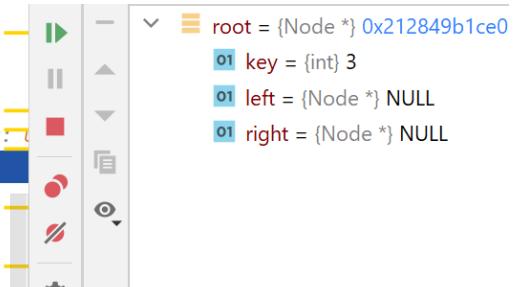


ထို့နောက် parent node ဖြစ်သည့် 2 ဆီသို့ ပြန်သွားပါမည် ပြီးလျှင် right ဘက်တွင် ရှိသော 7 ကိုလည်း ဆက်လက် စစ်ဆေးပါသည် ထို့သို့ စစ်ဆေးပြီးလျှင် 7 သည် လည်း right and left ၏ NULL များသာ ရှိနေသည့် အတွက် line 21 တွင် terminate ဖြစ်ပြီး true ကိုသာ return ပြန်ရပါမည်။ ယခု အခါတွင် true သုံးခု ရနေသည်ကို သတိပြုပါ။



Left ဘက်ခြမ်းတွင်ရှိသော leaf node အားလုံးကို စစ်ဆေးပြီးသွားသည် ဖြစ်တဲ့အတွက် ပထမ ညီးစွာ 2 မှ 7 သို့ ပြန်သွားသည့် 7 မှ တင်ဆင့် parent node ဖြစ်သည့် 2 သို့ ပြန်သွားသည့် 2 မှ တင်ဆင့် 10 သို့

ပြန်သွားပါသည်။ ရောက်ဆုံးတွင် root node ဖြစ်သည့် 8 သို့ပြန်ရောက်သွားပါသည်။ root node သို့ပြန်ရောက်သွားသောအ ခါတွင် right ဘက်ခြမ်းကို စစ်ဆေးရန် ကျွန်ုင်နေသည့် အတွက် 3 ဆီသို့



```

if (root == NULL)
    return true;

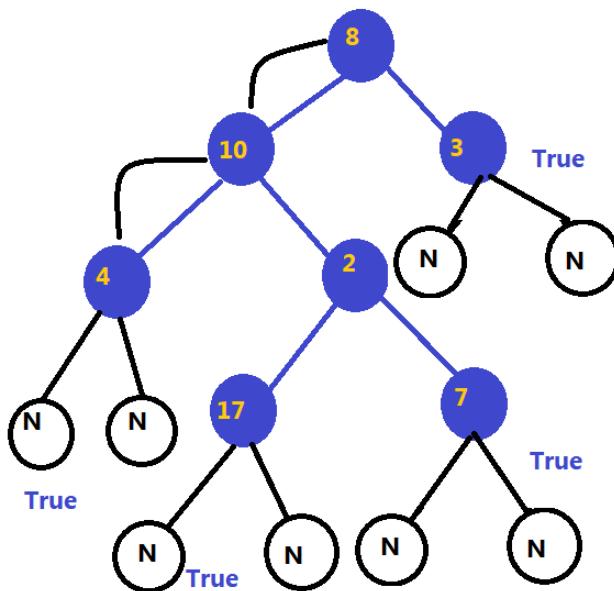
if (root->left == NULL && root->right == NULL) root: 0x212849b1ce0
    return true;

if ((root->left) && (root->right))
    return (isFullBinaryTree( root: root->left)
        && isFullBinaryTree( root: root->right));

```

ထပ်သွားပါသည်။

3 node သည်လည်း left and right တို့တွင် NULL များသာ ဖြစ်နေသည့် အတွက် line 21 တွင် terminate လုပ်ပြီး true ကို return ပြန်ပေးလိုက်ပါသည်။ ထို့ကြောင့် leaf node တစ်ခုချင်းစီရဲ့ return အားလုံးသည် true ဖြစ်နေသည့်အတွက် ထို function တစ်ခုလုံးရဲ့ return value သည်လည်း true ဖြစ်နေမှာပါ။ ထို့ကြောင့် final output အနေဖြင့် Full binary Tree ဆိုတဲ့ အဖြေကို ရရှိခြင်းဖြစ်ပါသည်။



Full Binary Tree Program

```
#include <iostream>
using namespace std;
```

```
struct Node {
```

```
int key;
struct Node *left, *right;
};

struct Node *newNode(char data) {
    struct Node *node = (struct Node *)malloc(sizeof(struct Node));
    node->key = data;
    node->right = node->left = NULL;
    return node;
}

bool isFullBinaryTree(struct Node *root) {

    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return true;

    if ((root->left) && (root->right))
        return (isFullBinaryTree(root->left)
                && isFullBinaryTree(root->right));

    return false;
}

int main() {
    struct Node *root = NULL;
    root = newNode(8);
    root->left = newNode(10);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(2);
    root->left->right->left = newNode(17);
    root->left->right->right = newNode(7);

    if (isFullBinaryTree(root))
        cout << "Full Binary Tree\n";
    else
        cout << "Is not Full Binary Tree\n";
}
```

Source code on github:

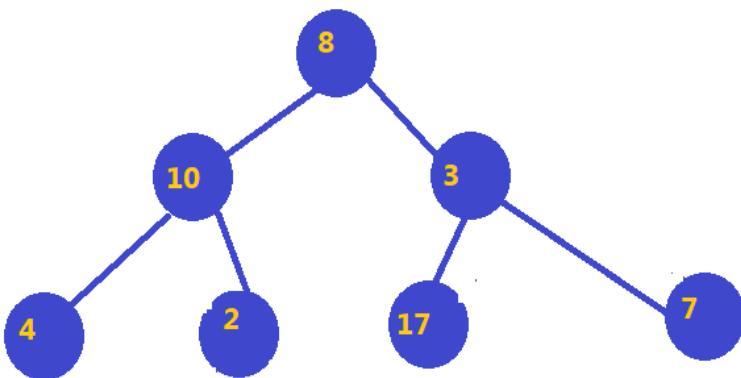
<https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/BinaryTree-FullBinaryTree.cpp>

Exercise အနေဖြင့် မိမိတို့ ဘာသာ အောက်ပါ အတိုင်း leaf node များ ထပ်ထည့်ပြီး full binary tree ဖြစ်မဖြစ်ကို စစ်ဆေးရပါမည်။

```
struct Node *root = NULL;
root = newNode(8);
root->left = newNode(10);
root->right = newNode(3);
root->left->left = newNode(4);
root->left->right = newNode(2);
root->left->right->left = newNode(17);
root->left->right->right = newNode(7);
root->left->right->right->left = newNode(70);
```

Perfect Binary Tree

Perfect Binary Tree ဆိုတာ internal nodes များတွင် two child အတိအကျ ရှိရမည် ဖြစ်ပြီး leaf nodes များအားလုံးသည်လည်း level အတူတူသာရှိရပါမည်။ အောက်တွင် ဖော်ပြထားသော ပုံမှု leaf node များ အားလုံးသည် same level များ ဖြစ်နေပြီး internal nodes များသည်လည်း leaf node နှစ်ခု ရှိနေသည့်အတွက် perfect binary tree ဖြစ်ပါတယ်။



```

bool isPerfect(Node *root) {
    int d = depth(node: root);
    return isPerfectR(root, d);
}

struct Node *newNode(int k) {
    struct Node *node = new Node;
    node->key = k;
    node->right = node->left = NULL;
    return node;
}

int main() {
    struct Node *root = NULL;    root: 0x19ceecc1eb0
    root = newNode(k: 8);
    root->left = newNode(k: 10);
    root->right = newNode(k: 3);
    root->left->left = newNode(k: 4);
    root->left->right = newNode(k: 2);
    root->right->left = newNode(k: 17);
    root->right->right = newNode(k: 7);

    if (isPerfect(root))    root: 0x19ceecc1eb0
        cout << "Perfect Binary Tree Found\n";
}

```

အထက်ပါ ပုံသည် လက်ရှိ data များ ထည့်ထားသော tree ပုံစံ ဖြစ်ပါသည် ယခု သင်ခန်းစာတွင် ထို tree သည် **perfect binary tree** ဖြစ်သလား မဖြစ်ဘူးလား ဆိတာကို စစ်ဆေးမှာ ဖြစ်ပါတယ်။ **main function** ထဲတွင် ရေးထားပုံမှာလည်း အောက်ပါ အတိုင်း ဖြစ်ပါတယ်။

```

41     int main() {
42         struct Node *root = NULL;
43         root = newNode(8);
44         root->left = newNode(10);
45         root->right = newNode(3);
46         root->left->left = newNode(4);
47         root->left->right = newNode(2);
48         root->right->left = newNode(17);
49         root->right->right = newNode(7);
50
51         if (isPerfect(root))
52             cout << "Perfect Binary Tree Found\n";
53         else
54             cout << "Perfect Binary Tree not Found\n";
55     }

```

Line 51 တွင် `isPerfect` ဆိုတဲ့ function ကို လှမ်းခေါ်ပါတယ်။ `isPerfect` function သည်လည်း `depth` နှင့် `isPerfectR` ဆိုသည့် function နှစ်ခုကို ပြန်ခေါ်ထားပါတယ်။

```

29     bool isPerfect(Node *root) {
30         int d = depth(root);
31         return isPerfectR(root, d);
32     }

```

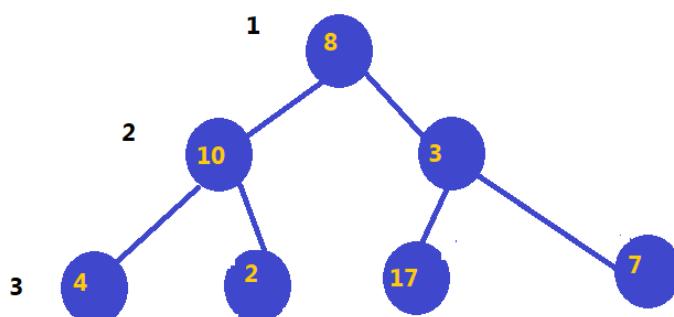
ပထမဆုံး အနေဖြင့် `root` ကို parameter အနေဖြင့် ထည့်ပေးလိုက်ကာ `depth` ကို ရှာပါတယ်။ `Depth` ကို ရှာရန် အတွက် tree ရဲ့ ဘယ်ဘက်ခြမ်း အဆုံးထိ သွားမှာ ဖြစ်ပါတယ်။

```

7     int depth(Node *node) {
8         int d = 0;
9         while (node != NULL) {
10            d++;
11            node = node->left;
12        }
13        return d;

```

`d` ဆိုသည့် variable ကိုသုံးပြီး `left` ဘက်ကို တစ်ခါ သွားတိုင်း `d` ကို တစ်ပေါင်းမှာ ဖြစ်ပါတယ် ထိုနည်းဖြင့် `left` သည် `NULL` မဖြစ်မခြင်း သွားမှာ ဖြစ်ပါတယ်။ `Left` သည် `NULL` ဖြစ်သွားသော အခါ `loop` ကို ရပ်လိုက်ပြီး `d` ကို `depth` တန်ဘိုး အနေဖြင့် `return value` ပြန် ပေးလိုက်မှာ ဖြစ်ပါတယ်။ ယခု tree structure တွင် `depth` တန်ဘိုး 3 ရှိမှာ ဖြစ်ပါတယ် အဘယ်ကြောင့် ဆိုသော် `root` ကနေ စပြီး `left` ဘက်ခြမ်းမှာ ရှိတဲ့ `node` အားလုံးကို ရေတွက်တဲ့ အခါ ၃ ခု ရှိသောကြောင့် ဖြစ်ပါတယ်။



နောက်တဆင့် အနေဖြင့် Line 31 တွင် isPerfectR ဆိုသည့် function ကို ထပ်ခေါပါတယ် ထို function ကို ခေါ်ရာတွင် depth function မှ ရလာသော depth data နှင့် root နှစ်ခုလုံးကို ထည့်ပေးလိုက်မှာ ဖြစ်ပါတယ်။

```
bool isPerfectR(struct Node *root, int d, int level = 0) {
    if (root == NULL)  root: 0x19ceec1eb0
        return true;

    if (root->left == NULL && root->right == NULL)
        return (d == level + 1);

    if (root->left == NULL || root->right == NULL)
        return false;
```

1. isPerfectR function ထဲတွင် ပထဲး အနေဖြင့် ဝင်လာသော node သည် NULL ဖြစ်နေသလား ဆိုတာကို စစ်ဆေးပါတယ် အကယ်၍ NULL ဖြစ်နေလျှင် true ကို return ပြန်ပေးလိုက်ပြီး perfect binary tree ဖြစ်ကြောင်းကို ပြန်ပြ ပေးပါမည်
2. ထိုသို့ မဟုတ်ပါက root left and right တို့သည် NULL ဖြစ်နေသလား ဆိုတာကို စစ်ဆေး ပါတယ် အကယ်၍ ဖြစ်နေခဲ့လျှင် depth data d နှင့် level တို့ကိုနှင့်ယူဉ်ကာ တူညီနေလျှင် true ကို return ပြန်ပေးပြီး မတူညီနေလျှင် false ကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။
3. ထိုပြင် root left သို့မဟုတ် right တစ်ခုခုသည် NULL ဖြစ်နေလျှင်လည်း false ကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။
4. အထက်ပါ အဆင့်များ တစ်ခုမှ မဟုတ်လျှင်တော့ root->left , depth , level+1 တို့ဖြင့် function call ပြန်ခေါ်မှာ ဖြစ်ပါတယ်။

```

bool isPerfectR(struct Node *root, int d, int level = 0) {
    if (root == NULL)  root: 0x19ceec41a0
        return true;

    if (root->left == NULL & root->right == NULL)
        return (d == level + 1);

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectR( root: root->left, d,  level: level + 1) &&
           isPerfectR( root: root->right, d,  level: level + 1);
}

```

Tree ရဲ left ထိ သွားပြီး တဲ့ အခါမှာ line 19 ရဲ condition နှင့် ညီသွားသည့်အတွက် line 20 အရောက်တွင် d တန်ဘိုးနှင့် level တန်ဘိုး တူညီ နေသလား ဆိတာကို စစ်ဆေးပါတယ် အကယ်၍ တူညီ နေခဲ့လျှင် true ကို return ပြန်ပေးပါတယ်။

အဘယ်ကြောင့် depth နှင့် level တို့ကို အသုံးပြုရ သနည်းဆိုလျှင် ယခု program ဒု ပထမ ဦးစာ tree ရဲ left ဘက်ခြမ်းမှ depth ကို အရင် ရာပါ ထိုနောက် tree တစ်ခုလုံးမှာ ရှိတဲ့ အခြားသော leaf node များကိုပါ depth ရှာဖြီး ငွေး depth ကို level အနေနဲ့ မှတ်ယူကာ မူရင်း left နဲ့ နှင့် ယူဉ်ပါတယ် အကယ်၍ တူညီ နေခဲ့လျှင်ထို node အတွက် true ဟု သတ်မှတ်ထားပါတယ်။ အကယ်၍ မတူညီတော့သော အခါတွင်မူး leaf node များအားလုံးသည် same level မဟုတ်တော့သည့် အတွက် perfect binary tree မဟုတ်တော့ပါဘူး။

ယခု အဆင့်တွင်မူး d သည်လည်း 3 ဖြစ်ပြီး level(2)+1 ဖြစ်သောကြောင့် 3 ဖြစ်နေသည့် အတွက် true ကို return ပြန်ရမှာ ဖြစ်ပါတယ်။ အထက်ပါ အဆင့်များ အားလုံးသည့် 4 node အတွက် ဖြစ်သည်။ နောက်တဆင့် အနေဖြင့် parent node ဖြစ်တဲ့ 10 သို့ ပြန်သွားပြီး right ကို သွားကာ 2 node ကို စစ်ဆေးပါတယ် 2 node တွင်လည်း left and right နှစ်ခုလုံးတွင် NULL ဖြစ်နေပါမည် ထို့ကြောင့် line 19 condition တွင် မှန်သွားပြီး line 20 ကို အလုပ် ဆက်လုပ်ပါတယ်။ ယခု နေရာတွင်လည်း level သည် (2+1) ဖြစ်နေသည့် အတွက် true ကိုသာ return ပြန်ပေးပါမည်။ ထို့ကြောင့် node 4 and 2 နှစ်ခုလုံးသည် true ချည်းသာ return ပြန်နေသည်ကို တွေ့ရမည်။

အကယ်၍ node တစ်ခုသည် data ရှိနေပြီး ကျော်node တစ်ခုသည် NULL ဖြစ်နေလျှင် line 22 condition နှင့် ကိုက်ညီသွားသည့် အတွက် false ကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။

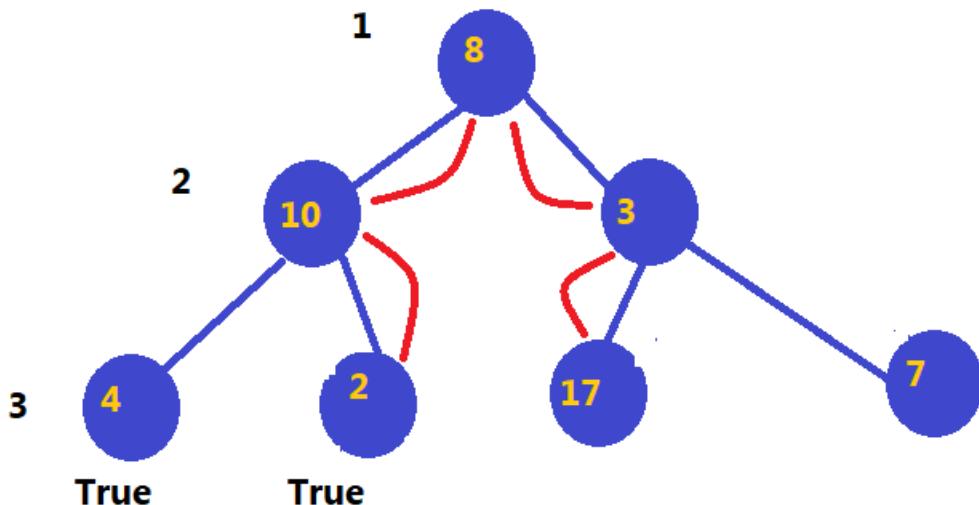
အထက်ပါ အဆင့်များ ပြီးသွားပါက root node ဖြစ်တဲ့ 8 ဆီသို့ 10 ကို ဖြတ်ကာ ပြန်သွားပြီး 3 node ဆီသို့ သွားပါတယ်။ 3 node သို့ ရောက်သော အခါတွင် leaf node ဖြစ်တဲ့ 17 ဆီသို့

ဆက်သွားပါတယ်။ 17 တွင်လည်း left and right တို့သည် null ဖြစ်နေသည့် အတွက် level တန်ဘိုးမှာ 3 ခု ဖြတ်လာရသောကြောင့် (2+1) ဖြစ်နေမည် ထို့ကြောင့် 17 node သည် true ကိုသာ return ပြန်ပေးမှာ ဖြစ်ပါတယ်။ တစ်ဖန် 3 node ဆီသို့ပြန်သွားပြီး right ဘက်တွင် ကျွန်ုင်သေးသော 7 ဆီသို့ဆက်သွားပါမည် 7 သည် လည်း right and left တို့မှာ NULL ဖြစ်နေပြီး level မှာလည်း 2+1 ဖြစ်သောကြောင့် d တန်ဘိုးနှင့် ညီကာ true ကိုသာ return ပြန်ရပါမည်။ leaf node များအားလုံးသည် depth တန်ဘိုးနှင့် တူညီနေပြီး true ဖြစ်နေသည့် အတွက် true ,true ,true ,true ဖြစ်နေကာ function တစ်ခုလုံးရဲ့ return value သည်လည်း true ကို ရရှိမှာ ဖြစ်ပါတယ်။ အကယ်၍

```

15     bool isPerfectR(struct Node *root, int d, int level = 0) {    root:
16         if (root == NULL)
17             return true;
18
19     →     if (root->left == NULL && root->right == NULL)    root: 0x17c3bc
20         return (d == level + 1);
21
22         if (root->left == NULL || root->right == NULL)
23             return false;
24
25         return isPerfectR( root: root->left, d,    level: level + 1) &&
26             isPerfectR( root: root->right, d,    level: level + 1);
27     }
28

```



თითოეულ `false` ပინელუნ `function` თითოეულ დროის დროის `return value` ვალილუნ: `false` ის ციფრულუნ რეტურნ დოკუმენტით ის.

ახალ დანართზე დაგენერირებულ `true` ციფრულუნ `Perfect Binary Tree found` შემთხვევაში `output` ის ციფრულუნ რეტურნ დოკუმენტით ის.

```

#include <iostream>
using namespace std;
struct Node {

```

```
int key;
struct Node *left, *right;
};

int depth(Node *node) {
    int d = 0;
    while (node != NULL) {
        d++;
        node = node->left;
    }
    return d;
}

bool isPerfectR(struct Node *root, int d, int level = 0) {
    if (root == NULL)
        return true;

    if (root->left == NULL && root->right == NULL)
        return (d == level + 1);

    if (root->left == NULL || root->right == NULL)
        return false;

    return isPerfectR(root->left, d, level + 1) &&
           isPerfectR(root->right, d, level + 1);
}

bool isPerfect(Node *root) {
    int d = depth(root);
    return isPerfectR(root, d);
}

struct Node *newNode(int k) {
    struct Node *node = new Node;
    node->key = k;
    node->right = node->left = NULL;
    return node;
}

int main() {
    struct Node *root = NULL;
    root = newNode(8);
    root->left = newNode(10);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(2);
    root->right->left = newNode(17);
```

```

root->right->right = newNode(7);

if (isPerfect(root))
    cout << "Perfect Binary Tree Found\n";
else
    cout << "Perfect Binary Tree not Found\n";
}

```

Perfect Binary Tree on github : <https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/BinaryTree-PerfectBT.cpp>

Complete Binary Tree

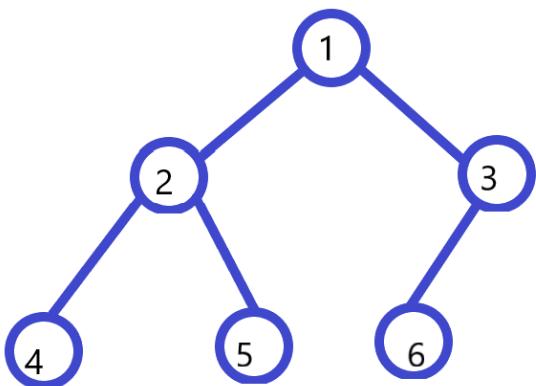
Complete Binary Tree ဆိတာ tree structure တစ်ခုလုံးမှာရှိတဲ့ level တွေအားလုံးသည် နောက်ဆုံး level မှ လွှဲ၍ node တွေ အားလုံး အပြည့်ရှိနေရမှာ ဖြစ်ပါတယ်။ နောက်ဆုံး Last level တွင်လည်း node များသည် from left to right ပုံစံဖြင့် တည်ရှိနေရမှာ ဖြစ်ပါတယ်။

```

35 int main() {
36     struct Node *root = NULL;
37     root = newNode(1);
38     root->left = newNode(2);
39     root->right = newNode(3);
40     root->left->left = newNode(4);
41     root->left->right = newNode(5);
42     root->right->left = newNode(6);
43
44     int numNodes = nodes(root);
45     int index = 0;
46
47     if (checkComplete(root, index, numNodes))
48         cout << "Complete Binary Tree Found \n";
49     else
50         cout << "Complete Binary Tree not Found\n";
51 }

```

အထက်ပါ အတိုင်းရေးပြီး program run ပြီးချိန်တွင် အောက်ပါ အတိုင်း tree structure ကို ရရှိမှာ ဖြစ်ပါတယ်။ အောက်ပါ ပုံသည် complete binary tree တွင်ရှိရမည့် ပိဿာများနှင့် ကိုက်ညီသည့် အတွက် complete binary tree ဟု ခေါ်ဆိုနိုင်ပါသည်။ အဘယ်ကြောင့် ဆိုသော် tree တွင် nodes အားလုံးသည် နောက်ဆုံး level မှ လွှဲ ၍ ပြည့်စုံ နေပါသည်။ နောက်ဆုံး level သည် မပြည့်စုံသော်လည်းပဲ nodes များသည် left to right ရှိရမည် ဆိုသောကြောင့် 4,5,6 တို့သည် left မှ စ၍ တည်ရှိနေပါသည်။



Line number 44 တွင် nodes ဆိုသည့် function ကို စခေါပါသည့် ငါး function သည် nodes အရေအတွက်ကို ရေတွက် ပေးမှာ ဖြစ်ပါတယ်။

```

15 // node အရေအတွက်ကို အရင်ဆုံး ရေတွက်ရန် nodes ဆိုတဲ့ function ကိုရေးထားပါတယ်။
16 int nodes(struct Node *root) {
17     if (root == NULL)
18         return (0);
19
20     return (1 + nodes(root->left)+ nodes(root->right));
21 }
  
```

root သည် null မဖြစ်မခြင်း function ကို ပြန်ခေါ်မှာ ဖြစ်ပါတယ်။ ထိုပြင် တစ်ကြိမ် ခေါ်တိုင်းမှာ 1 ကို stack ပေါ်တွင် မှတ်ထားမှာ ဖြစ်သည့်အတွက် အကြိမ်အရေ အတွက်လိုက် နောက်ဆုံးတွင် ပေါင်းပြီး ပြန်လည် ရရှိမှာ ဖြစ်ပါတယ်။ ယခု သင်ခန်းစာတွင်မူ nodes အရေအတွက်သည် 6 ခုသာရှိသည့်အတွက် 6 ကို ရရှိမှာဖြစ်ပါတယ်။ အကယ်၍ 1 + nodes ဟုရေးထားသည့် နေရာတွင် 2 + nodes ဟု ရေးကြည့်ပါက 12 ရသည်ကို တွေ့ရပါမည်။

Line number 47 တွင်မူ checkComplete ဆိုသည့် function ကို ခေါပါသည့် ထိုသို့ ခေါ်ရေတွင် root , index , numNodes တို့ကို ထည့်ပေးလိုက်ပါသည်။ root သည် tree structure

တစ်ခုလုံးအတွက် ထည့်ပေးခြင်း ဖြစ်ပြီး index , numNodes တို့ကို အဘယ်ကြောင့် ထည့်လိုက်ရသည် ကို အသေးစိတ် သိရှိရန် လိုပါသည်။

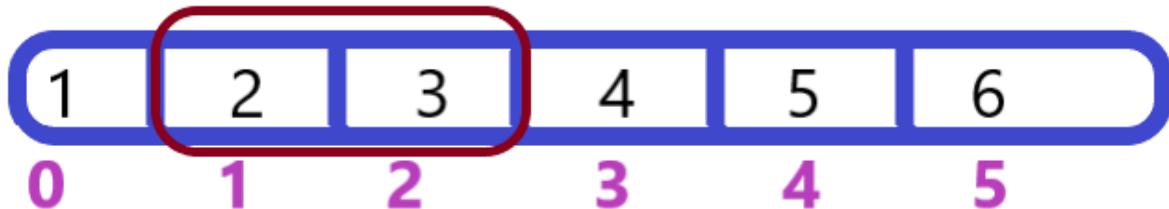
Line 25 တွင် root သည် NULL ဖြစ်သလား ဆိုတာကို ကြည့်ပါတယ် အကယ်၍ NULL ဖြစ်နေခဲ့လျှင် true ကို ပြန်ပေးပါတယ် ဆိုလိုသည်မှာ root ထဲတွင် မည့်သည့် data မှ ရှိမနေသော အခါး။

```
--  
23  bool checkComplete(struct Node *root, int index, int numberNodes) {  
24  
25      if (root == NULL)  
26          return true;  
27  
28      if (index >= numberNodes)  
29          return false;  
30  
31      return (checkComplete(root->left, 2 * index + 1, numberNodes) &&  
32          checkComplete(root->right, 2 * index + 2, numberNodes));  
33 }
```

complete binary tree ဖြစ်တယ်ဟု ယူဆခြင်း ဖြစ်သလို function call တစ်ခုအတွက် terminating point ထားပေးခြင်း လည်း ဖြစ်ပါတယ်။

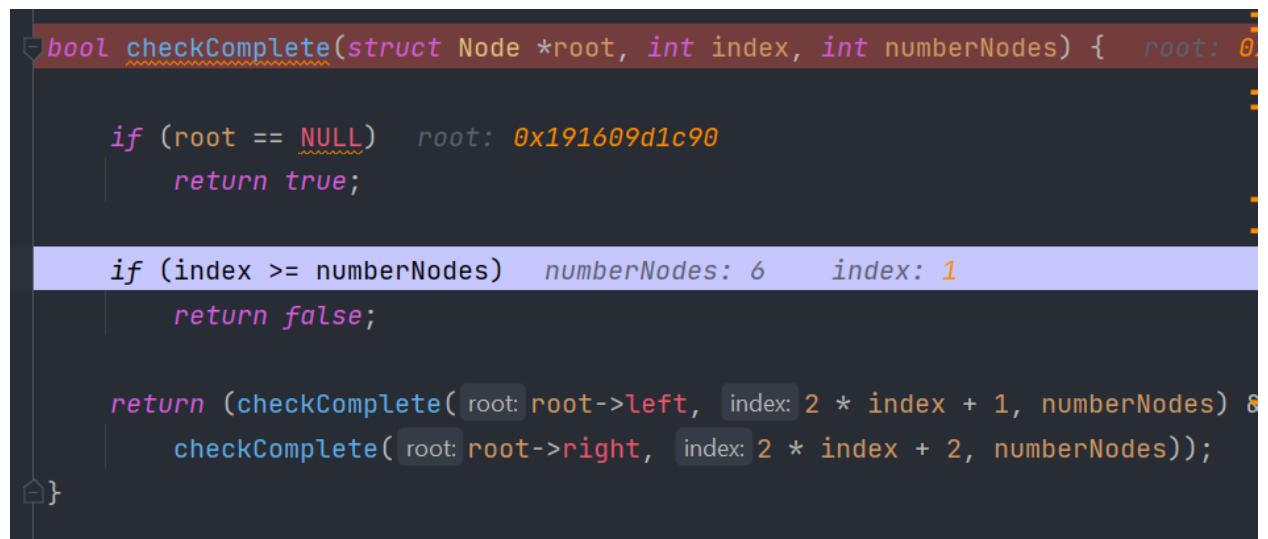
နောက် တဆင့် အနေဖြင့် index number သည် numberNodes ထက် ကြီးနေသလား ညီနေသလား စစ်ဆေးပါတယ် အကယ်၍ ကြီးနေလျှင် ညီနေလျှင် false ဟု သတ်မှတ်ပါတယ်။

Line number 31 မှာတော့ checkComplete ဆိုသည့် function ကို recursive နည်းလမ်းဖြင့် ပြန်ခေါ်ပါတယ်။ ထိုသို့ ပြန်ခေါ်ရာတွင် index ကို 2 ဖြင့် မြောက်ကာ 1 ဖြင့် ပေါင်းပါ သေးတယ် အဘယ်ကြောင့် ထိုသို့ လုပ်ရသည် ဆိုသော် index number ကို ရရှိရန် ဖြစ်တယ်။



ပထမဆုံး အကြိမ် recursive ပြန်ခေါ်သော အခါတွင် Index တန်ဘိုးသည် 1 ဖြစ်သွားမှာ ပါ အဘယ်ကြောင့် ဆိုသော 2*index + 1 တွင် index သည် 0 ဖြစ်သောကြောင့်ပါ။ အောက်ပါ ပုံတွင် ကြည့်ပါ။

ပထမဆုံး အကြိမ်ဝင်လာသည့် index 0 သည် root node ဖြစ်သည့် 1 ကို စစ်ဆေးခြင်း ဖြစ်ပြီး ဒုတိယ အကြိမ် ဝင်လာသည့် index 1 သည် node 2 ကို စစ်ဆေးခြင်း ဖြစ်သည်။ second time recursive ကို ပြန်ခေါ်သော အခါတွင် index သည် 2*1+1 ဖြစ်သောကြောင့် 3 ဖြစ်မှာပါ။ index သည် 3 ဖြစ်သောကြောင့် node 4 ကို သွားစစ်မှာ ဖြစ်ပါတယ် node 4 သည် root->left-> left ဖြစ်ပါတယ်။

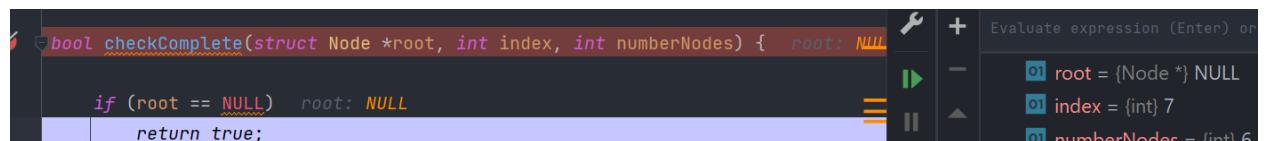


```
bool checkComplete(struct Node *root, int index, int numberNodes) {    root: 0
    if (root == NULL)    root: 0x191609d1c90
        return true;

    if (index >= numberNodes)    numberNodes: 6      index: 1
        return false;

    return (checkComplete( root: root->left, index: 2 * index + 1, numberNodes) &
            checkComplete( root: root->right, index: 2 * index + 2, numberNodes));
}
```

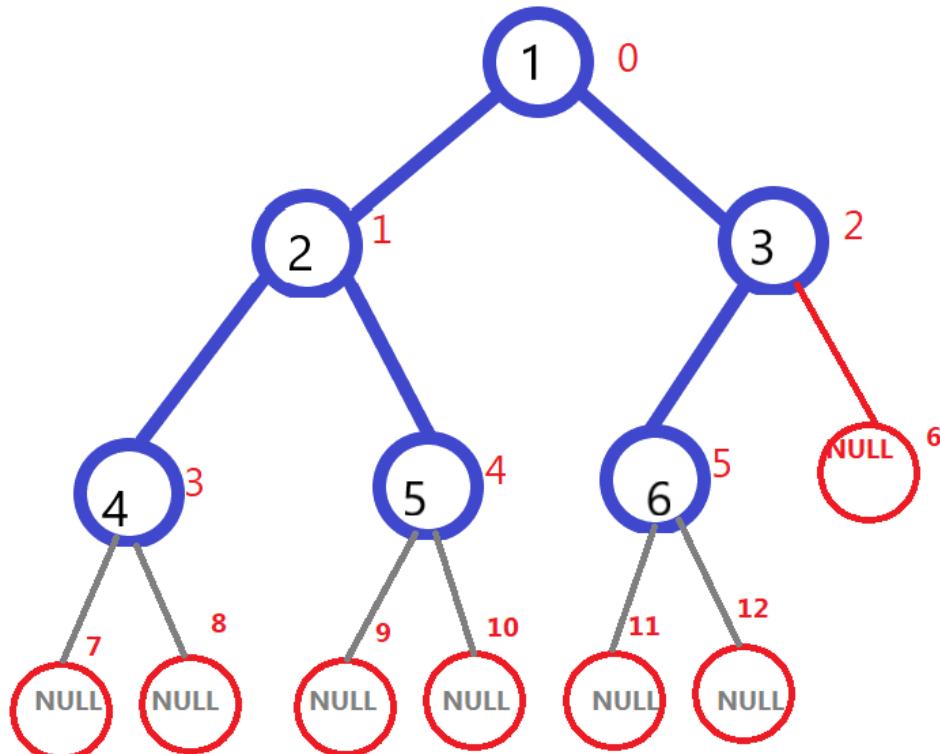
Third time recursive ကို ခေါ်သော အခါတွင်မူ index သည် 7 ဖြစ်နေမှာပါ။ ထို အခါတွင် root သည် node 4 ရဲ့ left ကို ထောက်ထားတာ ဖြစ်တဲ့ အတွက် NULL ဖြစ်နေမှာပါ NULL ဖြစ်နေလျှင် true ကို return ပြန်ပေးမှာ ဖြစ်ပါတယ်။



```
bool checkComplete(struct Node *root, int index, int numberNodes) {    root: NULL
    if (root == NULL)    root: NULL
        return true;
```

ထို့နောက် node 4 ရဲ့ left ကို ပြီးသွားသည့် အတွက် line 32 program အရ node 4 ရဲ့ right ကို သွားမှာ ဖြစ်ပါတယ်။ right သည် 2*index + 3 ပုံစံဖြင့် သွားပါတယ်။ အဘယ်ကြောင့် ဆိုသော left and right သည် ကပ်ရပ်ဖြစ်ပြီး left သည် အရင်လာပြီးမှ right ကို လာပါတယ်။ ထို့ကြောင့် left သည် +1 ပေါင်းပြီး right ကိုတော့ +2 ပေါင်းပါတယ် ထိုနည်းအားဖြင့် index number အမှန်ကို ရရှိနိုင်ပါတယ်။

Node 4 ရဲ့ right အတွက် index number သည် 8 ရှိမှုပါ အဘယ်ကြောင့်ဆိုသော် node 4 တွင် index သည် 3 ($2*1+1$) ဖြစ်သည်။ node 4 ရဲ့ right ကို ထပ်သွားသော အခါတွင် ($2*1+2$) ဖြစ်သည့်အတွက် index သည် 8 ဖြစ်နေမှုပါ သို့သော် NULL ဖြစ်နေသည့် အတွက် line 25 တွင် အဆုံး သတ်သွားဖြစ်ပါတယ် အောက်ပါ ပုံတွင် index number ချထားပုံကို ရရှိပြု၍ ကြည့်ပါ။

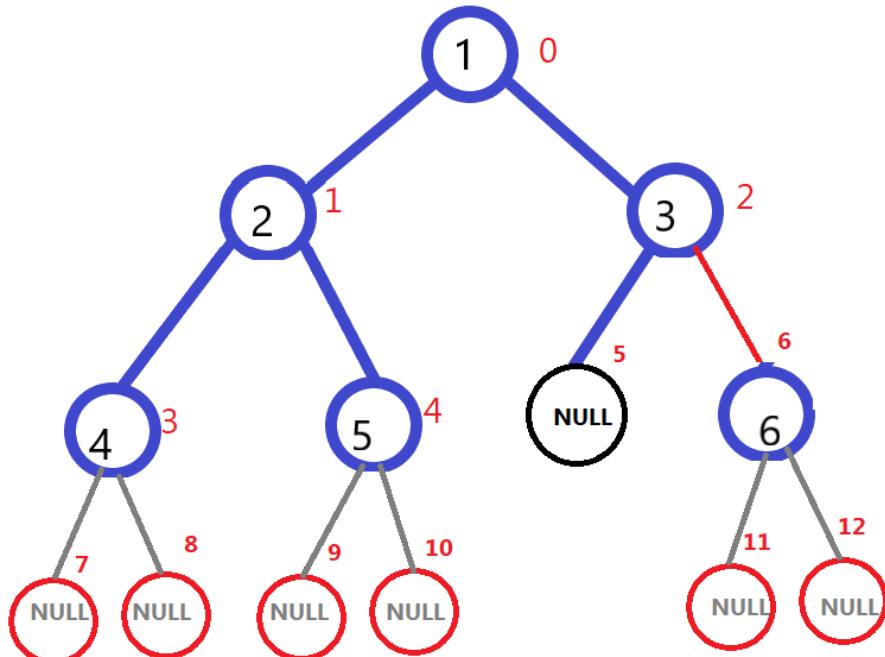


ထိနည်းတဲ့ node 5 ရဲ့ left and right တို့သည့်လည်း NULL ဖြစ်နေသည့် အတွက် line 25 တွင်သာ ရပ်တန်သွားကာ true ကို return ပြန် ပေးမှာ ဖြစ်ပါတယ်။ node 6 သည်လည်း ထိနည်း များ အတိုင်းပင် ဖြစ်သည်။ return value အားလုံးသည် true ဖြစ်နေသည့် အတွက် နောက်ဆုံး output တွင် true ကို ရရှိမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် complete binary tree ဖြစ်ကြောင်းကို တွေ့ရမှာ ဖြစ်ပါတယ်။

သို့သော် node 3 အောက်က child နှစ်ခု ဖြစ်သည့် left and right ကို နေရာ ချိန်းမည် ဆိုပါစို့ node 6 ရဲ့ index number သည် 6 ဖြစ်သွားမှုပါ။ ထိုကဲ့သို့ ဖြစ်သွားလျှင် line 28 မှ condition နှင့်

ညီသွားသည့် အတွက် false ကို return ပြန်ရမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် complete binary tree မဟုတ်ကြောင်း output ကို ပြန်ရမှာ ဖြစ်ပါတယ်။ အောက်ပါ ပုံတွင်ကြည့်ပါ။

Program အပြည့်အစုံ



```

#include <iostream>
using namespace std;

struct Node {
    int key;
    struct Node *left, *right;
};

struct Node *newNode(char k) {
    struct Node *node = (struct Node *)malloc(sizeof(struct Node));
    node->key = k;
    node->right = node->left = NULL;
    return node;
}

// node အရေအတွက်ကို အရင်ဆုံး ရေတွက်ရန် nodes ဆိုတဲ့ function ကိုရေးထားပါတယ်။
int nodes(struct Node *root) {
    if (root == NULL)

```

```

        return (0);

    return (1 + nodes(root->left)+ nodes(root->right));
}

bool checkComplete(struct Node *root, int index, int numberNodes) {

    if (root == NULL)
        return true;

    if (index >= numberNodes)
        return false;

    return (checkComplete(root->left, 2 * index + 1, numberNodes) &&
            checkComplete(root->right, 2 * index + 2, numberNodes));
}

int main() {
    struct Node *root = NULL;
    root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    root->right->left = newNode(6);

    int numNodes = nodes(root);
    int index = 0;

    if (checkComplete(root, index, numNodes))
        cout << "Complete Binary Tree Found \n";
    else
        cout << "Complete Binary Tree not Found\n";
}

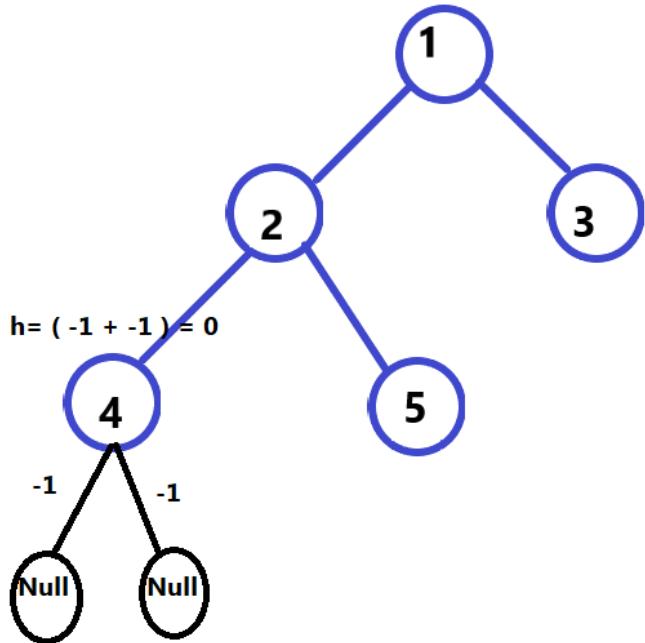
```

Github Link : <https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/BinaryTree-complete.cpp>

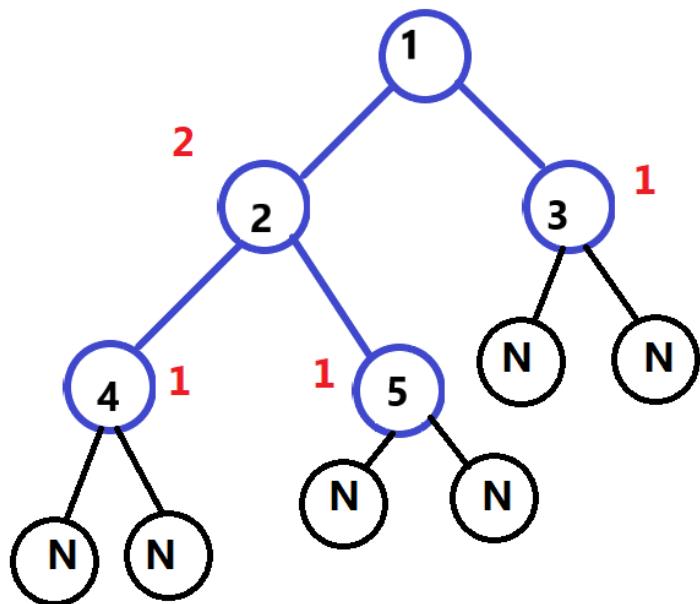
Balanced Binary Tree

Balanced Binary Tree ဆိုသော Null node တွင် parent node အတွက် height ၏

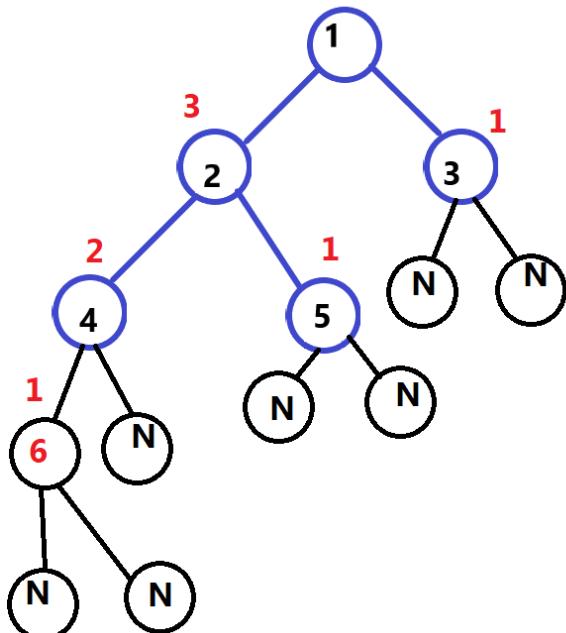
- 1 အဖြစ် သတ်မှတ်ပါတယ် ထို့ကြောင့် Null nodes များသာ ရှိသည့် node သည် balanced ဖြစ်ပြီး သူရဲ့ height သည်လည်း 0 ဖြစ်သည်။
- $-1 + (-1) = 0$ (height of a node)



အထက်ပါ ပုံတွင်မူ 2 node ရဲ့ height သည် 4 က ကြည့်မည် ဆိုလျှင် 1 ဖြစ်သည် အဘယ်ကြောင့် ဆိုသော် $\max(0 + 0) + 1 = 1$ ရရှိခြင်း ဖြစ်သည်။ ပထမ 0 တစ်ခုသည် 4 node မှ ရရှိပြီး ဒုတိယ 0 သည် 5 node မှ ရရှိခြင်း ဖြစ်သည်။ ယခုနည်း အတိုင်းကြည့်မည်ဆိုလျှင် အနည်းငယ် ရှုပ်ထွေးနိုင်သည် ထို့ကြောင့် အောက်ပါ နည်းလမ်းတစ်ခုကို ထပ်မံ ဖော်ပြ ပေးလိုက်ပါသည်။



အထက်ပါ ပုံအတိုင်း leaf node အားလုံးအား height 1 ဟုယူလျင်လည်း မှတ်ရလွယ် ကူပါမည်။ level 1 ရောက်ပြီဆိုလျင် left – right အား နှစ်ပေးခြင်းဖြင့် balanced ဖြစ် ဖြစ်ကို သိနိုင်ပါသည်။ အထက်ပါ ပုံတွင် left-node height သည် 2 နှင့် right-node height သည် 1 ဖြစ်သောကြောင့် balanced factor သည် 1 ဖြစ်ပါသည်(left – right)။ ထို့ကြောင့် အထက်ပါ tree သည် balanced ဖြစ်ပါသည်။



အထက်ပါ tree structure အတိုင်း ဆိုလျင်တော့ balanced ဖြစ်တော့မည် မဟုတ်ပါ အဘယ်ကြောင့်ဆိုသော် left သည် 3 ဖြစ်နေပြီး right သည် 1 ဖြစ်နေသောကြောင့် balanced factor မြားနားခြင်းသည် (left - right) 2 ဖြစ်နေပါပြီ။ Program အပြည့်အစုံနှင့် ရှင်းလင်းချက်ကို အောက်တွင် လေ့လာ ကြည့်ပါ။

```
#include <stdio.h>
#include <stdlib.h>
#define bool int
struct node {
    int item;
    struct node *left;
    struct node *right;
};
struct node *newNode(int item) {
    struct node *node = (struct node *)malloc(sizeof(struct node));
    node->item = item;
    node->left = NULL;
    node->right = NULL;
    return (node);
}

bool balanceChecking(struct node *root, int *height) {
    int leftHeight = 0, rightHeight = 0;
    int l = 0, r = 0;

    if (root == NULL) {
        *height = 0;
        return 1;
    }

    l = balanceChecking(root->left, &leftHeight);
    r = balanceChecking(root->right, &rightHeight);
    // calculate height if we got right and left height
    *height = (leftHeight > rightHeight ? leftHeight : rightHeight) + 1;
    // checking our balanced factor if we got two or that is unbalanced
    if ((leftHeight - rightHeight >= 2) || (rightHeight - leftHeight >= 2))
        return 0;

    else
        return l && r;
}
```

```

}

int main() {
    int height = 0;

    struct node *root = newNode(1);
    root->left = newNode(2);
    root->right = newNode(3);
    root->left->left = newNode(4);
    root->left->right = newNode(5);
    root->left->left->left = newNode(6);
    int data=balanceChecking(root, &height);
    if (data)
        printf("Balanced tree was found! ");
    else
        printf("Balanced tree was not found!");
}

39 int main() {
40     int height = 0;
41
42     struct node *root = newNode(1);
43     root->left = newNode(2);
44     root->right = newNode(3);
45     root->left->left = newNode(4);
46     root->left->right = newNode(5);
47     root->left->left->left = newNode(6);
48     int data=balanceChecking(root, &height);
49     if (data)
50         printf("Balanced tree was found! ");
51     else
52         printf("Balanced tree was not found!");
53 }

```

ပထမ ဆုံး အနေဖြင့် main function ထဲတွင် အထက်ပါ ပုံစံ အတိုင်း node 6 ခုကို
ထည့်သားပါတယ်။ ထို့နောက် line 48 မှာတော့ balanceChecking ဆိုတဲ့ function ကို root and
&height parameter များထည့်ပေးပြီး ခေါ်သားပါတယ်။ ပြန်ရလာသော return value အား data
ထဲသို့ ထည့်ပြီး ပြန်လည် စစ်ဆေးမှာ ဖြစ်ပါတယ် 1 ရလာပါက Balanced Tree ဖြစ်တယ်ဟု
မှတ်ယူပြီး 0 ရပါက Balanced Tree ဖြစ်သည်ဟု မှတ်ယူနိုင်ပါ။

```

18  bool balanceChecking(struct node *root, int *height) {
19      int leftHeight = 0, rightHeight = 0;
20      int l = 0, r = 0;
21
22      if (root == NULL) {
23          *height = 0;
24          return 1;
25      }
26
27      l = balanceChecking(root->left, &leftHeight);
28      r = balanceChecking(root->right, &rightHeight);
29      // calculate height if we got right and left height
30      *height = (leftHeight > rightHeight ? leftHeight : rightHeight) + 1;
31      // checking our balanced factor if we got two or that is unbalanced
32      if ((leftHeight - rightHeight >= 2) || (rightHeight - leftHeight >= 2))
33          return 0;
34
35      else
36          return l && r;
37  }

```

balanceChecking function ထဲသို့ ပထမဆုံး အကြိမ် ဝင်လာသောအခါတွင် ယခင် သင်ခန်းစာ များအတိုင်း root node ကနေ စတင် ဝင်လာမှုဖြစ်ပြီး height ကတော့ 0 ပါ။ ထို့နောက် leftHeight and rightHeight တို့ မှတ်သားရန် variable များ ကြော်ထားပြီး return ပြန်ရန်လည်း l and r ကို ကြော်ထားပါတယ်။

line 22 တွင် root သည် NULL မဖြစ်သည့် အတွက် line 27 ကို ဆက် အလုပ် လုပ်ပါတယ်။ နောက်တစ်ကြိမ် balanceChecking function ကို ပြန်ခေါ်သော အခါတွင်မူ root left ဖြင့် ခေါ်ပါတယ်။

