



HOW WE LEARN C PROGRAMMING

C PROGRAMMING LANGUAGE ကို နားလည်သဘောပေါက်စေလိုသော စေတနာဖြင့်

K ZAN HTIKE (MEC)

Contents	Page
<u>Introduction to C</u>	
1. How we learn C programming	1
2. History of C	7
3. Compilers ဆိုတာဘာလဲ	8
4. Structur of C program	8
5. Programming Rules (C program တွင် လိုက်နာရမည့် စည်းကမ်းများ)	12
<u>The C Declarations</u>	
1. C character Set များ	12
2. C Keywords များ	14
3. Identifiers ဆိုတာဘာလဲ	14
4. Constants ဆိုတာဘာလဲ	15
5. Variables ဆိုတာဘာလဲ	16
6. Variables သတ်မှတ်ပေးရာတွင် လိုက်နာရမည့် စည်းကမ်းများ	17
7. Data Types အမျိုးအစားများ	17
8. Declaring Variables (variables ကြေငြာခြင်း)	19
9. Variables သတ်မှတ်ခြင်း	20
10. data type ပြောင်းလဲခြင်း	21
11.Constants and Volatile Variables ဆိုတာဘာလဲ	22
<u>Operators and Expressions</u>	
1. Introduction	25
2. Arithmetic Operators	26
3. Relational Operators	33
4. Logical Operators	35
5. Priority of Operators and their Clubbing	37
<u>Input and Output in C</u>	
1. Introduction	40
2. Formatted Functions	41
3. Unformatted Functions	41
<u>Decision Statements</u>	
1. Introduction	55

2. The if Statement	55
3. The if ...else Statement	60
4. Nestef if...else Statement	64
5. The break Statement	70
6. The continue Statement	71
7. The goto Statement	71
8. The switch Statement	73
<u>Loop Control Statements</u>	
1. Introduction	81
2. The for loop	81
3. Nested for loops	-
4. The while loop	94
5. The do while loop	96
6. the do..while	96
<u>Arrays</u>	98
1. Array Initialization	98
2. Definition of Array	98
3. One Dimensiona Array	99
4. Two-Dimensional Arrays	107
<u>User defined functions</u>	111
1. Make user defined functions	111
2. Call by value and call by refefence	
<u>Pointers</u>	121
Pointer အသုံးပြုရေးနည်း	121

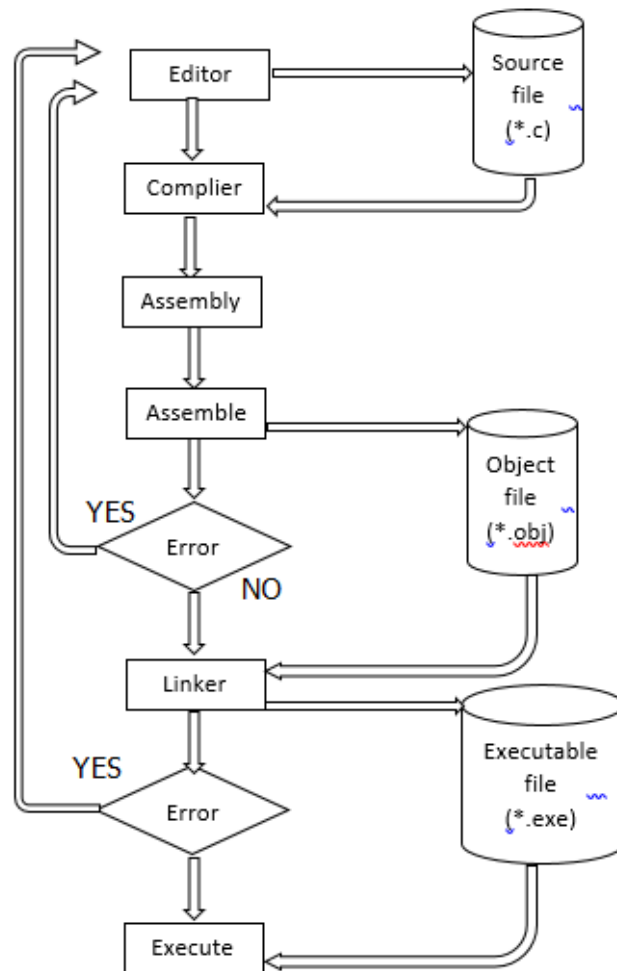
1. How we learn C programming

We are Order or Programmer

ဒီစာအုပ်ကို ရေးသားရတဲ့ အကြောင်းအရင်းကတော့ C Programming Language ကိုကောင်းကောင်း နားလည်သဘောကစလိုတဲ့ အတွက်ဖြစ်ပါတယ်။ Programming လို့ ပြောလိုက်ရင် အားလုံးသိပြီး သား ဖြစ်မှာ ပါ။ ဘယ်လိုအရာကို ခေါ်တာလဲဆိုရင်တော့ သေသေချာချာပြောပြနိုင်သူ နည်းပါးမယ်လို့ ထင်ပါတယ်။ အထူးသဖြင့် programming နဲ့ အကျွမ်းတဝင် မရှိသူတွေအတွက် ပြောပြနိုင်မယ်မထင်ပါဘူး။ programming ဘယ်လို ဖြစ်ပေါ်လာရလဲဆိုတာကို ပြောပြချင်ပါတယ်။ ကျွန်တော်တို့ အသုံးပြုနေတဲ့ ကွန်ပျူတာ ကို အဓိက အားဖြင့် Hardware and Software ပေါင်းပြီး တည်ဆောက်ထားတယ်လို့ အကြမ်းဖျင်း နားလည်ထားပါမယ်။ Hardware ဆိုတာကတော့ ထိတွေ့ကိုင်တွယ်နိုင်သော အရာတွေကို ခေါ်ဆိုခြင်းဖြစ်ပါတယ်။ဥပမာ Keyboard, Driver စတာတွေကို ဆိုလိုတာပါ။ နားလည်အောင် ပြောရရင် Electronic device တွေနဲ့ တည်ဆောက်ထားတဲ့ system တစ်ခုဖြစ်ပါတယ်။ Hardware သည် မှတ်ဉာဏ်သာရှိပြီး အသိဉာဏ်မရှိတဲ့ machine တစ်ခုဖြစ်ပါတယ်။ ဒါကြောင့် ထိုမှတ်ဉာဏ်သာရှိတဲ့ Hardware ကို task (လုပ်ငန်း) တွေ လုပ်ဆောင်နိုင်ဖို့အတွက် Software ဆိုတဲ့ အသိဉာဏ်နဲ့ ပေါင်းစည်းထားတာဖြစ်ပါတယ်။ “Hardware without software is similar to a body without soul” Software မပါတဲ့ hardware ဟာဆိုရင် “ခံစားချက်မရှိတဲ့ လူ” နဲ့တူပါတယ်ဆိုတဲ့ စကားလေး တစ်ခုရှိပါတယ်။ ဒါကြောင့် Hardware တွေကို ခိုင်းစေဖို့ အသိဉာဏ်နဲ့ တူတဲ့ software တွေကို ရေးသားခဲ့ကြပါတယ်။ ဒီ Software တွေ ဖြစ်အောင်ရေးသားတဲ့ စာတွေကို programming language လို့ခေါ်ပါတယ်။ ဥပမာ ကျွန်တော်တို့ ရုံးတွေ၊ လုပ်ငန်းခွင်တွေမှာ အသုံးပြုတဲ့ Adobe PageMaker, Excel, Power Point စတာတွေဟာ programming language ဖြင့်ရေးသားထားတဲ့ Software တွေဖြစ်ပါတယ်။ဒါကတော့ programming ဖြစ်ပေါ်လာရတဲ့ အကြောင်းအရင်းဖြစ်ပါတယ်။