1 node memory address 16e0

2 node memory address 1730

3 node memory address 1780

4 node memory address 17d0

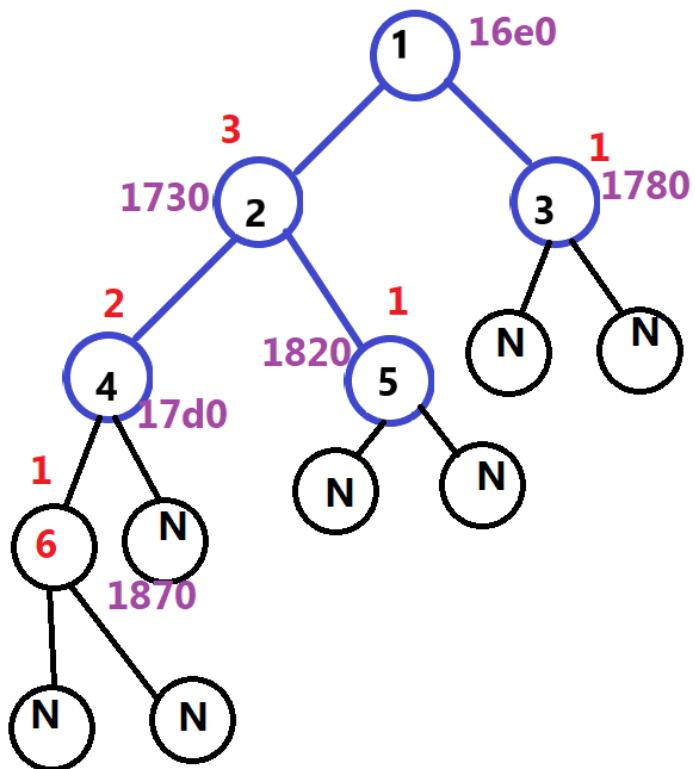
5 node memory address 1820

6 node memory address 1870 တို့ဖြစ်ပါတယ်။

```

01 height = {int} 0
-
02 root = {node *} 0x196c16916e0
03 item = {int} 1
-
04 left = {node *} 0x196c1691730
05 item = {int} 2
-
06 left = {node *} 0x196c16917d0
07 item = {int} 4
-
08 left = {node *} 0x196c1691870
09 item = {int} 6
0a left = {node *} NULL
0b right = {node *} NULL
0c right = {node *} NULL
-
0d right = {node *} 0x196c1691820
0e item = {int} 5
0f left = {node *} NULL
0g right = {node *} NULL
-
0h right = {node *} 0x196c1691780
0i item = {int} 3
0j left = {node *} NULL
0k right = {node *} NULL

```



1. balanceChecking function നിരീക്ഷാ അനുംതിപ്പിച്ച ഫൂണ്ടിഷ്യർ

2. balanceChecking function ကို ပြန်ခေါ်သော အခါတွင်မူ 17d0 ဖြင့်သွားပါတယ်
3. balanceChecking function ကို ပြန်ခေါ်သော အခါတွင် 1870 ဖြင့် သွားပြီး
4. balanceChecking function ကို နောက်ဆုံး ခေါ်သော အခါတွင် 1870->left ဖြစ်သော NULL ဖြင့် သွားပါတယ် ထို့ကြောင့် line 22 condition ကြောင့် အဆုံး သတ်သွားပြီး *height ကို 0 အဖြစ် သတ်မှတ်ပေးကာ 1 ကို return ပြန် ပေးပါတယ်။ 1870 ရဲ့ right သည်လည်း ထိုနည်းတူ 1 ကိုသာ return ပြန် ပေးပါတယ်။

ထို့ကြောင့် node 6 (1870) အတွက် return value တို့သည် 1 များသာ ဖြစ်သည့် အတွက်။ and r value တို့သည်လည်း 1 များသာ ဖြစ်နေပါမည်။

```

    *height = 0;
    return 1;

    l = balanceChecking( root: root->left, height: &leftHeight);
    r = balanceChecking( root: root->right, height: &rightHeight);
    // calculate height if we got right and left height
    *height = (leftHeight > rightHeight ? leftHeight : rightHeight);
}

```

ထို့နောက် Line 30 ကို ဆက်လက် အလုပ်လုပ်မှာ ဖြစ်သည်။ leftHeight သည်လည်း ယခု အခြေနေတွင် 0 rightHeight သည်လည်း ယခု အခြေနေတွင် 0 သာ ဖြစ်သောကြောင့် 0 ကို သာ return ပြန် ပေးမှာ ဖြစ်ပါတယ် ရလာသော 0 ကို 1 ပေါင်းလိုက်သောကြောင့် *height တန်ဘိုးသည် ယခု အခါတွင် 1 ကို ရရှိနေမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် node 6 အတွက် height ကို ရှာသော အခါ 1 ရနေဖြင့် ဖြစ်သည်။

Line number 32 ကတော့ unbalanced ဖြစ်မည့် အခြေနေကို စစ်ဆေးထားခြင်း ဖြစ်သည်။

ထို့နောက် line 36 ကို အလုပ်လုပ်သွားပြီး return အနေဖြင့် 1 and r တန်ဘိုးများကို ပြန်ပေးပါတယ် မှတ်သားရန်မှာ node 6 အတွက်သာ ဖြစ်သည်။

နောက်တစ်ဆင့် အနေဖြင့် Node 4 (17d0) ရဲ့ right ကို အလုပ် ဆက်လုပ်မှာ ဖြစ်ပါတယ်။

```

l = balanceChecking( root: root->left, height: &leftHeight);  LeftHeight: 1
r = balanceChecking( root: root->right, height: &rightHeight);  root: 0x196c16917d0
// calculate height if we got right and left height
*height = (leftHeight > rightHeight ? leftHeight : rightHeight) + 1;
// checking our balanced factor if we got two or that is unbalanced
if ((leftHeight - rightHeight >= 2) || (rightHeight - leftHeight >= 2))
    return 0;

else
    return l && r;

```

The screenshot shows a debugger interface with a code editor on the left and a variable viewer on the right. The code editor contains C-like pseudocode for a balance checking function. The variable viewer shows the following state:

- root**: `{node *} 0x196c16917d0`
- item**: `(int) 4`
- left**: `{node *} 0x196c1691870`
- right**: `{node *} NULL`
- height**: `(int *) 0x40713ff5a4`
- *height**: `{int} 0`
- leftHeight**: `{int} 1`
- rightHeight**: `{int} 0`
- l**: `{int} 1`
- r**: `{int} 0`

Node 4 ရဲ့ right သည်လည်း null ဖြစ်နေသည့် အတွက် line 22 ကို အလုပ်လုပ်သွားပြီး 1 ကို return ပြန်ရမှာ ဖြစ်ပါတယ်။ သတိပြုရန် အချက်မှာ Node 4 အတွက်လည်း height ကို တွက်ပေးရန် လိုအပ်ပါတယ် ထိုကြောင့် line 30 ကို အလုပ် ဆက်လုပ်မှာ ဖြစ်ပါတယ်။ node 4 ရဲ့ leftHeight သည် node 6 ရှိနေသည့် အတွက် 1 ပါ သို့သော် rightHeight သည် 0 ပါ။ line 30 တွင် condition စစ်ဆေးသောအခါ leftHeight ပြန်ရပြီး 1 ပေါင်း လိုက်သည့် အတွက် height တန်ဘိုးသည် 2 ဖြစ်သွားမှပါ။

Node 4 အတွက်လည်း l and r များကို ဆက်လက် return ပြန်ပေးသွားမည် ဖြစ်ပြီး 1 များသာ ဖြစ်နေမှာပါ။ ထို့နောက် 2 node (1730) ရဲ့ right ဘက်ကို ရှာကြည့်ရန် လိုအပ်ပါသေးသည် သို့မှသာ 2 node ရဲ့ height ကို သိရှိနိုင်မှာ ဖြစ်ပါတယ်။ 5 node ရဲ့ left and right သည် NULL များသာ ဖြစ်နေသည့် အတွက် 1 ကိုသာ ရရှိမှာ ဖြစ်ပါတယ်။ ထို့နောက် Line 30 တွင် left and right တို့သည် 0 များ သာ ဖြစ်နေသည့် အတွက် 1 ပေါင်းကာ height ထဲသို့ ထည့်ပေးလိုက်မှာ ဖြစ်ပါတယ်။

ထို့ကြောင့် Node 4 height သည် 2 ဖြစ်ပြီး node 5 height သည် 1 ဖြစ်နေသည်ကို တွေ့ရပါမည်။ ဤသို့ဖြင့် node 2 အတွက် Height ရှာသော အခါတွင် line 30 ၏ leftHeight သည် 2 ဖြစ်ပြီး rightHeight သည် 1 ဖြစ်နေသောကြောင့် line 30 တွင် leftHeight ကို 1 ထပ်ပေါင်းပါမည်။ ထို့ကြောင့် node 2 height တန်ဘိုးသည် 3 ရရှိနေပါမည်။

နောက်တစ်ဆင့် အနေဖြင့် node 1 (16e0) ရဲ့ height ကို သိရှိနိုင်ရန် node 3(1780) ရဲ့ height ကို ရှာရန် ကျဉ်နေပါသေးသည်။ node 3 သည် 1 သာ ရမည် ဖြစ်သောကြောင့် line 32 ကို စစ်ဆေးသော အခါတွက် leftHeight and rightHeight ပြားနားခြင်းသည် 2 ဖြစ်နေသည့် အတွက်

0 ကို return ပြန် ပေးလိုက်ပါသည် ထို့ကြောင့် output အနေဖြင့် Balanced tree was not found!
ဆိုတာကို တွေ့ရမှာ ဖြစ်ပြီး balanced tree မဟုတ်ကြောင်းကို ဖော်ပြသွားမှာ ဖြစ်ပါတယ်။
သတိပြုရန် အချက်များ။

1. Node 4 အတွက် left node ကို ခေါ် ပြီးသော အခါတွင် leftHeight တန်ဘိုးသည် 1 ရရှိသွားသည်ကို သတိပြုပါ။ right node ဖြစ်တဲ့ null ကို ခေါ် ပြီးသည့် အချိန်တွင်လည်း rightHeight သည် 0 ဖြစ်နေသည်ကို သတိပြုပါ။ ထို့ကြောင့် Node 4 အတွက် *height သည်လည်း 2 သွားပါမည်။(see line 30)
2. Node 2 အတွက် left node ကို ခေါ် ပြီးသော အခါတွင်လည်း leftHeight သည် 2 ဖြစ်နေသည်ကို သတိပြုပါ right node 5 ကို ခေါ် ပြီးချိန်မှာတော့ 1 ဖြစ်ပါတယ် ထို့ကြောင့် *height တန်ဘိုးသည် 3 ဖြစ်သွားပါမည်။

Program Github link : <https://github.com/WinHtut/DSA-Code-For-MyBook/blob/main/BinaryTree-Balanced.cpp>

Binary Search Tree ရေးရန်

AVL Tree

AVL tree ကို Soviet နှစ်ဦးဖြစ်တဲ့ Adelson-Velsky and Evgenii Landis တို့က 1962 မှတိထွင် ခဲ့တာ ဖြစ်ပါတယ် AVL tree ကို သူတို့ရဲ့ An algorithms for the organization of information ဆိုတဲ့ paper မှာ စတင် ဖော်ပြ ခဲ့တာပါ။ AVL tree သည် self-balancing Binary Search tree ဟုတ်လည်း နံမည် ကြီးပါတယ် ထို့ကြောင့် ထည့်လိုက်သော node အသစ်များအား သူဘာသာ နေရာ သတ်မှတ် ပေးသွားပြီး tree structure တစ်ခုလုံးကို သူဘာသာ တည်ဆောက်သွားမှာ ဖြစ်ပါတယ်။ အကယ်၍ Balanced မည့်ခဲ့ရင်လည်း အတက်နိုင်ဆုံး အညီဆုံး ဖြစ်အောင် သူဘာသာ auto rotate လုပ်ပြီး balanced ဖြစ်သွားအောင် လုပ်ဆောင်သွားမှာ ဖြစ်ပါတယ် ထို့ကြောင့် AVL tree အား self-balancing ဟု ခေါ်ခြင်း ဖြစ်ပါတယ်။ အသေးစိတ် ရှင်းလင်းချက်ကို အောက်တွင် ဖော်ပြ ပေးသွားမှာ ဖြစ်ပါတယ်။

1. ပထမဆုံး အနေဖြင့် ဝင်လာသည့် node ကို root node အဖြစ် ထားပါတယ်။

2. ဒုတိယ အနေဖြင့် ဝင်လာသည့် node သည် root node ထက် ကြီးသလား ငယ်သလား ကို စစ်ဆေးပါတယ် ထို့နောက် ကြီးရင် right ဘက်ခြမ်းသို့ ပို့ပြီး ငယ်လျှင် left ဘက် ခြမ်းသို့ ပို့ပါတယ်။
3. ထိုသို့ ပို့ပြီးသည့်နှင့် တပြုင်နက် height ကို ရှာပါတော့တယ်
4. Height ရှာ လို့ balanced factor နှင့် ညီနေသည် ဆိုလျှင် မည်သည့် အရာမှ ဆက်မလုပ်ပါ သို့သော်
5. Balanced factor နှင့် မညီဘူးဆိုလျှင် right rotate လှည့်မလား left rotate လှည့်မလား ကြည့်ရပါတယ်။ rotate လှည့်တယ် ဆိုတာ data တွေကို နေရာ ချိန်းပေးပြီး tree structure ကိုလည်း နေရာတွေ ချိန်းပေးလိုက်တာပါ။ အသေးစိတ်ကိုတော့ rotate အကြောင်းရှင်းရာတွင် ထည့်သွင်းရှင်းလင်းပါမယ်။

Sample Program

```
#include <iostream>
using namespace std;

// node တစ်ခု တည်ဆောက်ရာတွင် height ကိုပါ ထည့်ဆောက်ထားသည်ကို
// သတိပြုပါ

class Node {
public:
    int key;
    Node *left;
    Node *right;
    int height;
};


```

```
int max(int a, int b);

int height(Node *N) {
    if (N == NULL)
        return 0;
    return N->height;
}

int max(int a, int b) {
    return (a > b) ? a : b;
}

Node *newNode(int key) {
    Node *node = new Node();
    node->key = key;
    node->left = NULL;
    node->right = NULL;
    node->height = 1;
    return (node);
}

Node *rightRotate(Node *y) {
    Node *x = y->left;
    Node *T2 = x->right;
    x->right = y;
    y->left = T2;
    y->height = max(height(y->left),
                      height(y->right)) +
                 1;
    x->height = max(height(x->left),
                      height(x->right)) +
                 1;
    return x;
}

Node *leftRotate(Node *x) {
    Node *y = x->right;
    Node *T2 = y->left;
    y->left = x;
    x->right = T2;
    x->height = max(height(x->left),
                      height(x->right)) +
```

```

        1;
y->height = max(height(y->left),
                  height(y->right)) +
        1;
return y;
}

int getBalanceFactor(Node *N) {
    if (N == NULL)
        return 0;
    return height(N->left) -
        height(N->right);
}

Node *insertNode(Node *node, int key) {
    // root node ဘယ်ဘက်ခြမ်းမှာ data
    // ထည့်ရမလား ညာဘက် ခြမ်းမှာ data ထည့်ရမလား စစ်ဆေးခြင်း ဖြစ်ပါတယ်။
    if (node == NULL)
        return (newNode(key));
    if (key < node->key)
        node->left = insertNode(node->left, key);
    else if (key > node->key)
        node->right = insertNode(node->right, key);
    else
        return node;

    // node တစ်ခုခြင်း ရဲ့ height ပျော်ကို ပြန်တွက်ပြီး height သတ်မှတ်ပါတယ်
    // ယခု AVL tree program တွင် အခြား program များနှင့် မူတည့်သော အချက်မှာ
    // node တည်ဆောက်ကတည်းက height တန်ဘိုး 1 ဟု ပေးခဲ့ လိုက်ခြင်းပင် ဖြစ်ပါတယ်။
    node->height = 1 + max(height(node->left),
                           height(node->right));
    int balanceFactor = getBalanceFactor(node);
    // အကယ်၍ balance factor သည် 1 ထက်ကြီးနေပါက
    // node ထဲက left->key သည် key ထက် ကြီးရင် rightRotate နှင့်
    // ငါးနေလျှင် LR rotate တိုကို လုပ်ဆောင်ပါတယ်။
    if (balanceFactor > 1) {
        if (key < node->left->key) {
            return rightRotate(node);
        } else if (key > node->left->key) {
            node->left = leftRotate(node->left);
        }
    }
}

```

```

        return rightRotate(node);
    }
}

// အကယ်၍ balance factor သည် 1 ထောင်ပါက
// node ထဲမှာ right->key သည် key ထက် ငယ်ရင် leftRotate နှင့်
// ကြီးနေလျှင် RL rotate တိုကို လုပ်ဆောင်ပါတယ်။
if (balanceFactor < -1) {
    if (key > node->right->key) {
        return leftRotate(node);
    } else if (key < node->right->key) {
        node->right = rightRotate(node->right);
        return leftRotate(node);
    }
}
return node;
}

Node *nodeWithMinimumValue(Node *node) {
    Node *current = node;
    while (current->left != NULL)
        current = current->left;
    return current;
}

Node *deleteNode(Node *root, int key) {

    if (root == NULL)
        return root;
    if (key < root->key)
        root->left = deleteNode(root->left, key);
    else if (key > root->key)
        root->right = deleteNode(root->right, key);
    else {
        if ((root->left == NULL) ||
            (root->right == NULL)) {
            Node *temp = root->left ? root->left : root->right;
            if (temp == NULL) {
                temp = root;
                root = NULL;
            } else
                *root = *temp;
        }
    }
}

```

```

        free(temp);
    } else {
        Node *temp = nodeWithMinimumValue(root->right);
        root->key = temp->key;
        root->right = deleteNode(root->right,
                                   temp->key);
    }
}

if (root == NULL)
    return root;

root->height = 1 + max(height(root->left),
                        height(root->right));
int balanceFactor = getBalanceFactor(root);
if (balanceFactor > 1) {
    if (getBalanceFactor(root->left) >= 0) {
        return rightRotate(root);
    } else {
        root->left = leftRotate(root->left);
        return rightRotate(root);
    }
}
if (balanceFactor < -1) {
    if (getBalanceFactor(root->right) <= 0) {
        return leftRotate(root);
    } else {
        root->right = rightRotate(root->right);
        return leftRotate(root);
    }
}
return root;
}

void printTree(Node *root, string indent, bool last) {
    if (root != nullptr) {
        cout << indent;
        if (last) {
            cout << "R----";
            indent += "    ";
        } else {
            cout << "L----";
            indent += "|   ";
        }
        if (root->left != nullptr)
            printTree(root->left, indent + "    ", false);
        if (root->right != nullptr)
            printTree(root->right, indent + "    ", true);
    }
}

```

```

    }
    cout << root->key << endl;
    printTree(root->left, indent, false);
    printTree(root->right, indent, true);
}
}

int main() {
    Node *root = NULL;
    root = insertNode(root, 50);
    root = insertNode(root, 40);
    root = insertNode(root, 65);
    root = insertNode(root, 30);
    root = insertNode(root, 45);
    root = insertNode(root, 78);
    root = insertNode(root, 15);
    root = insertNode(root, 37);
    root = insertNode(root, 80);
    printTree(root, "", true);
    root = deleteNode(root, 13);
    cout << "After deleting " << endl;
    printTree(root, "", true);
}

```

AVL tree အသေးစိတ် ရှင်းလင်းချက်

ပထမဆုံး အနေဖြင့် အောက်ပါအတိုင်း data များထည့်ကာ AVL tree ကို စတင် တည်ဆောက်ပါတယ် line 195 ကနေ line 70 ဖြစ်တဲ့ insertNode ကို စတင် ခေါ်ဆိုပါတယ်။

```

70     Node *insertNode(Node *node, int key) {
71         // root node သယ်ယူကြခြင်းမှာ data
72         // ထည့်ရမလား ညာဘက် ခြင်းမှာ data ထည့်ရမလား စစ်ဆေးခြင်း ဖြစ်ပါတယ်။
73         if (node == NULL)
74             return (newNode(key));
75         if (key < node->key)
76             node->left = insertNode(node->left, key);
77         else if (key > node->key)
78             node->right = insertNode(node->right, key);
79         else
80             return node;
81

```

`insertNode` Function ထဲတွင် `node` သည် `NULL` ဖြစ်သလား ဆိုမဟုတ် ဘယ်ဘက်မှာ ထည့်ရမလား ညာဘက်မှာ ထည့်ရမလား ဆိုတာကို စစ်ဆေးပါတယ်။ `node` သည် `NULL` ဖြစ်နေသည့် အတွက် `newNode` function ကို line74 ကနေတဆင့် ခေါ်ဆိုပါတယ်။

```

26     Node *newNode(int key) {
27         Node *node = new Node();
28         node->key = key;
29         node->left = NULL;
30         node->right = NULL;
31         node->height = 1;
32         return (node);
33     }

```

`newNode` function သည် `*node` ဆိုသည့် `node` အသစ် တစ်ခု တည်ဆောက်ပြီး ထို `node` ထဲမှ `key` နေရာတွင် `key` နှင့် `left` and `right` တို့တွင် `NULL` များကို ထည့်ပြီး `height` ကိုလည်း 1 အဖြစ် သတ်မှတ်ကာ `node` ကို `return` ပြန်ပေးလိုက်ပါတယ်။ ထို့ကြောင့် ယခု line number 32 ထိ run ပြီးချိန်တွင် အောက်ပါအတိုင်း `root` node ထဲတွင် တည်ရှိနေမှာ ဖြစ်ပါတယ်။

```

root = {Node *} 0x1717c641eb0
    01 key = {int} 50
    01 left = {Node *} NULL
    01 right = {Node *} NULL
    01 height = {int} 1

```

နောက်တဆင့် မှာတော့ Line number 191 မှာ 40 ဆိုသည့် `data` ကို `insertNode` ကိုခေါ် ပြီးထည့်ပေးလိုက်ပါတယ်။ ယခု တစ်ခါမှာတော့ `node` သည် Null ဖြစ်မနေတော့သည့် အတွက် ကျွန်ုပ်code များကို အလုပ် ဆက်လုပ်မှာဖြစ်ပါတယ်။

```

    ...
    return insertNode(key, ...)

75 → if (key < node->key)  node: 0x1717c641eb0      key: 40
76     |   node->left = insertNode( node: node->left, key);
77     |   else if (key > node->key)
78         |       node->right = insertNode( node: node->right, key);
79     |   else
80         |       return node;

```

မူရင်း ရှိသည့် node တွင် key သည် 50 ဖြစ်ပြီး ယခု ဝင်လာသော key သည် 40 ဖြစ်သည့် အတွက် ဘယ် ဘက်ခြမ်းကို သွားမှာ ဖြစ်ပါတယ်။ ထို့ကြောင့် အထက်ပါ ပုံထဲတွင် line 76 ကို သွားပြီး insertNode ကို ပြန်သွားခေါ်မှာ ဖြစ်ပါတယ်။ Recursive နည်းဖြင့် ယခု function ကိုသာ ပြန်ခေါ်ခြင်း ဖြစ်ပါတယ်။ ယခု insertNode fun ကို ခေါ်သော အခါတွင်မူ node->left ကို ထည့်ပေးလိုက်သည့် 50 node ရဲ့ဘယ်ဘက်ခြမ်းကို သွားမှာ ဖြစ်ပါတယ်။ ထိုနေရာသည်လည်း null ဖြစ်နေသည့် အတွက် newNode function ကို သွားခေါကာ

B Tree

Order 4 ဖြစ်တယ်ဆိုရင်

Max no of children = 4

Max no of Keys = m-1 = 3

Min no of Children = m/2 = 4/2 = 2

Min no of keys = m/2 - 1 = 4/2 - 2 = 1

Deletion on BTREE

No1 အကယ်၍ ဖျက်ချင်သော key သည် leaf node တွင် ရှိနေသည် ဆိုလျှင် case နှစ်ခု ရှိပါမည်

No1.1 Key များကို ဖျက်သော အခါတွင် node တစ်ခုတွင် ရှိရမည့် minimum key properties ကို ချိုးဖောက်လို့ မပါဘူး။ ဆိုလိုသည်မှာ key ကို ဖျက်မည်ဆိုလျှင် minimum key ကိုပါ ထည့်တွက်ပေးရပါမည်။

အကယ်၍ minimum key properties ကို ချိုးဖောက်ခဲ့မည် ဆိုလျှင် neighboring sibling node ကနေ key တစ်ခုကို borrow လုပ်ရပါမည် ထိုသို့ လုပ်သော အခါတွင်လည်း left to right ပုံစံဖြင့် လုပ်ရပါမည်။

ပထမဆုံး အနေဖြင့် left sibling ကို သွားရမည် အကယ်၍ ထို node တွင် keys များသည် minimum number ထက် ကျဉ်နေမည်ဆိုလျှင် key တစ်ခုကို ငှားရမ်းနိုင်ပါသည်။ ထိုကဲ့သို့ မဟုတ်ဘူး ဆိုလျှင် right sibling node ကို check လုပ်ပြီး ငှားပါမည်။

ထိုသို့ ချေးငှားသော အခါတွင်လည်း ချေးငှားခံရသော key သည် ဖျက်ခံရသော key နေရာရဲ့ parent နေရာတွင် ထားရှိရမည် ဖြစ်ပြီး parent နေရာမှ key သည် ဖျက်ခံရသော key နေရာသို့ ဝင်လာရမည် ဖြစ်သည်။

အကယ်၍ sibling nodes နှစ်ခုလုံးသည် minimum number key properties ဖြစ်နေလျှင် merge လုပ်ရပါမည်။ ထိုသို့ merge လုပ်ရတွင်လည်း left sibling သို့မဟုတ် right sibling နဲ့ လုပ်ရပါမည်။ ထိုသို့ merge လုပ်တဲ့ နေရာမှာလည်း parent node ကို ဖြတ်ပြီး လုပ်ရပါမည်။

အကယ်၍ ဖျက်မည့် node သည် internal node ဖြစ်နေလျှင်

No1 ဖျက်ခံရသော internal node နေရာတွင် inorder predecessor ဖြင့် အစားထိုးရမည်

B tree insertion and splitting

B tree တွင် data များ insert လုပ်လိုသော အခါ မိမိ တို့ insert လုပ်မည့် node ရဲ့ maximum key အား ထည့်သွင်း စဉ်းစား ရပါသည်။ ထိုပြင် child nodes များ ရှိနေပါကလည်း ထို child nodes များတွင် မည့်သည့် child node တွင် ထည့်ရမည်ကိုလည်း ထည့်သွင်း စဉ်းစား ရမည်။

maximum key သို့ မရောက် သေးဘူး ဆိုလျင် ငယ်စဉ် ကြီးလိုက် ပုံစံဖြင့် စီစဉ်ပြီး တိုက်ရှိက် ထည့်နိုင်သော်လည်း အကယ်၍ max key ထက် ကျကျလွန် သွားပါက split လုပ်ရန် လိုအပ် နေပါသေးသည်။

Split လုပ်ရန် ပထမ ဦးစွာ node အသစ် တစ်ခု တည်ဆောက်ပါမည် ထို node ကို b ဟု နံမည် ပေးပါမည်။ split လုပ်ပြီး child အနေဖြင့်သာ ထားမှာ ဖြစ်တဲ့ အတွက် ထို node ထဲမှ properties တစ်ခုဖြစ်သည့် leaf အား true ဟု ပေးခဲ့ပါမည်။

ယခု အချိန်၌ split လုပ်ခြင်းသည် root node အား split လုပ်ခြင်း ဖြစ်သည့် အတွက် နောက်ထပ် node တစ်ခု တည်ဆောက်ရန် လိုအပ်ပါသေးသည်။ ထို node name အား s ဟု ပေးထားလိုက်ပါမည်။

ထို s node ထဲတွင် keys များထားရန် နေရာ တစ်ခုနှင့် child node များအား ချိတ်ဆက်ရန် C ဆိုသည့် pointers များရှိနေသည်ကို မှတ်ထားရမည်။

S node ထဲတွင် မည်သည့် key မျှ မထည့် သေးသော်လည်း ထို s node ထဲမှ child node များ အား ထောက်သော pointer ရဲ့ ပထမ ဆုံး တစ်ခု သည် ယခင် root အား စတင် ထောက်ထားပါပြီ။ ထို့ကြောင့် ယခု တစ်ကြိမ် ဆောက်လိုက်သော S node သည် root node ဖြစ်သွားကြောင်းကို သတိ ပြုရမည်။

ထို့နောက် ယခု root node မဟုတ်ပဲ ယခင် root node ထဲတွင် ရှိသော keys များထဲမှ နှစ်ခုအား b node ထဲသို့ ရွှေ့ပါမည်။ ထို့ကြောင့် ယခင် root node ထဲတွင် keys သုံးခု တည်းသာ ရှိတော့ကြောင်း မှတ်သားထားရမည်။ ထို key တစ်ခုအား s node ထဲမှ ပထမဆုံး keys များထားသော နေရာတွင် ထားပေးပါမည်။

အထက်ပါ အဆင့်များ ပြီးသွားသော အခါ တွင် root node အသစ်ဖြစ်လာသည့် s node ထဲတွင် key တစ်ခု ရှိနေမည် ဖြစ်ပြီး ယခင် root node အဟောင်းဖြစ်သည့် key နှစ်ခု ရှိနေမည်။
ထိုပြင် b node ထဲတွင်လည်း key နှစ်ခု ရှိနေမည်။

Root node ဖြစ်သည့် s node နှင့် **root node** အဟောင်းတို့ သည့် ချိတ်ဆက်ပြီးသော်လည်း b နှင့် ချိတ်ဆက် ရခြင်း မရှိသေးပေ ထို့ကြောင့် ချိတ်ဆက်ပေးရန် လိုအပ်သေးသည်။

S node ထဲမှ child nodes များကို ထောက်သည့် pointers c ရဲ့ဒုတိယ pointer သည် b node အား သွားရောက် ထောက်ထားမည် ဖြစ်သည်။

ယခု lesson တွင် b tree node အား array ကို သုံးပြီး implement လုပ်မည် ဆိုပါက number of keys ကို မှတ်သားထားရန် n ဆိုသည့် property တစ်ခုအား node ကို တည်ဆောက် ကတည်း က ထည့်ဆောက် ထားပေးရန် လိုအပ်ပါသည်။

Introduction To C Programming

Software installation

C Programming ရေးသားလို့ရမယ့် လိုအပ်တဲ့ **IDE** တွေအကြောင်းကို ရှင်းပြပါးတော့

Installation လုပ်နည်းအကြောင်းပြောပြပေးမှာဖြစ်ပါတယ်။ ပထမဆုံးအနေနဲ့

C Programming ကိုအရင်က **Code block IDE**မှာ ရေးကြပါတယ်။ ယနေ့ အချိန်ထိ

လဲသုံးလို့ရနေဆဲ ဖြစ်ပါတယ်။

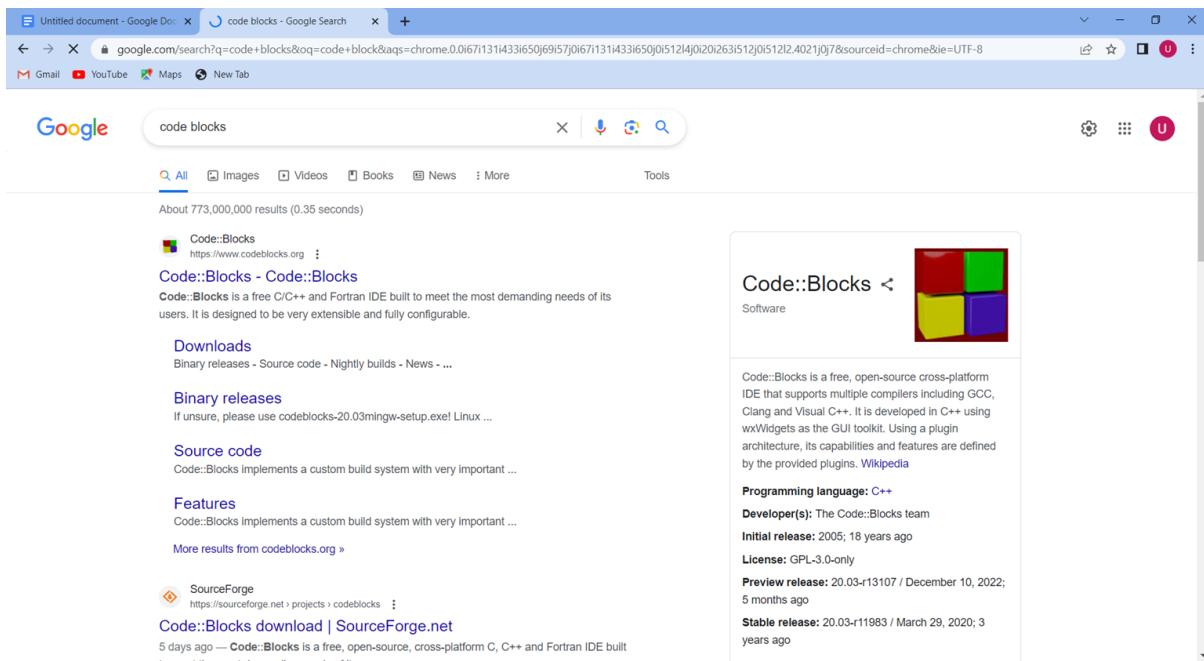
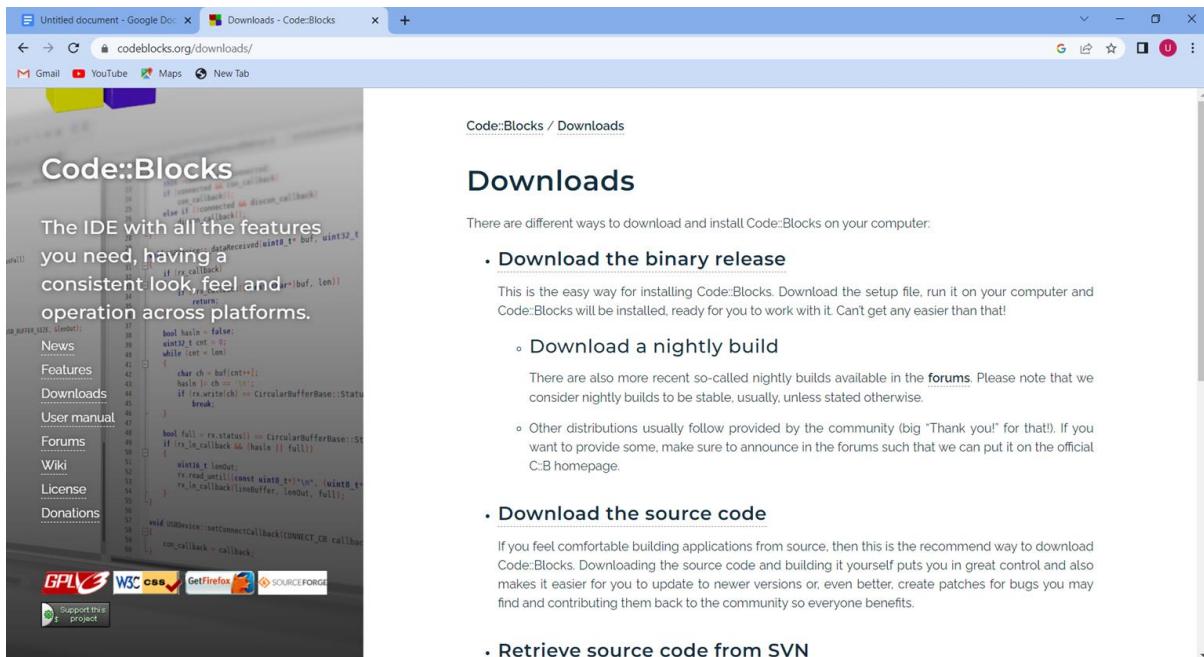


fig-1.installation code block.

fig-2.



ပီးခင် Download the binary release ထဲဝင်လိုက်ပါ။

fig-3 ၊ **codeblocks-20.03mingw-32bit-setup.exe** နေရာမှာ FossHUB သို့မဟုတ်

sourceforce.net ကိုနိုင်ပါး **install**လုပ်လိုက်ပါ။ ဒီနေရာမှာ **32-bits** ဘဲရှိပေမယ့် အဆင်ပြေပါ

တယ်။ **window** သမားတွေအတွက်အဆင်ပြေပါတယ်။ ကျွန်တော်ကတော့ **Sourceforce**.

net ကိုနိုင်ပါး **install** ပြေပါမယ်။ ငှင်းကိုနိုင်ပါးတဲ့ အခါမှာ **Download the latest version**

ထပ်ပါးနိုင်လိုက်ပါ။ **Download link** လေးကျလာပါ း**software** ကို အလိုအလျောက် **Download**

ဉဲသွားပါလိမ့်မယ်။ အကယ်လို့ **Download Link** မကျလာဘူးဆိုရင် **Download** ဆိုတာလေးကို

ထပ်နိုင်ပါး **install** လုပ်နိုင်ပါတယ်။

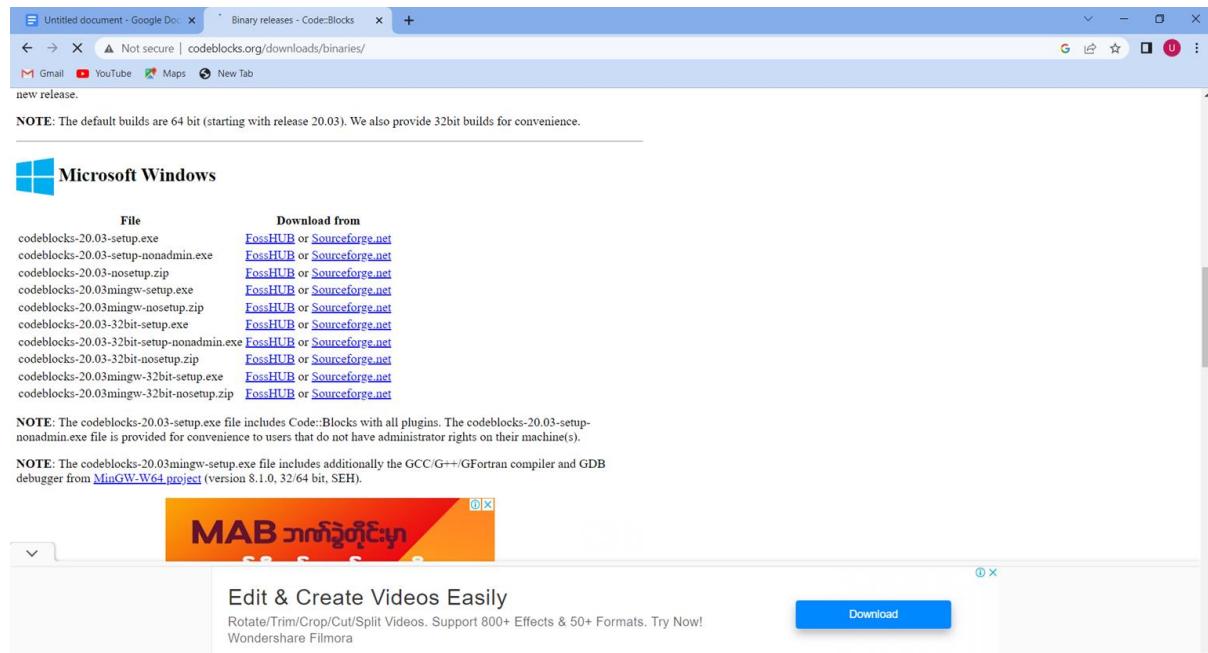


fig-3.

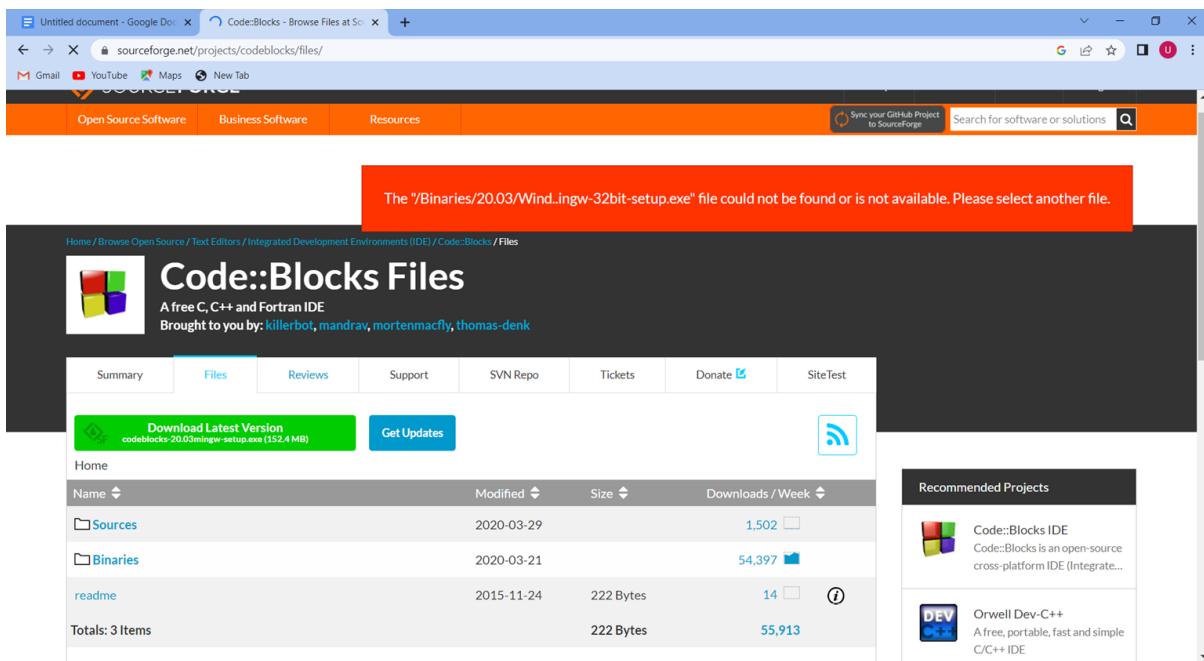


Fig-4

အထက်ပါအတိုင်း **download**လုပ်ပါက **install**ထားတဲ့ **File**ကို **click**ထောက်ပါ **install**လိုက်

ပါ။ပေါ်လာသမျှအကုန်လုံးကို **next** ကိုသာနိုင်ပါ။ **software** ကို **Install**လုပ်ပါက

fig-5မှာပြထားသလိုပေါ်လာပါလိမ့်မယ်။

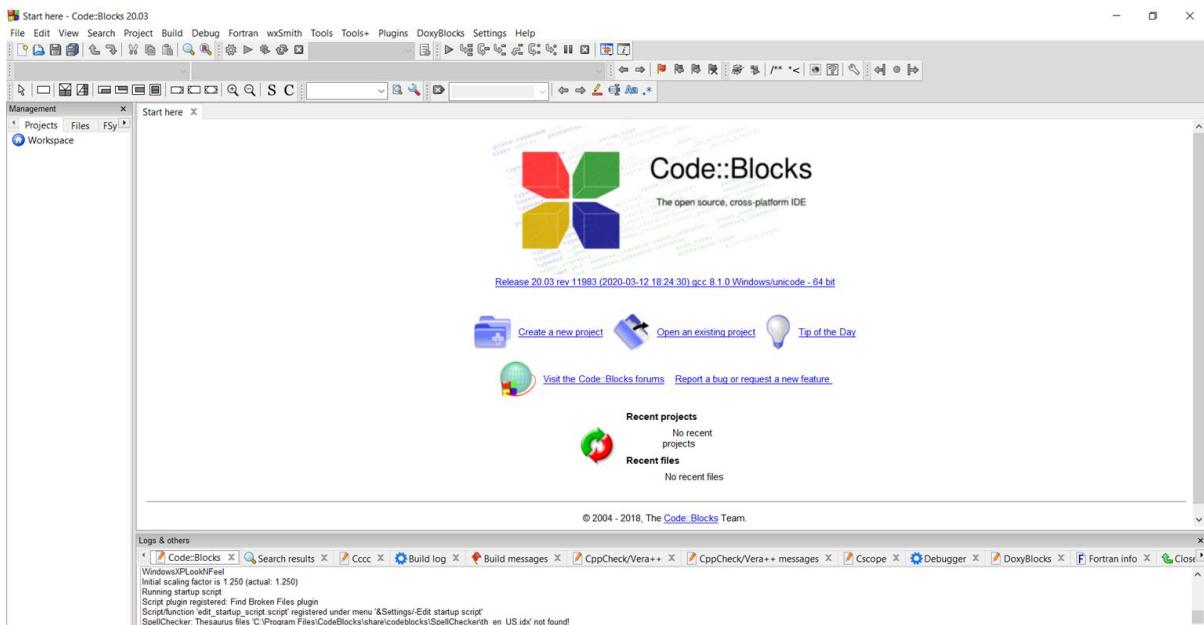
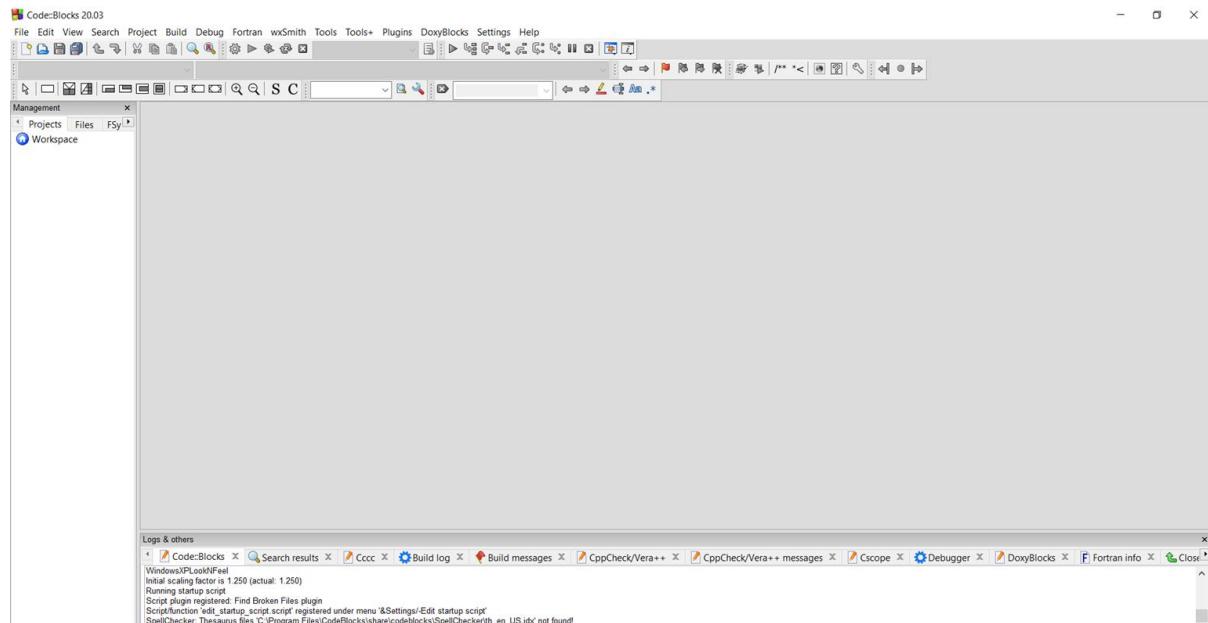


fig-5

Block ရဲ့ ဘယ်ဘက်ထောင့်မှုရှိတဲ့ **starthere** ကအမှားခြစ်လေးကိုနှုပ်လိုက်ပါက **block** ထဲမှုရှိတဲ့ စာများပေါ်က် သွားပါလိမ့်မယ်။**fig-6.**

**fig-6**

အထက်ပါအတိုင်းပါးသွားပါက **program** စရေးဖို့အတွက် **project file** တည်ဆောက်ရန်လိုအပ်ပါတယ်။ **screeen** ရဲ့ ဘယ်ဘက်ထောင့်မှု **File** ကိုနှုပ်ပါ။ ပါးပါက **New** ကိုနှုပ်ပါး **empty file** ကိုနှုပ်ပါး **project** တည်ဆောက်နိုင်ပါတယ်။

Example-

File->New->Empty File

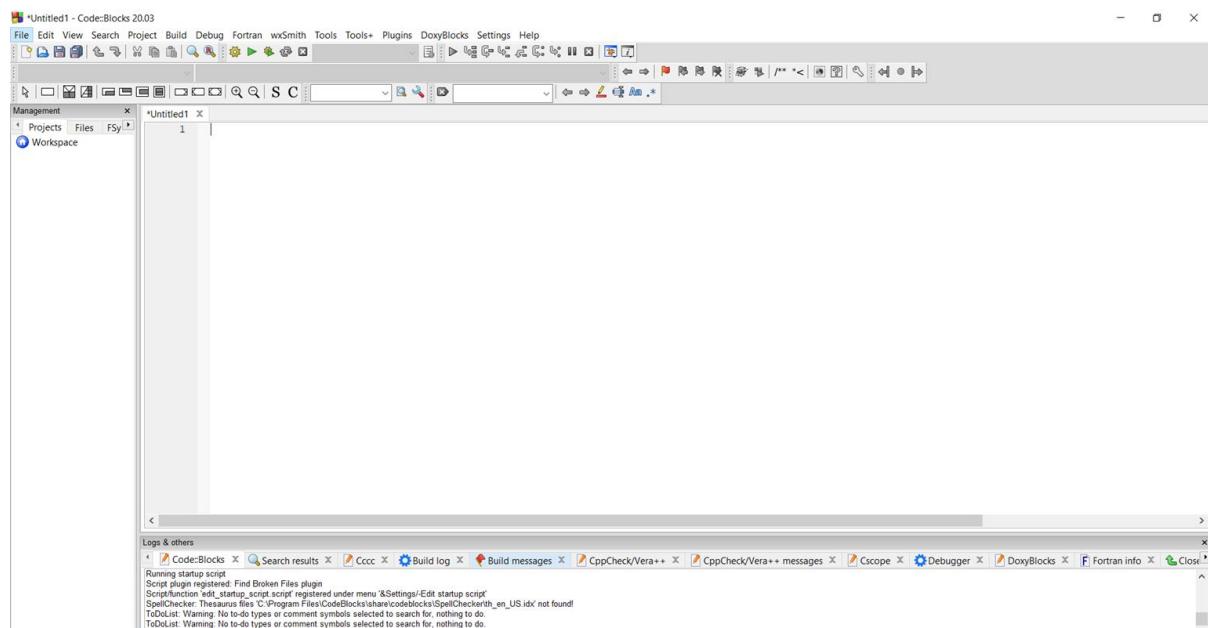


figure-7

Project တည်ဆောက်ပါးပါက **File** ကို **save** ပေးပါရန် လိုအပ်ပါတယ်။ **File save** ရန်

အောက်ပါအတိုင်း ပုံနှင့်တွေ ရှင်းပြထားပါတယ့်။ **File** ထဲကို နောက်တစ်ကြိမ်များပါး

save as file ကိုနှိပ်ပါ။ ထိုနေရာမှာ ကျွန်တော်ကတော့ **Desktop**ထဲမှာ **C Program** ဆို

တဲ့ **folder** ဆောက်ပါး **save** ပါတယ့်။ ပဲ **Figure-8** မှာ ကြည့်နိုင်ပါတယ်။ ပီးရင်တော့

Program စတင်ရေးသားလို့ရပါပါ။ အထက်ပါပြောပြပေးခဲ့တဲ့အကြောင်းအရာတွေ

ကတော့ **programming** ရေးသားဖို့အတွက် လိုအပ်တဲ့ **software install** လုပ်နည်းတွေ

အကြောင်း ပြောပြပေးထားတာဘဲဖြစ်ပါတယ်။

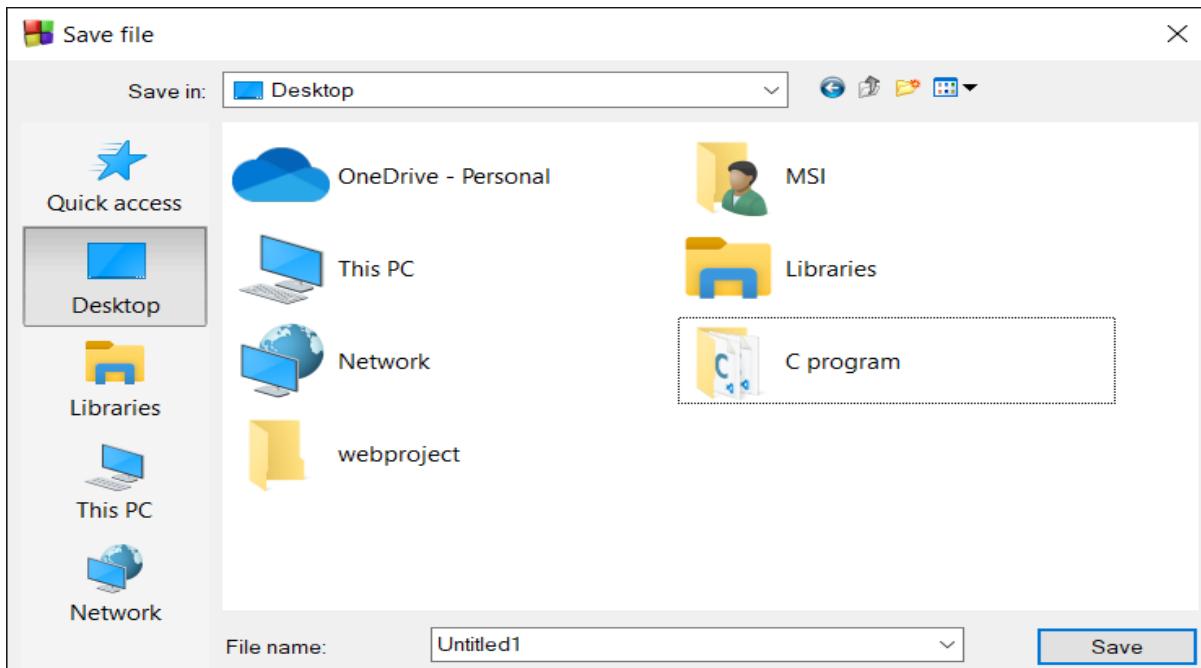


figure-8

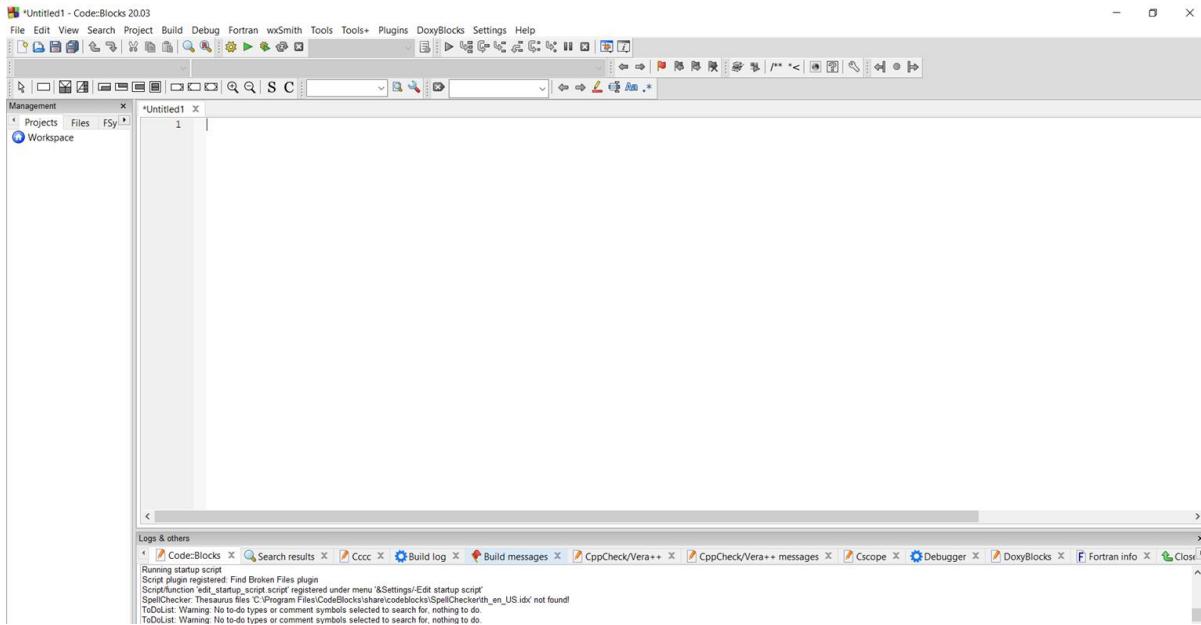


figure -9

C programm ඔත්ත්වා උද්ද:

```
#include<stdio.h>
```

```

int main(){

printf("Hello world");

return 0;
}

```

အထက်ပါအတိုင်း ရေးပါက **control+s** ကိန္ဒိယံး **program run**၏ **build & run**ဆိုတဲ့ **Icon** ကိန္ဒိယံးပါမယ်။

ဗုံး **figure -10**မှာပြထားပါတယ်။

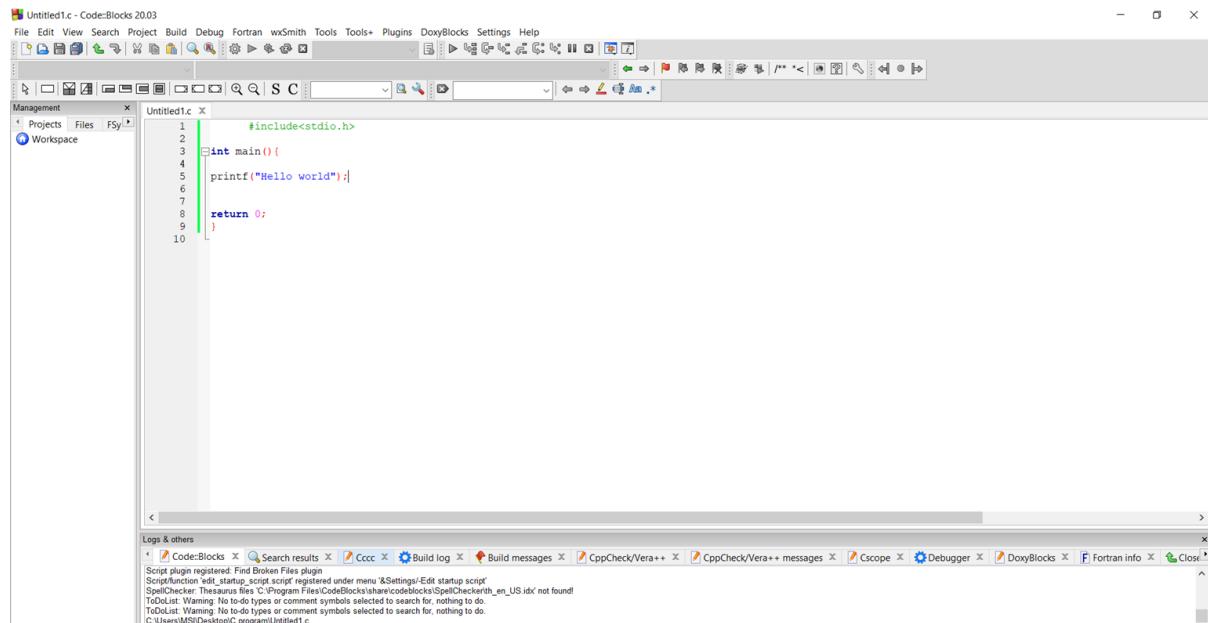


figure-10

output သည် **figure-11** အတိုင်းထွက်လာပါက ကျွန်တော်တို့က **C program** ကို ရေးသားလို့ရပါဖြစ်ပါတယ်။ ကျွန်တော်တို့ နောက်တစ်ခေါက် **lesson2** ဆိုတဲ့ **program**

တစ်ပုဒ်ထပ်ပါး ရေးသားကြည့်ကြပါမယ့် **File** ထဲကိုသွားပါး **empty file** ကိန္ဒိယံးဆောက်

ပါမယ်။ ပို့ရင် **save** ပို့**file** နာမည်ကို **Lesson 2**လို့ ပြောင်းပို့ **save** ပါမယ်။

ပို့ရင်တော့ **Figure-12**မှာ ပြထားတဲ့ အတိုင်း **program** ရေးသားပါမယ်။

The screenshot shows the Code::Blocks IDE interface. On the left, there is a code editor window containing the following C code:

```

1 #include<stdio.h>
2
3 int main()
4 {
5     printf("Hello World");
6
7     return 0;
8 }

```

To the right of the code editor is a terminal window titled "C:\Users\APerez\Documents\Verificador de nombres de archivo\HolaMundo.exe". The terminal displays the output of the program:

```

Hello World
Process returned 0 (0x0)   execution time : 0.216 s
Press any key to continue.

```

Below the terminal window, there is a tab bar with several tabs: "Logs & others", "Code::Blocks", "Search results", "Cccc", "Build log", "Build messages", "CppCheck", "CppCheck messages", "Cscope", and "Doxygen". The "Build messages" tab is currently active, showing the build log:

```

File Line Message
=====
==== Build file: "no target" in "no project" (compiler: unknown) ====
==== Build finished: 0 error(s), 0 warning(s) (0 minute(s), 0 second(s)) ====

```

Figure-11

The screenshot shows the Code::Blocks IDE interface. On the left, there is a code editor window containing the following C code:

```

1 #include<stdio.h>
2
3 int main()
4 {
5     printf("Hello lesson2");
6
7     return 0;
8 }

```

Below the code editor, there is a terminal window showing the output of the program:

```

Hello lesson2

```

At the bottom of the screen, there is a tab bar with several tabs: "Logs & others", "Code::Blocks", "Search results", "Cccc", "Build log", "Build messages", "CppCheck/Vera++", "CppCheck/Vera++ messages", "Cscope", "Debugger", "DoxygenBlocks", "Fortran info", and "Close". The "Build messages" tab is currently active, showing the build log:

```

Logs & others
Code::Blocks: Warning: No to-do types or comment symbols selected to search for, nothing to do.
C:\Users\MSH\Desktop\C program\Untitled1.c
Last compilation: C:\Users\MSH\Desktop\C program\Untitled1.c "C:\Users\MSH\Desktop\C program\Untitled1.exe"
NativeParser: DoFullParsing took: 0.504 seconds
NativeParser: CreateParser: Finish creating a new parser for project "NONE"
NativeParser: DoParseEnd: Project "NONE": parsing stage done!

```

Figure-12

Installation Visual Studio Code

ဒီသင်ခန်းစာမျကတော့ C program ရေးသားလို့ရမယ့် Visual Studio Code

အကြောင်းပြောပြောသွားမှာဖြစ်ပါတယ်။ VScode ကတော်ခြား Programming language

တွေလဲရေးသားလို့ရသလို C programming ကိုလဲ ကောင်းကောင်းရေးသားလို့ ရပါတယ့်။

သူ့ကို installation လုပ်ရတာ လွယ်ကူပေမယ့် C program ရေးသားဖို့လိုအပ်တဲ့ Extensions

တွေ install ရတာတော့ အနည်းငယ်ထွေးပါတယ်။ ဒါပေမယ့် စာရေးသူကတော့

VS code ထဲတွင် **C programming** ရေးသားလို့ရနိုင်ရန် လိုအပ်တဲ့ software / extensions

Installation လုပ်နည်း step by step ပြောသွားမှာ ဖြစ်ပါတယ့်။ ပထမဆုံးအနေနဲ့

Default browser ခဲ့ search bar မှာ **visual code download** လို့ရှိက်ရှာရပါမယ်။ ပုံ

Figure -13မှာပြထားပါတယ်။

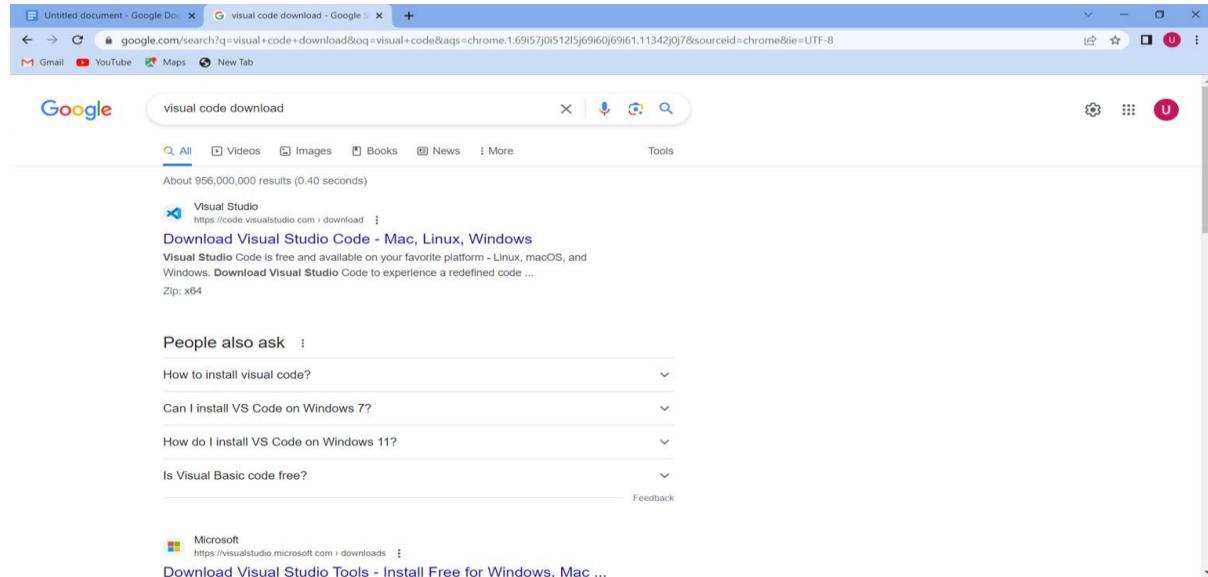


Figure -13

ထို့နောက် ပထမဆုံး link ထဲဝင်လိုက်ပါက **figure-14**တွင် ပြထားသည့် စာမျက်နှာသို့

ရောက်သွားပါလိမ့်မယ်။ ထို့နောက် မိမိတို့သုံးစွဲသည့် OS အလိုက် **VS code** ကို download

ရယူနိုင်ပါတယ်။ စာရေးသူအနေနဲ့ကတော့ windows ကိုအသုံးပြုပါး **VS code** ကို

Downloadသွားမှာဖြစ်ပါတယ်။

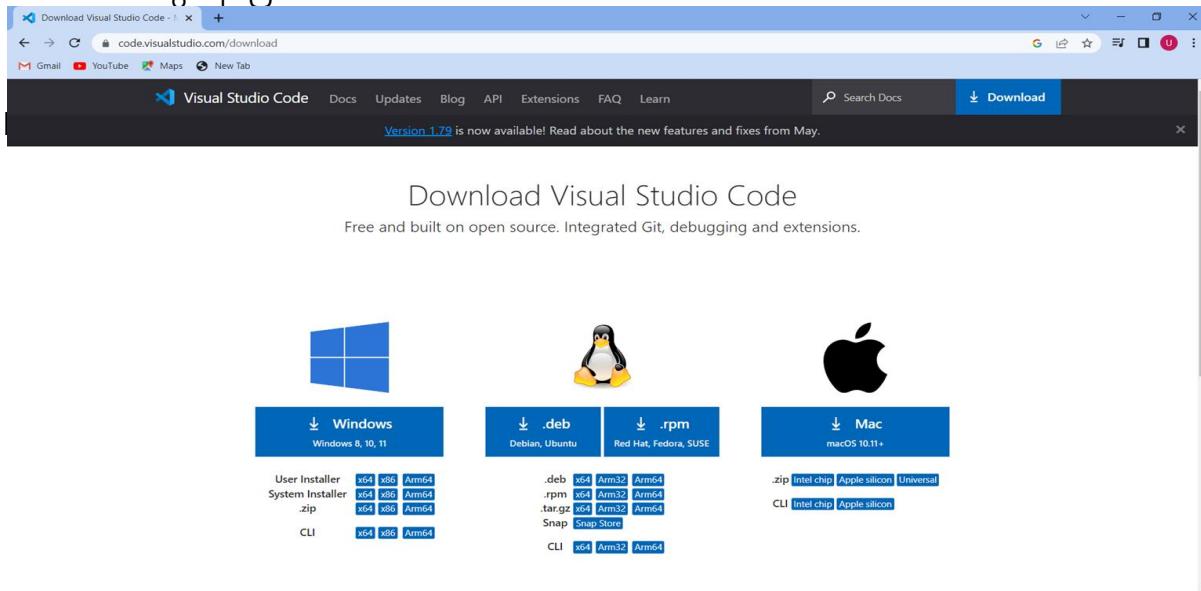
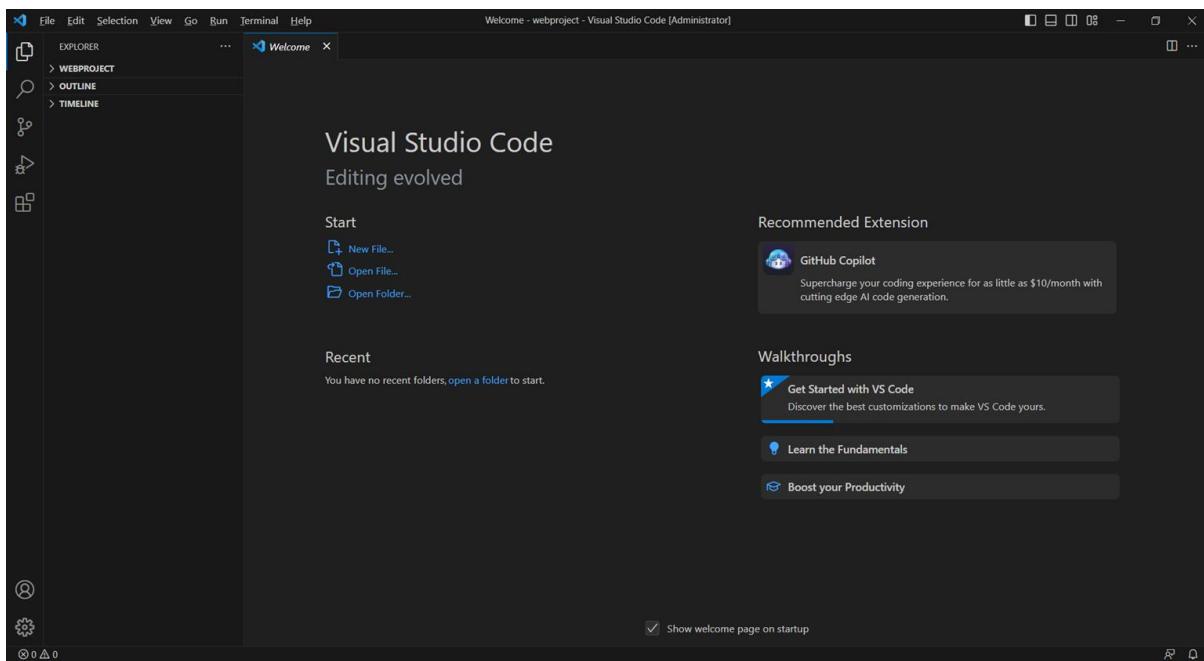
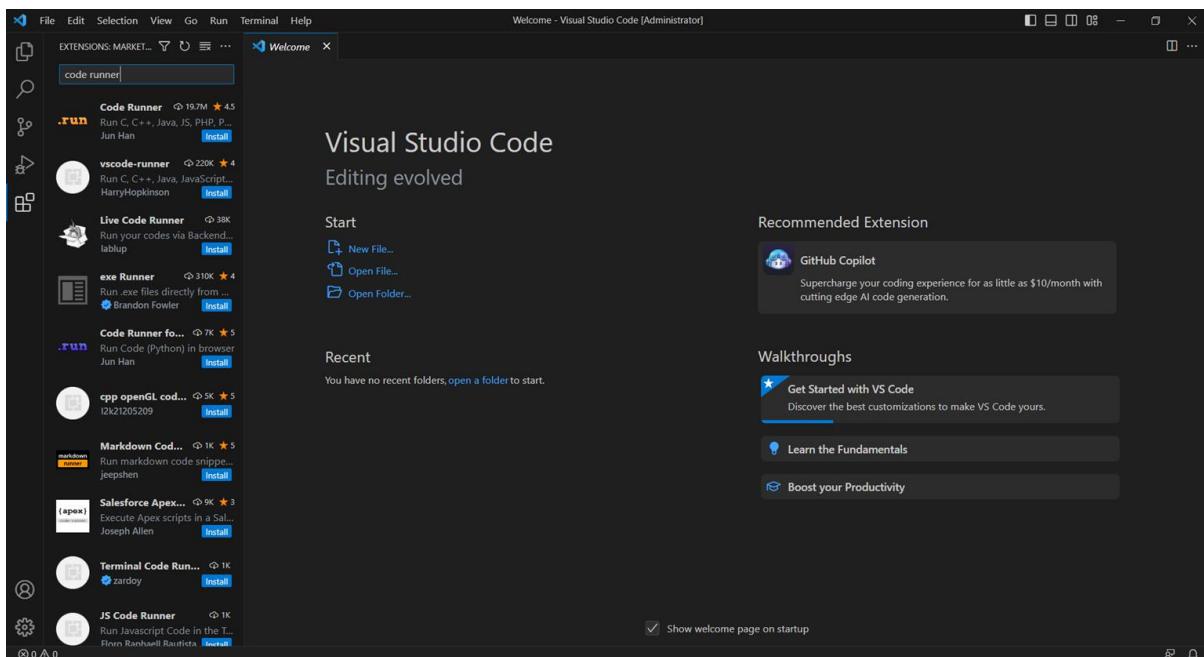


Figure-14

Windows ကိုနိုပ်ပီး **exe file**ကို **download**လုပ်လိုက်ပါ။ **Download**ပီးပါက ငှံ**exe file**ကို **click**နှစ်ချကနိုပ်ပီး **install** ရပါမယ်။ ထို့နောက် **I accept the agreement**ကို နိုပ်ပီး **Next**ကိုဆက်နိုပ်ပီး **install**ဆက်လုပ်ပါ။ **screen** ပေါ်မှာထားချွင်ပါက **create a Desktop icon**ကို **select**လုပ်ပါ။ **software**ကို ဖွင့်သောအခါ **figure-15** အတိုင်းပေါ်လာပါလိမ့်မယ်။

**Figure-15**

ထိုနောက် **ctrl+shift+X** ကိုနှိပ်ပိုး**Extensions**ထဲဝင်ရပါမယ်။သို့မဟုတ် ဘယ်ဘက်ခြမ်းမှာ ရှိတဲ့လေးထောင့်ကွက်ပုံစံလေးကို နိုးပို့လိုလဲရပါတယ်။ **figure-16**တွင်ပြထားသည့်အတိုင်း ပေါ်လာပါလိမ့်မယ်။

**Figure-16**

ထို့နောက် **search bar**မှာ **code runner**လိုဂိုက်ရှာရပါမယ်။ ထို့နောက် ငြင်း**extensions**ကို

Installလုပ်ရပါမယ်။

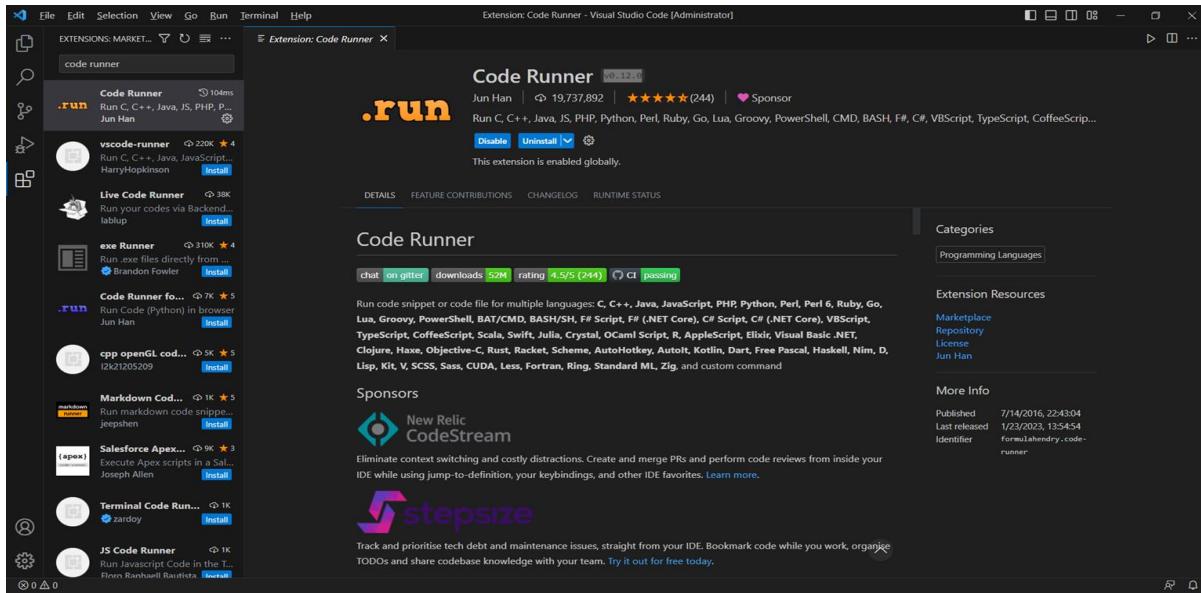


Figure-17

ထို့နောက် နောက်ထပ် **extensions** တစ်ခုဖြစ် **C C++**လို့ **search bar** မှာရိုက်ရှာရပါမယ်။

သူ့ရဲ့လက်ရှိ **version**က 1.15.4ဖြစ်နေမှာပါ။ **figure-18**တွင် ကြည့်ပါ။ ငြင်းကို **install**

လုပ်မှသာ **code** ကို **debugging and browsing**လုပ်နိုင်မှာဖြစ်ပါတယ်။ အထက်ပါ

အားလုံးပီးသွားပါက **ctrl+shift+E** ကိုနိုင်ပီးမူရင်းနေရာကိုပြန်သွားနိုင်ပါတယ့်

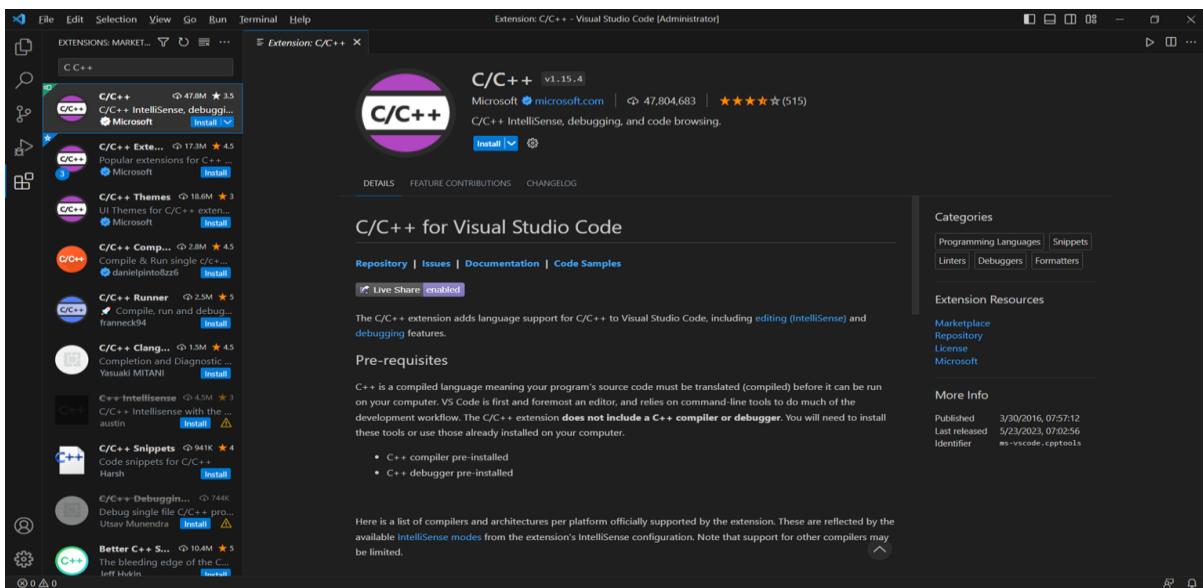


Figure-18

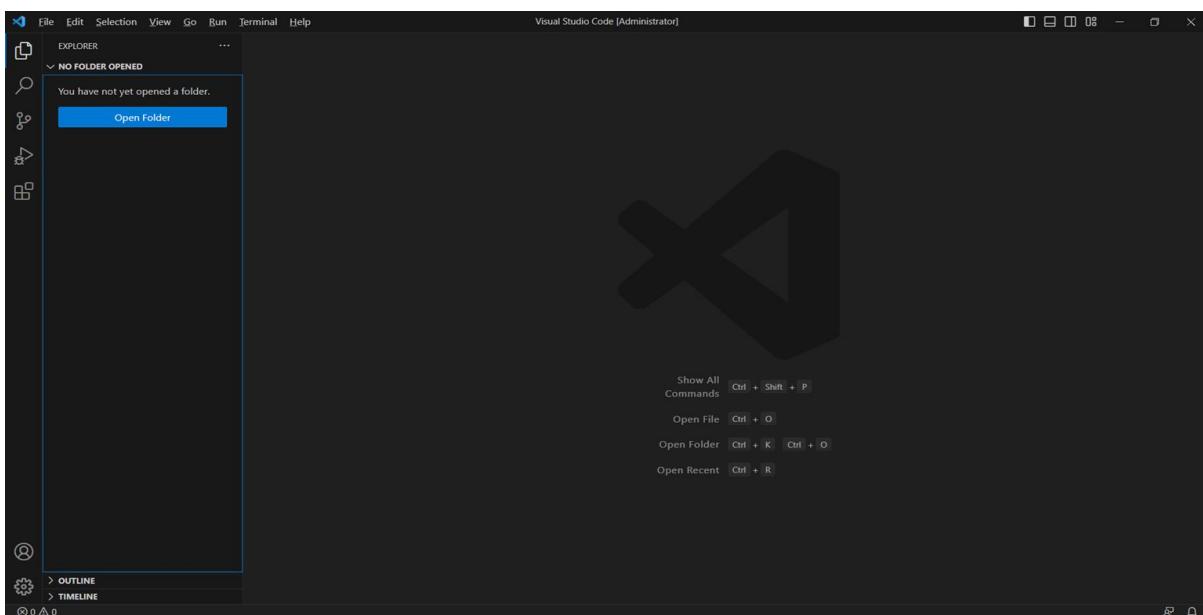
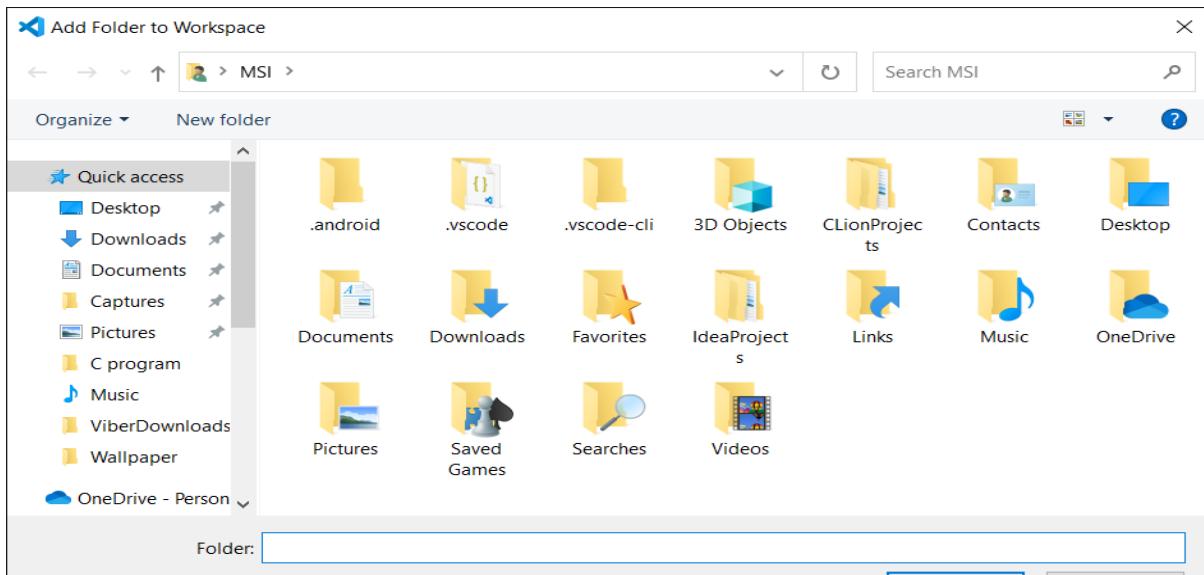


Figure-19

ထို့နောက် project ရေးသားရန်အတွက် folder ဆောက်ပေးလုပ်မယ်။ Folder ဆောက်ရန်

VS code ရဲ့ဘယ်ဘက်ထောင့်နားက file ကိုနှုပ်လိုက်ပါ။ ထို့နောက် Add folder to a

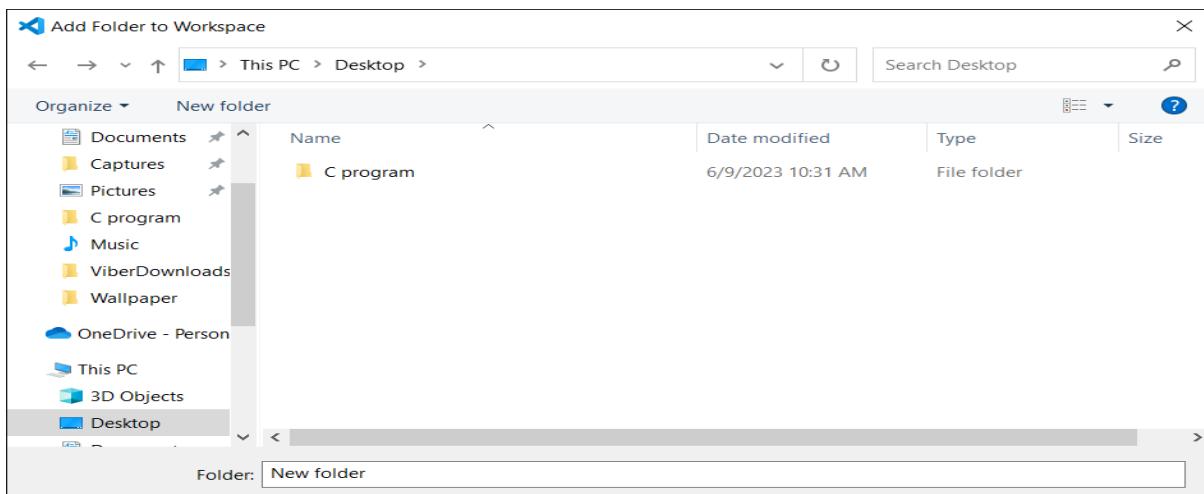
Workspace ကိုနှုပ်လိုက်ပါက figure-20 ထဲကအတိုင်းပေါ်လာပါလိမ့်။

**Figure-20**

თავეს:သავთებუ **Desktop** ထემა **folder** თან ეხება წერილი **folder** შედეგი **C program**

လისცე:თავ:პითაძე:თემა **ADD** კომისანი და **VS code** ထემა **C program** შედეგი

Folder თან ეცემა და დატენა შედეგი **Figure-22** თუ დანართი.

**Figure-21**

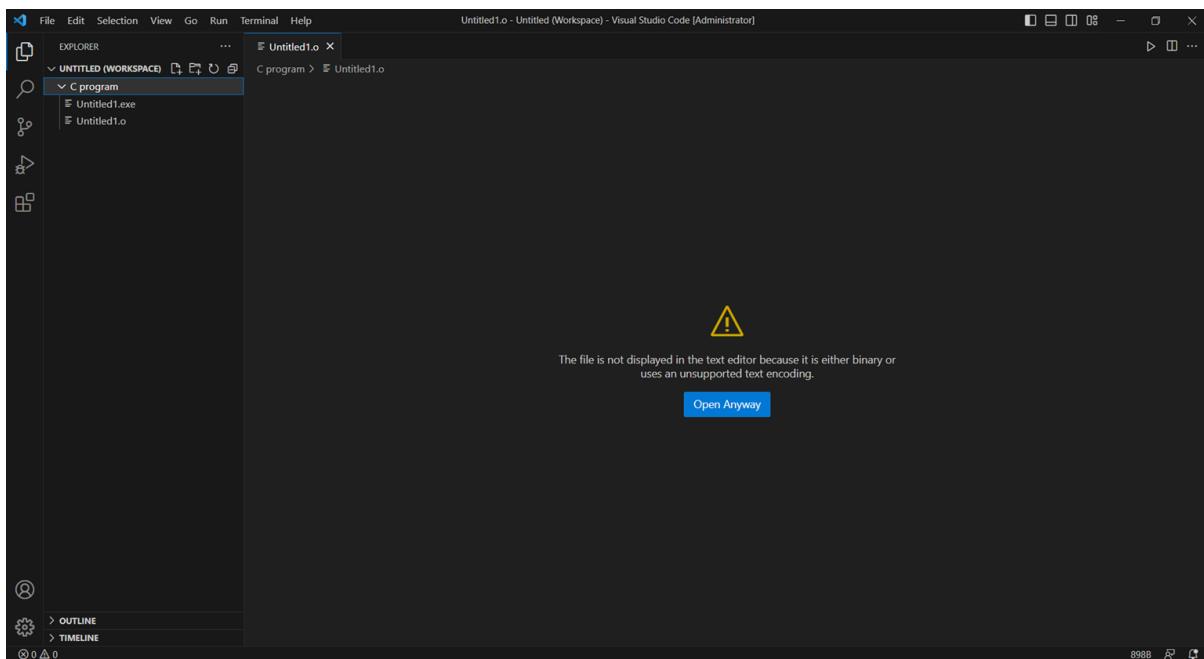


Figure-22

ထိုနောက် UNTITLED(WORKSPACED) ဘေးနားက Folderပုံစံလေးကိစ္စပြုပါ။ Project

တည်ဆောက်လိုက်ပါ။ project name ကို Hello.Cလိုပေးထားပါတယ့်။ ပြီးနောက်

Figure-23 တွင်ပြထားသည့်အတိုင်း HELLO WORLD program တစ်ပုဒ်ကို စမ်းရေးသား

ကျဉ်းပါက Error တက်ပါလိမ့်မယ့်။

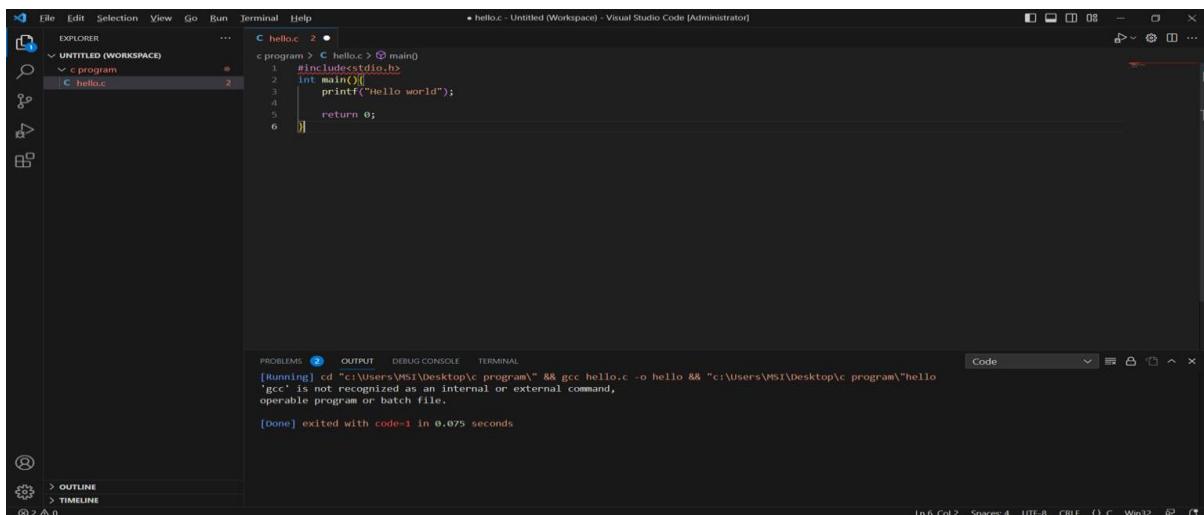


Figure-23

အဘယ်ကြောင့်ဆိုသော ကျွန်တော်တို့ **computer** ထဲတွင် **gcc compiler** မရှိသေးတာ

ကြောင့်ဖြစ်ပါတယ်။ထို့ကြောင့် **gcc compiler**ကို မိမိတို့ **computer** ထဲသို့ ထည့်သွင်းပေး

ရန် **browser** ရှိ**search bar**တွင် **mingw gcc compiler download**လို့ရှိက်ရှာပါး

Download ရယူနိုင်ပါတယ့်။ <https://sourceforge.net/projects/mingw-get/> linkမှလ

တိုက်ရှိက် **Download** ရယူနိုင်ပါတယ်။

Download ပီးပါက (**figure-24**) file ကို**click** နှစ်ချက်ထောက်ပီး**Install** လုပ်လိုက်ပါ။

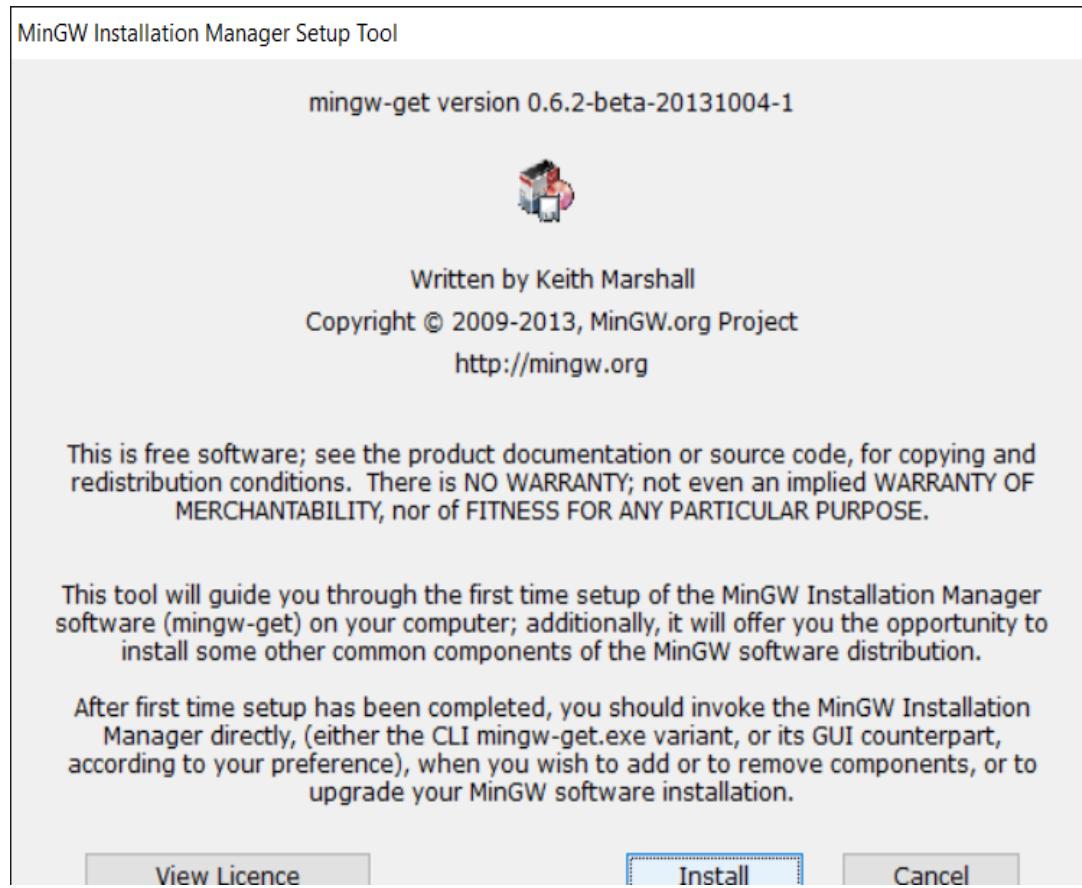
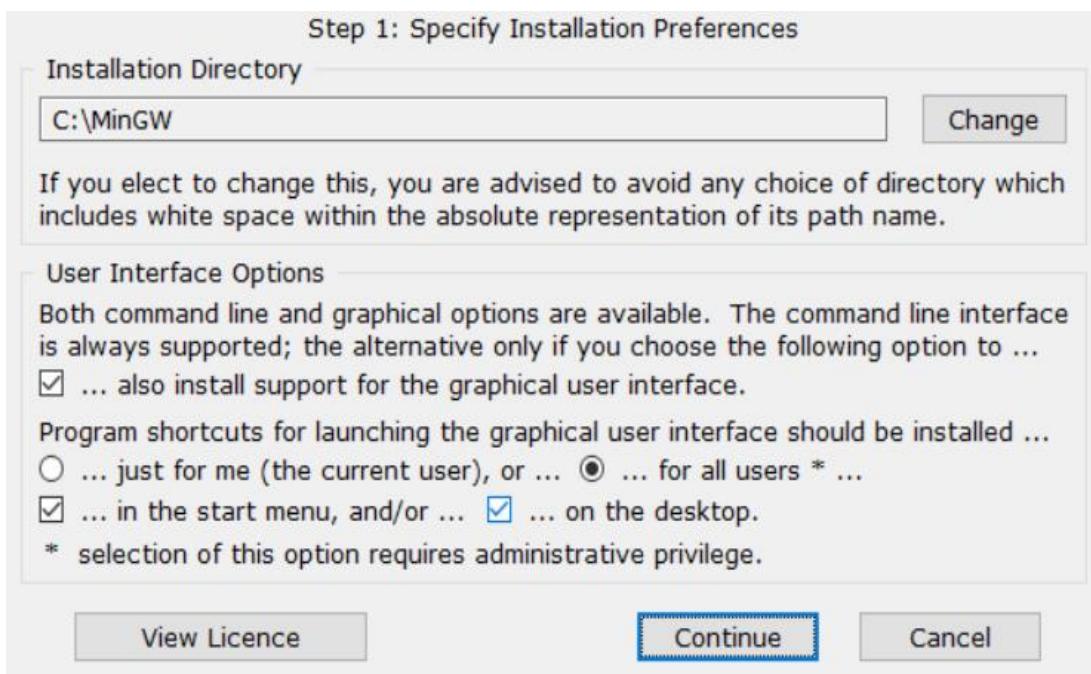
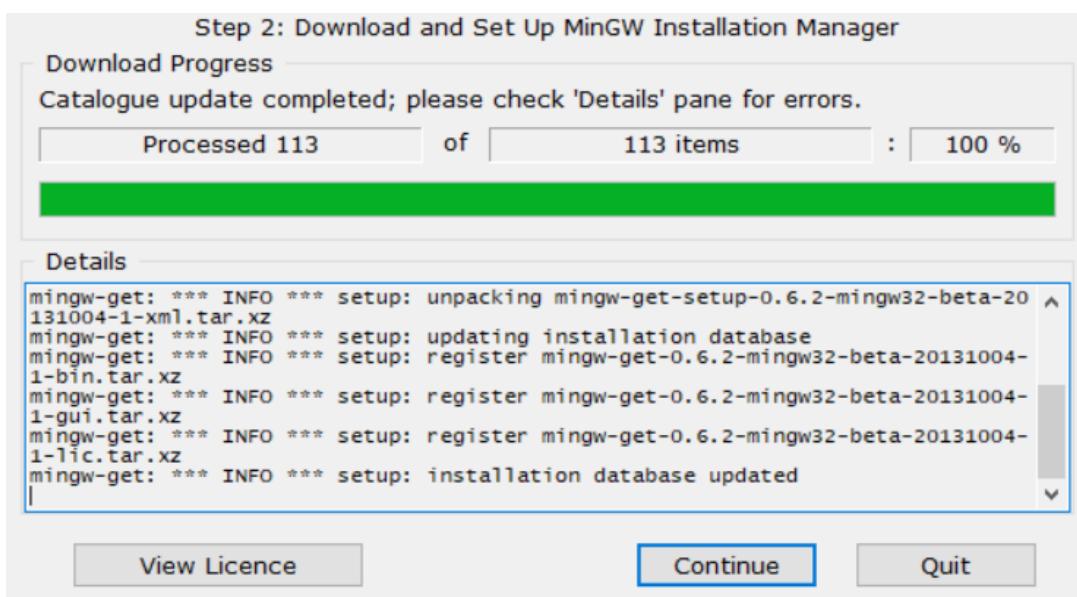


Figure-24

**Figure-25****Figure-26**

Installation ലൂട്ടാ 100% (Figure-26) പ്രൈവറ്റ്‌പിന് figure-27തട്ടിലെയും

അതീൻ:പേരിലാപിലിമുംഡ്യു!!