ဒီစာအုပ်မှာ programming languages များစွာထဲက C programming language နဲ့ပတ်သတ်ပြီး အခြေခံကို နားလည်အောင် ရှင်းပြထားပါတယ်။ C programming လို့ဆိုလိုက်တာနဲ့ ကျွန်တော်တို့ Academy မှာ မသိသူ မရှိပါဘူး။ အားလုံး သင်ဖူး၊လေ့လာဖူးကြပါတယ်။အရပ်ဘက် University တွေမှာလည်း ဘာသာရပ်အနေနဲ့ သင်ကြားပေးကြ ပါတယ်။ဘာကြောင့် C language ကို သင်ရတာလဲဆိုတာကို ပြောပြချင်ပါတယ်။ အကြောင်းမှာ ကျောင်းက ဘာသာရပ်တစ်ခုအနေနဲ့ ပြဋ္ဌာန်းတဲ့အတွက် သင်ယူရတယ် ဆိုတာမျိုး မဖြစ်စေချင်ပါဘူး။ programming language ပေါင်းများစွာရှိပါတယ်။ တိမ်မြုပ် ပျောက်ကွယ်သွားတဲ့ programming language တွေရှိသလို အသစ်ထပ်မံ ရေးသားကြတဲ့ language တွေလည်း ရှိပါတယ်။ C language သည် middle level language တစ်ခုဖြစ်ပါတယ်။ ယနေ့ programmer များအနေဖြင့် အများစုသည် C language ကိုလေ့လာကြသူ များပါ လိမ့်မည်။ အကြောင်းမှာ C language သည် hardware ပိုင်းကို

တိုက်ရိုက်ထိန်းချုပ်နိုင်ပြီး ကောင်းကောင်းနား လည်နိုင်သော language တစ်ခုဖြစ်ပါတယ်။ programming sense, programming flow ရဖို့လည်း အရမ်းရိုးရှင်းတဲ့ language တစ်ခုဖြစ်ပါတယ်။ အများစု သိချင်သော မေးခွန်းမှာ C language ကို သုံးပြီး software တွေ Game တွေရေးလို့ရလားဆိုသည့် မေးခွန်းဖြစ်ပါသည်။ ရေးလို့ရပါသည်။ သို့သော် အရမ်းခက်ခဲတဲ့ Language ဖြစ်ပါတယ်။ ရေးအိုးထဲက ရေတစ်ခွက်သောက်ဖို့ program တစ်ပုဒ် ရေး မယ်ဆို ရင် High level language တွင် ရေးအိုးဖုံးလုပ်ပြီး ခပ်သောက်ရုံ ရေးနိုင်သော်လည်း C language တွင် မန္တလေးကျုံးကို ဂျတ်လောက် ပတ်ပြေး ပြီးမှ ရေခပ်သောက် ရသလိုဖြစ်ပါတယ်။ Software တွေ Mobile application တွေတည်ဆောက်ဖို့ လွယ်ကူ တဲ့ high level language များရေးသားနိုင်ဖို့အတွက် C language လိုမျိုး structure language ကို သိထားနားလည်ထားရပါမည်။ သို့မဟုတ်လျှင် memory concept ဆိုတာမျိုးကိုတောင် high level language တွင်သိနားလည်မည်မဟုတ်ပါ။ တနည်းအားဖြင့် C language သည် Human language အမျိုးအစားဖြစ်ပြီး ထို language ကို လူက 100% နားလည် နိုင်သော ကြောင့် သင်ကြားပေးခြင်းဖြစ်ပါတယ်။ C programming ရေး သားရာတွင် အရင်ဆုံး ထို programming ကို computer မှာ ဘယ်လိုလုပ်ဆောင်ပြီး data တွေကို display ပြ တယ်ဆိုတာကို ပုံလေးဆွဲပြီးပြပေးပါမယ်။



အထက်ပါပုံကတော့ C program တစ်ပုဒ်ကို run ပြီး output result ကို ကျွန်တော်တို့ မြင်သာအောင် Monitor မှာ display ပြရန် computer တွင်အဆင့်ဆင့်လုပ်ဆောင်သွားပုံကို ဆွဲထားတဲ့ structure diagram ဖြစ်ပါတယ်။ ကျွန်တော်တို့ ပထမဆုံး programming language တစ်ခုကို ရေးမယ်ဆိုရင် ထို language အတွက် သတ်မှတ်ထားတဲ့ Software ရှိပါမယ်။ခု C programming language ကို ရေးသားဖို့အတွက်လည်း သတ်မှတ် ထားတဲ့ Software တွေရှိပါတယ်။ အများ စုအသုံး များတာကတော့ Code:Blocks ဖြစ်ပါတယ်။ အခြား Dev C++, Visual studio တွေကို သုံးပြီး လည်း ရေးကြပါ တယ်။ C program ရဲ့ source code တွေကို complie လုပ်လို့ရတဲ့ compiler ပါတဲ့ ဘယ် Software မဆို သုံး ပြီးရေးလို့ရပါတယ်။ C programming source code တွေ ကို ကျွန်တော်တို့ အသုံးပြုတဲ့ software ရဲ့ Editor မှာ ရေးရပါမယ်။ ဒီ source code တွေက ကျွန်တော်တို့ နားလည်တဲ့ programming language တွေဖြစ်ပါ တယ်။ တနည်းအားဖြင့်ပြောရရင် English like language ဖြစ်ပါတယ်။ ဘာကြောင့်လဲဆိုတော့ ဥပမာ printf() ဆိုတဲ့ function ၊ ဒီ function ကို C language မှာ data တွေကို Output display ပြဖို့အတွက်သုံး ပါတယ်။ ဒီ printf() မှာ English alphabet တွေကို code အဖြစ် အတိုကောက် သတ်မှတ်ထားတာ ဖြစ်ပါတယ်။ ဒါကြောင့် English like language လို့ခေါ်ဝေါ်ကြ တာဖြစ်ပါတယ်။ ဒီလိုရေးသားရတဲ့ programming language ကို ကျွန်တော်တို့ ကကောင်းကောင်း နားလည် နိုင်တယ်၊ဖတ်နိုင်ပါတယ်။ ဒီ source code တွေကတော့ computer ကို problem တွေ ကိုင်တွယ်ဖြေ ရှင်း ခိုင်းဖို့အတွက် ရေးထားတဲ့ instruction တွေဖြစ်ပါတယ်။ instruction ဆိုတာကတော့ ထမင်းစားမယ့် program တစ်ပုဒ်ရေးမယ်ဆိုပါတော့။ ဒါဆိုရင် ထမင်းချက်၊ဟင်းချက်၊ ကျက်ရင် စားမယ်၊ မကျက်သေးရင် မစားဘူး ဆိုတဲ့ လုပ်ဆောင်ချက်တွေကို အစီအစဉ်ရေးဆွဲထားတာကို ဆိုလိုတာပါ။ ဒါပေမယ့် Computer က ဒီ human language တွေကို နားမလည်ဘူး။ ဘာဖြစ်လို့လဲဆိုတော့ သူသိတာက pulse(1) or no pulse(0) ကိုပဲ နားလည်ပါတယ်။ ဒီတော့ ကိုင်တွယ်ဖြေရှင်းပေးနိုင်မှာ မဟုတ်ပါဘူး။ဒါကြောင့် ကျွန်တော်တို့ ရေးလိုက်တဲ့ code တွေကို နားလည်ဖို့အတွက် human language ကနေ machine language ကိုပြောင်းဖို့ ဘာသာပြန်လို အပ်ပါ တယ် ။ ဒီဘာသာပြန်ကို programming terms အရပြောရင် Compiler ဖြစ်ပါတယ်။ ကျွန်တော်တို့ ရေးလိုက်တဲ့ source code တွေကို (*.c) နဲ့ သိမ်းပေးထားပါတယ်။ ဒါကို extension dot c လို့ဖတ်ပါတယ်။ C source file တွေ ကို သိမ်းတာဖြစ်တဲ့အတွက် (.c) လို့ရေးတာဖြစ်ပါတယ်။ c++ source file ကို သိမ်းမယ်ဆိုရင်တော့ (*.cpp) ဆိုပြီးသိမ်းပေးမှ ဖြစ်ပါတယ်။ Compiler ရဲ့ အလုပ်က ခုလိုမျိုး (*.c) နဲ့ သိမ်းထားတဲ့ code တွေကို တစ်ကြောင်းချင်းစီ ဖတ်ပြီး Assembly language အဖြစ်ပြောင်းပေးဖို့ဖြစ်ပါတယ်။ ဖတ်တယ်ဆိုတာက ဘယ်ဟာ တွေက data ဘယ်ဟာတွေက command တွေဆိုပြီး Analysis လုပ်တာဖြစ်ပါတယ်။ တကယ်လို့ error (အမှား) တွေ့ခဲ့တယ်ဆိုရင်တော့ ဒီ instruction မှာ ဘယ်အရာတွေကတော့ မှားနေပါတယ်လို့ ချက်ချင်း ပြမပေးပါဘူး။ source file တစ်ခုလုံးကို ဖတ်ပြီးမှ ဒီ data တွေ command တွေကတော့ မှားနေပါတယ်ဆိုပြီး error တွေကို စုပြီးပြပေးတာဖြစ်ပါတယ်။ error မရှိတဲ့ source

code တွေကိုတော့ တစ်ခါတည်း Assembly language အဖြစ် ပြောင်းပေး ပါတယ်။ Assembly language က low level programming language ဖြစ်ပါတယ်။ ဒီ language ကို လူက 30% သာနားလည် နိုင်ပြီး machine က 70% နားလည်ပါတယ်။ ခုလိုမျိုး Assembly code တွေကို computer က အပြည့်အဝ နားမလည်သေးပါဘူး။ ဒါကြောင့် နောက်ထပ် computer အတွက် အပြည့်အဝနားလည်နိုင်တဲ့ machine code ပြောင်းဖို့ assemble လုပ်ပေးရ ပါမယ်။ ဘယ်လို Assemble လုပ်လဲဆိုတော့ ဥပမာ - ကိန်းနှစ်ခု(3 နဲ့ 4) ကို ပေါင်းဖို့အတွက် program လေး တစ်ပုဒ် ရေး တယ်ဆိုပါစို့။ ဒါဆိုရင် human language မှာ (3+4) လို့ရေးရပါမယ်။ ဒီ program မှာ ဆိုရင် 3 နဲ့ 4 ကို data လို့ခေါ်ပါတယ်။ သူတို့နှစ်ခုကို ပေါင်းဖို့အတွက်သုံးတဲ့ (+) sign ကိုတော့ command လို့ခေါ်ပါတယ်။ ခုလိုမျိုးရေးပြီးပေါင်းလိုက် တာကို computer က နားမလည်ပါဘူး။ ဒါကြောင့် Assembler က machine code အဖြစ်ပြောင်းပြီး ပေါင်းပေးပါတယ်။ ပိုနားလည်အောင် တွက်ပြပေးပါမယ်။

Decimal number 3 ရဲ့ binary code က 0011 ဖြစ်ပါတယ်။ Decimal number 4 ရဲ့ binary code က 0100 ဖြစ်ပါတယ်။ binary code မှာ 0 to 1 (2 digit) ပဲရှိပါတယ်။ ခုလိုမျိုး digit တွေကိုရေတွက်ဖို့အတွက် base 2 ကို အခြေခံပြီးတွက်ရပါတယ်။ 3 ဆိုရင် binary code အနေနဲ့ 0011 ဘာကြောင့်ရ တာလဲဆိုတာ ရှင်းပြပေးချင်ပါတယ်။ Decimal number က 0 to 9 (10 digit) ရှိပါတယ်။ 2^3 ဆိုရင် 8 digit ပဲရှိပါသေးတယ်။ 10 digit ဖြစ်ဖို့အတွက် 2^4 (16 digit) ဖြစ်မှရပါမယ်။ base 2 ရဲ့ power(ထပ်ကိန်း) ကိုတော့ bit လို့သတ်မှတ် ပါတယ်။ဒီတော့ 4 bit ဖြစ်ပါမယ်။ ဒါကြောင့် Decimal number တွေကို binary code ပြောင်းဖို့အတွက် 4 bit (8 4 2 1) code ကိုသုံးပြီး တွက်ရပါတယ်။ ဒါဆိုရင် decimal number 3 နဲ့ 4 ကို compiler က ဘယ်လို ပြောင်း လဲ ဆိုတာကို ကြည့်ရအောင်။

Decimal number 3 ဖြစ်ဖို့ဆိုရင် (8 4 2 1) code ထဲက 2 နဲ့ 1 ပေါင်းရင် 3 ဖြစ်ပါမယ်။ ဒီတော့ $3=2+1$ ဖြစ်ပါမယ်။ဒါဆိုရင် 8 နဲ့ 4 ကို 0 ပေးပြီး 2 နဲ့ 1 ကို 1 ဆိုပြီး (on) ပေးရပါမယ်။ ဒီလိုတွက်ချက်လိုက်တဲ့ အတွက် $3=0011$ ဖြစ်တာပါ။ 4 ဆိုရင်လည်းဒီအတိုင်းပါပဲ။ 4 ဖြစ်ဖို့အတွက် 4 ကိုပဲ 1 ထားပြီးကျန်တဲ့ 8,2,1 ကို 0(off) ထားလိုက်ပါမယ်။ ဒါကြောင့် $4=0100$ ကိုရတာဖြစ်ပါတယ်။ နောက်တနည်းအားဖြင့် ပြောင်းချင်တဲ့ decimal number ကို 2 နဲ့ စားမယ်။ စားလဒ် 0 မဖြစ်မချင်း အဆင့်လိုက် စားလဒ်တွေကို 2 နဲ့စားပြီး အကြွင်း ကို binary code အဖြစ်ယူတာလဲ ရှိပါတယ်။ နား လည်အောင်တွက်ပြပေးပါမယ်။

$$3/2=1 \text{ အကြွင်း } =1$$

$1/2=0$ အကြွင်း =1 ဖြစ်ပါမယ်။ ဒါဆိုရင် အောက်ကနေ အပေါ်ကို ယူပြီး အစဉ်လိုက် ရေးရတာဖြစ်တဲ့အတွက် 11 ဖြစ်ပါမယ်။ 4 bit ဆိုတဲ့အတွက် 11 ကို 4 bit ဖြစ်အောင် ရှေ့ကို 00 ဖြည့်ပေးရပါမယ်။ ဒါဆို $3 = 0011$ ဖြစ်သွား ပါမယ်။ decimal number 4 ကိုလည်း ဒီအတိုင်း တွက်ချက်လိုက်တာဖြစ်ပါတယ်။

$$4/2=2 \text{ အကြွင်း } =0$$

$2/2=1$ အကြွင်း =0

$1/2=0$ အကြွင်း =1 ဖြစ်ပါမယ်။ ဒီတော့ 100 လို့ရေးပေးရပါမယ်။ သူ့ကိုလည်း 4 bit ပြည့်ဖို့အတွက် 0 ဖြည့် ပေးရပါမယ်။ ဒီတော့ 0100 ဖြစ်ပါမယ်။ ဒါကို နားလည်မယ်လို့ထင်ပါတယ်။