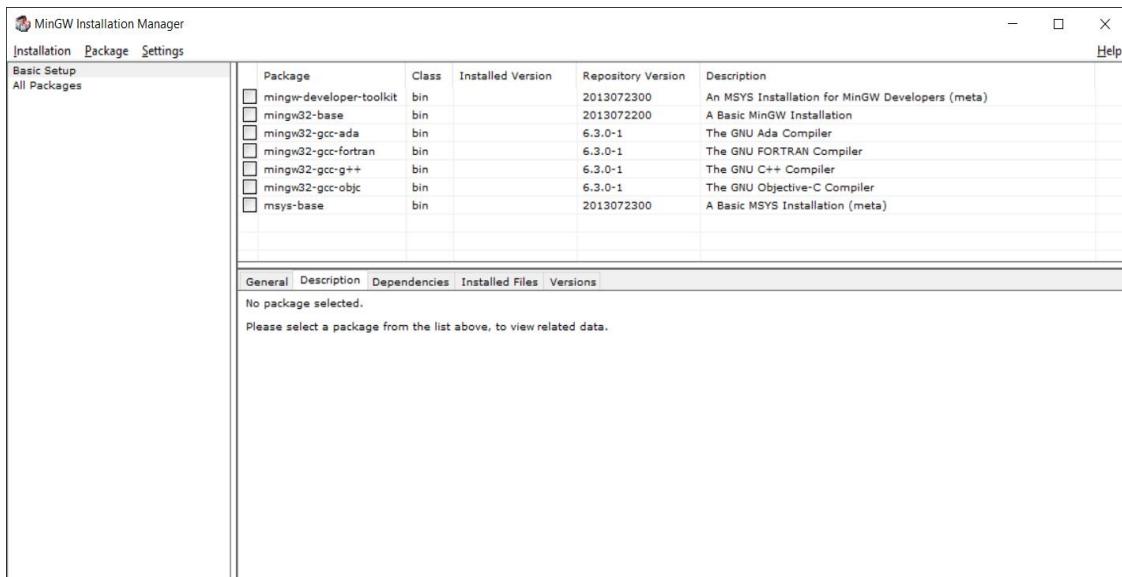
**Figure-27**

Figure-27တွင် **Package**ထဲက ဒုတိယမြောက်(**mingw32-base**)နဲ့ ငါးခုမြောက်

(**mingw32-gcc-g++**) ကို**select** မှတ်ပီး ဘယ်ဘက်ထောင့်က **installation** ကို

ထဲဝင်ပီး **Apply Changes**ကိုနိုပ်လိုက်ပါက **figure-28**တွင်ပြထားသည့်အတိုင်း

ပေါ်လာပါလိမ့်မယူ။ ညာဘက်နားက **Apply** ကိုနိုပ်ပီး လိုအပ်သည့်**package**များကို

Download လုပ်ပါလိမ့်မယ်။ ပီးပါက **desktop screen**တွင် **mingw icon**(**figure-28**)

ပေါ်လာပါလိမ့်မယူ။ ထို့နောက် **setting** ထဲက **search bar** တွင် **edit the systems**

environment(variable) ရိုက်ရှာပီး ထို့ထဲသို့ ဝင်လိုက်ပါ။ **figure-29**အတိုင်း ပေါ်လာ

ပါလိမ့်မယ်။



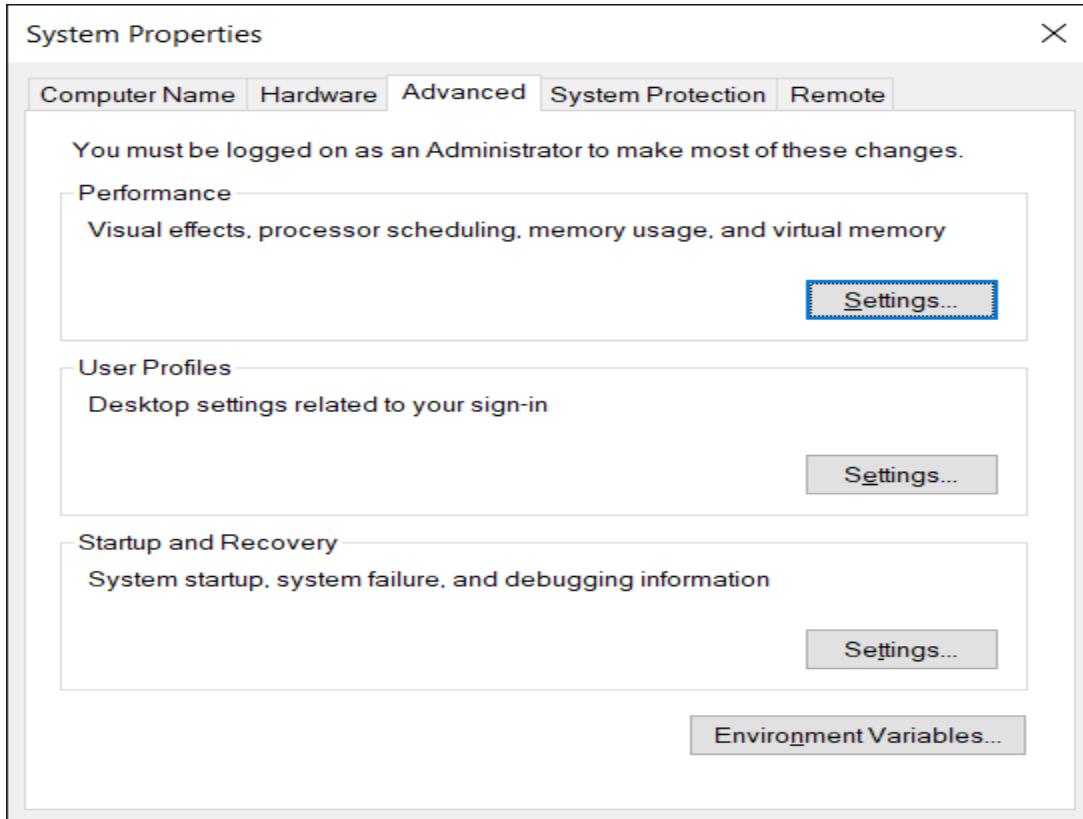
Figure-28**Figure-29**

Figure-29 ထဲက **Environment Variables** ကိုရှိပ်ပါး **figure-30**တွင် ပြထားသောနေရာ

သို့ရောက်သွားပါလိမ့်မယ်။ ထိုတဲ့သို့ရောက်သွားပါက **systems variable** ထဲက **path** ကို
တစ်ချက်ထောက်ကာ **edit** ကိုရှိပ်လိုက်ပါက **figure-31** ထဲက နေရာသို့ရောက်
သွားပါလိမ့်မယ့်**path** လမ်းကြောင်းယူဖို့အတွက် **This PC** ထဲက **local Disk(C)** ထဲသွားပါ
မယ်။ **Local Disk(C) (figure-32)**ထဲမှာ **Mingw** ဆိုတဲ့ **folder**ထဲဝင်ပါ။ ထို့နောက် **bin** ထဲသို့
ဝင်ပါ။ ထို့နောက် **Figure-33** တွင် ပြထားသည့်အတိုင်း **path** လမ်းကြောင်းကို **copy** ယူပါး
Figure-31နေရာသို့ပြန်သွားကာ **new** ကိုရှိပ်ကာ ခုနက **copy** ယူလာခဲ့တာကို **paste** လုပ်
လိုက်ပါ။ ပီးရင် အကုန်လုံးကို **OK** နို့ပ်ကာပြန်ထွက်လာလိုဂုပ္ပါ။ အထက်ပါ အချက်အားလုံး

လုပ်ပီးသွားရင်တော့ **vs code** မှာ **C program** ရေးသားလိုပါပါ။ သတိထားရမှာတစ်ချက်

က **program**တစ်ပုဒ်ရေးပီးတိုင်း **ctrl+s**နိုင်ဖို့ မမေ့ပါနဲ့။

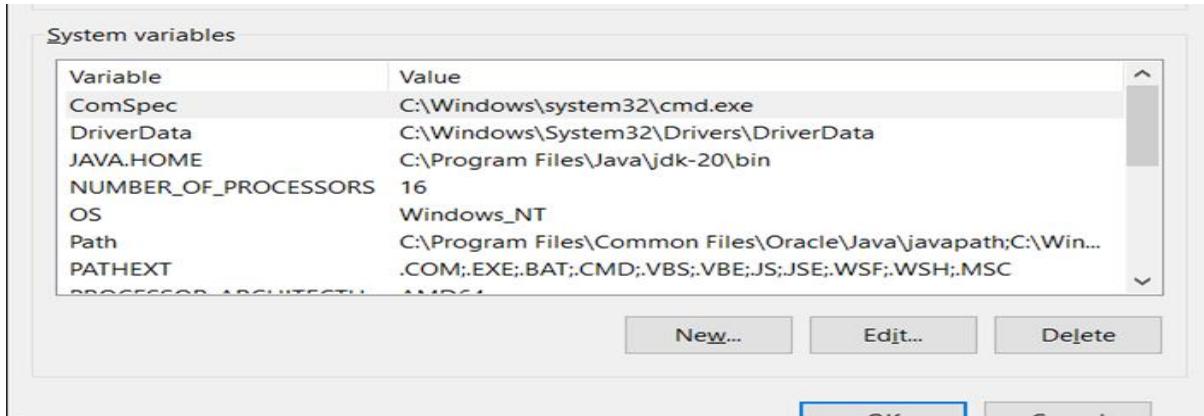


Figure-30

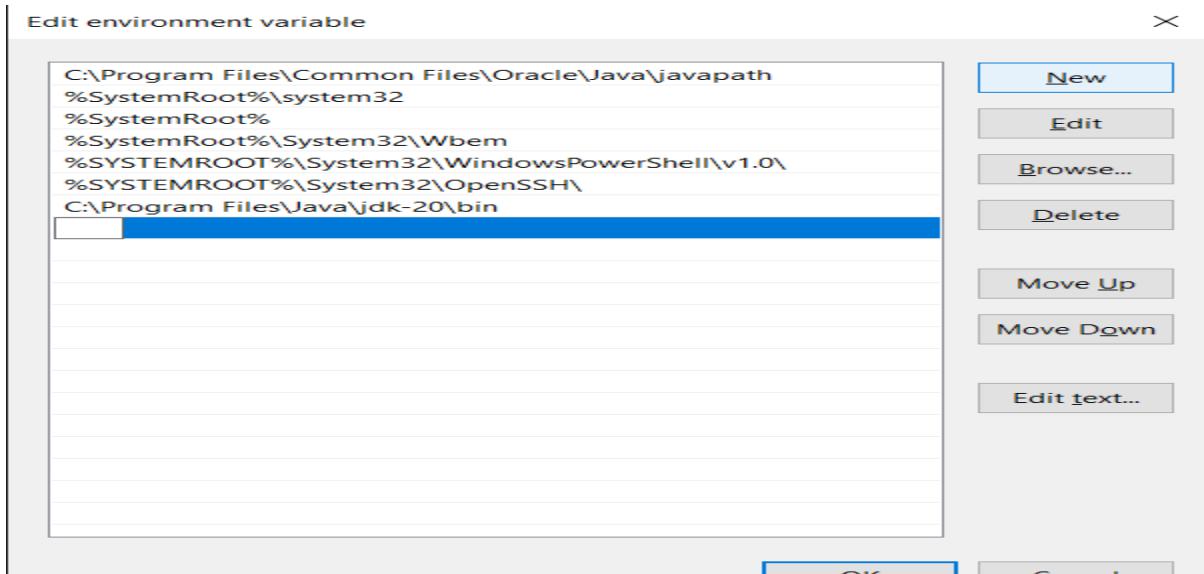


Figure-31

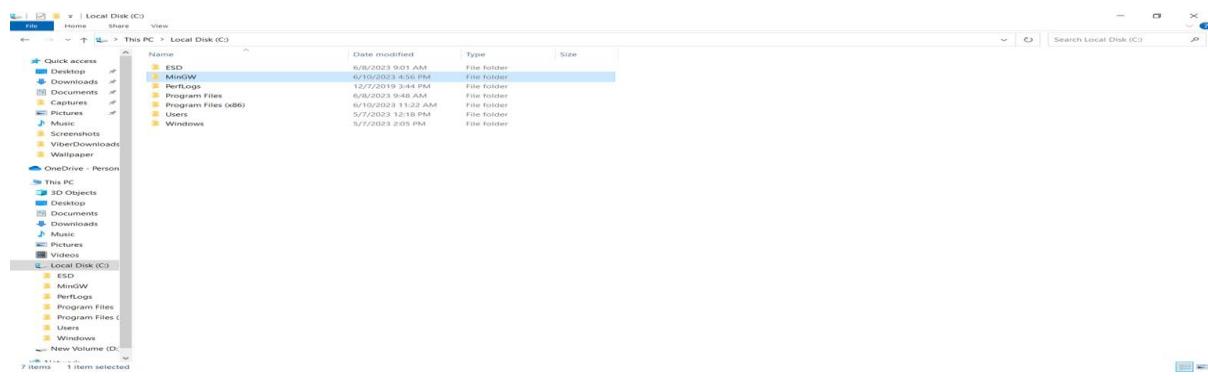


Figure-32

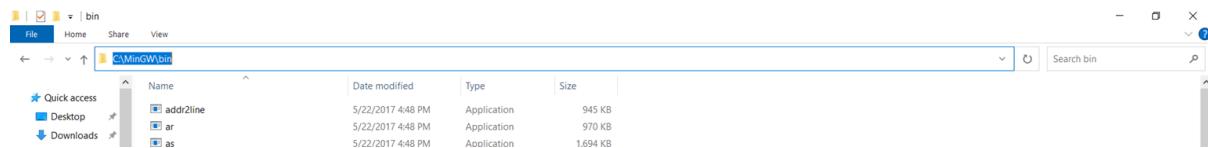


Figure-33

Program ተወስኝኩን እወጪ፡ይሸቻ፡

```
#include <stdio.h>

int main() {
    printf("Hello, World!\n");
    return 0;
}
```

እወጪበት የprogram ተወስኝኩን እወጪ፡ይሸቻ፡ የprogram ተጥቅምች እና የright click መመሪያ፡

run code የኩቻውን የrun የሚሰጥበትን፡

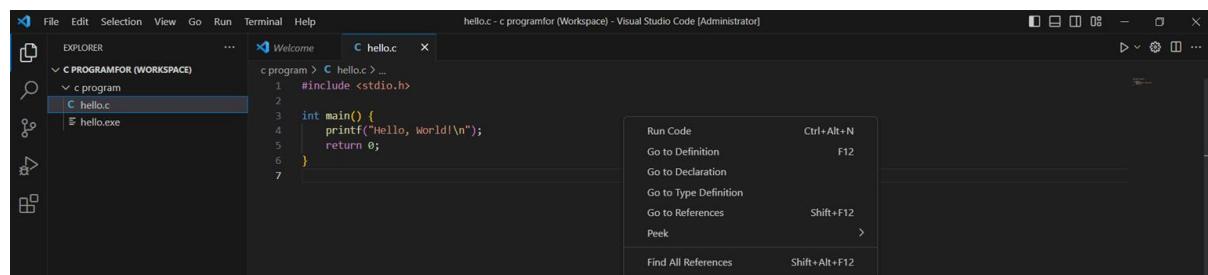


Figure-34

အောက်ပါအတိုင်း **output** ကို ရပါလိမ့်မယူ။

```
#include <stdio.h>
int main() {
    printf("Hello, world!\n");
    return 0;
}
```

Figure-35

Terminal မှ **run** ရန်အတွက် **set up** လုပ်ပီးပါက **C program** ရေးသားရန် **installation**

အပြည့်အစုံ လုပ်ပီးမှာဖြစ်ပါတယူ။**set up** လုပ်ရန် အောက်ပါအချက်မှားအတိုင်း တစ်ဆင့်

ချင်းလုပ်ဆောင်ပေးပါ။ **file** ထဲသို့သွားပါ။ ထို့နောက် **preferences** ထဲက **setting** ထဲ ဝင်

ရပါမယ်။

file->preferences->setting

ထို့နောက် **figure-36** ထဲကအတိုင်း **extensions** ထဲက **Run code configuration** ထဲဝင်ကာ **Run**

in Terminal ကို အမှန်ခြစ်ပေးရပါမယ်။

extensions->Run Code Configurations->Run In Terminal

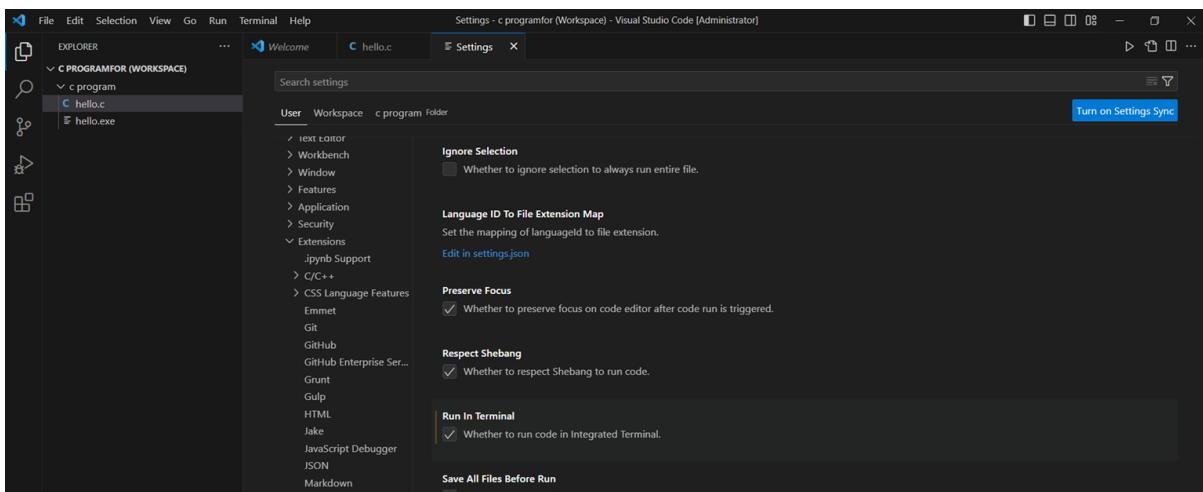


figure-36

Project Structure

ဒီသင်ခန်းစာကန္တစိုးတွေ **C programming** ကို သုံးပီးတွေ **Project**လေးတွေရေးသွား

မှာဖြစ်ပါတယ်။ နောက်ပိုင်း သင်ခန်းစာတွေကို ရေးသားနိုင်ဖို့ဆိုရင် **C** ရဲ့**basic concept**

တွေအကြောင်းကောင်းကောင်းသိထားဖို့လိုပါတယ်။

If else

Switch case break

For loop

While loop

Do while loop

Function

Array

Char Array

Int Array တွေ အကြောင်းအရင်သိဖို့ လိုပါမယ်။

Pointer

Structure

Dynamic memory allocation

File input/output တွေအကြောင်း အရင်သိမ့်လိုပါမယ်။ အထက်ပါအကြောင်းအရာများ

အကြောင်း မသိသေးပါက **National Cyber City page**မှာလာရောက်စုံစမ်းပီး

သင်တန်းတက်ရောက်နိုင်ပါတယ်။ <https://web.facebook.com/nationalcybercity>

အထက်ပါ **basic concepts**တွေအကြောင်းသိထားမှသာ အခုသင်ကြား

မယ့် **Lessons**တွေကိုနားလည်နိုင်မှာဘဲဖြစ်ပါတယ့်။

ဒီသင်ခန်းစာများတော့ ကျွန်တော်တို့ ရေးမယ့် **project structure**အကြောင်း ပြောပြပေးသွား

မှာဖြစ်ပါတယ်။ **online bank project**လို့ နာမည်ပေးထားပါတယ်။ ဒီသင်ခန်းစာများတော့

printf/scanf /file input outputကဲရင်ဘယ် **function call** ကိုမှ အသုံးမပြုဘဲနဲ့

ကျွန်တော်တို့ဘာသာ ကျွန်တော်တို့ ရေးသွားမှာဖြစ်ပါတယ့်။

Bank project ရေးဖို့ရင် **bank**တစ်ခု ရဲ့ လုပ်ငန်းဆောင်တာတွေ တစ်ချို့အကြောင်း အရင်

အကြမ်းဖျင်းသိမ့် လိုပါတယ်။ ဥပမာ **bank application** ထဲဝင်တဲ့အခါ **account**

ရှိထားဖို့လိုပါတယ်။ အကောင့်ရှိဖို့လို **login** ဝင်ရပါမယ်။ **account**မရှိပါက **register**လုပ်

ရပါမယ်။ **Register** လုပ်တဲ့အခါ **user** ရဲ့ **information** တွေထည့်တဲ့အခါ **gmail** ကို အသုံး

ပြုတဲ့အခါ **gmail format** နဲ့ကိုက်ညီမှု ရှိမရှိ စစ်ဆေးတာတွေ။ **Password** ထည့်သောအခါ

Strong password ဖြစ်အောင် **special character**တွေ သို့မဟုတ် ကဏ္ဍား စ လုံးအထိ

ထည့်ခိုင်းတဲ့ **features** တွေကို ရေးသားကြမှာ ဖြစ်ပါတယ်။ **user** ရဲ့အပိုင်းဘက်မှာ အ

ချက်အလက် တွေကို စစ်ဆေးပါးအတည်ပြုပေးမယ့် **Bank Admin** ဘက်ခြေမ်းကိုလဲ ရေး

သား သွားမှာ ဖြစ်ပါတယ့်။ **Bank Admin** ဘက်ခြေမ်းမှာကတော့ **user account** တွေ **suspend**

လုပ်တာတွေ၊ **account level** တင်ပေးတာတွေ၊ **account** ပိတ်တဲ့ **feature**တွေ၊

အစ ရှိတဲ့ အကြောင်းအရာ တွေ ကို ရေးသားသွားမှာ ဖြစ်ပါတယ်။ တခြားလိုအပ်တဲ့ အကြောင်းအရာများ ကိုလဲ **project** ရေးသားတရ်င်းနဲ့ ထပ်ပေါင်းထည့်ပါး ရေးသားသွားမှာ ဖြစ်ပါတယ့်။

ကျွန်တော်တို့ **project structure** အကြောင်းပြောပါးတဲ့အခါ **Project flow** တည်ဆောက်သွားမှာဖြစ်ပါတယ်။ အဲဒါမှသာ **Project** ရေးသားတဲ့အခါ အစီအစဉ်ကျ ရေးသားနိုင်မှာ ဖြစ်ပါတယ်။ **user** တစ်ယောက်မှာ ရှိသင့်ရှိထိုက်တဲ့ အချက်အလက်တွေ ဖြစ်တဲ့ **user id**, **name , nrc** နံပါတ်, **phone number, encryption_key, recovery_key,,account status(vip user or normal user),account_level,minimun_opening_deposit, current_amount, password, Email, pasernal account or business account, monthly income, loan amount(user)** က **bank**ကနေ ငွေချေးထားရင် ချေးထားတဲ့ပမာဏ),**loanrate**(အတိုးနှုန်း),**Address**အထက်ပါအကြောင်းအရာတွေကို ထည့်သွင်းရေးသားသွားမှာဖြစ်ပါတယ်။**program**ရေးသားသည့်အခါတွင်လဲအထက်ပါ အကြောင်းအရာများကို တစ်ချက်ချင်းပြပေးထားတာမို့လေ့လာသူတို့ အနေဖြင့် နားမလည်မှာ ကိုစိတ်ပူစရာ မလိုပါ။

- id**
- name**
- nrc**
- email**
- password**
- phoneNumber**
- encryption_key**
- recovery_key**

```

account_status
account_level
minimum_opening_deposit
currency
current_amount
loanStatus
monthly_income
loan_amount
loan_rate
address
TransRC

```

Figure-37

Program flow အကြောင်းပြောပီးသွားပါမည့် **program** စပီးရေးပါတွေ့မယ်။ ဒီနေရာမှာ စာရေးသူက **main file** နဲ့ **header file** ဆိုပီး ခွဲရေးပါမယ့်။ **header file**ထဲမှာဘဲ တကယ်အလုပ်လုပ်မယ့်၊ **code**တွေကို ရေးသွားမှာ ဖြစ်ပါတယ့်။ **header file** သုံးခြင်းဖြင့် တခြားနေရာကနေပါ **header file** ကို ခေါ်သုံးလို့ရပါတယ့်။ **header file**တည်ဆောက်ရန် **WORKSPACE**ဘေးနားက **new file**ကို တစ်ချက်နှင့်ကွဲ**onlinebank.h** လို့ နာမည်ပေးလိုက်ရင်ရပါပါ။ အဲဒီတွေ့ ကျွန်တော်တို့မှာ **main.c** နဲ့ **header.h** ဆိုတဲ့ **file** နှစ်ရှိပါမယ်။

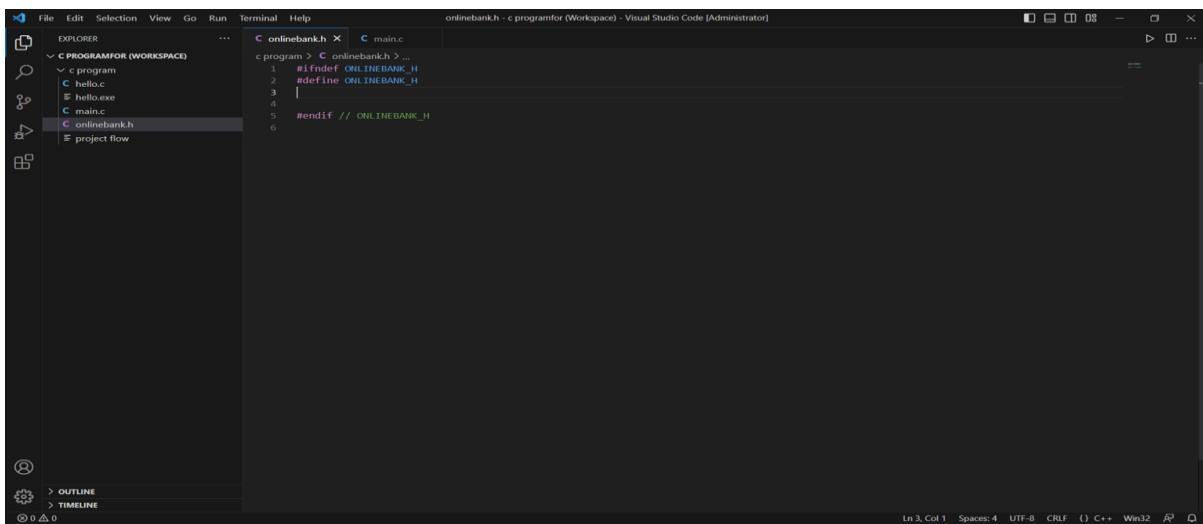


Figure-38

ဒီနေရာမှာ ကျွန်တော်တိသိထားရမှာတစ်ခုက **onlinebank.h**ထဲမှာရှိတဲ့စာသားတွေကို ကိုယ့်ပါ။

ဘာသာကိုယွည့်ပေးရမှာဖြစ်ပါတယ်။

```

#ifndef ONLINEBANK_H
#define ONLINEBANK_H

// Contents of your onlinebank.h header file go here

#endif // ONLINEBANK_H

```

Header file ဖန်တီးပါးရင်တော့ ကျွန်တော်တိ **main file** ထဲမှာ **code** စရေးပါမယ့်။

C program တစ်ပုဒ်မှာဆို **main function** သည် မပါမဖြစ်ပါရပါမယ်။အောက်တွင်

ကြည့်ရန်။

```

#include <stdio.h>
int main(){

```

```

    return 0;
}

```

ထိုနောက် အလုပ် လုပ်မယ့် **code**တွေကို **header file**ထဲသွားရေးပါမယ်။ ပထမဆုံးအနေနဲ့

Void main_menu() တစ်ခုရေးထားပါမယ့်။ ထိုနေရာမှာ **header file**ရဲ့အပေါ်မှာ

#include<stdio.h>, **#include<stdlib.h>** ကိုရေးပေးရပါမယ့်။ ဒါမှာသာ **input/output**

functionကိုသုံးလို့ ရမှာ ဖြစ်ပါတယ်။ ဒါဆိုရင်တော့ **Printf /scanf**သုံးလို့ရပါပြီ။

Main_menu functionထဲမှာ **figure-39**ထဲက အတိုင်း **Login/register** အတွက်စာသား

အနည်းငယ် ရေးပါမယ်။

```

File Edit Selection View Go Run Terminal Help
onlinebank - c programfor (Workspace) - Visual Studio Code [Administrator]
EXPLORER C onlinebank x C main.c
C program > C onlinebank > main.c
1 #ifndef ONLINEBANK_H
2 #define ONLINEBANK_H
3 #include<stdio.h>
4 #include<stdlib.h>
5 void main_menu(){
6     int option;
7     printf("Welcome to our bank\n");
8     printf("Press 1 to login\nPress 2 to Register\nPress 3 to Exit\n");
9     scanf("%d",&option);
10    if(option==1){
11        printf("This is login\n");
12    }else if(option==2){
13        printf("This is Register\n");
14    }else if(option==3){
15        exit (1);
16    }else{
17        main_menu();
18    }
19 }
20
21 #endif // ONLINEBANK_H

```

figure-39

ထိုနောက် **main.c** ထဲမှာ **main-menu function** ကို **figure-40**ထဲကအတိုင်း ခေါ်ပါမယ့်။

main.c ထဲမှာ **header file name**ကို ခေါ်ပေးရပါမယ့်။

```
#include "onlinebank.h"
```

```
main.c - C program - Visual Studio Code [Administrator]
File Edit Selection View Go Run Terminal Help
C PROGRAM
  C main.c > C main.c > C onlinebank.h
  main.c
  main.exe
  onlinebank.exe
  onlinebank.h
  main.c
  main.c > main.c
  1 #include <stdio.h>
  2 #include "onlinebank.h"
  3
  4
  5 int main()
  6 {
  7     main_menu();
  8 }
```

Line 6, Col 17 Spaces: 4 UTF-8 CRLF { } C Win32 /P

Figure-40

ထိုနောက် program ကို run ကြည့်ပါက အောက်ပါအတိုင်းပေါ်လာပါလိမ့်မယ့်။

Welcome to our bank

Press 1 to login

Press 2 to Register

Press 3 to Exit

1

This is login

1 ကိုနှိပ်လိုက်ပါက **This is login**လိုပေါ်လာမှာဖြစ်ပါတယ့်။ ၃ကိုနှိပ်လိုက်ပါက **exit-1**

နဲ့ **Program** ရပ်သွားမှာဖြစ်ပီး တစ်ခြား **Key**များကို နှိပ်ပါက **function** ကို ပြန်စမှာ

ဖြစ်ပါတယ်။

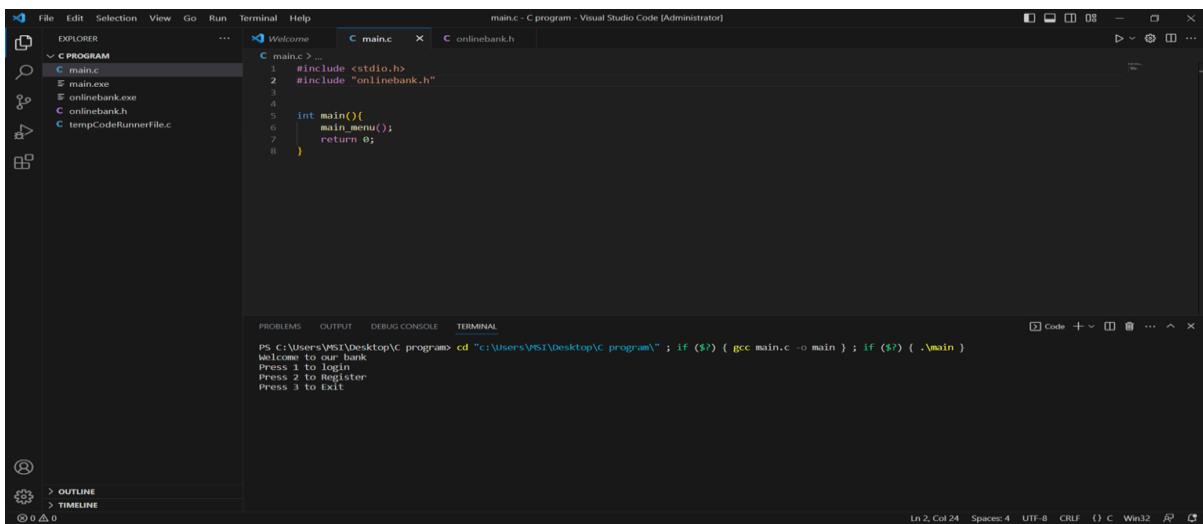


figure-41

ဒီနေရာမှာ **option input** ကို **Int** နဲ့ မတောင်းတွေ့ဘဲ **char** နဲ့ တောင်းပါမယ်။ ဒါကြောင့် **code**ကို

အနည်းငယ်ပြုပါမယ့်။ **int option** နေရာမှာ **char input[2]** လိုကြညာထား

လိုက်ပါမယ့်။ ထိုနောက် **int check_input()** ကို **parameter (char array)**ဖြတ်ပီးကြညာ

ပါမယ်။ **int** နဲ့ရေးထားတာမို့ **value**တစ်ခုကို**return** ပြန်ပေးမှာဖြစ်ပါတယ်။

```

// ascii value 0=48 & 9=57
int check_input(char input[2]){
    if(input[0]>=49 && input[0]<=57){
        return input[0];
    }else{
        return-1;
    }
}

```

ကျွန်ုတ်တိုက ဒီနေရာမှာ **number**တွေရဲ့ တန်ဖိုးကို **ascii value**ကို ကိုင်ပီး ဆံးဖြတ်ပါ

မယ့်။ **ascii table** မှာ **0 to 9** က **48 to 57** အထိရှိပါတယ်။ ကျွန်ုတ်တို့ထည့်လိုက်တဲ့

Input[0]ဟာ **1**ကနေ**9** ကြားရှိပါက **input[0]** ကျွန်ုတ်တို့ထည့်လိုက်တဲ့ တန်ဖိုးကိုဘဲ

return ပြန်ပေးမှာဖြစ်ပါတယ့်။ ထို့နောက် **function** ခေါ်သံဃားလို့ရစေရန် **main_menu**

ရဲ့အပေါ်ဘာက်မှာ **function declaration** လုပ်ပေးရပါမယ့်။ ထို့နောက် ပြန်လာတဲ့

return ကို **int variable** ထဲသိမ်းပါမယ်။ **line** နံပါတ် **30** က **input[1]=='\0'**

သည် **array** တွေရဲ့ဆုံးတဲ့နေရာသည် **\0** လို့ကြညာထားခြင်းဖြစ်သည်။

(figure-42)

```
#ifndef ONLINEBANK_H
#define ONLINEBANK_H
#include<stdio.h>
#include<stdlib.h>

// function declartion
int check_input(char input[2]);

void main_menu(){
    char input[2];
    printf("Welcome to our bank\n");
    printf("Press 1 to login\nPress 2 to Register\nPress 3 to Exit\n");
    scanf(" %[^\n]",&input);
    int option= check_input(input);
    if(option==49){
        printf("This is login\n");
    }else if(option==50){
        printf("This is Register\n");
    }else if(option==51){
        printf("exit");
        exit (2);
    }
}
```

```

    }else{
        main_menu();
    }

}

// ascii value 0=48 & 9=57

int check_input(char input[2]){
    if(input[0]>=49 && input[0]<=57 && input[1]=='\0'){
        return input[0];
    }else{
        return -1;
    }
}

#endif // ONLINEBANK_H

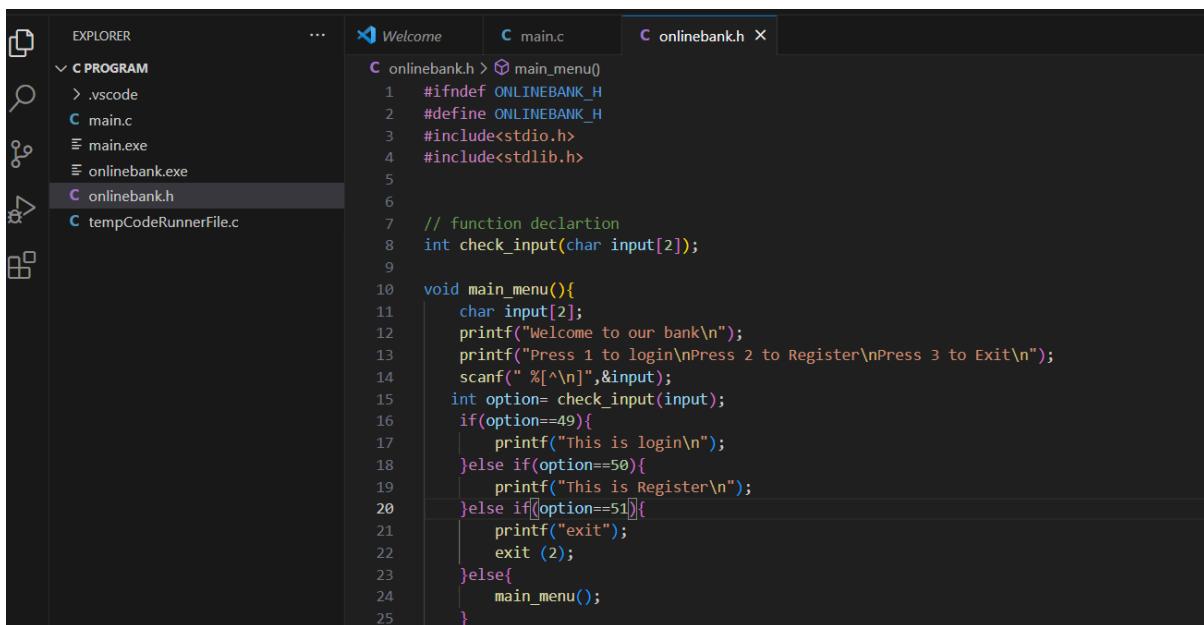
```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the project: .vscode, main.c, onlinebank.h, onlinebank.exe, tempCodeRunnerfile.c.
- Code Editor:** Displays the main.c file content, which includes the code provided in the text block above.
- Status Bar:** Shows "Ln 20, Col 24" and other standard status bar information.

Figure-42

ဒါပီးရင်တော့ ကျွန်တော်တို့ **Login** နဲ့ **register** ပိုင်းကို ဆက်ပီးရေးပါမယ့်။ **login**အတွက် ရေးတဲ့အခါ **email** အရင်တောင်းပါမယ်။ **email format** မှန်မမှန်စစ်ပီး **login** ဝင်ထဲ **password** တောင်းရပါမယ်။ သင်ခန်းစာတစ်ခုလုံးနဲ့ပါးကို **char array** ကိုဘဲ သုံးသွားပါမယ့်။ ထို့နောက် **register** ပိုင်းကို ဆက်ရေးပါမယ်။ (**figure-43**)



```

EXPLORER          ...      Welcome    main.c      onlinebank.h
C PROGRAM         .vscode   main.c      onlinebank.h > main_menu()
                    main.c      main.exe
                    onlinebank.exe
                    onlinebank.h
                    tempCodeRunnerFile.c

C onlinebank.h > main_menu()
1  #ifndef ONLINEBANK_H
2  #define ONLINEBANK_H
3  #include<stdio.h>
4  #include<stdlib.h>
5
6
7 // function declaration
8 int check_input(char input[2]);
9
10 void main_menu(){
11     char input[2];
12     printf("Welcome to our bank\n");
13     printf("Press 1 to login\nPress 2 to Register\nPress 3 to Exit\n");
14     scanf(" %[^\n]",&input);
15     int option= check_input(input);
16     if(option==49){
17         printf("This is login\n");
18     }else if(option==50){
19         printf("This is Register\n");
20     }else if(option==51){
21         printf("exit");
22         exit (2);
23     }else{
24         main_menu();
25     }
}

```

Figure-43

ပြီးရင်တော့ **main_menu**ထဲကနေခေါ်သုံးလို့ရအောင် ငြင်း၍အပေါ်ဘက်မှာ **function**

declaration လုပ်ပေးရပါမယ်။ ထို့နောက် **option 1** နေရာမှာ **login** ကိုခေါ်ပီး **option 2**

နေရာမှာ **rEgister** ကို ခေါ်ပါမယ့်။ **figure -44 (line-9 to 21)** ။

The screenshot shows a Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view of files and folders. The 'C PROGRAM' section is expanded, listing '.vscode', 'main.c', 'main.exe', 'onlinebank.exe', 'onlinebank.h' (which is selected), and 'tempCodeRunnerFile.c'.
- Terminal:** Shows the command 'onlinebank.h - C program - Visual Studio Code [Administrator]
- Code Editor:** Displays the 'main.c' file content. The code implements a simple menu system for a bank, allowing users to log in, register, or exit.

```
EXPLORER ... Welcome main.c onlinebank.h

C onlinebank.h > main_menu()
7 // function declaration
8 int check_input(char input[2]);
9 void login();
10 void rEgister();
11
12 void main_menu(){
13     char input[2];
14     printf("Welcome to our bank\n");
15     printf("Press 1 to login\nPress 2 to Register\nPress 3 to Exit\n");
16     scanf(" %[^\n]",&input);
17     int option= check_input(input);
18     if(option==49){
19         login();
20     }else if(option==50){
21         rEgister();
22     }else if(option==51){
23         printf("exit");
24         exit (2);
25     }else{
26         main_menu();
27 }
```

Figure-44

ဒီဇန်ရာမှာ ကျွန်တော်တို့သည် **program** တစ်ခါရေးပီးတိုင်း **ctrl+s** ကို အမြန်ပုံပေးနေရတယူမို့ **ctrl+s** နှင့်ဖို့ ခဏခဏ မွေ့တတ်ကြတာမို့ **program** ရေးပီးတိုင်း တန်း**run** လို့ရအောင် လုပ်ကြပါမယ်။ အဲလိုလိုပိုဖိုရင် **file** ထဲက **preferences** ထဲက **setting** ထဲကိုဝင်ကာ **search bar** မှာ **auto save**လို့ရိုက်ရှာပီး **off** နေရာမှာ **after delay**လို့ပြောင်းလိုက်ရင် ရပါပီ။ ပို့ဗုံးရင် **autosave delay** နေရာမှာ **1000**လို့ ထားလိုက်ရမှာပါ။ အဲဒီဇန်ရာမှာ **1000**သည် **millisecond**ကို ရည်ညွှန်းပါတယ့်။ **1000 millisecond**မှာ **1second**နဲ့သို့မျှပါတယ်။အထက်ပါအတိုင်းရေးပီးသွားပါက ကျွန်တော်တို့ က **Program** တစ်ခါ ရေးပီးတိုင်း **ctrl+s** နှုံပုံပေးနေစရာ မလိုတော်ပါဘူး။

rEgisterပိုင်းကို စဉ်းစားကြည့်ရအောင်။ကျွန်တော်တို့သည် **register** လုပ်ဖို့အတွက် **email** ထည့်တဲ့အခါ ဒီ**email**သည် နဂိုတည်းက ကျွန်တော်တို့ဆိုမှာ **account** ဖွင့်ပီးသားဆိုရင် **account** တစ်ခုကို **email**တစ်ခုဘဲလက်ခံမှာမို့ နောက်တစ်ခု ထပ်ထည့်ခိုင်းပါမယ်။ ကျွန်တော်တို့သည် **email**

ထည့်တဲ့ အခါမှာလဲ **email** စစ်ဆစ် ကို **email format** မှန်လားဆိုတာကို စစ်ဆေးရအုံမှာ ဖြစ်ပါတယ့်။

ပထမဆုံးအနေနဲ့ ကျွန်တော်တို့ **systems** ပေါ်မှာ **user** ထည့်လိုက်သည့် **email** သည် ရှိထားပီးသားလားဆိုတာ စွဲတဲ့ **function** ရေးပါမယ်။ **void email_exit_checking** လိုကြညာပီး ငှုံးထဲမှာ **char email[50]** ဆိုတဲ့ **parameter** တစ်ခုဖြတ်ပါမယ့်။ **email** ကိုစစ်ဖို့ဆိုရင် ဝင်လာတဲ့ **email** ရဲစာလုံးအရေအတွက်က စာလုံးရေမည်မျှရှိလဲဆိုတာ သိရ ဖို့**int char_counting**ကိုလဲ ရေးဖို့လိုပါတယ့်။ ဒီနေရာမှာ **parameter** ထဲက **char array** ထဲက အလုံးအရေအတွက်က **50** ဖြစ်နေရခြင်းသည် **looping** ပါတ်တဲ့အခါ မှာ အဆင်ပြုစေရန်အတွက်ဘဲဖြစ်ပါတယ်။

Email စာလုံးအရေအတွက် စစ်ရန် အောက်ပါအတိုင်း **function** ရေးသားပါမယ်။

```
int char_counting(char my_char[50]){
    int count=0;
    for(register int a=0;a<50;a++){
        if(my_char[a]=='\0'){
            break;
        }count++;
    }
    return count;
}
```

count ကို **0** လိုကြညာပီးသောအခါ **looping** ပါတ်ပီး **email** ထဲတွင်ပါသော စာလုံးအရေအတွက် ကို စစ်ပါမယ်။ **for loop** ထဲရှိ **register** သည် **storage class** ဖြစ်ပီး

ဂုဏ်ကို အသုံးပြုရချင်းမှာ မြန်မြန်ဆန်စွန် အလုပ်လုပ်စေရန်အတွက် ဖြစ်ပါတယ့်။ **Array** တွေရဲ့ နောက်ဆုံးအခန်းသည် “**10**” ဖြစ်တာမို့ **array** ရဲ့နောက်ဆုံးအခန်း ကိုရောက်လျှင် **break** လိုပေးလိုက်ပါမယ့်။ ဥပမာ- national@gmail.comလို့ ထည့်ကြည့်ရင် ၁၈လုံးမြောက်သည် “**10**” ဖြစ်နေပါလိမ့်မည်။ စာလုံးအရေအတွက် ကိုရောတွက်ရန် **count++** ကို သုံးလိုက်ပါတယ့်။ ပြီးရင်တော့ **count** ကို **return** ပြန်လိုက်ပါမယ့်။ ပြီးရင် **function decleartion** လုပ်ပီး **email_exit_checking** ထဲမှာ **char_counting** ကို လုမ်းခေါပါမယ်။ **return** ပြန်လာတဲ့ဟာကို **int counter**နဲ့ ယူပါမယ်။ **lesson** အစကနေ လက်ရှိရောက်နေသော နေရာထိရေးထားသော **git** **gub** **link**ကိုအောက်မှာဖော်ပြပေးထားပါတယ့်။ **lesson-**

1-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C>

DataBase for Bank

ဒီသင်ခန်းစာများကတော့ **structure** အကြောင်း ပြောပြပေးသွားမှာဖြစ်ပါတယ်။ **program** အလုပ်လုပ်နေတဲ့အချိန်မှာ **Data** တွေကို**file**ထဲမထည့်ခင် **structure** ထဲမှာ **store** လုပ်သွားမယ်။**file** ထဲကို **data** တွေသွားသိမ်းတဲ့အခါ တစ်လုံးချင်းစီသိမ်းရပါတယ်။ ပထမဆုံးအနေနဲ့ **info** လို့အမည်ပေးထားတဲ့ **struct** တစ်ခုကို **main_menu**ရဲ့အပေါ်မှာ တည်ဆောက်ပါမယ့်။

```
struct info{  
}  
}
```

ထို့နောက် **figure-37**တွင်ရှိသော **project flow** ရေးသားခဲ့သော စာမျက်နှာ၏ **copy** ကူးကာ **struct info** ထဲတွင် **paste** လုပ်ပါမယ်။ ပီးရင် သူတို့၏ **data type** တွေ ထည့်ပေးပါမယ်။

```

struct                                                 trans{
    char                                                 note[200];
};

struct                                                 info{
unsigned int id; // user id

    char name;
    char nrc;
    char email[50];
    char password[50];
    unsigned int phoneNumber;
    char encryption_key[50]; //ထည့်လိုက်သောpassword ကိုencrypt လုပ်ခြင်း

    char recovery_key[50];
    int account_type; //1 personal, 2 business, 3 other.
    char account_status[10];
    int account_level;
    int minimum_opening_deposit; //အနိမ့်ဆုံးငွေလွှဲနိုင်သည့်ပမာဏ

    char currency[5];
    unsigned long long int current_amount;
    char loanStatus[1];
    unsigned int monthly_income;
    unsigned int loan_amount;
    float loan_rate;
    char address[100];
Struct trans tr[300];

}

```

User တစ်ယောက်ချင်းဆီ ကနေ တစ်ယောက်ကို ငွေလွှဲတဲ့အခါ မှတ်ထားမယ့် ဗုဒ္ဓရ **record**

အရေအတွက်ကို အခါ ၃၀၀အထိ ထားပါမယ့်။ recordတစ်ခုဆိတိင်းမှာလဲ စာလုံး၏၀၀အထိလက်ခံထားပါမယ်။ အထက်ပါအတိုင်း ရေးပီးပါက userဘယ်နစ်ယောက် databaseထဲ ထည့်နိုင်မလဲဆိုတာရေးပေးရပါမယ်။

```
void email_exit_checking(char email[50]){
    int counter=char_counting(email);
    int same_counting=0;
    for(register int gcc=0;gcc<G_index;gcc++){
        int db_email_count=char_counting(db[gcc].email);
        if(counter==db_email_count){
            for(register int a=0;a<counter;a++){
                if(email[a]!=db[gcc].email[0]){
                    break;
                }
                same_counting++;
            }
        }
    }
}if(counter==same_counting){
    email_found=gcc;
}
```

အထက်ပါ program ဟာ register လုပ်ဖို့ထည့်လိုက်တဲ့ emailဘူး database

(structure) ထဲမှာ ရှိပီးသားလားဆိုတာ စစ်ဆေးတဲ့ program ဖြစ်ပါတယ်။

ပထမဆုံးအနေနဲ့ User ထည့်လိုက်တဲ့ email ကို အလုံးအရေအတွက် စစ်ဆေးဖို့အတွက်

```
int counter=char_counting(email); functionကို သုံးပီး counter ဆိုတဲ့ int
variable ထဲမှာ store လုပ်ပါမယ့်။ ကျွန်တော်တို့ looping ပါတယ်တဲ့နေရာမှာ
for(register int gcc=0;gcc<G_index;gcc++) G_index ကို သုံးထားရခြင်းက
```

database ထဲမှာ ရှိတဲ့ **user** အရေအတွက်အတိုင်း စစ်ဆေးချင်လို့ဘဲ ဖြစ်ပါတယ့်။ ဒီနေရာမှာလဲ **database** ထဲက **email**တစ်ခုချင်းဆီတိုင်းမှုရှိတဲ့ စာလုံးအရေအတွက် ကိုသိဖို့ရန်အတွက် **char_counting function** ကို သုံးပေးရမှာဖြစ်ပါတယ့်။

```
for(register int gcc=0;gcc<G_index;gcc++){
    int db_email_count=char_counting(db[gcc].email);
    if(counter==db_email_count){
```

အထက်ပါစာကြောင်းများမည်သို့အလုပ်လုပ်သွားသည်ကိုသိရှိနားလည်နိုင်ရန် **looping**

အကြောင်း ကောင်းကောင်း နားလည်သဘောပေါက်ရန် လိုအပ်ပါသည်။ကျွန်ုတ်တို့က

User ထည့်လိုက်တဲ့ **email** နဲ့ **database** ထဲက **email** နဲ့ အလုံးအရေအတွက်တူနေပါက ထို့ကို **email** နှစ်ခုဟာ တူညီလားဆိုတာ တစ်လုံးချင်းဆီတိုက်စစ်ပါမယ်။ အကယ်၍ မတူပါက **program** ကို **break** လုပ်ပစ်လိုက်မှာဖြစ်ပါး တူညီနေပါက တစ်လုံးချင်းဆီကို **same_counting** ထဲကို ထည့်ပေါင်းပေးမှာ ဖြစ်ပါတယ်။ **email** နှစ်ခုဟာတူညီနေခဲ့ပါ

က **email_found** ဆိုတဲ့ **global variable** တစ်ခု ကြေညာပီး **gcc(index)** ဘယ်လောက်မှာ တွေပါတယ်လို့ ရေးလိုက်ပါမယ့်။**Program** အစ ကနေ လက်ရှိရောက်နေသောရေးသားပီးသော **code** ၏ **github link**ဖြစ်သည်။

Lesson 2-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C>

Loading From File

ဒီသင်ခန်းစာမှာကတော့ **File(database)** ထဲမှာရှိတဲ့ **data** တွေကို ယူတဲ့အကြောင်းရှင်းပြပေးသွားမှာ ဖြစ်ပါတယ့်။ ဒီအပိုင်းမှာတော့ **db** ထဲကို **data** တွေထည့်တာ၊ **data** တွေထုတ်ယူတာ မျိုးလုပ်ပြသွားမှာ ဖြစ်ပါတယ်။

```
void loading_from_file(){

    FILE *fptr = fopen("database.txt","r");

    if( fptr != NULL){

        for(register int user=0; user < USERSIZE ; user++){

            fscanf(fptr ,"%u%s%s%s%u%s%s%d%d%d%s%llu%s%u%u%f%s",&db[user].id ,&db[user].name ,&db[user].nrc,&db[user].email,&db[user].password,&db[user].phoneNumber,&db[user].encryption_key,&db[user].recovery_key,&db[user].account_status,&db[user].account_type,&db[user].account_level,&db[user].minimum_opening_deposit,&db[user].currency,&db[user].current_amount,&db[user].loanStatus,&db[user].monthly_income,&db[user].loan_amount,&db[user].loan_rate,&db[user].address);

            for(register int trc=0; trc<= space_array[user]-19 ; trc++ ){

                fscanf(fptr , "%s",&db[user].tr[trc].note[0]);
            }

            if(db[user].id == 0){
                break;
            }

            G_index++;

        }

    } else{
        printf("File opening error at Loading form file function!\n");
    }

    fclose(fptr);
}
```

}

The screenshot shows the Visual Studio Code interface with two tabs open: 'onlinebank.h' and 'main.c'. The code in 'main.c' is as follows:

```
FILE *fptr = fopen("database.txt", "r");
if( fptr != NULL){
    for(register int user=0; user < USERSIZE ; user++){
        fscanf(fptr , "%s%s%s%s%s%s%d%d%s%lu%s%u%u%u", &db[user].id , &db[user].name , &db[user].nrc,&db[user].email,&db[user].password,&db[user].phoneNumber,&db[user].trc);
        if(db[user].id == 0){
            break;
        }
        G_index++;
    }
} else{
    printf("File opening error at Loading form file function!\n");
}
fclose(fptr);
#endif // ONLINEBANK_H
```