Assembler က ခုလိုမျိုး 3 နဲ့ 4 ကို binary code (machine code) ကိုပြောင်းပေးလိုက်တာ ဖြစ်ပါတယ်။ command က ဒီ data နှစ်ခုကို addition (+) လုပ်ခိုင်းတာဖြစ်တဲ့အတွက် ခုလိုမျိုး ပေါင်းပေးလိုက်ပါတယ်။

$$\begin{array}{r} 0011 \\ + 0100 \\ \hline 0111 \end{array}$$

ဒါကြောင့် binary code နှစ်ခုပေါင်းလို့ရတဲ့ result က 0111 ဖြစ်ပါတယ်။ ခုလိုမျိုး machine code ပြောင်းတာကို encoding လုပ်တာလို့လည်းခေါ် ပါသေးတယ်။ ဒါဆိုရင် computer က ဘာလုပ်ရမယ်ဆိုတာကို နားလည်သွား ပါမယ်။ပြီးရင် ကျွန်တော်တို့ နားလည်တဲ့ decimal တန်ဖိုးပြန်ရအောင် decode ပြန်လုပ်ပေးရပါတယ်။ 0111 ဆိုတော့ (8 4 2 1) မှာဆိုရင် 8 နေရာမှာပဲ 0 ဖြစ်နေပါတယ်။ဒီ တော့ 8 ကို ဖယ်လိုက်ပြီး ကျန်တဲ့ 1 ဖြစ်တဲ့ (4 2 1) ကိုပေါင်းပေးလိုက်တာပါ။ ဒီတော့ 7 ကို ရပါမယ်။ ခုလိုမျိုး encoding လုပ်ထားတဲ့ data တွေကို ချက်ချင်း decode လုပ်ပြီး execute လုပ်ပေးတာမဟုတ်ပါဘူး။ ဘာကြောင့် 7 လို့ display ပြရလဲဆိုတာကို သိစေချင်လို့ ပြောပြတာပါ။ ခုလိုမျိုး Assemble လုပ်ပြီးရလာတဲ့ machine code တွေကို object code လို့လည်းခေါ်ပါတယ်။ ဒါဆိုရင် Assembler ရဲ့ အလုပ်ကို နားလည်မယ်လို့ ထင်ပါတယ်။ ခုလိုမျိုး Source code ကို Assemble လုပ်ပြီးတဲ့အခါမှာ error ရှိမရှိ ထပ်စစ်ပါတယ်။တကယ်လို့ ရှိတယ်ဆိုရင်တော့ Editor ကို ပြန်သွားပြီး ပြင်ပေးရပါမယ်။ မရှိဘူးဆိုရင်တော့ object file (*.obj) အဖြစ်ပြောင်းပြီး သိမ်းပေးမှာဖြစ်ပါတယ်။ Object code တွေကို ရပြီးဆိုရင်တော့ linker က သူ့ရဲ့ အလုပ်ကို လုပ်ဆောင်မှာဖြစ်ပါတယ်။ Obj code ပြောင်းထားတဲ့ source code တွေကို linker ကနေပြီး C libraries file မှာ ရှိတဲ့ source file တွေနဲ့ linking လုပ်ပါတယ်။ linking လုပ်တဲ့ အခါမှာ linker ကနေပြီး error ရှိမရှိ ထပ်စစ်ပါတယ်။ ဘယ်လို error လဲဆိုတော့ libraries မှာ သတ်မှတ်ထားတဲ့ header file တွေ function တွေကို C source code program မှာ ရေးထားတယ်ဆို linker error တက်ပါတယ်။ C libraries မှာ သတ်မှတ်ထား တဲ့ file တွေရှိမှ header file တွေ function တွေကို သုံးလို့ရမှာ ဖြစ်ပါတယ်။ error ရှိမယ်ဆို ရင်တော့ Editor ကို ပြန်သွားပြီး error တွေကို ပြန်ပြင်ခိုင်းပါမယ်။ မရှိဘူးဆိုရင်တော့ object file တွေကို Executable file အဖြစ်ပြောင်းပါမယ်။ ဒီ executable file တွေကိုတော့ (*.exe) ဆိုပြီး သိမ်းထားပေးပါတယ်။ဒီ file တွေကတော့ object file တွေကို user နားလည်အောင် display ပြပေးဖို့အတွက် decode လုပ် ထားတဲ့ file တွေဖြစ်ပါတယ်။ဒါဆိုရင် $3+4 = 7$ လို့ user နားလည်အောင် display ပြပေးမှာ ဖြစ်ပါတယ်။ ကျွန်တော်တို့ရေးလိုက်တဲ့ C language program တွေကို Computer က ခုလိုမျိုး

အဆင့်ဆင့်လုပ်ဆောင်ပြီးမှ ကျွန်တော်တို့နားလည်တဲ့ executable code တွေကို ထုတ်ပေးတာ ဖြစ်ပါတယ်။

Programming language တိုင်းမှာ သူ့ရေးသားနည်းရေးသားဟန်တွေ ရှိပါတယ်။ ဘာသာစကား တောင်မှ China, India, Japan လိုမျိုး ရေးသားရတဲ့ ပြောဆိုရတဲ့ ဘာသာစကားတွေ ရှိသလိုပဲဖြစ်ပါတယ်။ ဒါကို programming paradigms လို့ခေါ်ပါတယ်။ C language ရေးသားနည်းက English language ရေးသားနည်း နဲ့ ဆင်တူပါတယ်။ English language မှာ paragraph တစ်ခုတည်ဆောက်ဖို့အတွက် sentence တွေရေး တတ်ဖို့လိုပါတယ်။ Alphabets တွေ Vowel တွေ Noun or Verb စတဲ့ အမျိုးအစားတွေ သိဖို့လိုပါတယ်။ C language မှာလည်းဒီအတိုင်းပါပဲ။ program တစ်ပုဒ်ရေးတတ်ဖို့အတွက် sentences လိုမျိုး instruction တွေ တည်ဆောက်တတ်ရပါမယ်။ Vowel, Noun, verb စတာတွေလိုမျိုး Data type, Keyword, Variables တွေကို သိဖို့လိုပါတယ်။ Alphabets လိုမျိုး Special symbols တွေ Digits တွေကို သိဖို့လိုပါတယ်။ C language ရေးသားဖို့အတွက် Alphabets တွေကိုလည်းပဲ သိထားရပါမယ်။ ဒီအချက်တွေကို သိမှ မိမိကိုယ်တိုင် program တစ်ပုဒ်ကို စဉ်းမျဉ်းတွေနဲ့ ကိုက်ညီအောင် ရေးတတ်မှာ ဖြစ်ပါတယ်။

C language မှာ instructions တွေကို တည်ဆောက်ဖို့အတွက် functions တွေ header file တွေကို နားလည်ဘောပေါက်ထားရပါမယ်။ functions တွေ header file တွေ ဆိုတာကို နားလည်အောင် ဗိုလ်လောင်း တစ်ယောက်ရဲ့ နေ့စဉ်လုပ်ဆောင်နေရတဲ့ လုပ်ငန်းတွေနဲ့ ဥပမာထားပြီး ပြောပါမယ်။ ဗိုလ်လောင်းတစ်ယောက်အတွက် တစ်ရက်တာ လုပ်ဆောင်ရတဲ့ လုပ်ငန်းတစ်ခု ကို အစုလေးတွေခွဲပြီး ပြပါမယ်။ မနက်အိပ်ယာ ထတာနဲ့ ကိုယ်လက်ကြံ့ခိုင်ရေးလုပ်ဖို့အတွက် ပြင်ဆင် ရတာတွေကို စာရွက်ပေါ်မှာ တခုချင်းစီ ချရေးကြည့်ရ အောင်။

၁။ မနက် ၅ နာရီ အိပ်ယာထ

၂။ ခြင်ထောင်သိမ်း

၃။ PT dress လဲ

၄။တန်းစီ

ဒီလုပ်ငန်းစဉ်ကို ခေါင်းစဉ်ပေးပါမယ်။ “နံနက်ခင်းကြံ့ခိုင်ရေးအတွက် ပြင်ဆင်မှု” လို့ပေးပြီး သိမ်းထားမယ်။ နောက်ထပ် “နံနက်ခင်း နိုင်ငံတော် အလံ အလေးပြုဖို့အတွက် ပြင်ဆင်မှု” ဆိုတဲ့ ခေါင်းစဉ်ပေးပြီး လုပ်ဆောင်ရ မယ့် အချက်တွေကို တခုချင်းစီရေးကြည့်မယ်။

၁။ ရေချိုး

၂။ သင်တန်း Dress လဲ

၃။ တန်းစီ

ဆိုပြီး စာရွက်တစ်ရွက်နဲ့သိမ်းထားမယ်။ နောက်ထပ် “ကျောင်းသွားဖို့အတွက် ပြင်ဆင်မှု” ဆိုတဲ့ ခေါင်းစဉ်ပေးပြီး အချက်အလက်တွေကို ရေးမယ်။

၁။ထမင်းစား

၂။တန်းစီ

၃။လမ်းလျှောက်

ဆိုတာကိုလည်း စာရွက်တစ်ရွက်နဲ့သိမ်းထားမယ်။ ဒါဆိုရင် "နံနက်ခင်း ကြံ့ခိုင်ရေးအတွက် ပြင်ဆင်မှု" ကို ကြည့်ချင်ရင် အချက်အလက်တွေ ထုတ်ပြီးကြည့်ယုံပဲ။ "နိုင်ငံတော်အလံ အလေးပြုဖို့အတွက် ပြင်ဆင်မှု" ကို သိချင်ရင်လည်း လွယ်လွယ်လေး စာရွက်ထုတ်ကြည့် ရုံပဲ။ မလွယ်ကူဘူးလား။ ခုလိုမျိုး လုပ်ငန်းစဉ် တစ်ခုရဲ့ လုပ်ဆောင်ရမယ့် အချက်အလက်တွေကို functions လို့ခေါ်ပါတယ်။ ဒီလုပ်ဆောင်ချက်တွေကို ခေါင်းစဉ်ပေးထားတာကိုတော့ header file လို့ခေါ်ပါတယ်။ ဒါကြောင့် functions တစ်ခုကိုသုံးချင်ရင် ဒီ function ကို သိမ်းထားတဲ့ header file ကို လည်းသိရပါမယ်။ C language မှာလည်း function တွေ header file တွေကို ထိုအတိုင်း တည်ဆောက်ထား တာဖြစ်ပါတယ်။ C language မှာ data တွေကို output display ပြုဖို့အတွက် printf() function (print function) ကို သုံးပါတယ်။ဒီ function ကို library file မှာ stdio.h(standard input /output header) လို့ အမည်ပေးပြီး သိမ်းထားတာဖြစ်ပါတယ်။ ဒီ header file မှာ အထက်က ဥပမာပေးခဲ့တဲ့ "နံနက်ခင်း ကြံ့ခိုင်ရေး အတွက်ပြင်ဆင်မှု" ဆိုတဲ့ ခေါင်းစဉ်မှာလိုပဲ functions တွေ များစွာပါဝင်ပါတယ်။ program မှာ function တွေကို သုံးမယ်ဆိုရင်တော့ ထို function ကို သိမ်းထားတဲ့ header file ကို ကြေငြာပေးရမှာ ဖြစ်ပါတယ်။ program တစ်ပုဒ်ရေးဖို့အတွက် အထက်မှာပြောခဲ့သလိုမျိုး Digit တွေ keyword တွေ Operator တွေကို သိဖို့ လိုပါမယ်။ ကြက်ဥကြော်ချင်ရင် ကြက်ဥကြော်ရမယ့် နည်းစနစ်တွေကို သိဖို့လိုသလို program တစ်ပုဒ်ရေးမယ် ဆိုရင် programming language paradigms ကို သိဖို့လိုပါတယ်။

အကျဉ်းချုပ် ပြောရရင် programming တစ်ပုဒ်ရေးတယ်ဆိုတာ Analog သဘာဝတရား ဖြစ်စဉ်တွေ၊ လုပ်ငန်းစဉ်တွေကို Digital စနစ် အသွင်ပြောင်းပြီး ရေးသားခြင်းဖြစ်ပါတယ်။ ဒါကြောင့် C programming language ကို လေ့လာမယ်ဆိုရင် အထက်ပါပြောပြထားတဲ့ အကြောင်းအရာတွေ ကို နားလည်သဘော ပေါက်အောင် လေ့လာပါ။ We are order လို့ခံယူပြီး သက်ရှိလူသား တစ်ယောက်အနေနဲ့ သက်မဲ့ ကွန်ပျူတာ တစ်လုံးကို အကျိုးရှိစွာ ညွှန်ကြားနိုင်တဲ့ အမိန့်ပေးနိုင်တဲ့ programmer တွေ ဖြစ်အောင် ကြိုးစားကြဖို့ တိုက်တွန်းချင်ပါတယ်။

2. History of C

1960 မှာ Computer language တွေကို ရေးသားတာဟာ ဓာတ်စားလာခဲ့ပါတယ်။ဥပမာ - COBL (ဒီ language ကို Commercial Application တွေ)ရေးသားဖို့အတွက် အသုံးပြုခဲ့ကြပါတယ်။ FORTRAN ကိုတော့ Engineering နဲ့ Scientific Application တွေရေးသားဖို့အတွက် အသုံးပြုခဲ့ကြပါတယ်။ဒါပေမယ့် လူတွေက ခုလိုမျိုး နယ်ပယ်တစ်ခုအတွက် သီးသန့်ရေးသားရတဲ့ language တွေအစား နယ်ပယ်အားလုံးအတွက် အဆင်ပြေမယ့် language ကိုရေးသားဖို့အတွက် ကြိုးစားခဲ့ကြပါတယ်။ဒါကြောင့် ALGOL 60 ဆိုတဲ့ language ကို ရေးသားခဲ့ကြ ပါတယ်။ဒါပေမယ့် ဒီ language ကလည်း အဆင်မပြေခဲ့ပါဘူး။ ဒါကြောင့် CPL(Combined Programming Language) ကို Cambridge University မှာ develop လုပ်ခဲ့ပါတယ်။ဒါပေမယ့် ဒီ language ကလည်း

နားလည်ရခက်ခဲတဲ့အတွက် အဆင်မပြေပြန်ပါဘူး။ နောက် Cambridge University က Martin Richards ဆိုသူကနေပြီးတော့ BCPL(Basic combined programming language) ကို develop လုပ်ခဲ့ပါတယ်။ ဒီ language ဟာဆိုရင် CPL language ထက်အများကြီး အဆင်ပြေခဲ့ပါတယ်။ BCPL language ပေါ်တဲ့အချိန်မှာပဲ AT & T's Bell labs မှာ B language ကို မိတ်ဆက်ခဲ့ပါတယ်။ Dennis Ritchie ဆိုသူက ဒီ language နှစ်ခုကို အခြေခံပြီး နောက် language တစ်ခုကို develop လုပ်ခဲ့ပါတယ်။ ဒီ language ကတော့ C programming language ဖြစ်ပါတယ်။ C Program ကို Dennis Ritchie က Bell Laboratories မှာ 1972 မှာ စတင်မိတ်ဆက်ရေးသားခဲ့ တာဖြစ်ပါတယ်။ C programming language ဟာ "Basic combined programming language " လို့ခေါ်တဲ့ "B" ကို developed လုပ်ထားတဲ့ language ဖြစ်ပါတယ်။ ဒါကြောင့် C language ကို Extended B language လို့လည်း ခေါ်စမှတ်ပြုကြပါတယ်။ C language သည် High level language လိုမျိုး ရေးသားနည်းအရမ်း မရှိ၊ ရှင်းသော်လည်း Low level language လိုမျိုး နားလည်ရခက်သော Language မဟုတ်ပါဘူး။ ဒါကတော့ C programming language ရဲ့ History အကျဉ်းချုပ်ဖြစ်ပါတယ်။