Figure-45

အထက်ပါအတိုင်း **data** တွေထည့်ဖို့အတွက် ဆိုရင် **database.txt** တစ်ခုကို ဆောက်ပေးသွားရမှာ

ဖြစ်ပါတယ့်။၎င်း file ကို database အနေနဲ့ သုံးသွားမှာဖြစ်ပါ

တယ့်။ အထက်ပါ programသည် database(database.txt) ထဲတွင်ရှိသော dataများကို

ဖတ်သည့် အဆင့်ဖြစ်သည်။ ကျွန်တော်တိုက **file** ထဲက **data** များကို **ကြည့ဖိုဆိုရင်** **file** ကို

အရင်ဖွင့်ရန် လို အပ်ပါသည်။ စာကြောင်းရေ 128 တွင်ရှိသော

`FILE *fptr = fopen("database.txt", "r");` သည် **file** ကိုဖွင့်တွေ့ဖြစ်ပါး **fopen** ဆဲ

တွင် မိမိတိုဖွင့်လိုသော file ၏ name နှင့် ရင်း file ကို မည်သည့် mood ဖြင့် ဖွင့်

မည် ဆိုတာကို ပြောခြင်းဖြစ်သည်။ ယခုတေကြောင်းတွင် **database.txt** ကို r(read)

Mood နှုဖတ်မည်ဟု ဆိုလိုခြင်းဖြစ်သည်။ ထို့နောက် **file** ကသာ **NULL** မဟုတ်ခဲ့ဘူးဆို၏။

ရင် **file** ထဲမှာရှိတဲ့ **data** တွေ သွားဖတ်ဖို့အတွက် **fscanf** ကို သုံးပါး **file** ကို စဖတ်ပါမယ်။

စာရေးသူက **database.txt** ထဲတွင် အောက်ပါ **data** အနည်းငယ်ကို ထည့်ထားပါသည်။

```

file Edit Selection View Go Run Terminal Help
EXPLORER C PROGRAM OUTLINE TIMELINE
database.txt
1 1 winhtut 9/paooia(N)80099 winhtutline@gmail.com admin123 959985890885 aaa123wh aaa123wh personal 1 1 10000 mmk 15000 T 5000 1 0.5 pyinoolwin online
2 2 winhtut 9/paooia(N)80099 winhtutline@gmail.com admin123 959985890885 aaa123wh aaa123wh personal 1 1 10000 mmk 15000 T 5000 1 0.5 pyinoolwin online
3 3 winhtut 9/paooia(N)80099 winhtutline@gmail.com admin123 959985890885 aaa123wh aaa123wh personal 1 1 10000 mmk 15000 T 5000 1 0.5 pyinoolwin online

```

Figure-46

အရင်သင်ခန်းစာမျာတုန်းက **transaction record** ကို အခု 300 အထိ ယူလိုခြတယ်လို့

ပြောခဲ့ဖူးပါတယ်။ **figure-46** ထဲက **pyinoolwin** နောက်က **onlinebankadmin**

သည် **transaction record** တစ်ခုဖြစ်ပါတယ်။ ကျွန်တော်တို့က ငွေလွှာတဲ့အခါမှာ **transaction record** တစ်ခုတိုးလာမှာ ဖြစ်ပါတယ်။ ထိုအခါ **onlinebankadmin** ရဲ့

အနောက်မှာ **data** အသစ်ထပ်ထည့်နိုင်ဖို့ စဉ်ဝါးစားရပါတော့မယ်။ **C programming** မှာ ရှိရင်းဖြူ **data** ထဲကို **data** အသစ်ပေါင်းထည့်တဲ့အခါ ငါးကြားထဲမှာရှိတဲ့ **space** ကိုပါထည့်သွင်းစဉ်ဝါးစားပေးရတာဖြစ်ပါတယ့်။ထို့ကြောင့် **id** 1ကနေ **onlinebankadmin** အထိ **space** ဘယ်နှစ်ခုပါလဲ စဉ်ဝါးစားရတော့မှာဖြစ်သလို **data** ထပ်ပေါင်းထည့်ပီးသော အခါတွင်လဲ **space** ဘယ်စုစုပေါင်းလဲ ရေတွက်ရတော့မှာ ဖြစ်ပါတယ့်။

လက်ရှိရေးသားထားပီးသော **code** ၏ **github link**ဖြစ်သည်။

lesson-3->[**https://github.com/NationalCyberCityNiST/Bank-Project-With-C**](https://github.com/NationalCyberCityNiST/Bank-Project-With-C)

Space Counter

Space ကိုရေတွက်ဖို့အတွက် **space_counter** ဆိုတဲ့ **function** ကိုတည်ဆောက်မည်

ဖြစ်ပါတယ့်။ **void** နဲ့ရေးမှာဖြစ်ပါတယ့်။

```
void space_counter(){
```

```
FILE *fptr = fopen("database.txt","r");

if(fptr != NULL){
    char c = fgetc(fptr);
    int index=0;

    while (!feof(fptr)){
        if(c != '\n'){

            if( c == ' '){
                space_array[index] +=1;

            }
            c = fgetc(fptr);

        } else{

            index++;
            c = fgetc(fptr);
        }
    }

} else{
    printf("File open error at space_counter fun()\n");
}

for(int i=0; i<DATA_COUNT; i++){

    printf(" %d",space_array[i]);
}
printf("\n");
```

```
}
```

အထက်ပါ **program** သည် **space** အရေအတွက်ကို စစ်ဖို့ရန်အတွက် ရေးသားထားခြင်း

ဖြစ်သည်။ **database.txt file** ကို **r mood** ဖြင့်ဖွင့်ပီးနောက်

```
char c = fgetc(fptr);
```

က ဘာကိုပြောတာလဲဆိုတော့ **pointer** ထောက်ထားတဲ့ **fptr** ထဲက **data** တွေကို တစ်လုံးချင်း

ယူဝါး **char c** ထဲကို ထည့်တာဖြစ်ပါတယ်။ ထို့နောက် **indexဆိုတဲ့ Integer variable** တစ်ခုကို

```
0 လိုပေးပီးကြညာထားပါတယ့်။ (!feof(fptr))
```

က ဘာကိုပြောချင်တာလဲ ဆိုတော့ **file** ရဲနောက်ဆုံးနေရာကို ရောက်ပီလားဆိုတာ

သိဖို့ဖြစ်ပါတယ့်။ ဆိုလိုတာက **file** မဆုံးမချင်း **while** ကို အလုပ်လုပ်နေဖို့ဖြစ်ပါ

တယ့်။ နောက်တစ်ကြောင်းဆင်းကြည့်ပါက **if-else condition** ကို တွေ့ရမှာ ဖြစ်

ပါတယ့်။ **figure-46** က ပထမစာကြောင်းက **onlinebankadmin** သို့ရောက်သောအခါ

နောက်တစ်ကြောင်းဆင်းရပါမယ့်။

ကျွန်တော်တို့ဟာစာတစ်ကြောင်းပီးလို့ဆင်းချင်တဲ့အခါ

“\n”ကိုသုံးရတာမှတ်မိမယ်ထင်ပါတယ့်။**figure-**

46ရဲပထမစာကြောင်းရဲအနောက်မှာ “\”လေးရှိပါတယ့်။ကျွန်တော်တို့ကမမြင်ရပေမယ့်**program**ကသိ

ပါတယ့်။**if condition**မှာ

ဘာပြောထားတာလဲဆိုရင် **char c**က \nမဟုတ်သေးဘူးဆိုရင်ကျွန်တော်တို့ယူလိုက်တဲ့**char c** က

“ “**space_array** ဆိုတဲ့ **array[0]** ထဲကို +1ပေါင်းထည့်ပေးမှာ ဖြစ်ပါတယ့်။ **space_array** ကို

global မှာ ကြညာပေးထားရပါမယ့်။

The screenshot shows the Visual Studio Code interface. The Explorer sidebar on the left lists files: .vscode, tasks.json, corn_decryption.h, corn_encryption.h, database.txt, encrypted_data.txt, main.c, main.exe, onlinebank.exe, onlinebank.h, project flow, secheader.h, and secmain.c. The main editor area has tabs for main.c, onlinebank.h (active), secheader.h, and database.txt. The code in onlinebank.h is:

```

#ifndef ONLINEBANK_H
#define ONLINEBANK_H
#include<stdio.h>
#include<stdlib.h>
#define USERSIZE 1000
#define DATA_COUNT 20
//global variable
int G_index=0; //for user counting
int email_found=-1; //for checking email exit
int space_array[DATA_COUNT];
//end of global variable

```

else condition මුළු වාච්‍යාතාල් ස්ථිරක් program ගැසැමතාගැනීමේදී space

මගුවෙනුවේ නිශ්චිත තර්ංගාදීම් නිර්මාණය වේ. index නිශ්චිත වූ

පෙන්සේ: එහි තාගැනීම් **while loop** නොකළ තර්ංගාදීම් පෙන්වනු ලබයි. index නිශ්චිත වූ

නිශ්චිත නිශ්චිත ප්‍රාග්ධනය වූ figure-46 ගැසැමතාගැනීමේදී space නිශ්චිත වූ

index 1 මුළු පෙන්සේ: නිශ්චිත ප්‍රාග්ධනය වූ

space_array[] = {19, 19, 0, 0, 0, ...};

The screenshot shows the Visual Studio Code interface with the terminal tab active. The terminal output is:

```

PS C:\Users\MSI\Desktop\C program> cd "c:\Users\MSI\Desktop\C program"
19 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Welcome to our bank
Press 1 to login
Press 2 to Register
Press 3 to Exit

```

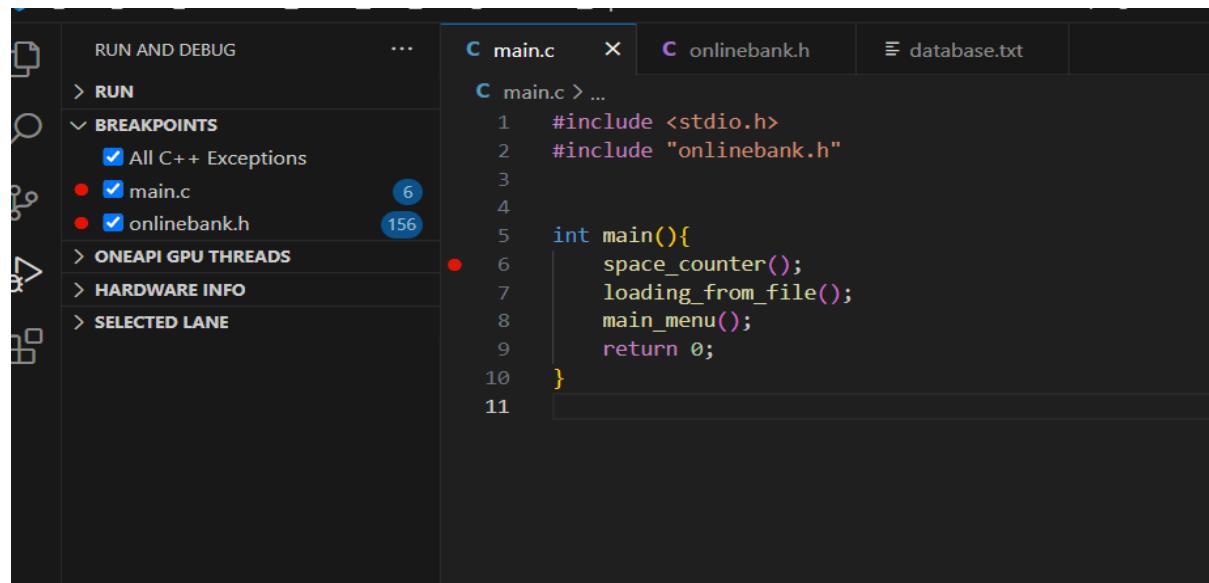
Figure-47

Debugging Space_counter

ဒီသင်ခန်းစာများ **Space_counter** ကို **debugging** ထောက်ပြသွားမှာ ဖြစ်ပါတယ့်။

တစ်ချို့အခြေအနေမှာ ကျွန်တော်တို့ **program** က ဘယ်လိုအလုပ်လုပ်လဲဆိုတာ အပြင် ပိုင်းကြည့်ယံနဲ့ မသိနိုင်ပါဘူး။ အဲလို အခြေအနေမျိုးမှာ **debugging** ထောက်ပီး စစ်ဆေးလို့ ရုံးနိုင်ပါတယ့်။

ကျွန်တော်တို့ **debug** ထောက်ချင်တဲ့ ဘေးနားမှာ **click** တစ်ချက်နှင့်ပိုး **debug** ထောက်ရမှာ ဖြစ်ပါတယ့်။



The screenshot shows a debugger interface with the following details:

- Left Sidebar:** RUN AND DEBUG menu is open, showing options like RUN, BREAKPOINTS, ONEAPI GPU THREADS, HARDWARE INFO, and SELECTED LANE. Under BREAKPOINTS, there are checkboxes for All C++ Exceptions, main.c (with 6 breakpoints), and onlinebank.h (with 156 breakpoints).
- Top Tab Bar:** The tabs are main.c, onlinebank.h, and database.txt. main.c is selected.
- Code Editor:** The code for main.c is displayed:

```

1  #include <stdio.h>
2  #include "onlinebank.h"
3
4
5  int main(){
6      space_counter();
7      loading_from_file();
8      main_menu();
9      return 0;
10 }
11 
```

figure-48

```

File Edit Selection View Go Run Terminal Help
RUN AND DEBUG ...
> RUN
> BREAKPOINTS
  ✓ All C++ Exceptions
    ● main.c
    ● onlinebank.h
> ONEAPI GPU THREADS
> HARDWARE INFO
> SELECTED LANE
C main.c C onlinebank.h database.txt
144     }
145     }
146     }
147     }
148   }
149   }
150   }
151   }
152   }
153   }
154   }
155   }
156 } else{
157   printf("File opening error at Loading form file function!\n");
158 }
159 FILE *fptr = fopen("database.txt","r");
160 if(fptr != NULL){
161   char c = fgetc(fptr);
162   int index=0;
163   while (!feof(fptr)){
164     if(c != '\n'){
165       if( c == ' '){
166         space_array[index] +=1;
167       }
168       c = fgetc(fptr);
169     } else{
170       index++;
171       c = fgetc(fptr);
172     }
173   }
174 } else{
175   fclose(fptr);
176 }
177 }
178 }
179 }
180 } else{
181 }

```

Ln 159, Col 1 Spaces: 4 UTF-8 CRLF {} C Win32

Figure-49

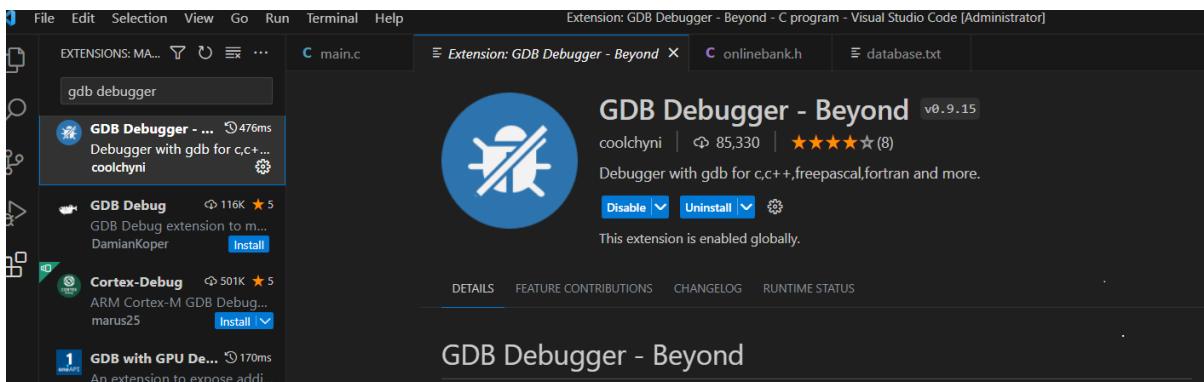
Debugging ထောက်ဖို့ဆိုရင် figure-48 ၏ main.c နေရာမှာ run ရမှာဖြစ်ပါတယ့်။

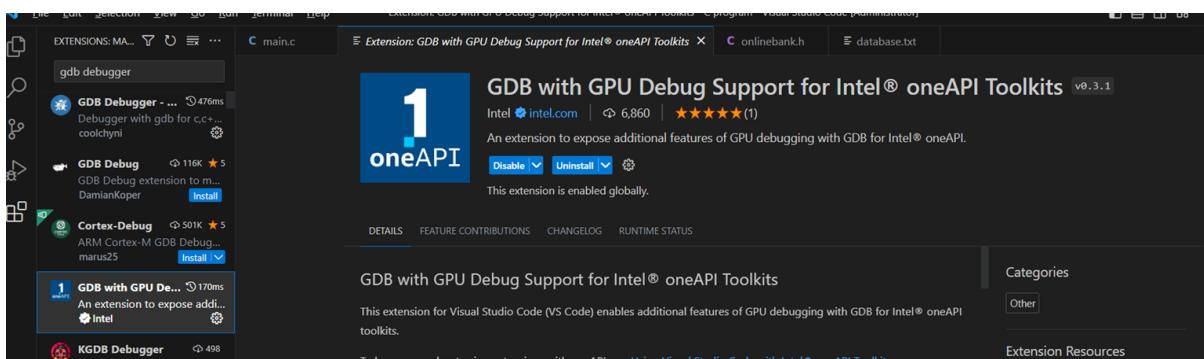
ကျွန်ုတ်တို့ debugging ထောက်ဖို့ရင် gdb debugger extensions install

ပေးဖို့လိုအပ်နေပါသေးတယ်။

ထိုနောက် extensions ထဲမှာ gdb debugger လို့ search bar မှာ ရိုက်ရှာပီးအောက်ပါ

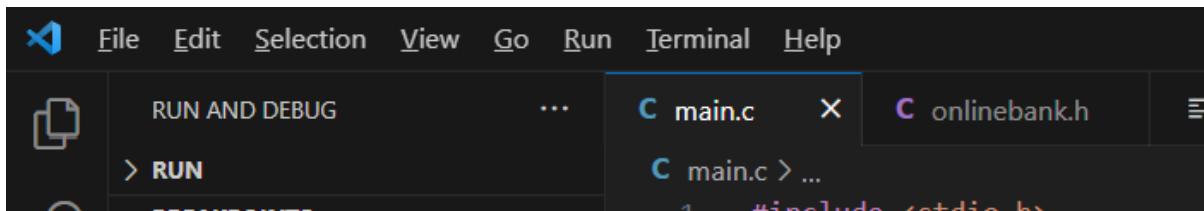
နှစ်ခုကို install လုပ်လိုက်ရင်ရပါမီ။





အထက်ပါ **extensions** တွေသုင်းပီးသွားရင်သောခါ **figure-48** ထဲက **BREAKPOINTS** ထဲမှာ

main.c နဲ့ **onlinebank.h** ဘေးနားက နံပါတ်များကကျွန်တော်တို့ **debug** လုပ်ချင်တဲ့ **linenumber**များဖြစ်ပါတယ့်။ထို့နောက် **Go** ရဲ့ဘေးနားက **Run** ကိုနိုပ်ကာ **Start** **Debugging** ကို
နိုပ်လိုက်ပါက **figure-50**အတိုင်း ပေါ်လာမှာ ဖြစ်ပါတယ့်။



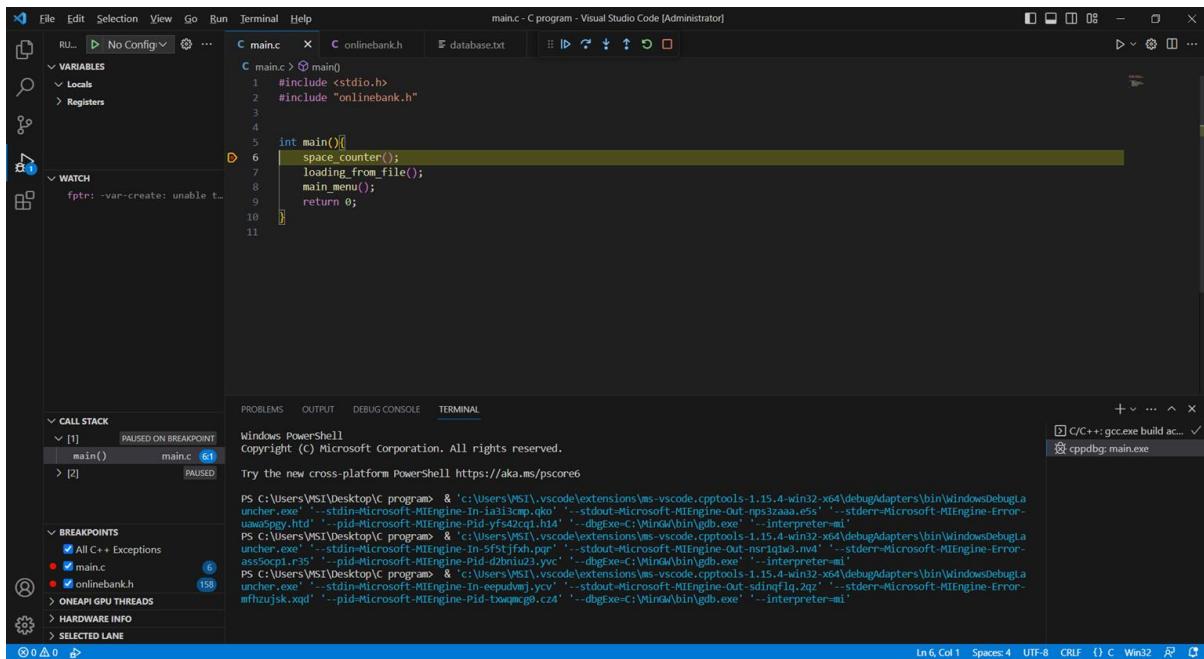


Figure-50

ດັ່ງນີ້ແມ່ນການສະແດງຂອງລາຍລະອຽດທີ່ມີຄວາມສຳເນົາ

မြောက် ခလုတ်ကို နိုပ်လိုက်ပါက **onlinebank.h** ထဲက **space_counter function** ကို

ခုနှင့်ကူးသွားမှုဖြစ်ပါတယ်။ထို့ခလုတ်သည် **Line by line debugging** ထောက်သည်

ခလုတ်ဖြစ်ပါတယ့်။

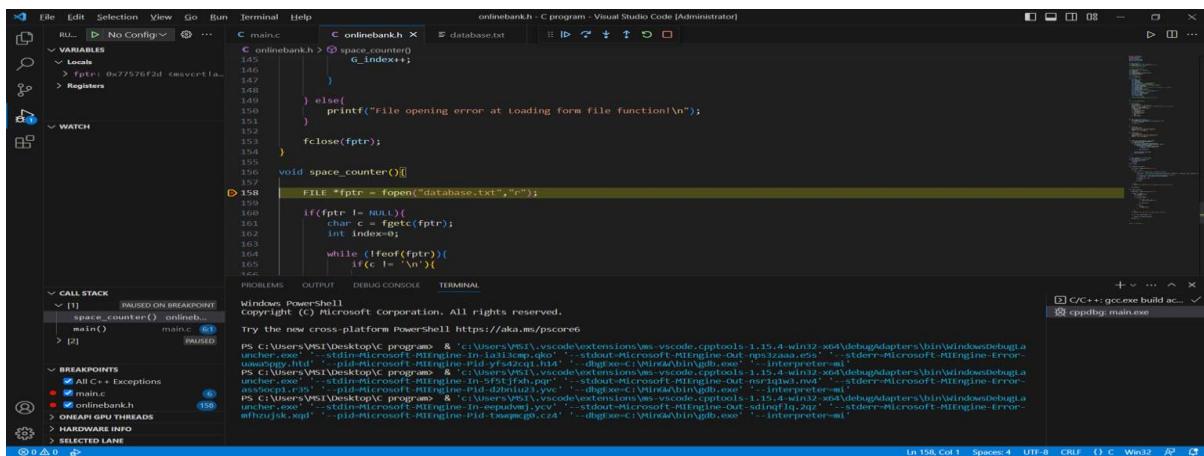


Figure-51

အဝါရောင် မျဉ်းကြောင်းသည် ကျွန်တော်တို့ လက်ရှိ debugging ထောက်ထားသောနေရာ ဖြစ်ပါတယ့်။ **fptr** ကို **copy** ယူကာ ဘယ်ဘက်မှာ ရှိတဲ့ **WATCH** ရဲ့သေးနားက + လေးကို နိုးပိုးပါ။
ထိနည်းတူ c ကိုလဲcopy ဆွဲကာ WATCH ထဲထည့်ပါမယ့်။ linebyline debug ထောက်သည့်ခလုတ်ကိုထပ်နိုးပိုးလိုက်ပါက အဝါရောင်လိုင်းလေးက

```
if(fptr != NULL){
    ကိုရောက်သွားမှာ ဖြစ်ပါတယ်။ fptr ရဲ့value သည် 0x775d4660 လိုပေါ်နေပါ
    တယ့်။ အကြောင်းအရင်းက fptr ဟာ NULLမဖြစ်နေသွားလို့ ဆိုလိုတာပါ။ ထို့ကြောင့်
    if condition အလုပ်လုပ်ကာ အောက်ကို တစ်ကြောင်းဆင်းဖို့ ထပ်နိုးပိုးလိုက်ပါက
    if(c != '\n') {
        ကိုရောက်သွားမှာ ဖြစ်ပါတယ်။
        ထိုနေရာမှာ c ရဲ့တန်ဖိုးက 0 '\000' ဖြစ်နေမှာဖြစ်ပါတယ်။နောက်တစ်ကြောင်းဆင်း
        ကြည့်ပါက c ရဲ့တန်ဖိုးသည် 1 ဖြစ်သွားမှာ ဖြစ်ပါတယ်။ အဘယ်ကြောင့်ဆိုတာ
Fgetc ကိုသုံးကာ fptrထဲက ပထမဆုံး data ကို ရယူလိုက်ပီး char c ထဲမှာ သိမ်း
    လိုက်တာ ဖြစ်ပါတယ်။ ထို့နောက် နောက်တစ်ကြောင်းဆင်းကြည့်တဲ့အခါမှာ
    while (!feof(fptr)){
```

ဒီနေရာမှာ နောက်တစ်ကြောင်းဆင်းကြည့်ပါက **file** ကသာ **feof** **file**ရဲ့အဆုံးသာ ဖြစ်နေခဲ့မယ့်ရင် **while loop** ကရပ်သွားမှာပါ။ **feof** မဖြစ်သေးဘူးဆိုရင်တော့
if else ကိုသွားအလုပ်မှာဖြစ်ပါတယ့်။

```
if(c != '\n') {
    ဒီနေရာကိုရောက်လာတဲ့အခါ c က '\n' နဲ့မညီသေးဘူးဆိုရင် နောက်တစ်ကြောင်းကို
```

ထပ်ဆင်းပါက c က “ ” နဲ့သိမလားကို စစ်မှာဖြစ်ပါတယ့်။

```
if( c == ' '){
```

အကယူ၍ c ကသာ space ဖြစ်နေခဲ့မယ်ဆိုရင် space_counter arrayထက် index

နံပါတ် 0 နေရာမှာ 1 ပေါင်းပေးမှု, တွေ့ဖြစ်ပါတယ်။ index ကို copyယူကာ watch

ထဲတွင်ထည့်ပါးကြည့်ပါ။ ဒီနေရာမှာ indexက 0ဘဲဖြစ်နေအုံမှာပါ။ ၁ပေါင်းထည့်နေတာ

က index 0နေရာမှာသာပေါင်းထည့်နေတာဖြစ်ပါး index နေရာပြောင်းလိုက်တာမဟုတ်

ပါဘူး။ space တစ်ခုတွေ့တိုင်း ၁ ပေါင်းထည့်နေရင်းနဲ့ '\n' ကိုတွေ့သွားခဲ့မယ်ဆိုရင်

တော့ else condition ထဲဝင်သွားပါး index++ ကိုရောက်သွားမှာပါ။ ဒီအချိန်မှာတော့

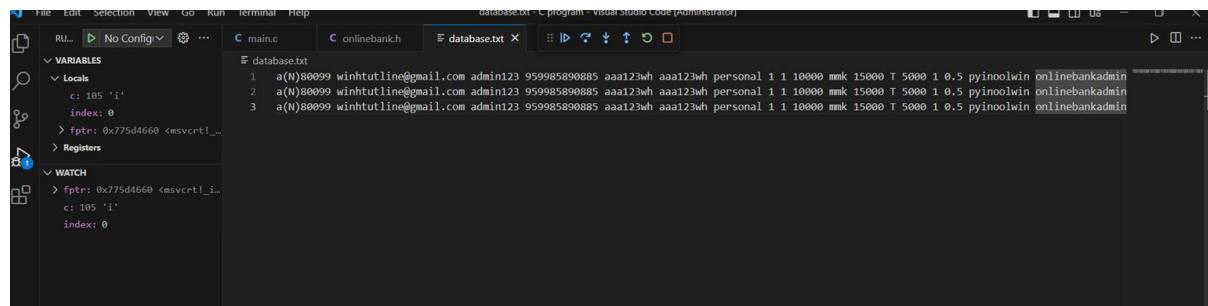
index က 1 လိုချိန်းသွားတာကိုမြင်ရပါလိမ့်မယ့်။ အဘယ်လိုင်းဆိုသော် နောက်တစ်

ကြောင်းဆင်းပါးနောက် ဂင်းစာကြောင်းတွင်ရှိသော space များကို index 1နေရာတွင်

ပေါင်းထည့်ချင်သောကြောင့်ဖြစ်ပါတယ်။ ဥပမာ - {20, 20, 0, 0, 0...}။ index 0 မှာ

ရှိသော 20 သည် ပထမစာကြောင်းတွင်ရှိသော space အရေအတွက်ဖြစ်ကာ index 1

နေရာမှာရှိသော 20 သည် ဒုတိယစာကြောင်းတွင်ရှိသော space အရေအတွက်ဖြစ်သည့်။



```
for(int i=0; i<DATA_COUNT; i++){
```

```
    printf(" %d",space_array[i]);
```

```
}
```

```
printf("\n");

```

ထိုနောက် database ထဲတွင်ရှိသော တစ်ကြောင်းချင်းဆီတိုင်း၏ spaceအရေအတွက်ကို Looping ပါတ်ပီးထုတ်လိုက်တာ ဖြစ်ပါတယ့်။

Printing_ALL_Data_From_File

အရင်သင်ခန်းစာမျာတုန်းက File ထဲမှာရှိတဲ့ DATA တွေကို ယူပီးဖတ်ခဲ့ကြပါတယ့်။

ယခုသင်ခန်းစာမျာ တော့ ငြင်းDATA များကို ထုတ်ပြမှာဘဲဖြစ်ပါတယ်။

Program ရှင်းလင်းချက်

```
void printing_all_data() {

    for (int user = 0; user < G_index; user++) {

        printf("%u-%s-%s-%s-%s-%llu-%s-%s-%d-%d-%d-%d-%s-%llu-%s-%u-%u-%f-%s",
               db[user].id, db[user].name, db[user].nrc, db[user].email, db[user].password,
               db[user].phoneNumber, db[user].encryption_key, db[user].recovery_key,
               db[user].account_status, db[user].account_type, db[user].account_level,
               db[user].minimum_opening_deposit, db[user].currency, db[user].current_amount,
               db[user].loanStatus, db[user].monthly_income, db[user].loan_amount, db[user].loan_rate,
               db[user].address);

        for (int gcc = 0; gcc <= space_array[user] - 19; gcc++) {
            printf("-%s", db[user].tr[gcc].note);
        }
        printf("\n");
    }
}
```

အထက်ပါ **program** ကိုတစ်ကြောင်းချင်းရှင်းပြုရမည် ဆိုပါက ပထမဆုံးအနေနဲ့ **void**

Function နဲ့ရေးထားတာကို မြင်ရမှာ ဖြစ်ပါတယ့်။ **G_index** အတိုင်း **looping** ပါတ်

တဲ့အခါ **G_index** က ဘာကိုဆိုလိုတာလဲဆိုတော့ **database** ထဲမှာ ရှိတဲ့ **user** အရေအ

တွက်ကို ဆိုလိုတာ ဖြစ်ပါတယ့်။ အရင်သင်ခန်းစာမှာတူန်းက **G_index** ကို **global**

ကြညာပေးခဲ့ဘူးပါတယ်။

```
for (int gcc = 0; gcc <= space_array[user] - 19; gcc++) {
    printf("-%s",
           db[user].tr[gcc].note);
}
```

အထက်ပါ **program** က ဘာကိုဆိုလိုတာလဲဆိုတော့ ကျွန်တော်တို့က ပထမ **printf**ထဲ

မှာ **db[user].address** အထိဘဲ **data** ထုတ်ခဲ့ပါးတော့ **transaction record** အတွက်ကို

သီးသန့်ရေးတာဘဲဖြစ်ပါတယ်။ **transaction record** အခု ၃၀၀ နဲ့ **record**တစ်ခုကို အချေဝေ

အထိ လက်ခံလို့ရတယ်ဆိုပါး ဖန်တီးထားခဲ့ပါတယ်။

loopingပါတ်တဲ့အခါ**Space_array[user]-19** ဆိုပါး ရေးထားတာက **id** ကနေစပီး

Address အထိ နှုတ်လိုက်တဲ့အခါ **transaction record** ကိုရေးလို့ရပါ ဖြစ်ပါတယ်။

ထို့နောက် **main.c** ထဲမှာ **function call** ခေါ်ပါး **program run** ကြည့်ပါက **database** ထဲက

data များ **terminal** မှာပေါ်လာမှာ ဖြစ်ပါတယ့်။ (**figure-51**)

The screenshot displays two instances of Visual Studio Code. The left instance shows the file structure of a C program named 'C PROGRAM' containing files like .vscode, launch.json, tasks.json, corn_decryption.h, corn_encryption.h, database.txt, encrypted_data.txt, main.c, main.exe, onlinebank.exe, onlinebank.h, project flow, secheader.h, and secmain.c. The main.c file is open in the editor, showing the following code:

```

1 #include <stdio.h>
2 #include "onlinebank.h"
3
4
5 int main(){
6     space_counter();
7     loading_from_file();
8     printing_all_data();
9     main_menu();
10    return 0;
11 }
12

```

The right instance shows the terminal output of the program running. The terminal window title is 'main.c - C program - Visual Studio Code [Administrator]'. The output shows the program's logic for loading data from a file and displaying a menu. The terminal also shows a welcome message and a prompt for user input (Press 1 to login, Press 2 to Register, Press 3 to Exit).

Figure-51

Data input/output အကြောင်း ရေးပီးတဲ့အခါ ကျွန်တော်တို့ loginနဲ့registration

ပိုင်းကိုပြပန်ရေးပါမယ်။ ကျွန်တော်တို့ **Login** လုပ်တာဘဲဖြစ်ဖြစ် **registration** လုပ်

တဲ့အခါ ဘဲဖြစ်ဖြစ် **email** တောင်းရပါတယ်။ ကျွန်တော်တို့က အကောင့်တစ်ခုကို **email**

တစ်ခုဘဲသုံးခွင့်ရှိတာမို့ **user**ထွေထွေလိုက်တဲ့ **email** ၊ **email** ၊ **format**

မှန်မှန်စစ်ဆေးပေးရပါမယ်။ ပြီးတော့ **email** က အကောင့်ဖောက်ပီးသားလားမဖောက်

ရသေးဘူးလားဆိုတာစစ်ဆေးပေးရပါမယ်။

Email Validation

ဒီသင်ခန်းစာမျာကတော့ **email format** မှန်မမှန်စစ်ဆေးတဲ့ **function** ရေးသားမှာဘဲ

ဖြစ်ပါတယ့်။ ပထမဆုံးအနေနဲ့ **email format** အကြောင်းအရင်သိတားဖို့လိုပါတယ်။

winhtut@gmail.com ဆိုရင် @gmail.com သည် **email format** အမှန်ဖြစ်ပါသည်။

Program ရှင်းလင်းချက်

```
void email_validation(char to_check[50]){

    int space=1;
    int format=1;
    char valid_form[10]={ '@', 'g', 'm', 'a', 'i', 'l', '.', 'c', 'o', 'm' };
    int check_counter = char_counting(to_check);

    int front_counter = check_counter-10;

    for(int i=0; i<front_counter ; i++){
        if(to_check[i]==' '){
            space=-1;
            break;
        }
    }

    for(int f=0; f<10; f++){
        if(to_check[front_counter] != valid_form[f]){
            format = -1;
            break;
        }
        front_counter++;
    }

    if(space == 1 && format==1){
```

```

    email_valid=1;
}
}

```

ပထမဆုံးအနေနဲ့ **email_valid** ဆိုတဲ့ **variable** တစ်ခုကို -1 ထားပါး **global**မှာ ကြော်သာပါမယ်။ **char array** တစ်ခုဖြတ်ထားတဲ့ **void function** တစ်ခုကိုရေး သားပါမယ်။ **space** နဲ့ **format** ဆိုတဲ့ **local variable** နှစ်ခုကို 1 လို့ထားပီးကြညာ ထားပါတယ်။ ပီးတော့ ကျွန်တော်တို့စစ်ချင်တဲ့ **@gmail.com**ကို **char array** ထဲမှာ သိမ်းထားပီး **user** ထည့်လိုက်တဲ့ **array** ရဲ့နောက်ဆုံးဝ လုံးကို တူမတူ စစ်မှာ ဖြစ်ပါတယ့်။ ကျွန်တော်တို့က ပထမဆုံးအနေနဲ့ **user** ထည့်လိုက်တဲ့ **email** ခဲ့စာလုံးအ ရေအတွက်ကို **char_counting function** သုံးပီးစစ်ပါတယ်။ **@gmail.com** ကိုဘဲစစ်ချင်တာမို့ **email.format** အရေအတွက်ဖြစ်တဲ့ ၁၀လုံးကို နှုတ်ပေးလိုက်ပါတယ့်။ **email**တွေက **space** မထည့်ပါဘူး။ဥပမာ **win h@gmail.com** ဆိုရင်မှားပါတယ်။
ထို့ကြောင့် **user** ကသာ **space** ပါတဲ့ **email** တစ်ခုကို ထည့်ခဲ့မယ်ဆိုပါက **email** မဟုတ်ကြောင်း ပြောပီး **Program** ပြန်စပါမယ်။

```

for(int i=0; i<front_counter ; i++){
    if(to_check[i]== ' '){
        space=-1;
        break;
    }
}

```

အထက်ပါ **code** သည် **space** ကိုစစ်ဆေးသည့်**code** ဖြစ်သည်။ ထို့နောက် **email format** ကိုစစ်ပါတွေ့မယ်။

```
for(int f=0; f<10; f++){
```

```

if(to_check[front_counter] != valid_form[f]){
    format = -1;
    break;
}
front_counter++;
}

```

အထက်ပါ program သည် email format ဘဲ စစ်မှာဖြစ်သည့်အတွက် looping ။

ကိုမြတ်ပါတယ်။ to_check ထဲက index front_counter ကဲ valid_form ထဲက index f နဲ့သာမညီခဲ့ဘူးဆိုရင် format ကို -1လို့ ပြောပြီး break လုပ်ပစ်လိုက်ပါမယ့်။ valid_form[f] ရဲနေရာဟာ စစ်ချင်း 0 ဖြစ်နေတာမို့ index 0 နေရာက data သည် @ဖြစ်နေပါသည့်။ တူညီနေခဲ့တယ်ဆိုရင်တော့ front_counter ကိုဘဲ တိုးတိုးသွားမှဖြစ်ပါတယ်။

```

if(space == 1 && format==1){
    email_valid=1;
}

```

အကယ်၍ user ထည့်လိုက်တဲ့ email မှာ space လဲမပါ format လဲမှန်တယ်ဆိုပါက Email_valid ကို 1 လို့ ကြေညာထားလိုက်ပါတယ့်။ email_valid =1 ကိုနောက်ပိုင်း Program တွေမှာ သုံးကြမှာ ဖြစ်ပါတယ်။

Email_Exit_Checking

ဒါprogram မှာတော့ register လုပ်တဲ့အခါ ကျွန်တော်တို့ထည့်လိုက်တဲ့ email က အကောင့်ဖွင့်ဖွံ့ဗီးပီးပီလား။ မဖွင့်ဖွံ့ဗီးသေးဘူးလားဆိုတာ စစ်တဲ့ functionအကြောင်းပြောမှာ ဖြစ်ပါတယ်။ email တွေစစ်ဖို့အတွက် database ထဲက email တွေ သွားယူဖို့ လိုပါမယ့်။ database ထဲက email တွေယူဖို့ အတွက် db[]email လိုပေးလိုက်ရင်ရပါတယ်။

```
void email_exist_checking(char email[50]){
    int counter = char_counting(email); //for user input email
    int same_counting=0;

    for(register int gcc=0; gcc<G_index ; gcc++){
        int email_count =char_counting(db[gcc].email); // for email from db,
        same_counting=0;
        if( counter == email_count ){
            for( register int a=0; a<counter ; a++){
                if(email[a] != db[gcc].email[a]){
                    break;
                }
                same_counting++;
            }
        }
        if( counter == same_counting){
            email_found=gcc;
        }
    }
}
```

ထိုးစံအတိုင်း **user** ထည့်လိုက်တဲ့ **email** ရဲ့ အလုံးအရေအတွက်ကို အရင်စစ်ပါမယ်။
အကောင့် တစ်ခုမှာ **email** တစ်ခုဘဲရှိတာမို့ **database** ထဲတွင် **email**ဘယ်နှစ်ခုရှိလဲ
ဆိုတာ **user** အရေအတွက်ကို ကြည့်ပီးသိနိုင်မှာ ဖြစ်ပါတယ့်။ အရင်သင်ခန်းစာမှာ
တုန်းက **G_index** သည် **user** အရေအတွက်နှင့် အတူတူလိုပြောပြဖူးပါတယ်။

```
for(register int gcc=0; gcc<G_index ; gcc++){
    int email_count =char_counting(db[gcc].email); // for email from db;
    same_counting=0;
```

ဆီမှာ ရှိတဲ့ စာလုံးအရေအတွက်ကို စစ်ပေးလိုက်ပါသည်။

```
if( counter == email_count ){
    for( register int a=0; a<counter ; a++){
        if(email[a] != db[gcc].email[a]) {
            break;
        }
        same_counting++;
    }
```

အကယ်၍ **user** ထည့်လိုက်တဲ့ **email** နဲ့ **database** ထဲက **gcc(index 0)** က

email ကသာစာလုံးအရေအတွက် အတိုင်းသာ တူညီခဲ့မယ်ဆိုပါက **email**စာလုံးတွေပါ
တူညီလားဆိုတာ အထဲကို ဝင်စစ်ပါမယ်။**email[a]** က **db[gcc].email[a]** နဲ့သာ တူညီနေပါက
same_counting ကို 1 ပေါင်းထည့်ပေးသွားမှာဖြစ်ပါတယ့်။ မတူခဲ့ဘူး
ဆိုရင်တော့ **break** ဖြစ်သွားမှာဖြစ်ပါတယ်။ ဒီနေရာမှာ **email[a]**မှာ **a** ရဲ့**index**
တန်ဖိုးက **looping** စစချင်းမှာ 0 ဖြစ်နေမှာကိုတော့ **for loop** ကို ကြည့်ပီး နား
လည်နိုင်လိမ့်မယ်လို့ ထင်ပါတယ်။

```
if( counter == same_counting){
    email_found=gcc;
}
```

အထက်ပါစာကြောင်းက ဘာကိုပြောတာလဲဆိုရင် အကယ်၍ **user**ထည့်လိုက်တဲ့
Email နဲ့ **database** ထဲမှာ ရှိနေတဲ့ **email** တွေထဲက တစ်ခုတာ တူညီနေခဲ့မယ်
ဆိုရင် **global variable** အနေနဲ့ ကြေညာထားတဲ့ **email_found** ကို **gcc** လို့
return ပြန်ထားလိုက်တာဖြစ်ပါတယ်။

Register

Email စစ်တာအတွက် ရေးပါးသွားပါက register မှာ အနည်းငယ် သွားရေးပါမယ်။

```
void rEgister(){
    char r_email[50];
    char re_email[50];
    printf("This is Online Bank Register !\n");
    email_valid=-1;
    while (email_valid== -1) {
        printf("Enter your email to register>>:");
        scanf(" %[^\n]", &re_email[0]);

        email_validation(re_email);
        if (email_valid == -1) {
            printf("Your email format was not valid\n");
        }
    }
    printf("Your email form was valid:\n");
    email_found=-1;
    email_exist_checking(re_email);
    if( email_found !=-1 ){

```

```

        printf("Your email was already register:\n");
        rEgister();
    } else{
        }
}

```

အထက်ပါ **program** သည် **register** ထဲတွင် **user** ထည့်လိုက်တဲ့ **email** ကို
စစ်ဆေးထားခြင်းဖြစ်သည်။ **program** ရှင်းလင်းချက်-> ပထမဗြို့ဆုံးအနေဖြင့် **User input**
လုပ်လိုက်တဲ့ **data** ကို **r_email** လိုက်ခေါ်တဲ့ **char array**
တစ်ခုကိုဖန်တီးထားပါတယ်။**while loop**အကြောင်းကိုတော့သိကြမယ် ထင်ပါတယ်။
ဘယ်လိုအလုပ်လုပ်လဲဆိုတော့ သူ့ထဲမှာရှိတဲ့ **condition** ကမမှားမချင်းအလုပ် လုပ်ပါတယ်။
email_valid ကို **global variable** အနေနဲ့ -1ဆိုပီးကြညာထားခဲ့ဘူးပါတယ်။ **user** ထည့်လိုက်တဲ့
email ကို

```
email_validation(re_email);
```

ကို သုံးပီးတော့စစ်လိုက်တဲ့အခါ **email_valid** ကသာ -1မဟုတ်ခဲ့ဘူးဆိုရင်

“Your email format was not valid” ဆိုပီးပေါ်လာမှာ ဖြစ်ပါတယ့်။ ထိုအကြောင်းကိုရှင်း

ပြရမဖို့ပါက ကျွန်တော်တို့ **email validation function** ဆောက်ခဲ့တုန်းက

```
if(space == 1 && format==1){
    email_valid=1;
```

Space နဲ့ **format** ကသာ 1 နဲ့သို့ခဲ့မယ်ဆိုပါက **email_valid** ကို 1 လိုက်ညာခဲ့ပါ

တယ်။ အဲဒါက ဘာကိုပြောတာလဲဆိုတော့ **true false state** လိုပါဘဲ။ ကျွန်တော်

တို့က **email_valid** ကို **global variable** မှာ -1လိုက်ညာထားခဲ့တာမို့ ငှုံးကသာ

-1ဖြစ်နေခဲ့မယ့်ရင် **true**ဖြစ်ပါး **email_validation** ထဲမှာတော့ **email format**

သာမျန်နေခဲ့မယ်ဆိုရင် 1လို့ ဆိုလိုထားခြင်ခဲ့တာဘဲ ဖြစ်ပါတယ်။

အကယူ၍ ကျွန်ုတ်တို့ထည့်လိုက်တဲ့ **email** က **format** မမှန်ခဲ့ဘူးဆိုပါက **email-valid**

က -1ဖြစ်သွားပါး “**Your email format was not valid**” ပေါ်လာခဲ့မှာ

ဖြစ်ပါတယ့်။ **while loop** ကို သုံးထားတာမို့ “Enter your email to register>>:”

နေရာကိုပြန်ရောက်သွားမှာဖြစ်ပါတယ့်။

Email format မှန်ခဲ့မယ်ဆိုရင်တော့ **email_exit_checking** ကိုထပ်ပီးစစ်မှာ

ဖြစ်ပါတယ်။ ကျွန်ုတ်တို့ထည့်လိုက်တဲ့ **email** က **database** ထဲမှာရှိပီးသားလားစစ်

တဲ့ အခါ **r_email** က **email_exit_checking** မှာ **parameter** အနေနဲ့ ဖြတ်ပီးဝင်

သွားမှာ ဖြစ်ပါတယ့်။

```
printf("Your email form was valid:\n");
email_found=-1;
email_exist_checking(re_email);
if( email_found !=-1 ){
    printf("Your email was already register:\n");
    rEgister();
} else{
}
}
```

email_found ကို **global variable** အနေနဲ့ ကြွေညာပါး -1 လို့ထားခဲ့ပါး **email_found** က

သာ -1 နဲ့သာမည့်ခဲ့ဘူးဆိုရင် “Your email was already register” လိုပေါ်လာမှာဖြစ်ကာ

rEgister () function ကို ပြန်ခေါ်ထားတဲ့မှာ **enter your email to register** နေရာကို

ပြန်ရောက်သွားမှာဖြစ်ပါတယ့်။ **email** က အသစ်ဖြစ်တယ်ဆိုရင်တော့ **else** ထဲမှာ ဆက်သွားမှာဖြစ်ပါတယ့်။

Email စစ်ပီးသွားပါဆိုရင် **nrc** ထပ်စစ်ရန် ကျွန်ုပ်သေးတယ်။ **nrc** စစ်ပို့ရန်အတွက်

Nrc_validation function တစ်ခုကို တည်ဆောက်သွားမှာ ဖြစ်ပါတယ့်။

Nrc validation

```
void nrc_validation(char nrc[20]){

    int nrc_counter = char_counting(nrc);

    for(int i=0; i<3; i++){

        twoCharArray=-1;
        compare_twoCharArray(nrc,db[i].nrc);
        if(twoCharArray ==1){
            nrc_valid=1;
            break;
        }

    }

}
```

Program ရှင်းလင်းချက်

NRC စစ်ဆေးဖို့အတွက် **void nrc_validation** ကို **nrc** လိုအမည်ပေးထားတဲ့ **char array**

တစ်ခုဖြတ်ထားလိုက်ပါတယ့်။ **user** ထည့်လိုက်တဲ့ **array** ကို ဘယ်နှစ်လုံးရှိလဲသိနိုင်ဖို့

char_counting function ကို သုံးပီးစစ်ပါမယ်။ **twoCharArray** ဆိုတဲ့ **variable** တစ်ခုကို -

1လိုထားပါး **global** မှာ ကြေညာထားလိုက်ပါမယ်။ ထိုနောက် **user** ထည့်လိုက်တဲ့ **string array** နဲ့ **database** ထဲမှာရှိတဲ့ **string** တွေကို တိုက်စစ်မှာဖြစ်ပါတယ်။ ဒီနေရာမှာ တစ်ခုသိထားရမှာက **C** မှာ **string** သီးသန့်မရှိပါဘူး။ အဲဒီအစား **char array** ကိုဘဲ **string** အနေနဲ့ အသုံးပြုကြပါတယ်။ **string** နှစ်ခုကို တိုက်စစ်ဖို့အတွက် **compare_twoCharArray** ဆိုတဲ့ **function** တစ်ခုကို တည်ဆောက်ပါမယ်။

```
void compare_twoCharArray(char first[200],char second[200]){

    int first_counter = char_counting(first);
    int second_counter = char_counting(second);

    int same_counter=0;

    if(first_counter == second_counter){

        for(register int i=0; i<first_counter; i++){

            if(first[i] != second[i]){
                break;
            }
            same_counter++;
        }

        if(first_counter == same_counter){
            twoCharArray = 1;

        }
    }
}
```

```
}
```

င်းထဲမှ **first** နဲ့ **second** ဆိုတဲ့ **char array** နှစ်ခုကို ထည့်ထားပါး ပထမ **array first** သည် **user** ထည့်လိုက်တဲ့ **nrc** ဖြစ်ပါး **second array** သည် **database** ထဲတွင်ရှိသော **nrc** များကို စစ်ရန်ဖြစ်သည်။ **parameter** အနေနဲ့ဖြတ်လာတဲ့ **array** နှစ်ခုကို **char_counting**စစ်ပါမယ့်။