3. Compilers ဆိုတာဘာလဲ

Compilers ဆိုတာကတော့ ဘာသာပြန်တစ်ယောက်ဗျ။ ဘာလို့လဲဆိုတော့ သူက Human language နဲ့ Machine Language ကြားမှာ Translate လုပ်ပေးလို့ ဖြစ်ပါတယ်။ ကျွန်တော်တို့ ရေးလိုက်တဲ့ C Source Codes တွေကို Compilers ကနေပြီး တစ်ကြောင်းချင်းစီ Run ပြီး Translate လုပ်ပေးပါတယ်။ Error ရှိခဲ့မယ်ဆိုရင်တော့ Error တွေကို List လုပ်ပြီး ပြပေးပါတယ်။ C Source Code တွေကို Compiler လုပ်ဖို့အတွက် GNU GCC Compiler ပါရှိရပါမယ်။ Code block software မှာတော့ Built in ပါပြီးသား ဖြစ်ပါတယ်။

4. Structure of C program

Include header file section
Global Declaration Section
/* comments */
main() function
{
/*comments*/
Declaration part
Executable part
}
User-defined functions
{
}

ဒါကတော့ C program တပုဒ်ရေးမယ် ဆိုရင် သိထားရမယ့် C language Style ဖြစ်ပါတယ်။ ခု ဒီ Structure လေးကို ရှင်းပါမယ်။ ပထမဆုံး -

1. include header file section

include header file section ကို ရှင်းပြပါမယ်။ C program တွေမှာ သုံးတဲ့ function တွေဟာ header file တွေပေါ်မှာ မှီခိုနေပါတယ်။ ဘာလို့အဲလိုပြောရတာလည်းဆို တော့ ဥပမာ(ကျွန်တော်တို့ "hello world " ဆိုတဲ့စာတန်းလေး output ထုတ်ချင်တယ်ဆိုတာ။ printf() ဆိုတဲ့ function ကို သုံးရပါမယ်။ ဒီအတိုင်းသုံးလို့ရလား ဆိုတော့ မရပါဘူး။ Input /Output function တွေအတွက် predefined လုပ်ထားတဲ့ standard Input/Output header file ကိုကြေငြာပေးမှ သုံးလို့ရပါမယ်။) အဲလောက် ဆို ဘာကြောင့် မှီခိုနေတာလဲဆိုတာကို နားလည်ပြီထင်ပါတယ်။ header file တွေကိုတော့ default အနေနဲ့ (extension dot h) (.h) နဲ့သိမ်းထားပေးပါတယ်။ ဟုတ်ပြီ header file တွေကို ဘာကြောင့်ကြေငြာပေးရလဲ ဆိုတာသိပြီဆို ဘယ်လိုကြေငြာပေးရမလဲဆိုတာ ရှင်းပြပါမယ်။ #include preprocessor directive ကို အသုံးပြုပြီး open/closed arrow ထဲမှာ header file ရဲ့ name ကို ကြေငြာပေးရပါမယ်။

#include<file name>

ဥပမာ- #include<stdio.h> or #include"stdio.h"

အဲလိုရေးပေးမယ်ဆိုရင်တော့ standar I/O ထဲက function တွေကို program မှာ ယူသုံးလို့ရပါပြီ။ header file နဲ့ ပတ်သတ်ပြီး အဲလောက်သိထားရင် ရပါပြီ။ Header file အမျိုးအစားတွေကိုတော့ program များများ ရေးမယ်၊လေ့လာမယ် ဆိုရင်တော့ သိလာပါလိမ့်မယ်။

2. Global Declaration

Global Declaration ဆိုတာ variable တန်ဖိုးတွေကို program တခုလုံးမှာရေးထားတဲ့ ဘယ် function မှာမဆို ခေါ်သုံးစေချင်တဲ့ အတွက် functions အားလုံးရဲ့ အပြင်မှာ ကြေငြာပေးရတာကို ဆိုလိုတာပါ။ဘာလို့လဲဆို တော့ variable တန်ဖိုးတွေဟာ သူတို့ကို ကြေငြာထားတဲ့ function ရဲ့ scope အတွင်းမှာပဲ ခေါ်သုံးလို့ ရတဲ့ အတွက်ဖြစ်ပါတယ်။ဒါကြောင့် ဘယ် function မှာမဆို ခေါ်သုံးလို့ရအောင် main() function ရဲ့ အပြင်မှာ Global အနေနဲ့ ကြေငြာပေးတာဖြစ်ပါတယ်။ အဲလို Global အဖြစ်ကြေငြာတဲ့ variable တွေကို Global variable တွေလို့ခေါ်ပါတယ်။ အသေးစိတ်ကိုတော့ Structure တခုလုံးရှင်းပြပြီးမှ program တပုဒ်ရေးပြီး ထပ်ပြီးရှင်းလင်းပေးပါမယ်။ ခုတော့ ဒီလောက်ပါပဲ။

3. Executable part

Executable part ဆိုတာကတော့ Operator တွေ Decision Statement တွေ functions တွေ စတာတွေကို သုံးပြီး program ကို ခိုင်းစေဖို့ instructions တွေရေးပေးရတဲ့ အပိုင်းပဲဖြစ်ပါတယ်။ အဲ့ဒီ statement တွေကို Curly brace { } နှစ်ခုရဲ့ အထဲမှာရေးပေးရပါမယ်။ program နဲ့ example ရေးပြီးရှင်းတဲ့အခါမှ နားလည်အောင်ပြန်ကြည့်ပါ။

4. User_defined function

User_define function ဆိုတာကတော့ user ကိုယ်တိုင် သတ်မှတ်ထားတဲ့ function တွေကို ဆိုလိုခြင်းဖြစ်ပါတယ်။ အဲ့ဒီ function တွေကိုတော့ ပုံမှန်အားဖြင့် main() function ရဲ့အောက်မှာ ရေးပေးရပါတယ်။ main() function ရဲ့ အပြင်မှာလည်း ကြေငြာပေးလို့လည်းရပါတယ်။ အဲ့ဒီ အပိုင်းကတော့ မဖြစ်မနေရေးပေးရမယ့် အပိုင်းတော့ မဟုတ်ပါဘူး။ user ရဲ့ စိတ်ကြိုက်ပါပဲ။ ဒါပေမယ့် syntax မှန်ကန်ဖို့တော့ လိုပါတယ်။ ဒီအကြောင်း ကို အခန်း ၈ မှာ ရှင်းပြပေးထားပါတယ်။

5. Comments

Comments ဆိုတာကတော့ program မှာ ဘယ် Statement တွေက ဘာကြောင့်ရေးထားလဲ ဆိုတာကို တေးမှတ် ချင်တဲ့အခါ၊ program ကို ဖတ်တဲ့သူတွေအတွက် နားလည်စေဖို့အတွက် ရေးထားတာကို ခေါ်တာပါ။ ဒါကလည်း မရေးမဖြစ်တော့ မဟုတ်ပါဘူး။ user ရဲ့ သဘောထားပါ။ ရေးတဲ့အခါမှာ `/* */` နှစ်ခုရဲ့အတွင်းမှာ ရေးပေးရမှာပါ။ နောက်တနည်းအားဖြင့် `//` နှစ်ခုခံပြီး ရေးလို့ရပါတယ်။ Comment တွေကိုတော့ program run တဲ့အခါမှာ Compilerက ထည့်ပြီး run ပေးမှာ မဟုတ်ပါ ဘူး။ ဒါကိုတော့ programmer က သိထားရပါမယ်။