User ထည့်လိုက်တဲ့ **nrc** နဲ့ **db[i].nrc** နဲ့သာ စာလုံးအရေအတွက် တူညီနေခဲ့မယ့် ဆိုရင် စာလုံးတွေတူညီသလားဆိုတာ ဝင်စစ်ပါမယ့်။

```
if(first_counter == second_counter){

    for(register int i=0; i<first_counter; i++){

        if(first[i] != second[i]){
            break;
        }
        same_counter++;
    }
}
```

First_counter နဲ့ **second_counter** တို့အလုံးအရေအတွက်တူညီနေတဲ့အခါတကယ်လို့များ

First[i] နဲ့ **second[i]** သာမတူညီခဲ့ဘူးဆိုရင် **break** ဖြစ်သွားမှာ ဖြစ်ပါးတူညီနေခဲ့မယ္ဗိုပါက

Same_counter ကို ++တိုးပေးလိုက်မှာ ဖြစ်ပါတယ်။ **first_counter** နဲ့ **samecounter** နှစ်

ခုသာတူညီနေခဲ့ပိုရင်တော့ **twoCharArray** ကို 1 လို့ **return** ပြန်လိုက်မှာ ဖြစ်ပါတယ့်။

```
if(twoCharArray ==1){
    nrc_valid=1;
    Break;
}
```

Nrc_validation ကိုပြန်သွားရပါဆိုရင် **twoCharArray** သာ1ဖြစ်ခဲ့မယ်ဆိုရင် **nrc_valid**

ကို 1လို့ **return** ပြန်ပါး **looping** ကို **break** လုပ်ပစ်လိုက်မှာ ဖြစ်ပါတယ့်။

```

printf("Enter your username:\n");
scanf(" %[^\n]",&re_name[0]);
nrc_valid=-1;

while (nrc_valid== -1){

    printf("Enter your NRC information:");
    scanf(" %[^\n]",&re_nrc[0]);

    nrc_validation(re_nrc);

    if(nrc_valid== -1){

        printf("Your NRC Format was incorrect!\n");
    }

}

printf("[+]Your NRC format was correct:\n");
printf(" del re_nrc test: %s",re_nrc);

```

ပထမစာကြောင်းမှာတော့ **username** တောင်းထားပါတယ့်။ **nrc_valid** ကို -1 ထားပါး **global** ကြေညာပေးထားပါတယ့်။ ပီးရင်တော့ **nrc** တောင်းတာကို **while loop** ထဲမှာ ထည့်ထားပါတယ်။ ထို့နောက် **NRC** စစ်ပါမယ့်။ ကျွန်ုတ်တို့**nrc_validation**မှာတူန်းက **Nrc** သာမှန်ခဲ့ရင် **nrc_valid** ကို 1လို့ **return** ပြန်ခဲ့တာကို မှတ်မိမယ်ထင်ပါတယ့်။ ဒါကြောင့်မူး
nrc_valid သာ -1 ဖြစ်ခဲ့မလို့ရင် “Your NRC Format was incorrect” ဆိုပီးပေါ်လာမှာ ဖြစ်ပါး ကျွန်ုတ်တို့ရဲ့ **NRC** နံပါတ်နဲ့အတူ “your NRC Format was correct” ဆိုတာ ပေါ်လာမှာ ဖြစ်ပါတယ့်။

လက်ရှိရောက်နေသောနေရာအထိရေးထားပီးသော **code** ၏ **github link**ဖြစ်သည်။

lesson-5->[**https://github.com/NationalCyberCityNiST/Bank-Project-With-C**](https://github.com/NationalCyberCityNiST/Bank-Project-With-C)

Strong_Password

ကျွန်တော်တို့ရှေ့က သင်ခန်းစာတွေမှာ အကောင့်ထဲဝင်ဖို့အတွက် **email** စစ်တာတွေ **Nrc**

စစ်တာတွေကို လုပ်ဆောင်ခဲ့ပီးနောက်မှာတော့ **password** ဖန်တီးတဲ့အပိုင်းကိုသွားပါမယ့်။

ကျွန်တော်တို့ **register** လုပ်တဲ့အခါမှာ **password** တောင်းတဲ့အခါ**strong password**

ဖြစ်နေဖို့လိုပါတယ့်။ ဘာလို့လဲဆိုတော့ ကျွန်တော်တို့က စကားဝါက်ကိုသာ **number** တွေ

ဘဲထည့်တာတွေ၊ **alphabet** တွေ့ကဲ ရေးမယ်ဆိုရင် **weak** ဖြစ်သွားပီး ခန့်မှန်းရလွယ်ကူ

သွားကြပါတယ့်။ ဒါကြောင့်မူး **strong** ဖြစ်သွားအောင်လို့ **number, alphabet, special**

Character နဲ့ **uppercase, lowercase** တွေပါအောင်ထည့်ပေးမှာ ဖြစ်ပါတယ်။

```
void strong_pass_validation(char pass[50]) {
```

```
    int i=0;
    int special=0;
    int number=0;
    int smallChar=0;
    int capChar=0;
```

```
    int pass_counter = char_counting(pass);
```

```
    if(pass_counter >=6) {
```

```
        while (strong_pass_valid == -1) {
            if( i == pass_counter){
                strong_pass_valid=-1;
```

```

        break;
    }
    if(pass[i] >=33 && pass[i]<=42){
        special++;
    } else if( pass[i] >=48 && pass[i]<=57){
        number++;
    } else if( pass[i]>=97 && pass[i]<=122){
        smallChar++;
    } else if(pass[i]>=65 && pass[i]<=90){
        capChar++;
    }
    i++;
}
if(special>0 && number>0 && capChar >0 && smallChar>0){
    strong_pass_valid=1;
}
}

}
} else{
    printf("[-]we need at least 6 characters!\n");
    strong_pass_valid=-1;
}
}

```

Program ရှင်းလင်းချက်

ပထမဆုံးအနေနဲ့ **strong_password_validation** ကို **pass** ဆိုတဲ့ **char array** တစ်ခုဖြတ်ပါမယ်။ပီးနောက်အောက်ပါ **Local variable** များကိုကြညာထားလိုက်ပါမယ်။

```

int i=0;
int special=0;
int number=0;
int smallChar=0;

```

```
int capChar=0;
```

ထိုနောက် User ထည့်လိုက်တဲ့ Password ကို အလုံးရောယ်လောက်ရှိလဲ ဆိုတာ

Char_counting ကို သုံးပီးစစ်ဆေးပါမယ့်။ ကျွန်တော်တို့က ဒီနေရာမှာ password

ကို ဖြည့်ခြင်းအောက်ထည့်ပါက လက်မခံဘဲ ဖြည့်အထက် သာလျှင် ရိုက်ထည့်ခွင့်ပေးထားပါ

တယ့်။

```
if(pass_counter >=6) {  
  
    while (strong_pass_valid == -1) {  
        if( i == pass_counter){  
            strong_pass_valid=-1;  
            break;  
        }  
    }  
}
```

ထိုနောက်မှာတော့ **ascii table** ကိုအသုံးပြုပါး user ထည့်လိုက်တဲ့ password ကို

Verify လုပ်ပါမယ့်။

```
if(pass[i] >=33 && pass[i]<=42){  
    Special++;
```

အထက်ပါ စာကြောင်းတွင် 33 နဲ့ 42 ကာဘာကိုပြောချင်တာလဲဆိုတော့ **special character** တွေကို

ရည်ညွှန်းတာဖြစ်ပါတယ့်။ အောက်ပါပုံကို ကြည့်ပါက နားလည်နှင့်

ပါလိမ့်မယ့်။

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0 000	000	NUL (null)	32	20 040	000	 	Space	64	40 100	000	@	Ø	96	60 140	000	`	'
1	1 001	001	SOH (start of heading)	33	21 041	001	!	!	65	41 101	001	A	A	97	61 141	001	a	a
2	2 002	002	STX (start of text)	34	22 042	002	"	"	66	42 102	002	B	B	98	62 142	002	b	b
3	3 003	003	ETX (end of text)	35	23 043	003	#	#	67	43 103	003	C	C	99	63 143	003	c	c
4	4 004	004	EOT (end of transmission)	36	24 044	004	$	\$	68	44 104	004	D	D	100	64 144	004	d	d
5	5 005	005	ENQ (enquiry)	37	25 045	005	%	%	69	45 105	005	E	E	101	65 145	005	e	e
6	6 006	006	ACK (acknowledge)	38	26 046	006	&	&	70	46 106	006	F	F	102	66 146	006	f	f
7	7 007	007	BEL (bell)	39	27 047	007	'	'	71	47 107	007	G	G	103	67 147	007	g	g
8	8 010	010	BS (backspace)	40	28 050	010	((72	48 110	010	H	H	104	68 150	010	h	h
9	9 011	011	TAB (horizontal tab)	41	29 051	011))	73	49 111	011	I	I	105	69 151	011	i	i
10	A 012	012	LF (NL line feed, new line)	42	2A 052	012	*	*	74	4A 112	012	J	J	106	6A 152	012	j	j
11	B 013	013	VT (vertical tab)	43	2B 053	013	+	+	75	4B 113	013	K	K	107	6B 153	013	k	k
12	C 014	014	FF (NP form feed, new page)	44	2C 054	014	,	,	76	4C 114	014	L	L	108	6C 154	014	l	l
13	D 015	015	CR (carriage return)	45	2D 055	015	-	-	77	4D 115	015	M	M	109	6D 155	015	m	m
14	E 016	016	SO (shift out)	46	2E 056	016	.	.	78	4E 116	016	N	N	110	6E 156	016	n	n
15	F 017	017	SI (shift in)	47	2F 057	017	/	/	79	4F 117	017	O	O	111	6F 157	017	o	o
16	10 020	020	DLE (data link escape)	48	30 060	020	0	0	80	50 120	020	P	P	112	70 160	020	p	p
17	11 021	021	DC1 (device control 1)	49	31 061	021	1	1	81	51 121	021	Q	Q	113	71 161	021	q	q
18	12 022	022	DC2 (device control 2)	50	32 062	022	2	2	82	52 122	022	R	R	114	72 162	022	r	r
19	13 023	023	DC3 (device control 3)	51	33 063	023	3	3	83	53 123	023	S	S	115	73 163	023	s	s
20	14 024	024	DC4 (device control 4)	52	34 064	024	4	4	84	54 124	024	T	T	116	74 164	024	t	t
21	15 025	025	NAK (negative acknowledge)	53	35 065	025	5	5	85	55 125	025	U	U	117	75 165	025	u	u
22	16 026	026	SYN (synchronous idle)	54	36 066	026	6	6	86	56 126	026	V	V	118	76 166	026	v	v
23	17 027	027	ETB (end of trans. block)	55	37 067	027	7	7	87	57 127	027	W	W	119	77 167	027	w	w
24	18 030	030	CAN (cancel)	56	38 070	030	8	8	88	58 130	030	X	X	120	78 170	030	x	x
25	19 031	031	EM (end of medium)	57	39 071	031	9	9	89	59 131	031	Y	Y	121	79 171	031	y	y
26	1A 032	032	SUB (substitute)	58	3A 072	032	:	:	90	5A 132	032	Z	Z	122	7A 172	032	z	z
27	1B 033	033	ESC (escape)	59	3B 073	033	;	:	91	5B 133	033	[[123	7B 173	033	{	{
28	1C 034	034	FS (file separator)	60	3C 074	034	<	<	92	5C 134	034	\	\	124	7C 174	034	|	
29	1D 035	035	GS (group separator)	61	3D 075	035	=	=	93	5D 135	035]]	125	7D 175	035	}	}
30	1E 036	036	RS (record separator)	62	3E 076	036	>	>	94	5E 136	036	^	^	126	7E 176	036	~	~
31	1F 037	037	US (unit separator)	63	3F 077	037	?	?	95	5F 137	037	_	_	127	7F 177	037		DEL

Source: www.LookupTables.com

Figure-52

Space ဆိတ္တေ အောက်နားမှာ ! ဆိတ္တေ ပုံစံလေးကနေ) အထိခိုရင် 33 ကနေ 42

အထိရှုတယ်ဆိတ္တေမြင်ရပါလိမ့်မယ်။ ထိနည်းတူစွာ 0 to 9 ဆိုရင်လဲ 48 to 57 ရှိပါ

တယ့်။ **Upper_letter** သည်လဲ A to Z အထိ 65 to 90 ရှိပါ။ **Lowercase** ဆိုရင်လဲ

a to z အထိ 97 to 122 အထိ ရှိပါတယ်။

```
if(pass[i] >=33 && pass[i]<=42){
    special++;
} else if( pass[i] >=48 && pass[i]<=57){
    number++;
} else if( pass[i]>=97 && pass[i]<=122){
    smallChar++;
} else if(pass[i]>=65 && pass[i]<=90){
    capChar++;
}
```

```

    }
    i++;
    if(special>0 && number>0 && capChar >0 && smallChar>0){
        strong_pass_valid=1;
    }
}

အထက်ပါ အတိုင်းစစ်ဆေးပါက user ထည့်လိုက်တဲ့ password က အထက်ပါ
အချက်လေးချက်လဲပါတယ့်။ ပီးရင် ဒဲလုံးလဲကျော်ပါက strong_pass_valid ကို 1
လို့ return ပြန်လိုက်တာဖြစ်ပါတယ်။ အထက်ပါ program ကို ရေးပါးပါခိုပါက
password ထည့်ဖို့ရန်အတွက် register ထဲကို ပြန်သွားပါမယ့်။

strong_pass_valid=-1;

while (strong_pass_valid== -1){

    printf("Enter your password:");
    scanf(" %[^\n]",&re_pass[0]);

    strong_pass_validation(re_pass);

    if(strong_pass_valid== -1){

        printf("[ - ]Your password format too weak and Try Again!\n");
    }
}

printf("[ + ]Your password format was correct and was saved!\n");

```

Re_pass နဲ့ password တောင်းပါက strong_pass_validation ကိုစစ်ပါမယ့်။

strong_pass_validationမှာတုန်းက strong_pass_valid ကို 1လို့returnပြန်ခဲ့တာမှို့ အခါ

၁-**Your password format too weak and Try Again!** လိုပြုမယ်။

Phone_Validation

ကျွန်တော်တိုက **password** စစ်ပီးတဲ့အခါမှာ **phone** နံပါတ်ထပ်ပီးစစ်ဆေးဖို့လိုအပ်ပါတယ့်။ ဒါကြောင့်မို့အောက်ပါ **function** ကို ရေးသားပါမယ့်။

```
void phone_validation(unsigned int phone){
    int phone_counter=0;
    for(int i=0; i<G_index ; i++){
        if(phone != db[i].phoneNumber){
            phone_counter++;
        } else{
            phone_valid=-1;
            break;
        }
    }
    if(phone_counter == G_index){
        phone_valid=1;
    }
}
```

အထက်ပါ**program** မှာ ဖြတ်လာတဲ့ **parameter** ကို **unsigned int** နဲ့ ကြညာပေး

ထားပါတယ်။ ပို့တော့ဖုန်းနံပါတ်တွေမှာ နိုင်ငံပေါင်းစုံရှိတာမို့ ကျွန်တော်တို့နိုင်ငံရဲ့ ဖုန်းနံ

ပါတ်ဆိုရင် +959 နဲ့ ရေးပါမယ်။ ပထမဆုံးအနေနဲ့ **database** ထဲမှာ ကျွန်တော်တို့ထည့်

လိုက်တဲ့ ဖုန်းနံပါတ်က ရှိပိုးသားလား မရှိသေးဘူးလားဆိုတာစစ်ဖို့လိုပါမယ့်။ **G_index**

အတိုင်း **looping** ပါတ်ပါမယ့်။ ထို့နောက် **database** ထဲက **db[i].phone number** နဲ့ **user**

ထည့်လိုက်တဲ့ဖုန်းနံပါတ်ကို တိုက်စစ်ပါမယ့်။ ဖုန်းနံပါတ်တွေအကုန်လုံးက user ထည့်လိုက်တဲ့ ဖုန်းနံပါတ်နဲ့သာ မတူညီခဲ့ဘူးဆိုရင် Phone_counter ကို 1လို့ return ပြန်ပါ။ ပါးတူညီနေခဲ့ပါက -1လို့ return ပြန်လိုက်မှာ ဖြစ်ပါတယ့်။ ပီးရင်တော့ register ကို ပြန်သွားပါး phone number စစ်ဆေးတာအတွက် စာအနည်းငယ်ရေးကြပါမယ့်။

```

phone_valid=-1;
while (phone_valid== -1){
    printf("[X]Enter your Phone Number:");
    scanf("%u",&re_phone);

    phone_validation(re_phone);

    if(phone_valid == -1){

        printf("[-]Your phone number is already registered:\n");
    }

}

printf("[+] Your phone is correct and saved!\n");

re_phone ဆိုတဲ့ character array တစ်ခုကို ဖန်တီးထားပါမယ့်။ ထို့နောက် re_phone
နဲ့ data လက်ခံပါးနောက် phone_validation လုပ်ပါမယ်။ ထို့နောက် phone_valid သာ -1နဲ့
ညီ့မယ်ဆိုရင် Your phone number is already registered လို့ပြောပါးwhile
loop ကို ဆက်ပါးပါတ်စေပါမယ့်။ အကယ်၍ -1မဖြစ်ခဲ့ဘူးဆိုရင် printf("[+] Your phone is
correct and saved!\n"); ဒီစာကြားကို အလုပ်လုပ်စေပါး while loop ကို ရပ်ပစ်လိုက်ပါမယ်။
လက်ရှိနေရာအထိ ရေးထားပါးသော code ၏ github link
```

ဖြစ်သည်။

lesson-6->[**Encryption_key**](https://github.com/NationalCyberCityNiST/Bank-Project-With-C)

Phone number တောင်းပါးတဲ့အခါ **Encryption key** တောင်းတဲ့ **function** ကို ဆက်

ပီးရေးသားကြပါမယ်။ **Encryption key** ကဘာကိုပြောတာလဲဆိုတော့ ကျွန်တော်တို့

ထည့်လိုက်တဲ့ **data** ကို **Encrypt** လုပ်ဖို့ရန်အတွက် **user** တစ်ဦးချင်းဆီတိုင်းကို သူ

တို့ရဲ့ကိုယိုင် **Key** ထားပေးထားမှာ ဖြစ်ပါတယ့်။ ကျွန်တော်အနေနဲ့ **encryption key**

ကို အနည်းဆုံး ငါလုံးကနေ အများဆုံး ဗြိုလ်ခိုင်းစေမှာ ဖြစ်ပါတယ်။

```

while (eKey_valid==-1){
    printf("[X]Enter your Encryption Key(from 4 to 6 char):");
    scanf(" %[^\n]",&re_encrypt_key[0]);
    encryption_key_validation(re_encrypt_key);

    if(eKey_valid==-1){
        printf("[-]Your Encryption Key do not correspond with our
pattern:\n");
    }

}
printf("[+]Your Encryption key valid and saved!\n");
}

```

အထက်ပါ Program အတိုင်း **encrypt key** ငါလုံးတောင်းပါမယ်။ ထိုနောက်မှာတော့

Encryption_kry_validation functionတည်ဆောက်ကပါး **user** ကို **encrypt**

Code ငလုံးဘဲထည့်လားဆိုတာ အောက်ပါအတိုင်း စစ်ဆေးပါမယ်။

```
void encryption_key_validation(char eKey[30]){

    int eKey_counter = char_counting(eKey);

    if( eKey_counter >=4 && eKey_counter<=6){
        eKey_valid=1;
    }

}
```

ထိုနောက်မှာတော့ **recovery key** ကိုဆက်ပီးရေးသားကြပါမယ့်။ အဲဒီ **recovery key** က
တစ်ချိန်လုံး **User** ကအမြတ်မှတ်မိနေရပါမယ့်။ ထိုနောက်မှာတော့ ကျွန်ုတ်တို့
program ရေးသားကြပါမယ်။

```
while (reKey_valid==1){

    printf("[X]Enter your Recover Key carefully:");
    scanf(" %[^\n]",&re_recovery_key[0]);

    recovery_key_validation(re_recovery_key);

    if(reKey_valid==1){

        printf("[-]Your recovery Key do not correspond with our
pattern:\n");
    }
}

printf("[+]Your recover key was saved!\n");

}
```

recovery key ကိုလဲ ဂဏန်း ငါလုံး ကနေ ဖြလိုးသဲထည့်ခိုင်းပါမယ့်။ ထို့နောက်

Key စစ်ဆေးဖို့ရန်အတွက် ငါ**recovery_key_validation function** ကို ဆောက်ပါမယ့်။

```
void recovery_key_validation(char reKey[30]){
    int reKey_couunter = char_counting(reKey);

    if(reKey_couunter >=6 && reKey_couunter<=10){
        reKey_valid=1;
    }

}
```

တကယ်လို့ **rekey_valid** သာ -1နဲ့သာညီနေခဲ့မယ်ဆိုရင် Your recovery Key do not correspond with our pattern ဆိုတဲ့စာကြောင်းကို ထုတ်ကပြမှာ ဖြစ်ပါတယ့်။

တကယ်လို့ **recovery key** ကို **format** မှန်မှန် ထုတ်ပြနိုင်ရင်တော့ Your recover key was saved ကိုထုတ်ပြပါး **register** လုပ်ဖို့ရန်လိုအပ်တဲ့ တဗြားအကြောင်းရာတွေ ဆက်လက်ရေးသားမှာ ဖြစ်ပါတယ်။

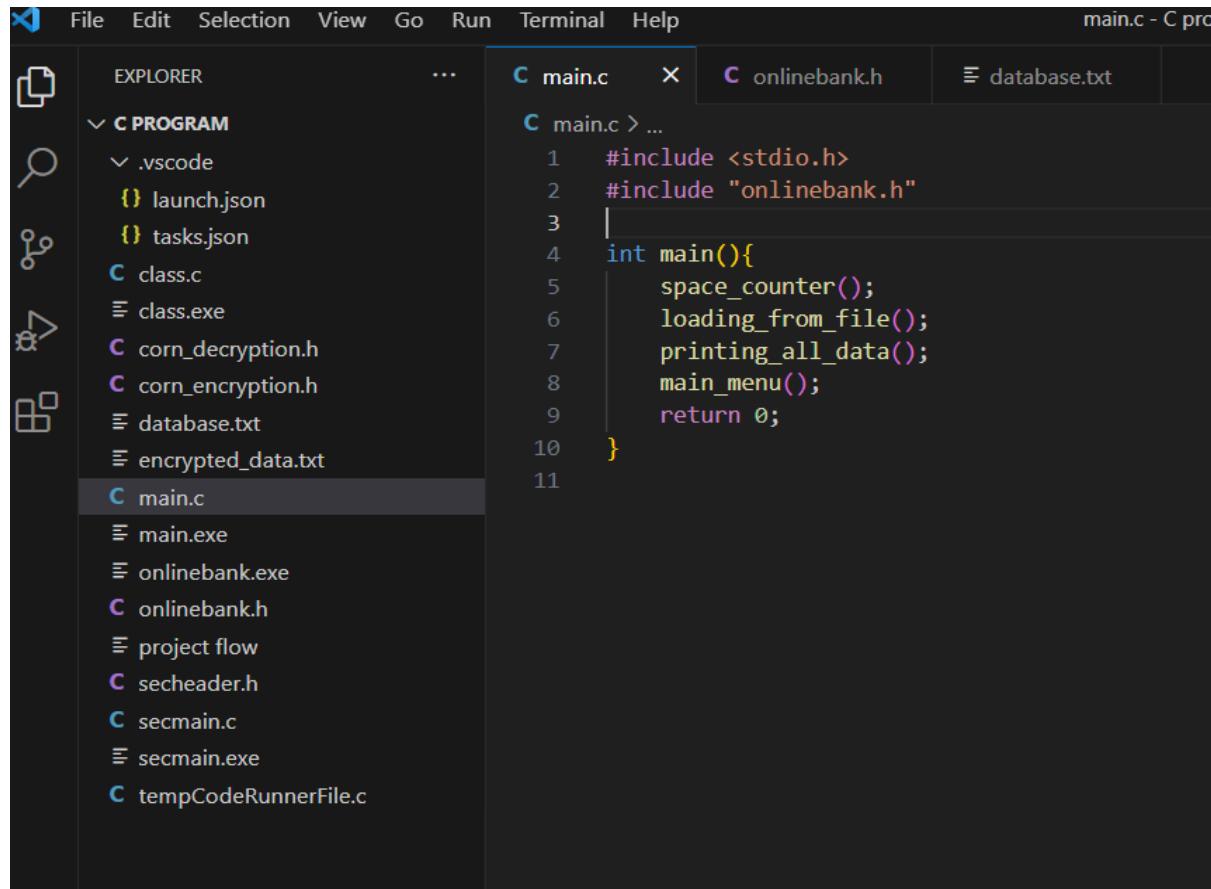
လက်ရှိရေးသားပီးသောနေရာအထိ ပီးထားသော **code** နဲ့**github link** ဖြစ်သည်။

lesson-7-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson%207>

ထို့နောက် တဗြားကျွန်ရှိနေသော **code** များကို ရေးသားကြပါမယ်။ **recovery key** အတွက်ရေးသားပီးနောက် **struct info** ထဲတွင်ကျွန်ရှိနေသော ကျွန်တဲ့ အချက်တွေကို

ထပ်ပီးရေးသားကြပါမယ့်။ လောလောဆယ် အဲဒီကျွန်ုတဲ့ အချက်မရေးပီးခင် အရင်

ရေးထားပီးသော **code** များအလုပ်လုပ်သလားဆိုတာစမ်းကြည့်ကြပါမယ့်။



The screenshot shows the Visual Studio Code interface with a dark theme. The left sidebar (Explorer) lists files and folders related to a C program, including .vscode, launch.json, tasks.json, class.c, class.exe, corn_decryption.h, corn_encryption.h, database.txt, encrypted_data.txt, main.c, main.exe, onlinebank.exe, onlinebank.h, project flow, secheader.h, secmain.c, secmain.exe, and tempCodeRunnerFile.c. The main editor area displays the main.c file:

```

1 #include <stdio.h>
2 #include "onlinebank.h"
3 |
4 int main(){
5     space_counter();
6     loading_from_file();
7     printing_all_data();
8     main_menu();
9 }
10
11

```

figure-53

အထက်ပါပုံကတော့ **main.c** ထဲမှာ ရှိတဲ့ **code** များဖြစ်ပါတယ့်။ အောက်က **link** ကတော့

Onlinebank.h ထဲက **code** ၏ **github link** ဖြစ်ပါတယ်။

<https://github.com/NationalCyberCityNiST/Bank-Project-With-C/blob/main/lesson%207/onlinebank.h>

ကျွန်ုတ်တို့ **Program** ၏ **run** တဲ့အခါ **main.c** ကနေ **run** ရပါမယ်။

```

File Edit Selection View Go Run Terminal Help
EXPLORER C PROGRAM ...
  .vscode
    launch.json
    tasks.json
  class.c
  class.exe
  corn_decryption.h
  corn_encryption.h
  database.txt
  encrypted_data.txt
  main.c
  main.exe
  onlinebank.exe
  onlinebank.h
  project flow
  sechader.h
  secmain.c
  secmain.exe
  tempCodeRunnerFile.c
C main.c x C onlinebank.h database.txt
C main.c >...
1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     space_counter();
6     loading_from_file();
7     printing_all_data();
8     main_menu();
9
10 }
11

```

Figure-54

အထက်ပါပုံရဲ့ညာဘက်က **Run Code** ကို နိုင်လိုက်ပါက အောက်ပါပုံအတိုင်း ပေါ်လာပါက

loading_from_file နဲ့ **printing_all_data** မှာ ပြသုနာ မရှိသေးပါဘူး။

```

File EDIT DOCUMENTS VIEW GO RUN TERMINAL HELP
EXPLORER C PROGRAM ...
  .vscode
    launch.json
    tasks.json
  class.c
  class.exe
  corn_decryption.h
  corn_encryption.h
  database.txt
  encrypted_data.txt
  main.c
  main.exe
  onlinebank.exe
  onlinebank.h
  project flow
  sechader.h
  secmain.c
  secmain.exe
  tempCodeRunnerFile.c
C main.c x C onlinebank.h database.txt
C main.c >...
1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     space_counter();
6     loading_from_file();
7     printing_all_data();
8     main_menu();
9
10 }
11

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS C:\Users\WSI\Desktop\C program> cd "C:\Users\WSI\Desktop\C program" ; if ($?) { gcc main.c -o main } ; if ($?) { ./main }
1-winhtut-9/paoola(N)80099-winhtutline@gmail.com-admin123-2208183876-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.000000-pynoolwin-onlinebankad
min
2-winhtut-9/paoola(N)80099-winhtutline@gmail.com-admin123-2208183875-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.000000-pynoolwin-onlinebankad
min
3-winhtut-9/paoola(N)80099-winhtutline@gmail.com-admin123-2208183874-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.000000-pynoolwin-onlinebankad
min
Welcome to our bank
Press 1 to login
Press 2 to Register
Press 3 to Exit

```

Figure-55

Terminal မှ 1 ကို နိုင်လိုက်ပါက **login** နေရာကို ရောက်သွားမှာ ဖြစ်ပါတယ့်။ ဒါပေမယ့်။

ကျွန်တော်က **register** အတွက်ဘဲပြောမှုမှာ 2 ကို နိုင်လိုက်တဲ့အခါ

This is Online Bank Register !

Enter your email to register>>:

အဲလိုပေါ်လာပီး **register** လုပ်ဖို့ရန်အတွက် **email** တောင်းပါမယ်။ အဲဒီမှာ ကျွန်တော်က

swan@gmai ဘဲထည့်လိုက်တဲ့ အခါအောက်ပါပုံအတိုင်းပေါ်လာမှာဖြစ်ပါတယ့်။အကြောင်း

အရင်းကတော့ **email format** ကို မှားထည့်မိလိုက်လိုဖြစ်ပါတယ့်။

Your email format was not valid

Enter your email to register>>:

The screenshot shows a terminal window with the following content:

```
PS C:\Users\MSI\Desktop\C program> cd "C:\Users\MSI\Desktop\C program\" ; if ($?) { gcc main.c -o main } ; if ($?) { ./main }
1-wihtut-9/paool(1)80099.wihtut@gmail.com-admin123-2208183876-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.000000-pyinoolwin-onlinebankadmin
2-wihtut-9/paool(1)80099.htutline@gmail.com-admin123-2208183875-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.000000-pyinoolwin-onlinebankadmin
3-wihtut-9/paool(1)80099-winline@gmail.com-admin123-2208183874-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.000000-pyinoolwin-onlinebankadmin
Welcome to our bank
Press 1 to login
Press 2 to Register
Press 3 to Exit
2
This is Online Bank Register !
Enter your email to register>>:win@mai
Your email format was not valid
Enter your email to register>>:|
```

Figure-56

ထိနေက် ကျွန်တော်တိုက **email format** ကို အမှန်ထည့်ပါး **database** ထဲမှ **register** လုပ်

ပီးထားတဲ့ email နာမည်ကို ရိုက်ထည့်ကြည့်ပါမယ့်။ winline@gmail.com ကို ရိုက်ထည့်

ကြည့်လိုက်မယ်ဆိုရင်

Your email form was valid:

Your email was already registered.

This is Online Bank Register !

Enter your email to register>>:

ထပ်ထည့်မိတာ ဖြစ်ပါတယ်။

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code + □  
  
19 19 19 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0  
1-winhtut-/paool(N)80099-winhtut@gmail.com-admin123-2208183876-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pyinolwin-onlinebankadmin  
2-winhtut-/paool(N)80099-htlntlg@gmail.com-admin123-2208183875-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pyinolwin-onlinebankadmin  
3-winhtut-/paool(N)80099-wlinle@gmail.com-admin123-2208183874-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pyinolwin-onlinebankadmin  
Welcome to our bank  
Press 1 to login  
Press 2 to Register  
Press 3 to Exit  
2  
This is Online Bank Register !  
Enter your email to register>:wir@mai  
Your email format was not valid  
Enter your email to register>:wlinle@gmail.com  
Your email form was valid:  
Your email was already register:  
This is Online Bank Register !  
Enter your email to register>:■
```

Figure-57

ကျွန်တော်တိုက ထပ်ခါထပ်ခါ မှားပီးရိုက်ထည့်မယ်ဆိုရင် **Enter your email to register** ဆိုကိုဘဲ

ပြန်ရောက်သွားမှ ဖြစ်ပါတယ်။ အဲဒီတော့ ကျွန်တော်တိ **email format** အမျိန်ထည့်

ထည့်ပါမယ့်။ nccuniversity@gmail.comလို့ ရိုက်ကြည့်လိုက်မယ်ဆိုရင် ကျွန်တော်တိုက်

Email ကို မှန်မှန်ကန်ကန် ထည့်လိုက်ပါမိမိလို **your email was saved** ဆိုပါတယ်

တော်တို့က NRC တောင်းတဲ့ နေရာကိုရောက်သွားမှာ ဖြစ်ပါတယ်။ 9/paoola(N)80099 ငါး NRC ကို

ထည့်လိုက်တဲ့အခါမှာတော့ **password** ထည့်တဲ့ နေရာကို ရောက်သွားမှာ ဖြစ်ပါ

ତାର୍ଯ୍ୟ

[+] Your NRC format was correct:

del re_nrc test: 9/paoola(N)80099

Enter your password:

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Code + v  
2  
This is Online Bank Register !  
Enter your email to register>>:win@mai  
Your email format was not valid  
Enter your email to register>>winlin@gmail.com  
Your email form was valid:  
Your email was already register:  
This is Online Bank Register !  
Enter your email to register>>:nccuniversity@gmail.com  
Your email form was valid:  
Del re_email nccuniversity@gmail.com  
Your email was saved:  
Enter your username:  
winthut  
Enter your NRC information:9/paoola(N)80099  
[+]Your NRC format was correct:  
del re_nrc test: 9/paoola(N)80099Enter your password:|
```

Figure-58

Password မှာဆိုရင် **strong password** နဲ့ **weak password** ဆိုပါတယ်။ **strong password** မှာ ဆိုရင် အချက်ငှာက်ရှိပါး အထက်ပါအချက်တွေနဲ့ ညီအောင်မထည့်သေးဘဲ **Weak password** ဘဲအရင်ထည့်ပါ မယူ။ **123hn** နဲ့ **1234hni** ကိုထည့်ကြည့်တဲ့အခါ ပထမ တစ်ခုအတွက်က

[-]we need at least 6 characters!

[-]Your password format too weak and Try Again!

ဆိုပေါ်လာမှာ ဖြစ်ပါတယ့်။ နောက်တစ်ကြိမ် **1234hni**ဆိုပါးထည့်မထွေ့ရင် ကျွန်တော်တို့ **password** ကဲ **strong password format** နဲ့ မည့်တာမှာ

Enter your password:1234hni

[-]Your password format too weak and Try Again!

Enter your password:

နောက်တစ်ကြိမ် **password** ကို ထည့်ခိုင်းပါလိမ့်မယ်။ ကျွန်တော်တို့အနေနဲ့ **passsword**

ကို **format** နှင့်အညီ ထည့်လိုက်တဲ့ အခါမှာတော့ **phone number** တောင်းတဲ့ နေရာကို

ရောက်သွားမှာ ဖြစ်ပါတယ်။ ဒီနေရာမှာ ဖုန်းနံပါတ်တဲ့နေရာမှာ **959** ခံပါးထည့်ပါမယ့်။

ထည့်လိုက်တဲ့ ဖုန်းနံပါတ်က **database** ထဲမှာ မရှိဘူးဆိုရင် **encryption key** တောင်းတဲ့

နေရာဆီ ခုန်ကူးသွားမှာ ဖြစ်ပါတယ်။

[X]Enter your Phone Number:959798652467

[+] Your phone is correct and saved!

[X]Enter your Encryption Key(from 4 to 6 char):

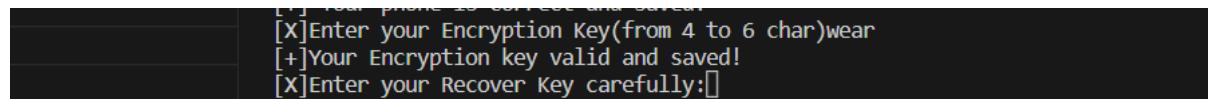
> OUTLINE > TIMELINE	Enter your password:kkjd@1@845jj [+]Your password format was correct and was saved! [X]Enter your Phone Number:959798652467 [+] Your phone is correct and saved! [X]Enter your Encryption Key(from 4 to 6 char):
---	--

Figure-59

ကျွန်တော်တို့အနေနဲ့ **encryption key** ကို **function** ထဲမှာ သတ်မှတ်ထားသလိုမျိုးဘဲ **key 4**

လုံးကနေ 7 လုံးအထိဘဲ တောင်းယူပါမယ်။ ကျွန်တော်က **wear** ဆိုတဲ့**keyword** ကို ထည့်ပါး

Database ထဲမှာ သိမ်းဆည်းထားလိုင်ကြပါတယ်။

**Figure-60**

ထို့နောက် **recovery key** ကို ထည့်ပါမယ့်။ **recovery key** ကို **wear12**ဆိုပါးထည့်လိုင်တဲ့ အခါ

X]Enter your Recover Key carefully:wear12

[+]Your recover key was saved!

ကျွန်တော်တို့ **program** က မှန်နေပါတယ်။ ဒါကြောင့်မို့လို့ ကျွန်ရှိနေတဲ့**features** တွေကို ဆက်လက်ရေးသားပီးတော့ **run** ကြည့်ကြပါမယ်။ ပထမဦးဆုံးအနေနဲ့ ကျွန်တော်တို့ က **mmk** လား **dollar** လားဆိုတာမေးပါမယ်။ 1 ကိုနို့ပို့ရင် **Personal** အကောင့်ဖြစ်သွားမှာ ဖြစ်ပါး 2 ကိုနို့ပို့ရင်တော့ **Business** ဖြစ်မှာ ဖြစ်ပါတယ်။ ပီးရင် **acc** ထဲကိုပောက်လာတယ်လောက်ထဲ ည့်ချင်လဲဆိုတာမေးပါမယ်။ ပီးရင်လစွဲဝင်ငွေကိုထည့်ပါမယ်။ ထို့နောက်မှာတော့ ကျွန်တော်တို့ **address** ကိုထည့်ပါမယ်။ ပီးရင် **user** တစ်ယောက်တိုးလာတိုင်း **id number** တိုးလာအောင် **Gindex** ကို +1 ပေါင်းပေးပါမယ့်။

```
printf("[X]Enter your currency to use:");
scanf(" %[^\n]",&re_currency[0]);

printf("[X]Enter your amount:");
scanf(" %llu",&re_current_amount);
```

```

printf("[X]Enter your monthly income:");
scanf(" %u",&re_monthly_income);
printf("[X]Enter your address:");
scanf(" %[^\n]",&re_address[0]);
printf("[X]Enter your opening note:");
scanf(" %[^\n]",&db[G_index].tr[0].note[0]);
db[G_index].id = G_index+1;

```

ဒါပေမယ့် တချို့ **status** ထွေ ထည့်ဖို့ကျွန်ပါသေးတယ်။ ကျွန်တော်တို့အရင်က

copy_twoCharArray function ကို ရေးပီးပိုလိုထင်ပါတယ်။

```
void copy_two_char_array(char receiver[200] ,char transmitter[200] ){
```

```
    int transmit_counter =char_counting(transmitter);
```

```
    for(int i=0; i<transmit_counter; i++){
```

```
        receiver[i] = transmitter[i];
```

```
}
```

```
}
```

ငှုံးက ဘာကိုပြောတာလဲဆိုတော့ **array** ထဲမှာ ရှိတဲ့ **data** ကို နောက်ထပ်**array**တစ်ခုထဲကို

ထည့်လိုက်ခြင်းဖြစ်ပါတယ်။ ဒီနေရာမှာ ကျွန်တော်တို့ က **user** ထည့်လိုက်တဲ့ **data**တွေကို

လက်ခံယုံဘဲရှိသေးပါး **database** ထဲကို မရောက်သေးဘူးဖြစ်နေပါတယ့်။ ဒါကြောင့်မှာ

ဒီ**function** ကို သံဃားပါး **copy** ကူးထည့်မှာ ဖြစ်ပါတယ့်။

```
copy_two_char_array(db[G_index].email,re_email);
```

```
copy_two_char_array(db[G_index].nrc,re_nrc);
```

```

copy_two_char_array(db[G_index].name,re_name);

copy_two_char_array(db[G_index].password,re_pass);
db[G_index].phoneNumber = re_phone;

```

db[G_index].current_amount = re_current_amount;
Re_current_amount ကို ကျန်တော်တိအစပိုင်းမှာတော်နဲ့က **user** ကို သတ်မှတ်ခြင်း
 ထားခဲ့ပါပဲ။ဖြစ်တာမို့ ထပ်ပီး **copy** ကူးစရာမလိုတော့ဘူး ဖြစ်ပါတယ်။

```

db[G_index].monthly_income = re_monthly_income;

copy_two_char_array(db[G_index].address,re_address);
//copy_two_char_array(db[G_index].tr[0].note , re_note);

```

```

copy_two_char_array(db[G_index].account_status,db[2].account_status);
db[G_index].account_level=db[2].account_level;
db[G_index].account_type = db[2].account_type;
db[G_index].minimum_opening_deposit=db[2].minimum_opening_deposit;
copy_two_char_array(db[G_index].loanStatus,db[2].loanStatus);
db[G_index].loan_amount = db[2].loan_amount;
db[G_index].loan_rate = db[2].loan_rate;

```

```
space_array[G_index]=19;
G_index++;
```

ထိုနောက် **printing_all_data** ကို သုံးပါ။ **database** ထဲကို **data** တွေထဲတိပြပါတော့မယ်။

```
printing_all_data();
main_menu();
```

Program နောက်တစ်ကြိမ်ထပ် **run** ကြည့်ပါ။ **data** ထည့်ကြည့်တဲ့ အခါ အောက်ပါအတိုင်း

Terminal မှ ပေါ်လာရင် ကျွန်တော်တို့ **program** က **data** တွေထဲတိပိုးသွားပါပဲ။

```
[X]Enter your amount:10000
[X]Enter your monthly income:100000
[X]Enter your address:mandalay
[X]Enter your opening note:hello
1-winhtut-9/paoola(N)80099-winhtut@gmail.com-admin123-2208183876-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pynoolwin-onlinebankadmin
2-winhtut-9/paoola(N)80099-hutline@gmail.com-admin123-2208183875-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pynoolwin-onlinebankadmin
3-winhtut-9/paoola(N)80099-winline@gmail.com-admin123-2208183874-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pynoolwin-onlinebankadmin
4-ohthisis-9/paoola(N)80099-eart@gmail.com-rre@#12KO-1510158880-wear-wear123-personal-1-1-10000-mmk-20000-T-200000-1-0.00000-mandalay-hello
5-name-9/paoola(N)80099-wer@gmail.com-kh2#F-979644994-->personal-1-1-10000-mmk-10000-T-100000-1-0.00000-mandalay-hello
Welcome to our bank
Press 1 to login
Press 2 to Register
Press 3 to Exit
```

figure-61

ငွေယောက်မြောက်နဲ့ ငွေယောက်မြောက်က **user registration** လုပ်လိုက်တဲ့ **data** တွေ ဖြစ်ပါတယ်။

တယ်။

```
EXPLORER PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
C PROGRAM .vscode launch.json tasks.json class.c class.exe corn.decryption.h corn.encryption.h database.txt encrypted.data.txt main.c main.exe onlinebank.exe onlinebank.h project flow seche.c secheader.h secmain.c secmain.exe tempCodeRunnerFile.c

PS C:\Users\WSI\Desktop\C program cd "C:\Users\WSI\Desktop\C program\" ; if ($?) { gcc main.c -o main } ; if ($?) { ./main }

1-winhtut-9/paoola(N)80099-winhtut@gmail.com-admin123-2208183876-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pynoolwin-onlinebankadmin
2-winhtut-9/paoola(N)80099-hutline@gmail.com-admin123-2208183875-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pynoolwin-onlinebankadmin
3-winhtut-9/paoola(N)80099-winline@gmail.com-admin123-2208183874-aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pynoolwin-onlinebankadmin
4-ohthisis-9/paoola(N)80099-eart@gmail.com-rre@#12KO-1510158880-wear-wear123-personal-1-1-10000-mmk-15000-T-5000-1-0.00000-pynoolwin-onlinebankadmin
5-name-9/paoola(N)80099-wer@gmail.com-kh2#F-979644994-->personal-1-1-10000-mmk-10000-T-100000-1-0.00000-mandalay-hello
Welcome to our bank
Press 1 to login
Press 2 to Register
Press 3 to Exit
```

Figure-62

အထက်ပါအားလုံး ရေးသွားပိုးသွားပါက ဤစာအုပ်ကို လွှဲလာနေသူတို့သည် **registration**

section ကို ရေးသားပိုးကြပါပါ။

လက်ရှိနေရာအတိရေးသားပိုးသော **code** ၏ **github link** ဖြစ်သည်။

lesson-8-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson-8>

Login Function

ကျွန်တော်တို့ **register** အတွက်ရေးပိုးပါက **Login** အတွက် ဆက်လက်ရေးသားကြပါမယ့်။

ကျွန်တော်တို့အနေနဲ့ **user** တစ်ယောက်က **Login** ဝင်တော့မထိခို့ရင် ပထမဦးဆုံးအနေနဲ့ **email** နဲ့

password တောင်းကြပါတယ်။ **user** ထည့်လိုက်တဲ့ **email** နဲ့ **Password** ကသာ

Database ထဲမှာရှိနေတယ်ဆိုရင် **login success** ဖြစ်သွားပါး တွေ့ခြား**status** တွေ့ကို အသုံး

ပြုလို့ရမှာ ဖြစ်ပါတယ်။

```
char l_email[50];
char l_pass[50];
email_found=-1;
twoCharArray=-1;
while (email_found == -1 || twoCharArray== -1) {
    printf("This is Online Bank Login!\n");
    printf("Enter your email to login!>>:");
    scanf(" %[^\n]", &l_email[0]);
    printf("Enter your password to login!>>:");
    scanf(" %[^\n]", &l_pass[0]);

    email_exist_checking(l_email);
    compare_twoCharArray(db[email_found].password,l_pass);
```

```

if(email_found == -1 || twoCharArray== -1){
    email_found=-1;
    twoCharArray=-1;
    printf("Your Login credential incorrect!\n");
}

}

```

user_sector();
Program ရှင်းလင်းချက်

I_email နဲ့ **I_pass** ဆိုတဲ့ **char array** နှစ်ခုကြညာပေးထားပါတယ်။ ထိုနောက် **user** ကို
Email နဲ့ **password** ထည့်ခိုင်းပါး **email_exit_checking** နဲ့ စစ်ကြည့်လိုက်လို့ **email** လဲ
Database ထဲရှိနေမယ် **password** ကို အဲဒီ **email** ထဲက **password** ဖြစ်နေခဲ့မယ္ဗိုရင်
login အောင်မြင်သွားပါး **profile** ကို ရောက်သွားပါမယ်။ **Profile** အတွက် **function** ကို
User_sector လို့ အမည်ပေးထားပါတယ်။

```

if(email_found == -1 || twoCharArray== -1){
    email_found=-1;
    twoCharArray=-1;
    printf("Your Login credential incorrect!\n");
}

```

```

tempCodeRunnerFile.c

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
1:winhtut@poola(0)~$00999-winhtut@gmail.com-admin123-2208183876-aaa123wh-aaa123wh-personal-1-1-10000-mmk_15000-T-5000-1-0-000000-pyinoolwin-onlinebankadmin
2:winhtut@poola(0)~$00999-htruline@gmail.com-admin123-2208183875-aaa123wh-aaa123wh-personal-1-1-10000-mmk_15000-T-5000-1-0-000000-pyinoolwin-onlinebankadmin
3:winhtut@poola(0)~$00999-winhtut@gmail.com-admin123-2208183874-aaa123wh-aaa123wh-personal-1-1-10000-mmk_15000-T-5000-1-0-000000-pyinoolwin-onlinebankadmin
Welcome to our bank
Press 1 to Login
Press 2 to Register
Press 3 to Exit
1
This is Online Bank Login!
Enter your email to login!>:winhtut@gmail.com
Enter your password to login!>:admin123
Welcome M/s : winhtut
Your Current Amount : 15000
PS C:\Users\MSI\Desktop\C program>

```

Figure-63**User_sector function**

ကျွန်တော်တို့၏ **function** မှာတော့ **user** အနေနဲ့ ငွေထုတ်တာတွေ၊ ငွေလွှာတာတွေ၊ ငွေသွင်းတာတွေကို အခိုကထားရေးသွားမှာ ဖြစ်ပါတယူ။

```
void user_sector(){
    printf("Welcome Mr/s : %s\n",db[email_found].name);
    printf("Your Current Amount : %llu\n",db[email_found].current_amount);
    char user_option[2];

    printf("Press 1 To Transfer Money:\nPress 2 to Withdraw :\n");
    printf("Press 3 To Cash In:\nPress 4 to get your Transaction
Record :\n");
    printf("Press 5 To Loan:\nPress 6 to get Main Menu:\n");
    printf("Press 7 To Exit:\nEnter your option:");
    scanf(" %[^\n]",&user_option[0]);

    int option = check_input(user_option);

    if(option == 49){
        transfer_money();
    } else if(option==50){
        user_withdraw();

    } else if(option==55){
        recording_all_data_to_file();
        main_menu();
    }
}
```

```
}
```

User အနေနဲ့ **profile** ထဲရောက်သွားတဲ့ အခါ **Press-1** ကို နိုပ်ခဲ့မယ်ဆိုရင် **to transfer** တွေား**button** တွေကို နိုပ်ရင်တော့ သူနဲ့ သတ်ဆိုင်ရာ နေရာတွေဆီရောက်သွားမှာ ဖြစ်ပါတယ့်။

```
int option = check_input(user_option);
```

Option ကသာ **49**နဲ့ ညီရင် ဆိုတာက ဘာကို ပြောတာလဲဆိုတော့ **49**က **ascii value** နဲ့ ဆိုရင် **1**ဖြစ်တာမို့ **user** ကသာ **1** ကိုနိုပ်လိုက်မယ်ဆိုရင် **transfer money** ကိုရောက်သွားမှာ ဖြစ်ပါတယ်။ **2** ကိုနိုပ်ရင်တော့ **user_withdraw** ကိုရောက်သွားမှာ ဖြစ်ပါတယ့်။
လောလောဆယ် **transfer_money** အတွက် **function**ကို အရင်ရေးပါမယ်။

```
void transfer_money(){

    unsigned int amount_toTrans=0;
    unsigned int receiver_phone=0;
    char trans_pass[50];
    phone_found=-1;
    while (phone_found== -1) {
        printf("Enter receiver phone number>>:");
        scanf("%u", &receiver_phone);
        phone_number_finding(receiver_phone);

        if(phone_found== -1){
            printf("This is phone number is Not found in our bank\n");
        }
    }

    while (1) {
        printf(" Enter amount to send for %s : %s >>:",

```

```
db[phone_found].name, db[phone_found].email);  
  
    scanf("%u", &amount_toTrans);  
  
    if(amount_toTrans < db[email_found].current_amount-1000){  
        break;  
    }  
}  
twoCharArray=-1;  
int wrong_counter=0;  
while (twoCharArray== -1){  
    printf("Your current amount %llu\nTransfer  
amount %u :\n",db[email_found].current_amount,amount_toTrans);  
    printf("Enter your password to confirm for transaction>>:");  
    scanf(" %[^\n]",&trans_pass[0]);  
  
    compare_twoCharArray(db[email_found].password,trans_pass);  
    if(twoCharArray== -1){  
        wrong_counter++;  
        if(wrong_counter==3){  
            transfer_money();  
            break;  
        }  
    }  
}  
}
```

Program ရှင်းလင်းချက်

ကျွန်တော်တို့က ငွေလွှဲချင်တဲ့ အခါမှာ **Phone number** အသံးပြုကြပါတယ်။ အဲဒီတော့
ပထမဆုံးအနေနဲ့ ကျွန်တော်တို့ ငွေလွှဲချင်တဲ့ **Phone_number** ကို ရိုက်ထည့်ပါး အဲဒီ **phone**
နံပါတ်နဲ့ အကောင့်ဟာ အကောင့်ရှိပါးသားဆိုရင် ငွေလွှဲလို့ ရမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် ဖုန်း
နံပါတ် စစ်ဆေးတဲ့ **function** ဆောက်ပေးရပါမယ်။

```
void phone_number_finding(unsigned int to_find_ph){

    for(int i=0; i<G_index; i++){

        if(to_find_ph == db[i].phoneNumber){
            phone_found=i;
            break;
        }
    }

}
```

အထက်ပါ **function** သည် **phone** နံပါတ်စစ်ဆေးသည့် **function** ဖြစ်ပါတယ့်။

```
while (1) {
    printf(" Enter amount to send for %s : %s >>:", db[phone_found].name,
db[phone_found].email);
```

အချက် အလက်တွေသာမှန်နေခဲ့မယ္ယာရင် အဲဒီ**user** ရဲ့နာမည်နဲ့ **phone** နံပါတ်ကိုဖော်ပြပါး

User လွှဲချင်တဲ့ ပမာဏ ကိုထည့်ခိုင်းမှာ ဖြစ်ပါတယ်။ ကျွန်တော်တို့ ငွေလွှဲတဲ့ အခါမှာ အ

ကောင့်ထဲမှာ အနည်းဆုံးပမာဏ ကျွန်ရှိနေပါးမှ လွှဲလို့ရအောင် ရေးသားရပါမယ်။

```
scanf("%u", &amount_toTrans);
if(amount_toTrans < db[email_found].current_amount-1000){
    break;
}
```

အထက်ပါ **Program** ကဘာကိုပြောတာလဲဆိုရင် **user** လွှဲချင်တဲ့ ပမာဏက အကောင့်ထဲ

မှာ လက်ကျွန်ပမာဏ ၁၀၀၀ ကျွန်မှ လွှဲလို့ရမှာ ဖြစ်ပါတယ့်။ အထိုး အကောင့်ထဲမှာ **1000**

လဲ ကျွန်သေးတယ်ဆိုရင် တော့ **password** တောင်းပီး ငွေလွှဲခွင့်ပေးမှာ ဖြစ်ပါတယ့်။

အထက်ပါ **program** တွေက ငွေလွှဲခွင့်ရမရဆိုတာစစ်ဆေးတဲ့ **function** ဘဲရှိပါသေးတယ့်။

နောက် **program** မှာတော့ ငွေလွှဲတဲ့ **status** ကို ရေးမှာ ဖြစ်ပါတယ်။

လက်ရှိ ရေးသားပီးသောနေရာ ရှိသော **code** ၏ **github link** ဖြစ်သည်။

lesson-9-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson-9>

Money_transcation

ဒါ **status** မှာတော့ ငွေလွှဲပေးမယ့် အကြောင်းအရာတွေကို ရေးသားတော့မှာဘဲ ဖြစ်ပါတယ်။

```
void money_transaction(int transmit , int receiver , unsigned int amount)
```

ဒို့ **function** မှာ ဆိုရင်တော့ **parameter** ၃ ခုရှိပါပီးတော့ ပထမတစ်ခုက လွှဲပေးမယ့်သူ၊

အလွှံခံမယ့်သူနဲ့ လွှဲပေးမယ့် ပမဏ ကို ထည့်ထားတာဖြစ်ပါတယ့်။

```
void money_transaction(int transmit , int receiver , unsigned int amount){//  
amount is to transfer  
  
//before data  
printf("This is from money transfer\n");  
db[transmit].current_amount= db[transmit].current_amount-amount;  
db[receiver].current_amount = db[receiver].current_amount+amount;  
printf("transcation complete\n");  
printf("Your current amount: %s :%llu\n",db[transmit].name,  
db[transmit].current_amount);  
printing_specific_data(transmit);
```

```
    user_sector();
```

```
}
```

Transmit ရဲ **current_amount** ထဲကနေ လွှာပေးချင်တဲ့ ပမာဏကို နှုတ်ပီး **reciever**

ရဲ **current_amount** ထဲကို **amount** ပြောင်းထည့်ပေးရုပါဘဲ။ ထို့နောက် **data**

တွေရောက်သွားလားဆိုတာသိဖို့ရန်အတွက် **Printing_specific_data** ဆိုတဲ့ **function**

လေးကို ထပ်ပီးရေးသားထားပါတယ်။

```
void printing_specific_data(int user){

    printf("%u-%s-%s-%s-%u-%s-%s-%s-%d-%d-%d-%s-%llu-%s-%u-%u-%f-%s",
db[user].id, db[user].name, db[user].nrc,
        db[user].email, db[user].password, db[user].phoneNumber,
db[user].encryption_key, db[user].recovery_key,
        db[user].account_status, db[user].account_type,
db[user].account_level, db[user].minimum_opening_deposit,
        db[user].currency, db[user].current_amount, db[user].loanStatus,
db[user].monthly_income,
        db[user].loan_amount, db[user].loan_rate, db[user].address);
    for (int gcc = 0; gcc <= space_array[user] - 20; gcc++) {
        printf("-%s", db[user].tr[gcc].note);
    }
    printf("\n");
}
```

ဒဲ **function** ၊ **printing_all_data** ၊ **code** များကို ယူသုံးထားခြင်းဖြစ်ပါတယ်။

လက်ရှိနေရာအထိရေးသားပီးသော **code** ၏ **github link** ဖြစ်ပါသည်။

lesson10-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson10>

User_withdraw

ကျွန်တော်တို့ သူများကို ငွေလွှဲတဲ့ **status** ပီးရင် ကိုယ့် အကောင့်ထဲက ပိုက်ဆံကို ပြန်ထုတ် တဲ့ **function** ကို ရေးသားကြပါမယ့်။ ပထမဆုံးအနေနဲ့ **option** ရွှေးတဲ့နေရာမှာ အရင် သက်မှတ်ပေးပါမယ့်။

```
if(option == 49){
    transfer_money();
} else if(option==50){
    user_withdraw();
```

User ၏ **option2** ကို သာ နိုုပ်ခဲ့ရင် ငွေထုတ်တဲ့ **function**ကို ရောက်သွားမှာ ဖြစ်ပါတယ့်။

```
void user_withdraw(){
    char with_pass[50];
    unsigned int withdraw_amount=0;
    printf("Your current amount: %s : %lu\n",db[email_found].name ,
db[email_found].current_amount);
    printf("Enter amount to withdraw>>:");

    scanf("%u",&withdraw_amount);

    if(withdraw_amount<db[email_found].current_amount-1000){
        printf("Enter your password to proceed withdraw>>:");
        scanf(" %[^\n]",&with_pass[0]);
        twoCharArray=-1;
```

```

compare_twoCharArray(db[email_found].password , with_pass);
if(twoCharArray==1){
    printf("Get your withdraw amount: %u\n",withdraw_amount);
    printf("Your current
amount: %llu\n",db[email_found].current_amount-withdraw_amount);
    user_sector();
} else{
    printf("[ - ]Wrong credential:\n");
    user_withdraw();
}
} else{
    printf("Insufficient Balance:\n");
    user_withdraw();
}

}

```

Program ရှင်းလင်းချက်

With_pass ဆိုတဲ့ **char_array** နဲ့ **Unsigned int variable** တစ်ခုကို တည်ဆောက်ထားပါး နောက်တစ်ကြောင်းက **user** မှာရှိတဲ့ **current_amount** ကို ပြသထားပါတယ်။ ပီးရင် **user** ထုတ်ချင်တဲ့ ပမာဏကို ထည့်ခိုင်တဲ့အခါ **user withdraw** လုပ်ချင်တဲ့ပမာဏ က ငှါးရဲ့ လက်ကျွန်ပမာဏမှာ ၁၀၀၀အောက်ရောက်နေမယ်ဆိုရင် **insufficient Balance** လိုပြီး အစကိုပြန်ရောက်သွားမှာ ဖြစ်ပါတယ့်။ တကယ်လိုင်းထုတ်ဖို့လုံလောက်တယ်ဆိုရင် **user** ကို **password** တောင်းလိုက်ပါး ငွေထုတ်ခွင့်ပေးသွားမှာ ဖြစ်ပါတယ့်။ ကျွန်တော်တို့ အနေနဲ့ ဒီနေရာမှာ တစ်ခုသိထားရမှာက ငွေထုတ်တဲ့အခါ သို့မဟုတ် ငွေလွှဲတဲ့အခါ မှာ သူ့ရဲ့ **Limit** ရှိပါတယ်။ အဲဒီ **limit** ကိုသတ်မှတ်ပေးဖို့ ကျွန်တော်တို့ **function**တွေထပ်ဆောက်ပေးရမှာ ဖြစ်ပါတယ့်။

`printing_all_data,printing_specifics_data,loading_from_file` တွေရဲ့

`db[user].address` နောက်မှာ `db[user].trans_amount_limit_perday` ဆိုတဲ့ function

တစ်ခုကို ထပ်ရေးပေးရမှာ ဖြစ်ပါတယ်။

Recording_all_data_to_file

ဒီသင်ခန်းစာများတော့ `Recording_all_data_to_file` ဆိုတဲ့ functionကို တည်ဆောက်ပါး

ရေးပြသွားမှာ ဖြစ်ပါတယ်။ ဒီ functionတည်ဆောက်ရတဲ့ ရည်ရွယ်ချက်က ကျွန်တော်တို့

`data` အချက်အလက် တွေပြင်ချင်တဲ့ အခါ `terminal` မှာ ပြင်လိုက်ပေမယ့် `file` ထဲမှာ က

ပြောင်းလဲသွားတာမျိုး မဖြစ်ဘဲ `terminal` မှာဘဲပြောင်းသွားတာဖြစ်ပါတယ့်။ ထို့ကြောင့်

Program ပြန်စတဲ့အခါမှာလည်း `File` ထဲမှာ `data` တွေပြောင်းလဲသွားမှသာ ကျွန်တော်တို့က

ငွေလွှဲခဲ့တဲ့ မှတ်တမ်းတွေ`user`ရဲ့ `Information`တွေပြောင်းလဲတာတွေကိုမှတ်တမ်းတင်ထား

နိုင်မှာ ဖြစ်ပါတယ်။

```
void recording_all_data_to_file(){

    FILE *fptr = fopen("database.txt", "w");

    if(fptr == NULL){
        printf("[-]File opening error at recording to file function():\n");
    } else{

        for(int user=0; user<G_index; user++){

            fprintf(fptr, "%u%c%s%c%s%c%s%c%s%c%s%c%s%c%s%c%s%c%s%c1lu%c%s%c
            %u%c%u%c%f%c%s%c%d", db[user].id , ' ', db[user].name , ' ', db[user].nrc, '
```

```
' ,db[user].email, ' ',db[user].password, ' ',db[user].phoneNumber, '
',db[user].encryption_key, ' ',db[user].recovery_key, '
',db[user].account_status, ' ',db[user].account_type, '
',db[user].account_level, ' ',db[user].minimum_opening_deposit, '
',db[user].currency, ' ',db[user].current_amount, ' ',db[user].loanStatus, '
',db[user].monthly_income, ' ',db[user].loan_amount, ' ',db[user].loan_rate, '
',db[user].address, ' ',db[user].trans_amount_limit_perday);

    for(register int trc=0; trc<= space_array[user]-20 ; trc++ ){
        fprintf(fptr , " %s",db[user].tr[trc].note);
    }

    fprintf(fptr,"%c",'\\n');

}

printf("Recording complete to 'encrypted_data.txt!' \\n");

}
```

Program ရှင်းလင်းချက်

ဒါ **program** မှာတော့ **file** ကိုကိုင်တွယ်ရမှာ ဖြစ်ပါတယူ။ ထို့ကြောင့် **database.txt file** ကို

W mood နဲ့ဖွင့်ပါမယ်။ ထို့နောက် file ကို NULL ဖြစ်ခင် file opening error at recording to file

function ဆိုတဲ့စသားပေါ်အောင်ရေးသားထားပါး **NULL** မဖြစ်ခဲ့ဘူးဆိုရင် **else**

Condition ဆဲက **code** တွေ အလုပ်လုပ်မှာ ဖြစ်ပါတယ့်။

```

',db[user].minimum_opening_deposit,' ',db[user].currency,
',db[user].current_amount,' ',db[user].loanStatus,' ',db[user].monthly_income,
',db[user].loan_amount,' ',db[user].loan_rate,' ',db[user].address,
',db[user].trans_amount_limit_perday);

    for(register int trc=0; trc<= space_array[user]-20 ; trc++ ){
        fprintf(fptr , " %s",db[user].tr[trc].note);
    }

    fprintf(fptr,"%c",'\\n');
}

}

```

fprintf ကိုသုံးပါး **file** ထဲကို **data** တွေထည့်တဲ့ နေရာမှာ တစ်ကြောင်းဆင်းချင်တဲ့အခါ

\n ကို **double column** မသုံးထားတာကို သတိထားရမှာ ဖြစ်ပါတယ်။ အကယ်၍

Double column ကိုသာသုံးခဲ့မယ်ဆိုရင် ငါင်းက **string** ဖြစ်သွားမှာ ဖြစ်ပါတယ့်။

Structure ထဲမှာ **loan_rate** ကို **double** နဲ့ကြညာထားတာကို **float** လိုပြန်ပြောင်းထားပါတယ့်။

လက်ရှိရောက်နေသောနေရာအထိရေးသားပါးသော **code** ၏ **github link**ဖြစ်သည်။

lesson-11-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson-11>

Checking Program

ဒီသင်ခန်းစာမှာတော့ အပေါ်မှာရေးခဲ့ **recording_all_data_to_file** က အလုပ်လုပ်လားဆိုတာ စစ်ဆေးကြည့်မှာဘဲ ဖြစ်ပါတယ်။ ပထမဆုံးအနေနဲ့ **main.c** မှာထားပါး **Program run** ကြည့်တဲ့အခါ အောက်ပါအတိုင်း **terminal** မှာ ပေါ်လာမှာ ဖြစ်ပါတယ့်။

```

1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     space_counter();
6     loading_from_file();
7     printing_all_data();
8     main_menu();
9 }
10
11

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

PS C:\Users\MSI\Desktop\C_program> cd "C:\Users\MSI\Desktop\C_program"
1-winhtut-9/paoola(N)80099-winhtut@gmail.com-admin123-2208183876-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
2-winhtut-9/paoola(N)80099-huttleline@gmail.com-admin123-2208183875-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
3-winhtut-9/paoola(N)80099-winline@gmail.com-admin123-2208183874-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
Welcome to our OnlineBank-NCC!
Press 1 to Login!
Press 2 to Register!
Press 3 to Exit!>>>

```

> OUTLINE > TIMELINE

figure-64

ဒီနေရာမှာ ကျွန်တော်တို့က **register** လုပ်လိုက်တဲ့အခါ အသစ်ထည့်လိုက်တဲ့ **user** ရဲ့**inform**

ကို **file** ထဲရောက်ခဲ့ပါက ကျွန်တော်တို့ **function** က အလုပ်လုပ်တွေ့ဖြစ်တဲ့
သိရမှာ ဖြစ်ပါ

တယူ။ ဒါကြောင့် **2ကိုနှိပ်ပီး register** ကိုသွားပါမယူ။ ထိုနောက်အောက်ပါအချက်အလက်

များကို **register** လုပ်ဖို့ရန်အတွက်ဖြည့်သွင်းကြပါမယ်။

This is Online Bank Register !

Enter your email to register>>:mear@gmail.com

Your email form was valid:

Del re_email mear@gmail.com

Your email was saved:

Enter your username:

mear

Enter your NRC information:9/paoola(N)80099

[+]Your NRC format was correct:

del re_nrc test: 9/paoola(N)80099Enter your password:rg#@#53KFC

[+]Your password format was correct and was saved!

[X]Enter your Phone Number:099858904

[+] Your phone is correct and saved!

[X]Enter your Encryption Key(from 4 to 6 char):wear

[+]Your Encryption key valid and saved!

[X]Enter your Recover Key carefully:wearis

[+]Your recover key was saved!

[X]Enter your currency to use:mmk

[X]Enter your amount:100000

[X]Enter your monthly income:100000

[X]Enter your address:pyinoolwin

[X]Enter your opening note:hellonewuser

del re_email mear@gmail.com

1-winhtut-9/paooola(N)80099-winhtut@gmail.com-admin123-**2208183876-**
aaa123aaa123wh-aaa123wh-personal-**1-1-10000-mmkk-15000-T-5000-1-**
0.500000-pyinoolwin-5000-onlinebankadmin

2-winhtut-9/paooola(N)80099-htutline@gmail.com-admin123-**2208183875-**
aaa123aaa123wh-aaa123wh-personal-**1-1-10000-mmkk-15000-T-5000-1-**
0.500000-pyinoolwin-5000-onlinebankadmin

3-winhtut-9/paooola(N)80099-winline@gmail.com-admin123-**2208183874-**
aaa123aaa123wh-aaa123wh-personal-**1-1-10000-mmkk-15000-T-5000-1-**
0.500000-pyinoolwin-5000-onlinebankadmin

4-mear-9/paooola(N)80099-mear@gmail.com-rg@#53KFC-**99858904-wear-**
wearis-personal-**1-1-10000-mmkk-100000-T-100000-1-0.500000-**
pyinoolwin-5000-hellonewuser

Recording complete to 'encrypted_data.txt'!

[X] Registration Success: Mr/s: mear

Welcome to our OnlineBank-NCC!

Press 1 to Login!

```
Press 2 to Register!  
Press 3 to Exit!>>:3  
Recording complete to 'encrypted_data.txt'!  
Good Bye!  
PS C:\Users\MSI\Desktop\C program>
```

Figure-65

အထက်ပါအတိုင်း **data** ထည့်ပါက **main_menu** ကို ပြန်ရောက်သွားပါက **exit**ထွက်ဖို့ရန်

အတွက် 3ကို နိုဝင်ပီးပါက database.txt ထဲသွားကြည့်တဲ့အခါ ငယောက်မြောက်userဝင်လာ

တာကို တွေ့ရမှာ ဖြစ်ပါတယ်။

```

C main.c      E database.txt  C onlinebank.h
database.txt
1 1 winhtut 9/paoola(N)80099 winhtut@gmail.com admin123 2208183876 aaa123aaa123wh aaa123wh personal 1 1 10000 mmk 15000 1 0.500000 pyinoolwin 5000 onlinebankadmin
2 2 winhtut 9/paoola(N)80099 htutline@gmail.com admin123 2208183875 aaa123aaa123wh aaa123wh personal 1 1 10000 mmk 15000 T 5000 1 0.500000 pyinoolwin 5000 onlinebankadmin
3 3 winhtut 9/paoola(N)80099 winline@gmail.com admin123 2208183874 aaa123aaa123wh aaa123wh personal 1 1 10000 mmk 15000 T 5000 1 0.500000 pyinoolwin 5000 onlinebankadmin
4 4 mear 9/paoola(N)80099 mear@gmail.com rg@#5KFC 99858904 wear wearis personal 1 1 10000 mmk 100000 T 100000 1 0.500000 pyinoolwin 5000 hellonewuser
5 |

```

Figure-66

Error Checking

လေ့လာသူတို့အနေနဲ့ **lesson_10** က **code** အတိုင်းရေးထားပါး **register** လုပ်တဲ့ အခါ **error** တက်နေပါလိမ့်မယ်။ ထိုကဲ့သို့ဖြစ်ရတဲ့ အကြောင်းအရင်းက **money_transcation** လုပ်ခဲ့တဲ့ အနေးက **transcation_amount_limit_per_day** ဆိုတဲ့ **feature** အသစ်ထပ်ထည့်တဲ့အခါ **space_array** ကို 19လိုဘဲထားခဲ့ပါတယ့်။ အရင်သင်ခန်းစာတွေတဲ့ အနေးက **feature** က ၁၉ခုဘဲ ရှိပေမယ့် အခုက ၂၀ရှိတဲ့ အခါ **register function** ထဲမှာ **space_array** ကို ၂၀ ပြောင်းပေးရပါမယ်။

Get_Time

ဒီသင်ခန်းစာမှာကတော့ အချိန်တွေနဲ့ပါတယ်သတ်ပါး ရေးသားသွားမှာ ဖြစ်ပါတယ့်။ ငါး

Function ကို ရေးနိုင်ဖို့ **header file** ခဲ့အပေါ်မှာ **#include "time.h"** လို့အရင်ကြညာပေး

ရပါမယ့်။

```

void get_time(){
    time_t tm;
    time(&tm);
    int time_space_counter=0;

    printf("Current Date/Time = %s", ctime(&tm));
}

```

အထက်ပါ အတိုင်းအရောင်ရေးသားပီးတော့ **main.c** မှာ ရေးသားကြည့်တဲ့အခါ အောက်ပါပုံ

ကအတိုင်း **output** ထွက်လာမှာ ဖြစ်ပါတယ့်။

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer (Left):** Shows project files including `main.c`, `onlinebank.h`, `secheader.h`, `main.c` (selected), `main.exe`, `onlinebank.exe`, `onlinebank.h`, `project flow`, `seche.c`, `secheader.h`, `secmain.c`, `secmain.exe`, `tempCodeRunnerFile.c`, and `user_setting.h`.
- Editor (Center):** Displays the `main.c` file content:

```
1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     // space_counter();
6     // loading_from_file();
7     // printing_all_data();
8     // main_menu();
9     get_time();
10    return 0;
11 }
```
- Terminal (Bottom):** Shows command-line output:

```
PS C:\Users\MSI\Desktop\C program> cd "c:\Users\MSI\Desktop\C program\" ; if ($?) { gcc main.c -o main } ; if ($?) { .\main }
Current Date/Time = Wed Jun 28 09:26:16 2023
PS C:\Users\MSI\Desktop\C program>
```

Figure 67

Current Date/Time = Wed Jun 28 09:26:16 2023

အထက်ပါအတိုင်း **current date** နဲ့ **time** ကို **file**ထဲသို့ထည့်ပါးစာလုံးတွေကြားထဲက **space** တွေကို

ဖျောက်ပါး တစ်ခုချင်းကို **array** တွေထဲကိုထည့်ပါမယူ။ ထို့ကြောင့် **structure** တစ်ခုကိုအောက်ပါး

თაღუ: გაყიდვების მიზანი და მიზანის მიზანი

ପିଷ୍ଟା

```
struct currentTime{
```

```
char curTime[25];
```

date እና **time** በ የ **file** ተሰርዞ ይታረም ይችላል ማለት የ **mytime.txt** ተሰርዞ ይችላል

ပါတယ်။ ထို့နောက် **get_time function** ထဲမှာ mytime.txt ကို w mood ဖြင့် ဖွင့်လိုက်ပါး

Date နဲ့ time ကို file ထဲသိတည်လိုက်ပါသည်။

The screenshot shows the Visual Studio Code interface. On the left is the Explorer sidebar with project files like .vscode, launch.json, tasks.json, class.c, main.c, mytime.txt, onlinebank.exe, onlinebank.h, seche.c, secheader.h, secmain.c, secmain.exe, tempCodeRunnerFile.c, and user_setting.h. The main area has four tabs open: main.c, onlinebank.h, secheader.h, and mytime.txt. The code in main.c is as follows:

```

void get_time(){
    time_t tm;
    time(&tm);

    printf("Current Date/Time = %s", ctime(&tm));
    FILE *fptr=fopen("mytime.txt","w");
    fprintf(fptr,"%s",ctime(&tm));

    int index= 0;
    fclose(fptr);
    getCTime[0].current_time[index]='-';

    FILE* fptr2=fopen("mytime.txt","r");
    char c=fgetc(fptr2);
    while(!feof(fptr2)){
        if(c==' '){
            getCTime[0].current_time[index]='-';
    }
}

```

The terminal at the bottom shows the command line output:

```

PS C:\Users\MSI\Desktop\C program> cd "c:\Users\MSI\Desktop\C program"
Current Date/Time = Wed Jun 28 09:55:34 2023
PS C:\Users\MSI\Desktop\C program>

```

Figure_68

```

void get_time(){

    time_t tm;
    time(&tm);

    printf("Current Date/Time = %s", ctime(&tm));
    FILE *fptr=fopen("mytime.txt","w");
    fprintf(fptr,"%s",ctime(&tm));

    int index= 0;
    fclose(fptr);
    getCTime[0].current_time[index]='-';

    FILE* fptr2=fopen("mytime.txt","r");
    char c=fgetc(fptr2);
    while(!feof(fptr2)){
        if(c==' '){
            getCTime[0].current_time[index]='-';
    }
}

```

```

        c=fgetc(fptr2);
        index++;
    }else{
        getCTime[0].current_time[index]=c;
        c=fgetc(fptr2);
        index++;
    }
}
printf("\nremoved space: %s",getCTime[0].current_time);
}

```

Program ရှင်းလင်းချက်

data ကို **figure-68** ထဲကအတိုင်းရေးပါက**header file** ရဲ့အပေါ်ဘက်မှာ **structure** ဆောက်ပါမယ်။

```

struct currentTime{

    char current_time[25];
};

struct currentTime getCTime[1];

```

ထို့နောက် **file** ကို နောက်တစ်ကြိမ် **read mood** နဲ့ဖွင့်ပါမယ်။ **file** က**feof** မဖြစ်မချင်း
while loop ပါတယ်။ **if condition** ထဲတွင် အကယ်၍ **char c** သည် **space** နဲ့ညီးမထွေ့ပါက ငါး
space နေရာတွင်”-“ကို အစားထိုးလိုက်မှာ ဖြစ်ပါတယ်။

```

getCTime[0].current_time[index]=c;
c=fgetc(fptr2);
printf("\nremoved space: %s",getCTime[0].current_time);

```

```

File Edit Selection View Go Run Terminal Help
main.c - C program - Visual Studio Code [Administrator]
C main.c X C onlinebank.h C secheader.h E mytime.txt
C main.c > ...
1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     // space_counter();
6     // loading_from_file();
7     // printing_all_data();
8     // main_menu();
9     get_time();
10    return 0;
11 }
12

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\MSI\Desktop\C program> cd "c:\Users\MSI\Desktop\C program" ; if ($?) { gcc main.c -o main } ; if ($?) { ./main }
Current Date/Time = Wed Jun 28 11:13:45 2023
removed space: [Wed-Jun-28-11:13:45-2023]
PS C:\Users\MSI\Desktop\C program>

```

figure-70

program နောက်တစ်ကြိမ် စလိုက်တဲ့ အခါ **space** တွေကို ဖယ်ထုတ်ပီးသားဆိုတာကို သိရ

မှာ ဖြစ်ပါတယ့်။

lesson-12-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson%2012>

Transication_record

ဒီသင်ခန်းစာများတော့ **user** က ငွေလွှဲတဲ့အခါ သို့မဟုတ် ငွေထုတ်ပိုးတဲ့အခါ မှတ်တမ်းတင်

ထား တဲ့ **function**ကို ရေးသားသွားမှာ ဖြစ်ပါတယ့်။ ငွေလွှဲတဲ့အခါမှာလဲ **transmitter** နဲ့

receiver ဘက်မှာပါ **transaction record** တွေကို ပြုသထားနိုင်တဲ့အထိ ရေးသွားမှာ ဖြစ်

ပါတယ့်။ နောက်တစ်ချက်က **user** အနေနဲ့ ငြင်းခဲ့ **transaction record** ကိုဘဲသီးသန့်လို

ချင်ပါက ငြင်းတစ်ယောက်ထဲကိုဘဲ သီးသန့်ပြတဲ့ **function** ကို ရေးသားသွားမှာ ဖြစ်ပါတယ်။

ပထမဆုံးအနေနဲ့ **function** ကို **parameter** ၃ခုဖြတ်ထားပါမယ်။ ပထမတစ်ခုက **transmitter,receiver** နဲ့ နောက်တစ်ခုက ဘယ်သူကို ဘယ်လောက်လွှဲလိုက်လဲဆိုတော်ပြဖို့အတွက် **char array** တစ်ခု ဆောက်ပါမယ်။

```
void transacitonRecord(int userDBIndex,int uFound,char who){
    int transfer = char_counting(db[userDBIndex].name);
    int accept = char_counting(db[uFound].name);

    if(who == 't') {

        char toInsert1[4] ={'F','r','o','m'};
        char toInsert2[2] ={'t','o'};

        int indexPoint=0;
        for(int x=indexPoint; x<4; x++){
            db[userDBIndex].tr[space_array[userDBIndex] - 20].note[x]=toInsert1[x];
            indexPoint++;
        }
    }
}
```

အထက်ပါ **function** ကို ရှင်းပြရမယ်ဆိုရင် **char_counting** ကိုသုံးပီး **userDBIndex** နဲ့

uFound ခဲ့**name** တွေကို **count** လိုက်ပါတယ်။ **who=='t'** ကဘာကိုပြောတာလဲဆိုရင်

t က **transfer** ကို ရည်ညွှန်းတာဖြစ်ပီး ငှါးအောက်က **char array** နှစ်ခုဖြစ်သည့် **f,r,o,m**

နှင့် **t,o** သည် **from** ခဲ့နောက်မှာတော့ **transmitter** လိုက်ပီးတော့ **to** ခဲ့နောက်မှာတော့ **receiver** လိုက်မှာ ဖြစ်ပါတယ်။ ယခု **transmitter** အတွက်ရေးနေသည့်အတွက် **space_array** ကိုအလုံး**19**နှင့်ထားတာရဲ့နောက်မှာ **toInsert1 array** ကိုပေါင်းထည့်မည်ဖြစ်

ပါတယ့်။ **user** ရဲ့ **id** ကနေ **transaction record** အထိကြားထဲမှာရှိတဲ့ **space** တွေကို
ရေတွက်ကြည့်မယဆိုရင်အလုံး **19** ရှိတာကို မြင်ရမှာ ဖြစ်ပါတယ့်။
ထိုနောက် **transaction_money** ထဲမှာ **printing_specifies_data** ကို ထည့်ပါးနောက်
user_sector ကို ပြန်သွားပါမယ့်။

```
void money_transaction(int transmit , int receiver , unsigned int amount){//
amount is to transfer
```

```
    db[transmit].current_amount= db[transmit].current_amount-amount;
    db[receiver].current_amount = db[receiver].current_amount+amount;
    printf("This is from money transfer\n");
    printing_specific_data();
    user_sector();
```

}

```
void transacitonRecord(int userDBIndex,int uFound,char who){
    int transfer = char_counting(db[userDBIndex].name);
    int accept = char_counting(db[uFound].name);

    if(who == 't') {

        char toInsert1[4] ={ 'F','r','o','m'};
        char toInsert2[2] ={ 't','o'};

        int indexPoint=0;
        for(int x=indexPoint; x<4; x++){
            db[userDBIndex].tr[space_array[userDBIndex] -
```

```

19].note[x]=toInsert1[x];
    indexPoint++;
}
int nameIndex=0;
int endPoint=indexPoint+transfer;
for(int x=indexPoint;x<endPoint;x++){
    db[userDBIndex].tr[space_array[userDBIndex]-
19].note[indexPoint]=db[userDBIndex].name[nameIndex];
    nameIndex++;
    indexPoint++;
}
}

```

from ဆိုတဲ့စာသားကို **space 19** လုံးမြောက် မှာထည့်ပါးနောက် ငွေး၏အနောက်မှာ လဲပေးမယူ။

သူ၏ နာမည် ပေါ်စေချင်တာမို့ အောက်ပါ **code** များထပ်ဖြည့်ထားပါတယ်။

```

int nameIndex=0;
int endPoint=indexPoint+transfer;
for(int x=indexPoint;x<endPoint;x++){
    db[userDBIndex].tr[space_array[userDBIndex]-
19].note[indexPoint]=db[userDBIndex].name[nameIndex];
    nameIndex++;
    indexPoint++;
}

```

nameIndex ဆိုတဲ့ **integer variable** တစ်ခုကို ကြေညာထားပါး **endpoint** ကို **indexPoint** နဲ့

transfer ကို ပေါင်းတဲ့ တန်ဖိုးကို သိမ်းထားလိုက်ပါတယ့်။ အပေါ် **looping** မှာတူန်းက

indexPoint တန်ဖိုးသည် **4** ဖြစ်နေရမှာ ဖြစ်ပါတယ့်။ **transfer** ကတော့ **user** ခဲ့ **name** တွင်ရှိသည့်

စာလုံးအရေအတွက်အတိုင်း ဖြစ်နေမှာ ဖြစ်ပါတယ့်။ **userDBIndex** ခဲ့ **tr**ထဲက

Space _array ကို ၁၉နှင့်ထားတဲ့နေရာက **note** ခဲ့ **indexPoint** မှာ **userDBIndex** ၏ **name**

ဝင်လာမှာဖြစ်ပါတယ့်။ ထို့နောက် **Looping** တစ်ကြိမ်ပါတ်ပီးတိုင်း **nameIndex** နှင့်

indexPoint ကို ++ တိုးပေးသွားမှာ ဖြစ်ပါတယ်။ **from+transmitter** အတွက် ရေးပီးသော

အခါ **to+receiver** အတွက်ရေးပေးဖို့ကျန်နေပါသည်။

```

nameIndex=0;
endPoint = indexPoint+2;
for(int x=indexPoint; x<endPoint; x++){

    db[userDBIndex].tr[space_array[userDBIndex] - 
19].note[indexPoint]=toInsert2[nameIndex];
    nameIndex++;
    indexPoint++;
}

nameIndex=0;
endPoint=indexPoint+accept;
for(int x=indexPoint; x<endPoint; x++){
    db[userDBIndex].tr[space_array[userDBIndex] - 
19].note[indexPoint]=db[uFound].name[nameIndex];
    nameIndex++;
    indexPoint++;
}

nameIndex=0;

```

nameIndex ကို 0 လို့ ပြန်ထားပါးနောက် **endpoint** တန်ဖိုးကို **indexpoint+2** လို့ တန်ဖိုး

သက်မှတ်ထားလိုက်ပါမယ်။ ထို့နောက် **to** ကို **transmitter name** ရဲ့ထည့်ဖို့ရန် အတွက်

toInsert2 array ကို **db[userDBIndex]** ထဲက **tr space array** ကို 19 နှင့်ပါး **note** ရဲ့**indePoint**

ထဲကို ထည့်လိုက်ပါတယ်။ **to** ပေါင်းထည့်ပါးပါက ငှင်းနောက်မှာ **receiver** ပေါင်းထည့်ဖို့ရန်အတွက်

nameIndex တန်ဖိုး ကို 0 လို့ ပြန်ထားပါး **endPoint** ကိုတော့ **indexPoint+ accept**

လို့သတ်မှတ်ပေးလိုက်ပါတယ့်။ ထို့နောက် **note** ရဲ့**indexPoint** နေရာမှာ **db[uFound].name** ရဲ့

nameIndex ကို ပေါင်းထည့်ပေးလိုက်မှာ ဖြစ်ပါတယ့်။

from+UserDBIndex+to+uFound+get_time

Adding_Time_To_Transication_record

ငွေလွှဲတဲ့ **status** ပီးတဲ့အခါ လွှဲခဲ့တဲ့အချိန်ကို သိဖော်ရန်အတွက် **get_time function** ကိုအသံးပြုမှာ ဖြစ်ပါတယ်။ အချိန်ကို **receiver** ရဲနောက်မှာ ထည့်ပေါင်းမှာ ဖြစ်ပါတယ့်။

```
get_time();
```

```
for(int win= endPoint; win<25+endPoint ; win++ ){
    db[userDBIndex].tr[space_array[userDBIndex]-
19].note[win]=getCTime[0].current_time[nameIndex];
    nameIndex++;
}
space_array[userDBIndex] +=1;
}
```

get_time ကို အရင်ခေါ်သုံးထားပီး**for loop** ပါတ်သောအခါ **endpoint** ကို**25** ပေါင်းရခြင်း၏ ရည်ရွယ်ချက်မှာ **Wed Jun 28 11:13:45 2023** သည် အားလုံးပေါင်း ၂၅ ဖြစ်သောကြောင့် ဖြစ်ပါသည်။ ထို့နောက် **loop** ၏ **endPoint** ကနေ စထားတာကို မြင်ရမှာ ဖြစ်ပါသည်။ **db** ထဲက **transmitter** ရဲ့ **tr** ထဲက **space_array** ထဲက **note** ရဲ့ **win(endPoint)**တွင် **getCTime[0]** ရဲ့ **current_time[nameIndex]** ကို ပေါင်းထည့်ပေးပါ သည်။ **getCTime** ရဲ့ **Index** နေရာကို **0** ဖြစ်ရခြင်းသည် အပါသင်ခန်းစာတွေမှာတူန်းက ငှုံး၏ **structure array** အခန်းကို **1**လို့ကြော်ထားခဲ့ခြင်းကြောင့် ဖြစ်ပါသည်။

ထိုနောက် **transmitter** အတွက်ရေးပါက **receiver** အတွက် ထပ်ပိုးသားကြပါမယ့်။

User ကငွဲလွှာအခါ **transmitter** ဘက်မှာဘဲ **transaction record** ကိုဖော်ပြရုံမက

Receiver ဘက်မှာလည်း **transaction record** ကို ပြသမှာ ဖြစ်ပါတယ့်။

```

else{

    charreceiveFrom[13]={'-',
',','R','e','c','e','i','v','e','F','r','o','m','-'};
    int indexPoint=0;
    int endpoint=0;
    int reIndex=0;
    for(int i=0;i<accept;i++){
        db[uFound].tr[space_array[uFound]-
19].note[i]=db[uFound].name[i];
        indexPoint++;
    }
    endpoint=accept+13;
    for(int i=indexPoint;i<endpoint;i++){
        db[uFound].tr[space_array[uFound]-
19].note[i]=receiveFrom[reIndex];
        reIndex++;
    }

    reIndex=0;
    int toendpoint=endpoint+transfer;
    for(int i=endpoint;i<toendpoint;i++){
        db[uFound].tr[space_array[uFound]-
19].note[i]=db[userDBIndex].name[reIndex];
        reIndex++;
    }

    reIndex=0;
}

```

```

    get_time();
    for(int win=toendpoint; win<25+toendpoint ; win++ ){

        get_time();
        db[uFound].tr[space_array[uFound]-
19].note[win]=getCTime[0].current_time[reIndex];
        reIndex++;

    }
    space_array[uFound] +=1;
}

```

Receiver ဘက်ခြင်းကိုတော့ **else condition** ဘက်မှာ ရေးသားထားမှာ ဖြစ်ပါတယ့်။ ထို့နောက် ငြင်းအတွက် **function call** ကို တော့ **money_transication** ထဲမှာ ခေါ်မှာဘဲဖြစ်ပါတယ့်။

```
transacitonRecord(transmit,receiver,'r');
```

Receiver_form ဆိုတဲ့ **char array** တစ်ခုနဲ့ **local variable** ခုခုကြညာထားတာကိုမြင်ရမှာ ဖြစ်ပါတယ်။ **ufound** က **money_transcation** ထဲမှာ **receiver** ကို ကိုယူးပြပါတယ်။ **db[ufound]** ထဲက **space_array** ကို **19** နှင်းထားတဲ့ **note** ထဲကို အလွှံခံရမယ့် သူရဲ့ **name** ကို ဘဲဖြစ်ပါတယ်။ ထို့နောက် ငြင်း၏ နောက်တွင် **reciever_form** ကို **-receive-from-** **array** ကို ထည့်ပါတယ်။ ထည့်ပါတယ်။ ထို့နောက် တွင် **space_array** ကို **1** ပေါင်းပေးရသည့် လွှဲပေးခဲ့သည့် အချိန်ကို

ထည့်လိုက်မှာ ဖြစ်ပါတယ့်။ ထို့နောက် တွင် **space_array** ကို **1** ပေါင်းပေးရသည့် အကြောင်းအရင်းမှာ နောက်တစ်ကြိမ် **information** အသစ်ထည့်သည့်အခါ

ထပ်ပေါင်းထည့်နိုင်အောင်ဖြစ်ပါတယ့်။ သို့မှသာ **error** မတက်မှာ ဖြစ်ပါတယ့်။

lesson-13-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson%2013>

Integer_To_Char_array

Transication_record အတွက်ရေးသားပါက ငွေလွှဲသည့်အခါတွင် **user** အနေဖြင့် တစ်နောက်
ငွေပမာဏ မည်မျှလွှဲနိုင်သည်ဆိုသည်ကို သတ်မှတ်ပေးရအုံမည်ဖြစ်ပါသည်။

ကျွန်တော်တို့ က အကောင့် **level** တွေကို သတ်မှတ်ပေးခဲ့ ကြပီး ဖြစ်ပါသည်။ အကယူဇုံ **user** ၏
account level သည် **1**ဖြစ်နေခဲ့ရင် ငှုံး၏ ငွေလွှဲနိုင်သည့် နှစ်းကို ကန့်သတ်ထားလိုက်ပီး ငှုံးအနေဖြင့်
level တင်မှသာလျှင် ငွေအကန့်အသတ် မရှိလွှဲနိုင်မှာ

ဖြစ်ပါတယ့်။ လက်ရှိတွင် ကျွန်တော်တို့က **program** ရေးနည်းသင်နေကြတာဖြစ် သည့်အတွက် **24**
hour ကိုတာ ယူပီး **program** ကို ရေးပါက တစ်ရက်ကို တစ်ကြိမ်သာ လွှဲခွင့်ရမှာဘဲ ဖြစ်သည့်အတွက်
ကျွန်တော်အနေဖြင့် **24 hour = 3 minutes** နဲ့ ဘဲတွက်ယူမှာ

ဘဲ ဖြစ်ပါတယ်။ ဆိုလိုသည်မှာ **3minute** ပြည့်ပီးပါက နောက်တစ်ကြိမ်ခွင့်ပေးမှာဘဲဖြစ်ပါ
တယ့်။ ထိုနေရာတွင် **transcation_amount_limit_per_day** ရေးသည့်အခါ **data** ထည့်ရတာ
ခက်နိုင်သည့် အတွက် **integer** ကနေ **char_array** ကို ပြောင်းပစ်လိုက်ပီး တစ်လုံးချင်းစီ **data**
ဖတ်ပီးထည့် ဖို့ ဆုံးဖြတ်ခဲ့ပါသည်။ ထိုနောက် **char_array** မှုလဲ **integer** သို့ပြောင်းလဲသည့် **function**
ကို လဲရေးသားသွားမှာ ဖြစ်ပါတယ်။ ထိုသို့ရေးသားရန် အတွက် **whw.txt** ဆိုသည့် **file** တစ်ခုကို
တည်ဆောက်သွားမှာ ဖြစ်ပီး ငှုံး

ထဲသို့ user ထည့်မည့် amount ကို ထည့်ကာ integer မှာ array သို့ပြောင်းလဲသွားမှာ ဖြစ်ပါတယ်။

```
void integer_to_charArrayFun(unsigned int integer){
    int index=0;
    FILE *fptr=fopen("whw.txt","w");
    if(fptr==NULL){

        printf("[ - ]Error at integer_to_charArrayFun\n");
    }else{
        fprintf(fptr,"%d",integer);
    }
    fclose(fptr);

    FILE *fptr2=fopen("whw.txt","r");
    char c=fgetc(fptr2);

    while(!feof(fptr2)){
        int_toCharArray[index]=c;
        c=fgetc(fptr2);
        index++;
    }
    fclose(fptr2);
}
```

Program ရှင်းလင်းချက်

integer_to_charArrayFun ဆိုသည့် function ကို parameter unsigned int စာစ်ခု ထားသည်ကို မြင်ရမှာ ဖြစ်ပါတယ့်။ ထို့နောက် whw.txt file ကို 'w' mood ဖြင့် ဖွင့်လိုက ပါတယ်။ **w mood** ဖြင့် ဖွင့်ရသည့် အကြောင်းအရင်းမှာ file တစ်ခါ ဖွင့်သည့်အခါတိုင်း

File ထဲတွင်ရှိသည့် **data** များကို **over write** လုပ်ရန် ဖြစ်ပါသည်။ ထုံးစံ အတိုင်း **file** ကို **NULL** ဖြစ်မဖြစ် စစ်ဆေးပါမယ့်။ အကယ့်၍ **file** က **NULL** ဖြစ်နေခဲ့ပါက **[-]Error at integer_toCharArrayFun\n** ထိုစာကြောင်းကို ပြသမှာ ဖြစ်ပါး **NULL** မဖြစ်ခဲ့ဘူးဆိုပါက **file** ထဲသို့ **fprintf** ဖြင့် **integer** ကို ရှိက်ထည့်လိုက်မှာ ဖြစ်ပါတယ့်။ ထို့နောက် **file** ကို ပိတ်လိုက်မှာ ဖြစ်ပါတယ်။ ထို့နောက် **FILE fptr2** ကို **whw.txt** ကို နောက်တစ်ကြိမ် **read mood** ဖြင့် ဖွင့်လိုက်ပါး **file** ထဲတွင်ရှိသော **data** များကို **char c** ထဲသို့ **fgetc** ကို သုံးပီး တစ်လုံးချင်းဆီ ထည့်လိုက်မှာ ဖြစ်ပါတယ့်။ တစ်လုံးချင်းစီ ဝင်လာသော **data** များကို **while loop** ဖြင့် **int _toCharArray** ထဲကို ထည့်ထားလိုက်မှာ ဖြစ်ပါတယ့်။ **int_toCharArray** ကို **global** တွင်ကြညာထားပါးဖြစ်ပါသည်။

```
int_toCharArray[index]=c;
c=fgetc(fptr2);
Index++;
```

ထို့နောက် **file** ကို ပိတ်လိုက်ပါး ပါက **integer** ကို **char array** သို့ပြောင်းလဲလိုက်ပါး ဖြစ်ပါသည်။ **program** အလုပ်ဖြစ်မဖြစ်ကို သိရှိစေရန် **program** ကို အစမ်း **run** ကြည့်ကြပါမယ့်။ ထိုသို့ **program** စရန်အတွက် **main.c** ထဲတွင် အောက်ပါအတိုင်း ပြင်ထားလိုက်ပါပီးက **program run** လိုက်သောအခါ **char array** အနေဖြင့် ထွက်လာပါက အဆင်ပြေမှာ ဖြစ်ပါတယ်။

```
#include <stdio.h>
#include "onlinebank.h"

int main(){
    // space_counter();

    // loading_from_file();
    // printing_all_data();
```

```

// main_menu();
// get_time();

integer_to_charArrayFun(1500);
printf("char array %s",int_to_charArray);
return 0;
}

```

The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows files in the project directory, including `main.c`, `database.txt`, `onlinebank.h`, `class.c`, and `whw.txt`.
- Code Editor:** Displays the `main.c` file with the provided C code.
- Terminal:** Shows the command-line interface with the following session:


```
PS C:\Users\MSI\Desktop\C program> cd "c:\Users\MSI\Desktop\C program"
PS C:\Users\MSI\Desktop\C program> if ($?) { gcc main.c -o main }
PS C:\Users\MSI\Desktop\C program> ./main
char array 1500
```

Figure-78

Terminal တွင် char array: 1500 ပေါ်လာသောအခါ 1500 သည် integer မဟုတ်တော့ဘဲ

Array တစ်ခုပြောင်းလဲသွားပါ ဖြစ်ပါသည်။ ထိုနောက် တစ်ဆက်ထဲအနေဖြင့် **char array**

ကနေ **integer** သို့ ပြောင်းလဲသည့် **function** ကို ဆက်လက် ရေးသားသွားမှာ ဖြစ်ပါသည်။

ဒီ **function** မှာတော့ **return** ပြန်လိုက်ရန်အတွက် **unsigned int** ဖြင့် ရေးသားသွားမှာ ဖြစ်ပါတယ့်။

```

unsigned int char_to_interger_fun(char char_array[50]){
    unsigned int char_to_int_data=0;
    FILE *fptr=fopen("whw.txt","w");
}

```

```

if(fptr==NULL){
    printf("[ - ]Error\n");
}else{
    fprintf(fptr,"%s",char_array);
}
fclose(fptr);

FILE *fptr2=fopen("whw.txt","r");

if(fptr2==NULL){
    printf("[ - ]Error at char_to_int_data");
}else{
    fscnaf(fptr2,"%d",&char_to_int_data);
}
return char_to_int_data;
}

```

Program ရှင်းလင်းချက်

Function တွင် **paramaeter** အနေဖြင့် **char array** တစ်ခု ဖြတ်သွားမှာ ဖြစ်ပါး **whw.txt**

file ကို **w mood** ဖြင့် ဖွင့်ကာ **char array** တစ်ခုကို **fprintf** ဖြင့် **char array** တစ်ခုကို **file**

ထဲသို့ ထည့်လိုက်မှာ ဖြစ်ပါသည်။ ထို့နောက် **file** ကို ပိတ်ပါး **fptr2**ဆုံးပါး နောက်တစ်ကြိမ် ဖွင့်လိုက်မှာ ဖြစ်ပါသည်။ ထို့နောက် **fscanf** ဖြင့် **file** ထဲတွင်ရှိသော **data** များကို ဖတ်ပါးနောက် **char_to_int_data** ကို **return** ပြန်လိုက်မှာ ဖြစ်ပါတယ်။ **program** အလုပ်လုပ်လားဆိုတာ **test** လုပ်ကြည့်ပါမယ့်။

```

File Edit Selection View Go Run Terminal Help
main.c - C program - Visual Studio Code [Administrator]
EXPLORER C main.c database.txt C onlinebank.h whw.txt
C main.c > main()
1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     // space_counter();
6     // loading_from_file();
7     // printing_all_data();
8     // main_menu();
9     // get_time();
10    // integer_to_charArrayFun(1500);
11    // printf("char array %s",int_toCharArray);
12
13    char myarr[5]={'1','2','3','4','5'};
14    unsigned int data=char_to_integer_fun(myarr);
15    printf("data %u",data);
16    return 0;
17
18
19
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
PS C:\Users\MSI\Desktop\C program> cd "c:\Users\MSI\Desktop\C program" ; if ($?) { gcc main.c -o main } ; if ($?) { ./main }
data 12345
PS C:\Users\MSI\Desktop\C program>

```

Figure-72

Data 12345 လိုပေါ်လာပါက ကျွန်တော်တို့ ရေးထားတဲ့ **function** သည် **myarr** ဆိုသည့် **array** ကို **int** အဖြစ်ပြောင်းလဲလိုက်တာ ဖြစ်ပါတယ်။

lesson-14-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson%2014>

Time_class

ဒီသင်ခန်းစာမှာကတော့ **time** နဲ့ ပါတ်သတ်တဲ့ သင်ခန်းစာအခါးကိုပြောပြပေးမှာ ဖြစ်ပါတယ့်။ ဥပမာ

Transcation_amount_limit_per_day ကို စစ်ဆေးဖို့ရန်အတွက် **time** နဲ့

ပါတ်သတ်တာရေးတဲ့အခါ အပြင်မှာ **24 hour** ဆိုရင် **program** မှုလဲ**24 hour** ကြာမှာ ဖြစ်ပါတယ့်။

ပထမဦးဆုံးအနေဖြင့် **time function**များကို တစ်စုတစ်စည်းထဲ ဖြစ်စေရန် **time class function**

တစ်ခုကို ဖန်တီးထားလိုက်ပါတယ့်။

```

void time_class(int user_index){
    time_class_last_record(user_index);
}

void time_class_last_record(int user_index){
    int last=0;
    int records=space_array[user_index]-19;
    for(int i=0;i<=records;i++){
        last=i;
    }
    printf("\n User Last Record: %s\n",db[user_index].tr[last].note);

}

```

အထက်ပါ program သည် User ၏ last record ကို ဖော်ပြသည့် program ဖြစ်ပါသည်။

User ၏နောက်ဆုံး record ကို လိုချင်တဲ့ အခါ user ၏ space array ထဲကနေ -1 ပါ။

နှင့်လိုက်တဲ့အခါ address ရဲအနောက်တွင်ရှိသော record အရေအတွက်ကို သိရမှာ ဖြစ်ပါတယ်။

ထိုအခါ records ကို looping ပါတ်ပီး နောက်ဆုံး record ကို သိအောင်လုပ်ပါမယ်။ ထို့နောက် printf