ကဲ..အခု Structure ကိုရှင်းပြီးပြီဆိုတော့ Example program လေးနဲ့ ရှင်းသွားအောင် ထပ်ရှင်းပြပါမယ်။

Hello World program လေးနဲ့ စတာပေါ့....

```

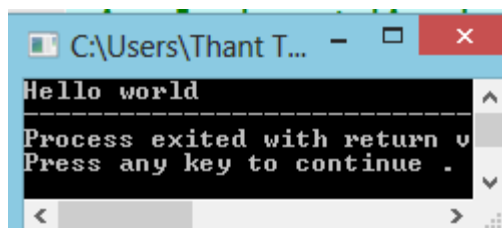
1  #include<stdio.h>
2  /* Hello World Program*/
3  main()
4  {
5      printf("Hello world");
6  }
```

Include header file section

Comments

Main function

Executable part



Structure ပိုင်းအားလုံးကို ရှင်းမပြထားပါဘူး။ဘာလို့လဲဆိုတော့ Beginner တစ်ယောက် အတွက် အရမ်းရှုပ် ထွေးသွားမှာ စိုးလို့ပါ။ ဒါကတော့ ဘယ် programming language ကိုဘဲ စလေ့လာပါ။ Hello World ကနေပဲ စပြီး လေ့လာကြရပါတယ်။ ဒါမှ Programming ရဲ့ သဘောသဘာဝကို တီးမိခေါက်မိရှိမှာမို့ပါ။ ဟုတ်ပြီ... ပထမ ဦးဆုံး Program လေ့လာတဲ့အချိန်မှာ အကြံပေးချင်တာက ဘယ် Statement တွေကို ဘာကြောင့်ရေးပေး ရ တာလဲဆိုတာကို သိအောင် line by line စဉ်းစားပေးစေချင်ပါတယ်။ ခု program ကို ကြည့်ရအောင်....

- ◆ ပထမဦးဆုံး `#include <stdio.h>` ဆိုတဲ့ Standard Input/Output Header file ကို ကြေညာပေးရမှာပါ။ အဲလို ကြေညာပေးမှ ဒီ Header file မှာရှိတဲ့ Function တွေကို current program မှာ ခေါ်သုံးလို့ရမှာမို့ပါ။ အိုကေ.. ရှင်းပြီနော်။
- ◆ နောက်တကြောင်းကတော့ Comment ပါ။ ဘာကြောင့်ရေးပေးတာလဲဆိုတာကို ရှင်းပြပြီးနော်။ ကျွန် တော်တို့ ရေးမှာက Hello world ဆိုတဲ့ စာသားလေးကို screen ရဲ့ monitor (ဖန်သားပြင်) မှာ output အနေနဲ့ပြချင်တာလေ။ ဒါကြောင့် programmer က ခုprogram ဟာ Hello world program လေးပါဆိုပြီး တေး မှတ်ထားပေးခြင်းဖြစ်ပါတယ်။
- ◆ နောက်တစ်ကြောင်းကတော့ `main()` function ပိုင်းပါ။ ဒီအပိုင်းကတော့ C language မှာ မပါလို့မရပါဘူး။ `main` function မပါဘဲနဲ့ ရေးထားတဲ့ program တပုဒ်က အကျုံးမဝင်ပါဘူး။ ဆိုလိုတာက run လို့မရ ဘူးပေါ့ဗျာ။ C program တိုင်းက `main` function ကနေပဲ စပြီး အခြား Statement တွေ function တွေကို run တာမို့ပါ။ ဒီအပိုင်းကိုတော့ မရေးဘဲချန်ထားလို့မရပါဘူး။ ပြောရရင် C language ရဲ့ အသက်ပေါ့ဗျာ။ နဲ့နဲ့ over ဖြစ်သွားလားမသိဘူး.....။ `main` function ရေးပြီးတာနဲ့ program ကို စပါတော့မယ် လို့ကြေညာ လိုက်တာ ကတော့ opening brace ({)လေးဖြစ်ပါတယ်။ ဆုံးသွားရင်တော့ closing brace (}) လေးနဲ့ ပြန်ပိတ်ပေးလိုက်ရုံပါပဲ။ ရှင်းတယ်ဟုတ်။
- ◆ `printf("Hello world");` ဟုတ်ပြီ ဒီ Statement လေးက ဘာလုပ်ဖို့ ရေး တာလဲ။ ရှင်းပါမယ်။ အပေါ်မှာပြောခဲ့ပြီးပြီနော်။ Hello world ဆိုတဲ့ စာတန်းလေး output ထုတ်ချင်လို့ဆိုတာ။ `printf(" ");` ဆိုတဲ့ function ကို ဘာကြောင့်ရေးပေးတာလဲဆိုတော့။ `printf` function ကသူ့ရဲ့ double quote(" ") ထဲမှာ ရေးသမျှကို Monitor မှာ ပြပေးတာမို့ပါ။ ဥပမာ - Hello world လို့ မရေးဘဲ I love U ဆိုပြီး double quote ထဲမှာ ရေးမယ်ဆိုရင် I love U ဆိုပြီး screen monitor မှာ မြင်ရမှာဖြစ်ပါတယ်။ ကိုယ်တိုင် စမ်းသပ်ကြည့်ပါ။ ဒီ `printf()` function လေးကို သုံးလို့ရတာကလည်း program ရဲ့ အစမှာ အဲ့ဒီ functionရဲ့ library file ဖြစ်တဲ့ Standard I/O header file ကို include လုပ်ထားလို့ပါ။ ဟုတ်ပြီ နောက် တချက် သတိထားရမှာက semicolon (;) ပါ။ Statement တစ်ကြောင်းဆုံးတိုင်း မရေးမဖြစ်ရေးပေးရမှာပါ။ မရေးပေးဘူးဆိုရင်တော့ အပေါ်မှာ ပြောခဲ့ သလိုပဲ Compile error တက်ပြီပေါ့ဗျာ။ အဆင်ပြေတယ်နော်။
- ◆ ကဲ..ခု program လေးပြီးပြီဆိုတော့ ပြီးပြီလို့ကြေညာလိုက်ရအောင်။ ဘယ်လိုကြေညာမလဲ။ လွယ်လွယ်လေးပေါ့။ closing brace (}) လေးနဲ့ ပိတ်ပေးလိုက်ရုံပဲ။ ကဲ ဒါဆို success program လေး တပုဒ်ဖြစ်သွားပြီပေါ့နော်။ ခုကျွန်တော် Example လေးတွေပေးမယ်ဗျာ။ ကိုယ်တိုင်ရေးကြည့်ပါ။
 My name is Mg Mg
 I Love U Baby !
 Hi! Candy.
 အိုကေ... ဒီစာသားလေးတွေကို screen မှာပေါ်အောင်ရေးကြည့်ပါ။ အဆင်ပြေပါစေ။

5. Programming Rules (C Program တွင်လိုက်နာရမည့် စည်းကမ်းများ)

ဘယ် programming languages မှာမဆို program ရေးသားတဲ့နေရာမှာ သတ်မှတ်ထားတဲ့ စည်းကမ်းတွေရှိ တယ်ဗျ။ ခု C programming နဲ့ပတ်သက်ပြီး လိုက်နာရမယ့် စည်းကမ်းတွေကို ပြောပြပါမယ်။