ဖွင့် db[user_index] ရဲ့ tr[last].note ကို ထုတ်လိုက်မှာ ဖြစ်ပါသည်။ program

အလုပ်လုပ်လားဆိုတာ ကျွန်တော်တို့စမ်းသပ်ကြည့်

ပါမယ့်။ အောက်ပါပုံအတိုင်း Main.c ထဲတွင် program ကို အနည်းငယ် ပြင်ကြည့်ပါး program ကို run ကြည့်ပါက user ၏ နောက်ဆုံး record result ထွက်လာမှာ ဖြစ်ပါတယ်။

The screenshot shows the Visual Studio Code interface. The left sidebar displays project files including `main.c`, `onlinebank.h`, and `whw.txt`. The main editor window shows the `main.c` code. The terminal window at the bottom shows the output of the program, which is a series of lines of text. The code in `main.c` includes comments and function definitions related to file operations and data processing.

```

main.c  C onlinebank.h  whw.txt
C main.c > main()
1  #include <stdio.h>
2  #include "onlinebank.h"
3
4  int main()
5  {
6      space_counter();
7      loading_from_file();
8      printing_all_data();
9      // main_menu();
10     // get_time();
11
12     // integer_to_charArrayFun(1500);
13     // printf("char array %s",int_to_charArray);
14
15     // char myarr[5]={'1','2','3','4','5'};
16     // unsigned int data=char_to_integer_fun(myarr);
17     // printf("data %u",data);
18     time_class();
19
20 }

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```

1-winhtut-9/paoola(N)80099-winhtut@gmail.com-admin@123-2208183876-aaa123aa123wh-aaa123wh-personal-1-1-10000-mmk-13000-T-5000-1-0_500000-pyinoolwin-5000-onlinebankadmin
2-winhtut-9/paoola(N)80099-htutline@gmail.com-admin@1d02-2208183875-aaa123aa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0_500000-pyinoolwin-5000-onlinebankadmin
3-winhtut-9/paoola(N)80099-winline@gmail.com-admin@1d02-2208183874-aaa123aa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0_500000-pyinoolwin-5000-onlinebankadmin
4-meir-9/paoola(N)80099-meir@gmail.com-rp@53KFC-99858904-wear-wearis-personal-1-1-10000-mmk-100000-T-100000-1-0_500000-pyinoolwin-5000-hellonewuser
5-kari-9/paoola(N)80099-kari@gmail.com-wg@#43D-875865-wear-wearis-personal-1-1-10000-mmk-10000-T-10000-1-0_500000-mandala-5000-hellonew
6-main-9/paoola(N)80099-main@gmail.com-sdsSAF12@#3-9984184-MEAR-meiris-personal-1-1-10000-mmk-12000-T-10000-1-0_500000-mandalay-5000-hello

last: record onlinebankadmin
PS C:\Users\MSI\Desktop\C program>

```

figure-73

Terminal တွင် **last record** **onlinebankadmin** လိုပေါ်လာသောအခါ **onlinebankadmin** သည်

index နံပါတ် **0** ၏ နောက်ဆုံး **last record** ဖြစ်ပါသည်။ **last record** ကို

သိရမည့်အကြောင်းအရင်းသည် ငါး **last record** တွင်ရှိသော အချိန်ပေါ်မှုတည်ပါး နောက်တစ်ကြိမ်

ငွေလွှဲနိုင်သည့် အချိန်ကို သတ်မှ တိနိုင်မှာဘဲ ဖြစ်ပါသည်။

Mon-Jul-03-11:00:03-2023

အထက်ပါစာကြောင်းထဲက **day,month,year** ကို ဆွဲထုတ်ပါး နောက်တစ်ကြိမ်ငွေလွှဲတဲ့အ

ပါ **get_time** ထဲကို ရောက်လာမယ့် အချိန်နှစ်ဦးယူး ငွေလွှဲနှင့်ကန့်သတ်တာတွေ အကောင် **level**

တင်ခိုင်းတာမျိုးလုပ်ခိုင်း လုပ်မှာဘဲဖြစ်ပါတယ်။ ပထမဆုံးအနေဖြင့် ငါးတိုက ကိုင်တွယ့် **get_time**

ထဲတွင် **code** အနည်းငယ်ပြောင်းလဲပါမည်။

```

while(!feof(fpstr2)){
    if(c=='
        '){

```

```

        time_space_counter++;

        if(time_space_counter==1){
            getCTime[0].current_time[index]='!';
            c=fgetc(fpstr2);
            index++;
        }else
            if(time_space_counter==4){
                getCTime[0].current_time[index]='@';
                c=fgetc(fpstr2);
                index++;
            }
        else{
            getCTime[0].current_time[index]='-';
            c=fgetc(fpstr2);
            index++;
        }
    }else{
        getCTime[0].current_time[index]=c;
        c=fgetc(fpstr2);
        index++;
    }
}

```

အရင်တုန်းက ကျွန်တော်တို့ **get_time function** တွင် **space** ကို တွေပါက နောက်ထပ်

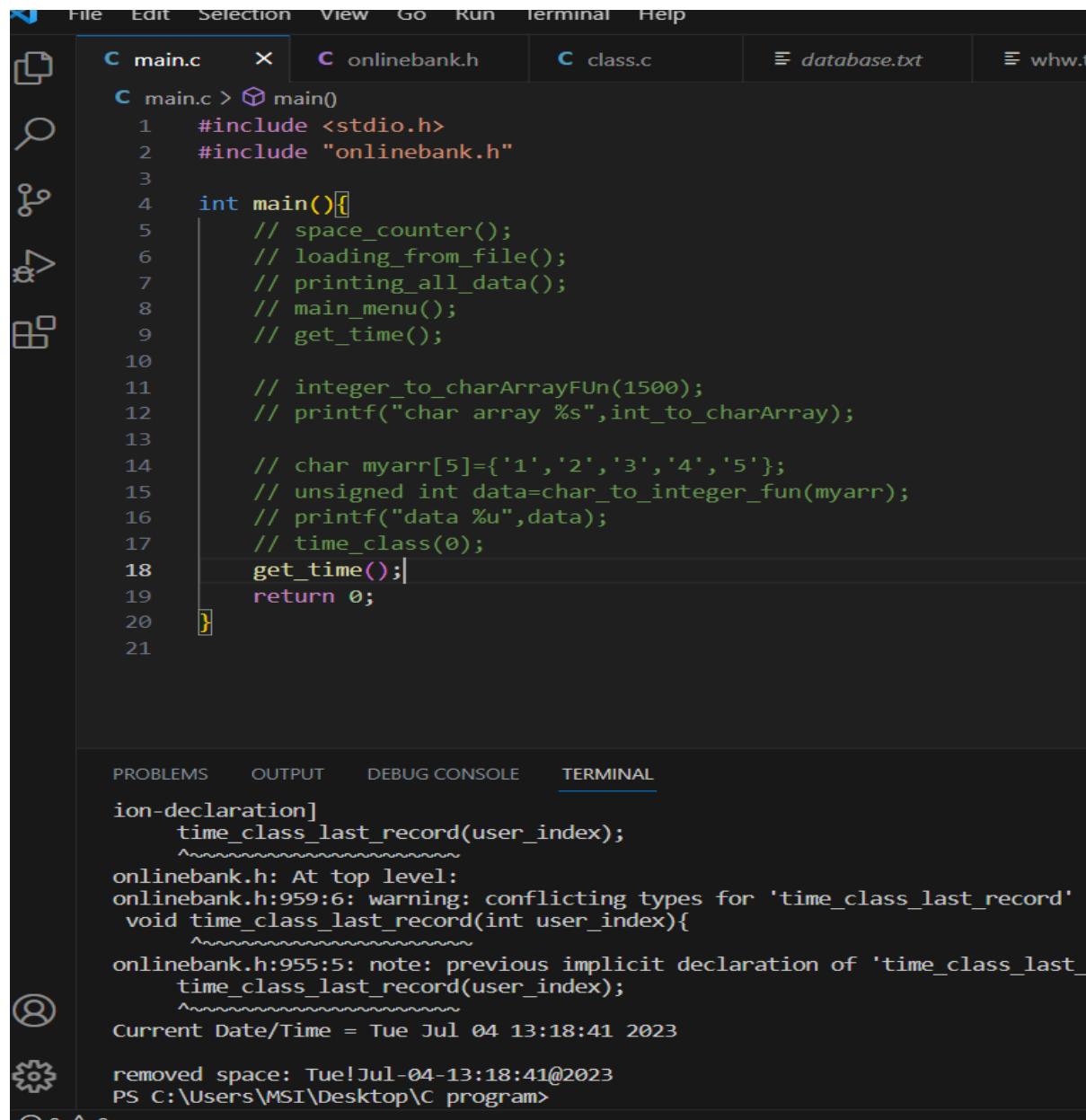
data(c) တစ်ခုကို ထပ်ယူလိုက်တာဖြစ်ပါး ယခုအခါမှာတော့ **space** ကို တွေ့တာနှင့်

time_space_counter ကို +1 ပေါင်းပေးလိုက်မှာ ဖြစ်ပါး အကယ်၍ **time_spce_counter**

သည် 1 နှင့်သာ ညီခဲ့မည်ဆိုပါက ငါး နေရာတွင် ! ကိုထားလိုက်မှာ ဖြစ်ပါသည်။ ဆိုလိုရင့်ဝေးသည်**Mon-**

Jul-03-11:00:03-2023တွင်**Mon**၏အနောက်တွင် ! ထားလိုက်ခြင်း

ဖြစ်ပါသည်။ ထို့နောက် **time_space_counter** သည် 4 နှင့် ညီလဲပါက @ ကို ဂုဏ်ရေးနေရာတွင် အစားထိုးလိုက်ပါမည်။ ငွေမြောက် **space** သည် 2023 ၏အရှေ့တင်ရှိသော နေရာဖြစ်ပါသည်။



```

File Edit Selection View Go Run Terminal Help
main.c x onlinebank.h class.c database.txt whw.t

main.c > main()
1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     // space_counter();
6     // loading_from_file();
7     // printing_all_data();
8     // main_menu();
9     // get_time();
10
11    // integer_to_charArrayFun(1500);
12    // printf("char array %s",int_toCharArray);
13
14    // char myarr[5]={'1','2','3','4','5'};
15    // unsigned int data=char_to_integer_fun(myarr);
16    // printf("data %u",data);
17    // time_class();
18    get_time();
19    return 0;
20}
21

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL
ion-declaration]
    time_class_last_record(user_index);
    ^
onlinebank.h: At top level:
onlinebank.h:959:6: warning: conflicting types for 'time_class_last_record'
    void time_class_last_record(int user_index){
    ^
onlinebank.h:955:5: note: previous implicit declaration of 'time_class_last_
    time_class_last_record(user_index);
    ^
Current Date/Time = Tue Jul 04 13:18:41 2023
removed space: Tue!Jul-04-13:18:41@2023
PS C:\Users\MSI\Desktop\C program>

```

Figure-74

main.c ထဲတွင် **Program** ကို အထက်ပါအတိုင်း အနည်းငယ် ပြင်ဆင်ပါး **run** ကြည့်သောအ

၅။ Terminal တွင် removed space :Tue!Jul-04-13:18:41@2023 ဆိုပါးပေါ်လာသည်ကို
မြင်ရပါလိမ့်။ အရင်ရေးသားထားပီးသော time_class_last_record function တွင် code
ထပ်ထည့်ပေါင်းကြပါမည်။

```
void time_class_last_record(int user_index){
    int last=0;

    int records=space_array[user_index]-19;
    for(int i=0;i<=records;i++){
        last=i;
    }
    printf("\n User Last Record: %s\n",db[user_index].tr[last].note);

    time_class_get_date(db[user_index].tr[last].note);}
}
```

အထက်ပါ function တွင် အသစ်တည်ဆောက်ထားသည့် time_class_get_date ကို ခေါ်လိုက်မှာ
ဖြစ်ပါသည်။

```
void time_class_get_date(char last_record[]) {
    // to get month name , day , year
    int last_record_counter = char_counting(last_record);
    int i = 0;
    for (i = 0; i < last_record_counter; i++) {

        if (last_record[i] == '!') {
            break;
        }
    }
    i++;
    for (int a = 0; a < 3; a++) {
```

```

month[a] = last_record[i];
i++;
}

printf("\nget_month %s\n", month);

day[0] = last_record[i + 1];
day[1] = last_record[i + 2];

unsigned int get_day = char_to_integer_fun(day);

printf("get day: %d\n", get_day);

```

ဒီ**Function** ၏ရည်ရွယ်ချက်တော့ **user** ၏ **last record** တွင် **day,month,year** ကို
ထုတ်ဖော်သည့် **function** ဖြစ်ပါသည်။ ငှင့် **function** ကို **parameter** အနေဖြင့် **char array** တစ်ခု
ဖြတ်ထားပါသည်။ ငှင့် **parameter** သည် **time_class_last_record** မှ **user**
၏ နောက်ဆုံး **record** မှ ဖြတ်လာခြင်း ဖြစ်သည်။ ထို့နောက် ငှင့် **record** တွင် စာလုံးအရေအတွက်
မည်မျှရှိသည်ဆိုတာ သိရှိရန်အတွက် **char counting function** ကို သုံးပီး ရရှိလာသည့် **data** ကို
integer **last_record_counter** ထဲတွင် သိမ်းဆည်းထားလိုက်ပါသည်။ ထို့နောက်
last_record_counter အတိုင်း **Looping** ပါတယ်ပါး **last_record[i]** သည် ‘!’ နှင့် တူညီခဲ့ပါက **break**
လုပ်လိုက်မှာ ဖြစ်ပါသည်။

Wed Jul-05-08:35:54@2023

ထို့နောက် ! ၏အနောက်တွင်ရှိသော **month** ကို ထုတ်ယူမှာဘဲ ဖြစ်ပါသည်။
i++ လုပ်ပေးရခြင်းအကြောင်းအရင်းမှာ **i**၏ တန်ဖိုးသည် ! ကိုတွေ့တုန်းက ရပ်တန်ထားပါး
နောက် ငှင့်တန်ဖိုးကို ဆက်လက်တိုးစေရန်ဖြစ်ပါသည်။ **month** တွေကို ဖော်ပြရာတွင် အများဆုံး

စာလုံး ၃လုံးဖြင့်သာ ဖော်ပြပါသည်။ ဥပမာ-(**jul,fab,mar**)

ထို့ကြောင့် **loop** ကို ၃လုံးသာပါတ်ပီး ရလာသော **last_record[i]** ကို **month []** ထဲတွင် သိမ်းဆည်းလိုက်ပါသည်။

```
char month[3];
char day[2];
char year[4];
```

အထက်ပါ **char array** ၃ခုကို **global variable** အဖြစ် အပေါ်တွင် ကြေညာထားလိုက်ပီး ဖြစ်ပါသည်။

```
printf("\nget_month %s\n", month);
ထို့နောက် month ကို %s ဖြင့် output ထုတ်ထားလိုက်ပါသည်။ Month ကို ရှာပီးပါက day အားဆက်လက်ရှာဖွေပါမည်။ day သည် ကိုန်းကဏ္ဍားဖြစ်သည့်အတွက် ငှါးကို char array ဖြင့် မထုတ်တော့ဘဲ integer အဖြစ် ထုတ်ယူမှာဘဲဖြစ်ပါသည်။ char array မှ integer အဖြစ်သို့ပြောင်းလဲသည့် function ကို အရင်သင်ခန်းစာတွင် ပြသခဲ့ပါးဖြစ်သည့်အတွက် ငှါး function အကြောင်းမသိသေးပါက အရင်သင်ခန်းစာများကို ကြည့်ရှုပါးလေ့လာနိုင်ပါသည်။ day ၏ အများဆုံးကိုန်းကဏ္ဍားသည် ၂လုံးသာဖြစ်ပါသည်။ day[0] နေရာတွင် last_record[i+1] ပေါင်းထည့်ပါး day[1] တွင် last_record[i+2] ပေါင်းထည့်ပါး day array အား char_to_integer ဖြင့် ပြောင်းလဲကာ unsigned int get_day ထဲတွင် သိမ်းထားလိုက်ပါသည်။


ထို့နောက် year ကို ဆက်လက်ထုတ်ယူမှာ ဖြစ်ပါသည်။


```

```
// for finding year
```

```
int z = 0;
```

```

for (z = 0; z < last_record_counter; z++) {

    if (last_record[z] == '@') {
        break;
    }

}

for (int x = 0; x < 4; x++) {
    z++;
    year[x] = last_record[z];
}
unsigned int get_year = char_to_integer_fun(year);

printf("get_year :%d\n", get_year);

```

Year ကိုလည်း **char array** မှ **Integer** သို့ ပြောင်းလဲမှာ ဖြစ်ပါတယ်။ **loop** ကို **last_record_counter** အတိုင်း ပါတ်ပီး အကယ်၍ **last_record[z]** သည် @ နှင့်သာ ညီးပါက **break** လုပ်လိုက်ပါမည်။ ထို့နောက် နောက်တစ်ကြိမ် **loop** ပါတ်ပီး **year[x]** ထဲသို့ **last record[z]** ကို ပေါင်းထည့်ထားလိုက်မည် ဖြစ်ပါသည်။ ထို့နောက် **array** မှ **integer** သို့ ပြောင်းလဲရန်အတွက် **char_to_integer function** ကို သုံးပီး **get_year** ထဲသို့ သိမ်းလိုက်ပါမည်။

main.c ထဲတွင် **program** ကို အနည်းငယ်ပြင်ပီး **run** ကာ **program** အလုပ်ဖြစ်လားဆိုတာ စမ်းသပ်မှာ ဖြစ်ပါသည်။

```

1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     space_counter();
6     loading_from_file();
7     printing_all_data();
8     // main_menu();
9     // get_time();
10
11    // integer_to_charArrayFun(1500);
12    // printf("char array %s",int_to_charArray);
13
14    // char myarr[5]={'1','2','3','4','5'};
15    // unsigned int data=char_to_integer_fun(myarr);
16    // printf("data %u",data);
17    time_class();
18
19    return 0;
20 }

```

figure-75

```

1-winhtut-/paola(N)80099-winhtut@gmail.com-admin123-2208183876-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-10000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin->From winhtut-to-main-Wed!Jul-05-08:35:54@2023
2-winhtut-/paola(N)80099-hthtling@gmail.com-admin123@02-2208183875-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
3-winhtut-/paola(N)80099-winhtut@gmail.com-admin123@02-2208183874-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
4-mean-9/paola(N)80099-mean@gmail.com-rg@53KFC-99858904-wear-wearis-personal-1-1-10000-mmk-10000-T-10000-1-0.50000-pyinoolwin-5000-hellonewuser
5-kari-9/paola(N)80099-kari@gmail.com-w@#143DD-875865-wear-wearis-personal-1-1-10000-mmk-10000-T-10000-1-0.50000-mandala-5000-hellonew
6-main-9/paola(N)80099-main@gmail.com-sdsSAF12@#3-9984184-MEAR-meanis-personal-1-1-10000-mmk-15000-T-10000-1-0.50000-mandalay-5000-hello-main->ReceiveFrom winhtut-Wed!Jul-05-08:35:54@2023

User Last Record: From winhtut-to-main-Wed!Jul-05-08:35:54@2023
get_month Jul
get_day: 5
get_year :202305
PS C:\Users\WSI\Desktop\C program>

```

Figure-76

program အား run ကြည့်လိုက်သောအခါတွင် figure-76 တွင် ပေါ်လာသည့်အတိုင်း

user_last_record တွင် ! နှင့် @ သည် space 1 နှင့် 4 နေရာတွင်ပေါ်လာပါ။ ။day,month

နှင့် year မှာလဲ Jul,5,2023 ဆိုပါး ပေါ်လာသည်ကို မြင်ရမှာ ဖြစ်ပါသည်။

lesson-15-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson-15>

Amount_transcation_record

Time အတွက် လိုအပ်သည့် **function**များ ရေးသားပါက **user** မှ **receiver** ထံသို့ လွှဲပေးသည့် **function** ကို ဆက်လက်ရေးသားပါမည်။ ပထမဦးစွာ **transmitter** ဘက်တွင် အရင်ရေးသားသည့်အခါ ငွေလွှဲသည့် **process** ကို **transcation_record** ထဲတွင် ရေးသားသည့်အတွက် လွှဲပြောင်းသည့် ငွေပောက် **record** ကိုလဲ ငြင်းထဲတွင် **time** ၏အနေကိုမှာ ရေးသားမှာ ဖြစ်ပါသည်။

```
void transacitonRecord(int userDBIndex,int uFound,char who,unsigned int amount){

    integer_to_charArrayFUn(amount);
    int money_quant_counter=char_counting(int_toCharArray);
```

အရင် **transcation_record** ရေးသားခဲ့တုန်းက **parameter** ၃ခု ထဲသာလျှင် ပါရှိတာဖြစ်ပါး ဒီ **function** ကို ခေါ်ခဲ့တာက **transfer_money** မှ ခေါ်ခဲ့တာ ဖြစ်ပါသည်။ ထို့ကြောင့် **amount** ကို သိဖို့ရန်အတွက် **transcationrecord** ထဲတွင် **unsigned int amount** ကို ထပ်ပါး ထည့်ထားပါသည်။ ဒီ **function** တွင် ပြင်ထားသလို **global** တွင် ကြညာထားတာ ကိုလည်း ပြင်ပေးရန် လိုအပ်ပါသည်။ ထို့နောက် ငြင်း **amount** ကို **integer** မှ **char array** သုံးဝို့ ပြောင်းလဲပေးရန် အတွက် **integer_toCharArray** ဖြင့် ပြောင်းလဲလိုက်ပါသည်။ **array** အဖြစ်သို့ပြောင်းလဲပါးသောအခါ ငြင်း **array** ထဲတွင် **data** မည်မျှရှိနေလဲဆိုတာ သိဖို့ရန်အတွက် **char_counting function** ကို သုံးပါး **money_quant_counter** ထဲတွင် သိမ်းထား လိုက်ပါသည်။

```

= encrypted_data.txt      882
c main.c                 883
E main.exe                884
E mytime.txt              885
E onlinebank.exe          886
C onlinebank.h            887
project flow               888
c seche.c                  889
C secheader.h              890
c secmain.c                  891
E secmain.exe              892
E secmain.exe              893
C tempCodeRunnerFile.c     894
E whw.txt                  895
896
897
898
899
900
901
902
903

```

```

    get_time();
    db[userDBIndex].tr[space_array[userDBIndex]-19].note[endPoint]=':';
    endPoint++;
    for(int win=endPoint; win<25+endPoint ; win++ ){
        db[userDBIndex].tr[space_array[userDBIndex]-19].note[win]=getCTime[0].current_time[nameIndex];
        nameIndex++;
    }
    endPoint=endPoint+25;
    db[userDBIndex].tr[space_array[userDBIndex]-19].note[endPoint]='$';
    endPoint++;
    nameIndex=0;
    for(int wa=endPoint;wa<endPoint+money_quant_counter;wa++){
        db[userDBIndex].tr[space_array[userDBIndex]-19].note[wa]=int_toCharArray[nameIndex];
        nameIndex++;
    }
    db[userDBIndex].tr[space_array[userDBIndex]-19].note[endPoint+money_quant_counter]='-';
    space_array[userDBIndex]= space_array[userDBIndex]+1;
} else{
    char receiveFrom[12] = {0x41, 0x42, 0x43, 0x44, 0x45, 0x46, 0x47, 0x48, 0x49, 0x4A, 0x4B, 0x4C};
}

```

Figure-77

```

endPoint=endPoint+25;

db[userDBIndex].tr[space_array[userDBIndex]-19].note[endPoint]='$';
endPoint++;

nameIndex=0;
for(int wa=endPoint;wa<endPoint+money_quant_counter;wa++){
    db[userDBIndex].tr[space_array[userDBIndex]-19].note[wa]=int_toCharArray[nameIndex];
    nameIndex++;
}

db[userDBIndex].tr[space_array[userDBIndex]-19].note[endPoint+money_quant_counter]='-';
space_array[userDBIndex]= space_array[userDBIndex]+1;

```

Program የደንብ ማረጋገጫ

time አይቀ኏ል፡፡ የሸፍ፡፡ ዘመኑ፡፡ የሚቀርቡት ተክና ይፈጸማል፡፡ **endPoint** ጥሩ 25 በፊርድ፡፡ ዘመኑ፡፡ የሚቀርቡት ተክና ይፈጸማል፡፡

db[userDBIndex] ነገር **tr[space_array]** ጥሩ 19 ቀናት፡፡ **note** ነገር **endPoint** አለው፡፡ **\$ sign** ጥሩ በፊርድ፡፡ ዘመኑ፡፡ የሚቀርቡት ተክና ይፈጸማል፡፡ **endpoint** ጥሩ ++ ተከታታለሁ፡፡ **nameIndex** ጥሩ 0 በፊርድ፡፡

endPoint ቀናት የ**Money_quant_counter** ጥሩ በፊርድ፡፡ ፍርድ፡፡ ዘመኑ፡፡ የሚቀርቡት ተክና ይፈጸማል፡፡

የተቀባዩ የ**note[endPoint+money_quant_counter]** አለው፡፡ **int_toCharArray[nameIndex]**

```
 923     reIndex=0;
 924     get_time();
 925     for(int win=toendpoint; win<25+toendpoint ; win++ ){
 926
 927         db[uFound].tr[space_array[uFound]-19].note[win]=getTime[0].current_time[reIndex];
 928         reIndex++;
 929     }
 930
 931     toendpoint=25+toendpoint;
 932     db[uFound].tr[space_array[uFound]-19].note[toendpoint]='$';
 933     toendpoint++;
 934
 935     for(int gcc=0;gcc<money_quant_counter;gcc++){
 936         db[uFound].tr[space_array[uFound]-19].note[toendpoint]=int_toCharArray[gcc];
 937         toendpoint++;
 938     }
 939
 940     db[uFound].tr[space_array[uFound]-19].note[toendpoint]='-';
 941     space_array[uFound] +=1;
 942
 943     }
 944
 945 }
 946
 947 void integer_to charArrayFun(unsigned int integer){
```

figure-78

```
toendpoint=25+toendpoint;

    db[uFound].tr[space_array[uFound]-19].note[toendpoint]='$';

    toendpoint++;



for(int gcc=0;gcc<money_quant_counter;gcc++){
    db[uFound].tr[space_array[uFound]-
19].note[toendpoint]=int_toCharArray[gcc];
    toendpoint++;
}

db[uFound].tr[space_array[uFound]-19].note[toendpoint]='-';
```

അയന്ത്രിക്കുന്ന **program** കൂടി **receiver** വരുത്തുന്ന **time** എൻ അയന്ത്രിക്കുന്ന ഭേദമിലെ **program**

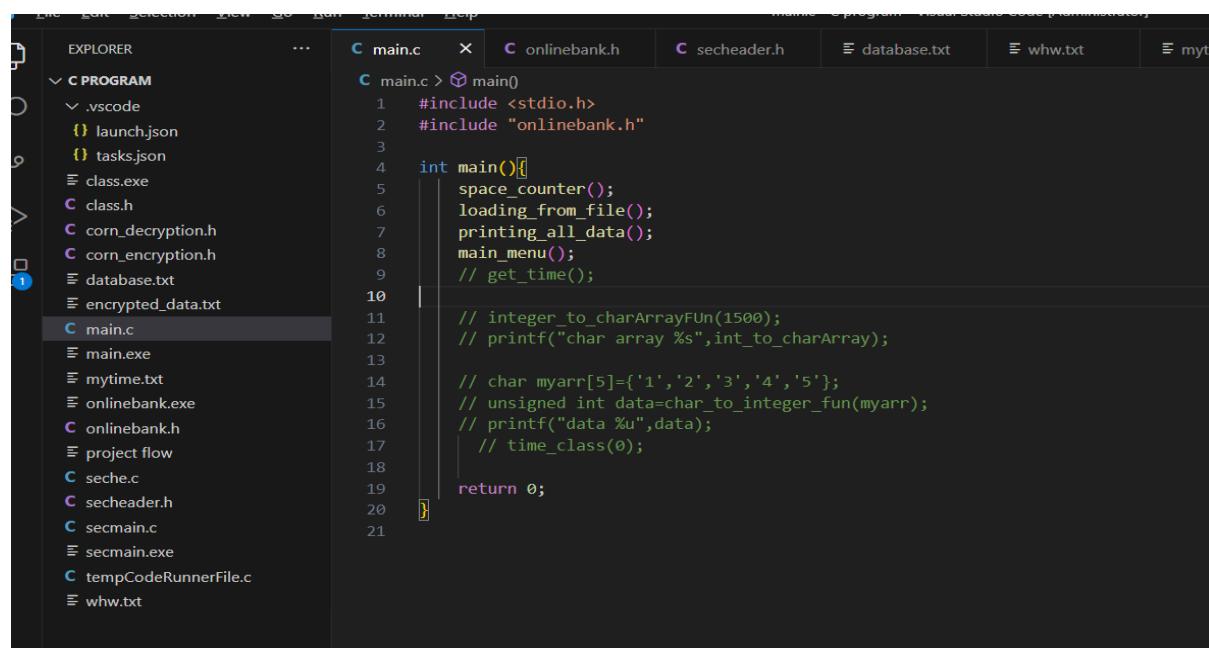
Run രീതി: അലൂപ്പുത്തുവലാഃ ഫൈൽ ഒരു വർഷിക്കുന്ന പിബൽ|| **main.c** തുടർന്ന് അയന്ത്രിക്കുന്ന ആവശ്യിക്കുന്ന

പ്രോഗ്രാം രീതി: **program run** പിബൽ|| തീരുമാനിക്കുന്ന **Login** ഓട്ടോമേറ്റിക് **user** ഫോറോന്റ്

Login ഓട്ടോമേറ്റിക് **user** കൂടിയിൽ ചേരുന്നതിനും വൈഡോം ഫോറോന്റ് **figure -80** തുടർന്ന് **program**

അലൂപ്പുത്തുവലാഃ മുൻപുമുന്ന് പ്രോഗ്രാം രീതി

lesson-16-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson16>



The screenshot shows the Visual Studio Code interface with the following details:

- File Explorer:** Shows a tree view of files under the folder "C PROGRAM". Files listed include .vscode, launch.json, tasks.json, class.exe, class.h, corn_decryption.h, corn_encryption.h, database.txt, encrypted_data.txt, main.c, main.exe, mytime.txt, onlinebank.exe, onlinebank.h, project flow, seche.c, secheader.h, secmain.c, secmain.exe, tempCodeRunnerFile.c, and whw.txt.
- Code Editor:** The main editor window displays the code for "main.c". The code is as follows:

```

1 #include <stdio.h>
2 #include "onlinebank.h"
3
4 int main(){
5     space_counter();
6     loading_from_file();
7     printing_all_data();
8     main_menu();
9     // get_time();
10
11    // integer_to_charArrayFun(1500);
12    // printf("char array %s",int_toCharArray);
13
14    // char myarr[5]={'1','2','3','4','5'};
15    // unsigned int data=char_to_integer_fun(myarr);
16    // printf("data %u",data);
17    // time_class(0);
18
19    return 0;
20}
21

```

figure-78

```

File Edit Selection View Go Run Terminal Help main.c - C program - Visual Studio Code [Administrator]
EXPLORER ... C main.c x C onlinebank.h C secheader.h database.txt whw.txt mytime.txt
C main.c > main()
1 #include <stdio.h>
2 #include "onlinebank.h"
3
n
3-winhtut:9/paoola(N)80099-winline@gmail.com-admin123@2-2208183874-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
4-mear-9/paoola(N)80099-mear@gmail.com-rge@53FC-99858904-wear-wearis-personal-1-1-10000-mmk-10000-T-10000-1-0.50000-pyinoolwin-5000-hellonewuser
5-kari-9/paoola(N)80099-kari@gmail.com-wg@#A3D-875865-wear-wearis-personal-1-1-10000-mmk-10000-T-10000-1-0.50000-mandala-5000-hellonew
6-main-9/paoola(N)80099-main@gmail.com-sdsAF12@#3-9984184-MEAR-mearis-personal-1-1-10000-mmk-16000-T-10000-1-0.50000-mandalay-5000-hello
Welcome to our OnlineBank NCC!
Press 1 to Login!
Press 2 to Register!
Press 3 to Exit!>:1
This is Online Bank Login!
Enter your email to login!>:winhtut@gmail.com
Enter your password to login!>:admin123
Welcome My/S : winhtut
Your Current Amount : 9000
Press 1 To Transfer Money;
Press 2 To Withdraw :
Press 3 To Cash In:
Press 4 to get your Transaction Record :
Press 5 To Loan:
Press 7 to get Main Menu:
Press 6 To Exit: nEnter your option:1
Enter receiver phone number:>:9984184
Enter amount to send for main : main@gmail.com:>:1000
Your current amount 9000
Transfer amount 1000 :
Enter your password to confirm for transaction:>:admin123
Current Date/Time = Thu Jul 06 08:59:35 2023
Current Date/Time = Thu Jul 06 08:59:35 2023
1-winhtut:9/paoola(N)80099-winhtut@gmail.com-admin123-2208183874-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-8000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin-Fr
on-winhtut-to-main-Thu Jul 06 08:59:35@2023$1000
2-winhtut:9/paoola(N)80099-huttleline@gmail.com-admin1ld02-2208183875-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
n
3-winhtut:9/paoola(N)80099-winline@gmail.com-admin123@2-2208183874-aaa123aaa123wh-aaa123wh-personal-1-1-10000-mmk-15000-T-5000-1-0.50000-pyinoolwin-5000-onlinebankadmin
4-mear-9/paoola(N)80099-mear@gmail.com-rge@53FC-99858904-wear-wearis-personal-1-1-10000-mmk-10000-T-10000-1-0.50000-pyinoolwin-5000-hellonewuser
5-kari-9/paoola(N)80099-kari@gmail.com-wg@#A3D-875865-wear-wearis-personal-1-1-10000-mmk-10000-T-10000-1-0.50000-mandala-5000-hellonew
6-main-9/paoola(N)80099-main@gmail.com-sdsAF12@#3-9984184-MEAR-mearis-personal-1-1-10000-mmk-17000-T-10000-1-0.50000-mandalay-5000-hello-main-ReceiveFrom-winhtut-Thu
u1-06-08:59:35@2023$1000.
Welcome My/S : winhtut
Your Current Amount : 8000
Ln 15, Col 53 Spaces:2 UTF-8 CRLF {} C Win32 R

```

Figure-79

```

yinoolwin 5000 onlinebankadmin From-winhtut-to-main:-Thu!Jul-06-08:59:35@2023$1000-
00 pyinoolwin 5000 onlinebankadmin
pyinoolwin 5000 onlinebankadmin
hellonewuser
n
lo main-ReceiveFrom-winhtut-Thu!Jul-06-08:59:35@2023$1000-

```

Figure-80

Transcation_amount_limit_per_day

Transcation_amount_limit_per_day အကြောင်း အနည်းငယ် ရှင်းပြပါက အကယ်၍

User ၏ အကောင့် level သည် 1 ဖြစ်နေပါက သတ်မှတ်ထားသည့် limit ကိုသာ လွှဲရမှာ ဖြစ်ပါသည်။

ထိုသို့ Limit ကို သိရန်အတွက် ကျွန်တော်တို့သည် user ၏ နောက်ဆုံးလွှဲခဲ့သည့် record ကိုဘဲ သိရမှာ

မဟုတ်ဘဲ 24 hour အတွင်းလွှဲခဲ့သည့် record အရေအတွက် သိဖို့ရန်လိုအပ်ပါသည်။ ထိုအခါမှသာ

user မှ ယနေ့အတွင်း ငွေ amount မည်မျှလွှဲခဲ့သည်ကို သိနိုင်မှာ ဖြစ်ပါသည်။ ငွေလွှဲသည့် process

ကို ရေးသားထားသည်မှာ

money_transcation ထဲတွင် ရေးသားထားသည့်အတွက် **amount_limit** ဖို့ရန်အတွက် ငှါးထဲတွင်

code အနည်းငယ် ထည့်သွင်းကြပါမည်။

```
time_class_last_record(transmit);
unsigned int last_day = char_to_integer_fun(day);
unsigned int last_year = char_to_integer_fun(year);
unsigned int last_money = char_to_integer_fun(money);
copy_two_char_array(_month[0].str_month,month);

//  
/*To calculate transaction record for all same days....battery30% */
```

```
int total_amount_tranferred =
total_money_for_same_days(transmit,last_day);
total_amount_tranferred = total_amount_tranferred+amount;
```

ပထမဦးစွာ **money_transcation** ၏ အပေါ်ဘက်တွင် **Month** အတွက် **structure**

တစ်ခုဆောက်ထားပီး ငှါး **Month** ထဲတွင် **char array** တစ်ခုထည့်ထားလိုက်ပါသည်။

time_class_last_record function ကို ခေါ်ပီးနောက် ငှါးထဲမှ **day,year** ကို **integer** သို့

Function သုံးပီး ပြောင်းလဲလိုက်ပါသည်။ **money** အတွက် **global** တွင် **char money[10];**

ဆိုပီး ကြေညာထားပီး ဖြစ်ပါသည်။ **month** အတွက်ကိုလဲ **monyh[0].str_month** ထဲသို့

copy ကူးထည့်ထားလိုက်ပါသည်။ ထို့နောက်မှတော့ **user** ၏ တစ်နေ့လုံးတွင် သုံးစွဲခဲ့

သည့် **amount** ပမာဏကို သိဖို့ရန်အတွက် **total_money_for_same_day** ဆိုသည့် **function** ကို
ရေးသားပါမညှာ။

```

int total_money_for_same_days(int transmit,int last_day){

    int to_return_money=0;
    int index_counter=0;

    // to calculate temp day

    int to_get_all_records=space_array[transmit]-20;

    for(int wa=to_get_all_records-1; wa>0; wa--){
        //to calcuate temp day

        char temp_day[2];

        //to count char for current note
        int current_trans_counter =char_counting(db[transmit].tr[wa].note);

        for(int aw=0; aw<current_trans_counter; aw++){

            if(db[transmit].tr[wa].note[aw]=='!'){
                break;
            }
            index_counter++;
        }

        temp_day[0]=db[transmit].tr[wa].note[index_counter+5];
        temp_day[1]=db[transmit].tr[wa].note[index_counter+6];
    }
}

```

```
unsigned int temp_day_int=char_to_integer_fun(temp_day);
printf("\n temp day for %d record = %d\n",wa,temp_day_int);

if(temp_day_int!=last_day){
    break;
} else{

    for(int m=index_counter; m<current_trans_counter; m++){
        if(db[transmit].tr[wa].note[m]=='$'){
            break;
        }
        index_counter++;
    }

    int quantity_of_money=0;
    char current_amount_char[10];
    for(int ko=index_counter; ko<current_trans_counter; ko++){
        if(db[transmit].tr[wa].note[ko]=='-'){
            break;
        }
        quantity_of_money++;
    }

    index_counter++;
    for(int v=0; v<quantity_of_money; v++){

        current_amount_char[v]=db[transmit].tr[wa].note[index_counter];
        index_counter++;
    }

}
```

```

        unsigned int amount_for_current_trans =
char_to_integer_fun(current_amount_char);
        printf("Amount for current
transaction: %d\n",amount_for_current_trans);
        index_counter=0;
        to_return_money = to_return_money+amount_for_current_trans;

    }

}

printf("\nTotal Money except last record's money for all same
days: %d\n",to_return_money);
return to_return_money;
}

```

Program ဂျင်းလွင်းချက်

```

int to_get_all_records=space_array[transmit]-20;
for(int wa=to_get_all_records-1; wa>0; wa--){
    //to calculate temp day
    char temp_day[2];
    //to count char for current note
    int current_trans_counter =char_counting(db[transmit].tr[wa].note);

    for(int aw=0; aw<current_trans_counter; aw++){
        if(db[transmit].tr[wa].note[aw]=='!'){
            break;
        }
        index_counter++;
    }
}

```

```
}
```

င်းကို **parameter** ခုဖြတ်ထားပါး **record** အားလုံးသိမ့်ရန်အတွက် **space_array** မှ 20 ကို နှစ်ထားလိုက်ပါသည်။ **loop** စဉောအခါ **to_get_all_record-1** မှ စပီး 0 ထက် ကြီးသည်အထိ **looping** ပါတ်ထားပါမည့်။ ထို့နောက် **db[transmit].tr[wa]** ၏ **note** သည် စာလုံးအရေအတွက်မည်မျှ ရှိသည်ကို သိမ့်ရန်အတွက် **char_counting** ကို သုံးပါး **current_trans_counter** ထဲတွင် သိမ်းထားလိုက်ပါသည်။ အထက်ပါ **code** သည် ယနေ့တွင် **user** ၏ပထမဆုံးလွှဲထားသည့် **record** ဖြစ်ပါသည်။ **current_trans_counter** အတိုင်း **loop** ပါတ်သောအခါ '!' ကို တွေ့ပါက **break** လုပ်လိုက်မှာ ဖြစ်ပါး '!' ကိုမတွေ့မချင်း **index_counter** ကို ++ တိုးပေးနေမှာ ဖြစ်ပါသည်။

```
temp_day[0]=db[transmit].tr[wa].note[index_counter+5];
temp_day[1]=db[transmit].tr[wa].note[index_counter+6];
unsigned int temp_day_int=char_to_integer_fun(temp_day);
printf("\n temp day for %d record = %d\n",wa,temp_day_int);

if(temp_day_int!=last_day){
    break;
} else{

    for(int m=index_counter; m<current_trans_counter; m++){
        if(db[transmit].tr[wa].note[m]=='$'){
            break;
        }
        index_counter++;
    }
}
```

ထို့နောက် **user** အနေဖြင့် မိမိလွှဲသည့် နေ့ရက်ကို သိစေရန်အတွက် **temp_day**၏

Index 0 နှင့် **1** နှင့် **5** နှင့် **6** ကို ထပ်ပေါင်းလိုက်သောအခါ လက်ရှိ **current** နေ့ရက်
ကို ရရှိမှာ ဖြစ်ပါသည်။ ထို့နောက် ငြင်းကို **char_array** မှ **integer** သို့ ပြောင်းလဲပီး **temp_day_int**
ထဲသို့ ပေါင်းထည့်ထားလိုက်ပါသည်။ ငြင်း **temp_day_int** နှင့် **last_day**
သည် မတူညီခဲ့ပါက **break** လုပ်လိုက်မှာ ဖြစ်ပီး **current** ရက်စွဲသာ တူညီနေခဲ့ပါက **Loop**
ကို **index_counter** အတိုင်းစပီး **current_trans_counter** အောက် ငယ်သည်အထိ **m++**
တိုးသွားမည် ဖြစ်ပါသည်။ **loop** ထဲတွင် အကယ်၍ **db[transmit].tr[wa].note[m]** သည်
'\$' နှင့်သာ တူညီသည့် နေရာသို့ရောက်သွားပါက **break** လုပ်လိုက်မှာ ဖြစ်ပီး မတူမချင်း
Index_counter ကို ++ တိုးသွားမှာ ဖြစ်ပါသည်။

```
int quantity_of_money=0;
char current_amount_char[10];
for(int ko=index_counter; ko<current_trans_counter; ko++){
    if(db[transmit].tr[wa].note[ko]=='-'){
        break;
    }
    quantity_of_money++;
}

index_counter++;
```

အထက်ပါ **code** တွင် **int variable** တစ်ခုနှင့် **current_amount_char array** တစ်ခုကို
တည်ဆောက်ထားပီး နောက် **loop** ကို **index_counter** မှ စကာ **current_trans_counter**
အောက်ငယ်သည်အထိ **ko** ကို ++တိုးပေးထားလိုက်ပါသည်။ ထို့နောက် **note[ko]** သည် '-'

နှင့်သာ ညီခဲ့ပါက **break** လုပ်လိုက်မှာ ဖြစ်ပါသည်။ **quantity_of_money** နှင့် **index_counter** ကို
++ တိုးပေးလိုက်မှာ ဖြစ်ပါသည်။

```
for(int v=0; v<quantity_of_money; v++){
    current_amount_char[v]=db[transmit].tr[wa].note[index_counter];
    index_counter++;
}
```

ထို့နောက် **looping** ပါတ်ပိုး **current_amount_char** ထဲသို့ **db[transmit].tr[wa].note[ind**
ex_counter] ကို ပေါင်းထည့်လိုက်မှာ ဖြစ်ပါသည်။

```
unsigned int amount_for_current_trans =
char_to_integer_fun(current_amount_char);
printf("Amount for current
transaction: %d\n",amount_for_current_trans);
index_counter=0;
to_return_money = to_return_money+amount_for_current_trans;

}

}

printf("\nTotal Money except last record's money for all same
days: %d\n",to_return_money);
return to_return_money;
}
```

ထို့နောက် ငါး **current_amount_char** ကို **integer** အဖြစ်ပြောင်းလိုက်ပါမည်။ ထို့နောက် အားလုံးပေါင်း ငွေပမာဏကို

သို့ ရန် အတွက် **to_return_money** ထဲသို့ **to_return_money+amount_for_current_**

Trans ကို **loop** မပါးမချင်း(**record** မကုန်မချင်း) ပေါင်းထည့်ပေးနေမှာ ဖြစ်ပါသည်။

အထက်ပါ **function**ရေးသားပါက **user** ၏ တစ်နေ့လုံးငွေလွှဲခဲ့သည့် **record** စုစုပေါင်းကို ရှိမှာ ဖြစ်ပါသည်။

lesson-17-><https://github.com/NationalCyberCityNiST/Bank-Project-With-C/tree/main/lesson-17>

```
get_amount(transmit); // to get trans amount limit per pay
to_transfer_money=total_amount_transferred;
```

ထို့နောက် **get_amount function** ကို ဆက်လက် ရေးသားလိုက်ပါမည်။ ငါး **function** သည် **user** ၏ **account type** ပေါ်မှုတည်ပါး **user** က တစ်ရက်လျှင် ငွေမည်မျှ သုံးစွဲနှင့် သည်ကို ဖော်ပြုမည့် **function** ဖြစ်ပါသည်။

```
void get_amount(int user_index){

    int p_or_b = db[user_index].account_type;
    int acc_level = db[user_index].account_level;

    switch (p_or_b) {
        case 1:
            if(acc_level==1){
                trans_limit=5000;
            } else if(acc_level==2){
                trans_limit=20000;
            } else{
                trans_limit = 50000;
            }
    }
}
```

```
break;

case 2:
    if(acc_level==1){
        trans_limit=50000;
    } else if(acc_level==2){
        trans_limit=200000;
    } else{
        trans_limit = 500000;
    }
    break;

case 3:
    if(acc_level==1){
        trans_limit=500000;
    } else if(acc_level==2){
        trans_limit=2000000;
    } else{
        trans_limit = 5000000;
    }
    break;

default:
    break;
}
```

P_or_b ဆိုသည့် integer variable တစ်ခုကို user ၏ account_type ကို store လုပ်ထားလိုက်ပါမည်။ ထို့နောက် switch case ထဲတွင် p_or_b သည် **case 1 (personal)**ဖြစ်ခဲ့ပါက account level ကိုထပ်မံစစ်ဆေးပါး **level 1** ဆိုပါက ၅၀၀၀၊ **level 2** ဆိုပါက ၂၀၀၀၀၊ နှင့် အမြင်ဆုံး

level ဖြစ်ပါက ၅၀၀၀၀ အထိလွှံခွင့်ပေးထားမှာ ဖြစ်ပါသည်။ ထိုနောက် **case 2** ဖြစ်ပါက **user**၏ **account type** သည် **business** ဖြစ်ပါသည်။ **case 3** ဆိုပါက **other type** ကိုပြောတာဖြစ်ပါသည်။

```
get_time();
time_class_get_date(getCTime[0].curTime);

ထိုနောက် time function ကို ခေါ်လိုက်ပါး time_class_get_date ထဲတွင်လက်ရှိ current time
(getCTime[0].curTime) ကို ဖြတ်ထားလိုက်ပါသည်။ ထိုနောက် အချိန်ကွာဟချက်ကို
ရှာပါး user သည် ငါး၏ level နှင့် ကိုက်ညီသည့် ပမာဏကို တစ်ရက်အတွင်း လွှာထားပါက
နောက်ထပ်လွှာလို့မရဲကြောင်းပြောနိုင်ရန်အတွက် time_different ရှာသည့်
function ကို ဆက်လက်ပါး ရေးသားမည်ဖြစ်ပါသည်။
```

```
void calculate_time_dif(char last_month[], unsigned int last_day, unsigned
int last_year, unsigned int last_money, unsigned int limit_amount){
    unsigned int current_day=char_to_integer_fun(day);
    unsigned int current_year= char_to_integer_fun(year);
    unsigned int current_money= to_transfer_money;//global money
    twoCharArray=-1;
    printf("\n\n\n----last month %s",last_month);
    compare_twoCharArray(last_month,month);

    if(current_day==last_day && current_year ==last_year &&
twoCharArray==1) {

        if(limit_amount<current_money+last_money){
            transaction_pass=-1;
            transaction_amount_over = limit_amount-
            (current_money+last_money);
```

```

    return;

} else{
    transaction_pass=1;
    return;

}

} else{

    transaction_pass=1;
    return;
}
}

```

Program ရှင်းလင်းချက်

ဒီ **program** မှာ ဆိုရင်တော့ **parameter** ၅ခုအထိဖြတ်ထားပါသည်။ **program** အစတွင် **day** နှင့် **year** ကို **char array** မှ **integer** သို့ ပြောင်းလဲလိုက်ပါး **transfer_money** ကိုလဲ **current_money** တွင် **store** လုပ်ထားလိုက်ပါသည်။

```

printf("\n\n\n----last month %s",last_month);
compare_twoCharArray(last_month,month);

if(current_day==last_day && current_year ==last_year && twoCharArray==1) {

    if(limit_amount<current_money+last_money){
        transaction_pass=-1;
        transaction_amount_over = limit_amount-(current_money+last_money);
        return;
}

```

```

    } else{
        transaction_pass=1;
        return;
    }

} else{

    transaction_pass=1;
    return;
}

```

ထိုနောက် month ကို compare_two_char_array ထဲတွင် နိုဝင်းယူဉ်ပါးနောက်
 အကယ်၍ last_day နောက်ဆုံးလွှဲခဲ့သော ရက်စွဲသည် current_day ,current_year သည်
last_year && twoCharArray သည် 1 ဖြစ်ခဲ့ပါက if else ကို ထပ်ခေါကာ သတ်မှတ်ထားသည့်
 limit amount သည် current_money+last_money ကျော်နေပါက
 transcation_pass ကို -1 သို့ return ပြန်ထားလိုက်ပါမည်။ **transcation_amount_over**
 ကို unsigned int အဖြစ် **global variable** အဖြစ် ကြေညာထားပါသည်။

```

    transaction_amount_over = limit_amount-(current_money+last_money);
} else{
    transaction_pass=1;
    Return;
}

```

limit_amount ထက်မကျော်ပါက transcation_pass ကို 1 လို့ return ပြန်ထားလိုက်မှာ
 ဖြစ်ပါသည်။ ထိုနောက် money_transcation ထဲတွင် calculate_time_diff function
 ကို ခေါ်လိုက်ပါမည်။

```
if(transaction_pass==1){  
    db[transmit].current_amount=      db[transmit].current_amount-amount;  
    db[receiver].current_amount = db[receiver].current_amount+amount;  
    //for insert amount to transaction record will get character array;  
    integer_to_charArrayFun(amount);  
  
    transacitonRecord(transmit,receiver,'t');  
    transacitonRecord(transmit,receiver,'r');  
  
    printf("Transaction complete:\n");  
    printf("Your current amount: %s : %llu\n",db[transmit].name ,  
db[transmit].current_amount);  
  
    printing_all_data();  
    user_sector();  
} else{  
  
    printf("\nYou are exceeded over your  
limit %d:\n",transaction_amount_over);  
    transfer_money();  
}
```

Transcation_amount_limit_per_day function ആംഗീംഡിനാംഗിലും നിന്ന് കൊടുക്കുന്നതിൽ പരിപാലിക്കുന്നതിനായി ഉപയോഗിച്ചുവരുന്നു.