၁။ Statements အားလုံးကို lower case letter နဲ့ပဲရေးရပါမယ်။ upper case letter တွေကိုတော့ symbolic constants တွေအတွက်ပဲ ရေးလို့ရပါမယ်။ Symbolic constants ဆိုတာကို နားလည်အောင် ဥပမာ ပြပေးပါ မယ်။

```
#define AGE 19
```

```
#define NAME "Mg Mg"
```

ဒီဥပမာမှာဆိုရင် AGE, NAME တွေဟာ Symbolic constant တွေဖြစ်ပါတယ်။

၂။ စာလုံးတစ်ခုနဲ့ တစ်ခုကြားမှာ Blank spaces တွေထားပြီးရေးလို့ရပါတယ်။ ဒါပေမယ့် variable name တွေ keywords, constant and function တွေကြောင့် အခါမှာတော့ Blank spaces တွေထားလို့မရပါဘူး။

၃။ open brace { နဲ့ close brace } အတွင်းမှာ Statement တွေကို မိမိကြိုက်နှစ်သက်ရာ နေရာမှာ ရေးလို့ ရပါတယ်။ ဘယ်နေရာမှာ ရေးရမယ်လို့ သတ်မှတ်ထားတာမရှိပါဘူး။ ဒါပေမယ့် Syntax မှန်အောင်တော့ ရေးဖို့ လိုပါတယ်။ နောက်တချက် programmer က Statement တွေကို တစ်ကြောင်းတည်းမှာပဲ စုပြီးရေးလို့ရပါတယ်။ ဒါပေမယ့် statement ရဲ့ အဆုံးမှာတော့ semicolon(;) ပိတ်ပေးရပါမယ်။

ဥပမာ-

```
a =b+c;
```

```
d=b*c;
```

(or)

```
a =b+c; d=b*c; နားလည်တယ်ဟုတ်...။
```

၄။ opening brace { နဲ့ closing brace } ရဲ့ အရေအတွက် ညီမျှရပါမယ်။ ဆိုလိုတာကတော့ဗျာ open brace ခုပါတယ်ဆိုရင် close brace ကလည်း ခုပါတယ်။ ဒီလောက်ဆိုလုံလောက်ပါပြီ။ နောက်ထပ်စည်းကမ်း တွေကိုတော့ သက်ဆိုင်ရာ lesson လိုက် ရှင်းပြပေးပါမယ်။

2. The C Declarations

1. C character Set များ

ဒီအခန်းမှာတော့ C language မှာသုံးတဲ့ character set တွေနဲ့ ပတ်သက်ပြီး ပြောပြပါမယ်။

1. Letters

2.Digits

3.White spaces

4.Special characters တွေဖြစ်ပါတယ်။ ပိုရှင်းသွားအောင် Table လေးနဲ့ ပြပါမယ်။

1. Letters	2.Digits	3. White Spaces
Capital A to Z	All decimal digits 0 to 9	Blank space
Small a to z		Horizontal tab
		Vertical tab
		New line
		Form feed

4. Special Characters

,	Comma	&	Ampersand
.	Period or dot	^	Caret
;	Semi-colon	*	Asterisk
:	Colon	-	Minus
'	Apostrophe	+	Plus
"	Quotation mark	<	Less than
!	Exclamation mark	>	Greater than
	Vertical bar	()	Parenthesis left/right
/	Slash	[]	Bracket left/right
\	Back slash	{ }	Braces left/ right
~	Tilde	%	Percent
_	Under score	#	Number sign or Hash
\$	Dollar sign	=	Equal to
?	Question mark	@	At the rate

ဒီ Character Set တွေရဲ့ အသုံးပြုပုံကိုတော့ Program တွေရေးတဲ့ အခါမှာ ရှင်းပြပါမယ်။ တချို့အသုံးနှုန်းတွေ ကိုတော့ သိပြီးသားဖြစ်မှာပါ။ ဒါတွေကို ကျက်ဖို့တော့ မလိုပါဘူး။ program တွေရေးသားရင်းမှတ်မိသွားပါမယ်။

အခု အသုံးများတဲ့ Special Characters တွေကို ပြောပြပါမယ်။

Delimiters	Use
------------	-----

:	Colon	Use for label
;	Semicolon	Terminates statement
()	Parenthesis	Used in expression and function
[]	Square bracket	Used for array declaration
{ }	Curly Brace	Scope of statement
#	Hash	Preprocessor directive
,	Comma	Variable separator

ဒါကိုတော့ ရှင်းပြစရာ လိုမယ်မထင်ဘူးဗျ။ အသုံးလေးတွေ ရှင်းထားတာက ရှင်းရှင်းလေးဆိုတော့။

2. C Keywords များ

Program တစ်ပုဒ်ရေးတော့မယ် ဆိုရင်သိထားရမယ့် အရေးကြီးတဲ့ C keywords တွေကို ပြောပြပါမယ်။ ဘာလို့အရေးကြီးတယ်ဆိုတော့ ဒီ keywords တွေကို C Compiler က သိပြီးသားဖြစ်နေလို့ပါ။ သိတော့ ဘာဖြစ်လဲ။ ကျွန်တော်တို့ data တွေကို ကွန်ပျူတာရဲ့ memory ပေါ်မှာ သိမ်းဆည်းဖို့အတွက် variable name တွေ ကြေငြာပေးရတယ်ဗျ။ အဲလို အမည်ပေးတဲ့ အခါမှာ C compiler ကသိပြီးသားဖြစ်တဲ့ keyword တွေရဲ့ အမည်တွေနဲ့ သွားပြီးတူလို့မရဘူးဗျ။ တူမယ်ဆိုရင်တော့ error တက်ပြီပေါ့ဗျ။ ဒါကြောင့်မို့ keyword တွေကို သေချာမှတ်ထား ပေးဖို့ပြောချင်တယ်ဗျ။ ကျက်ဖို့တော့လိုမယ်မထင်ဘူးဗျ။ ဘာလို့လဲဆိုတော့ အိုင်းစတိုင်း ပြော ထားတာလေးရှိတယ်ဗျ။ “စာအုပ်ထဲမှာ ပြန်ရှာလို့ရတဲ့ အရာတွေကို ဦးနှောက်ထဲမှာ မှတ်ထားစရာမလိုဘူး” တဲ့ဗျ။ ဒါကြောင့်မို့ ဘယ်အချိန်ဘဲ သိချင် သိချင် ကြည့်လို့ရအောင် Table လေးနဲ့ ပြပေးပါမယ်။

C Keywords

auto	do	Double	If	Int	static	Struct	while
Break	Default	Else	Goto	Long	Sizeof	Switch	Volatile
Case	Continue	Enum	For	Register	Signed	Typedef	Void
char	Const	extern	Float	return	Short	union	Unsigned

3. Identifiers ဆိုတာဘာလဲ

Identifiers ဆိုတာက variables name, functions name, arrays name တွေကို ခေါ်ဆိုခြင်းဖြစ်ပါတယ်။ Identifiers တွေမှာ under score(_) တွေကိုလည်း သုံးလို့ရပါတယ်။ ဘယ်နေရာမှာ သုံးလဲဆိုတော့ စကား လုံးတစ်ခုနဲ့ တစ်ခု ကို ဆက်စပ်ပေးချင်တဲ့နေရာပေါ့ဗျ။ ဥပမာ- F_name, sum_of ဆိုပြီး အမည်တွေပေးတဲ့ နေရာမှာပေါ့ဗျ။ ခု user သတ်မှတ်ပေးတဲ့ identifiers လေးတွေကို example တစ်ပုဒ်နဲ့ ရှင်းပြပါမယ်။

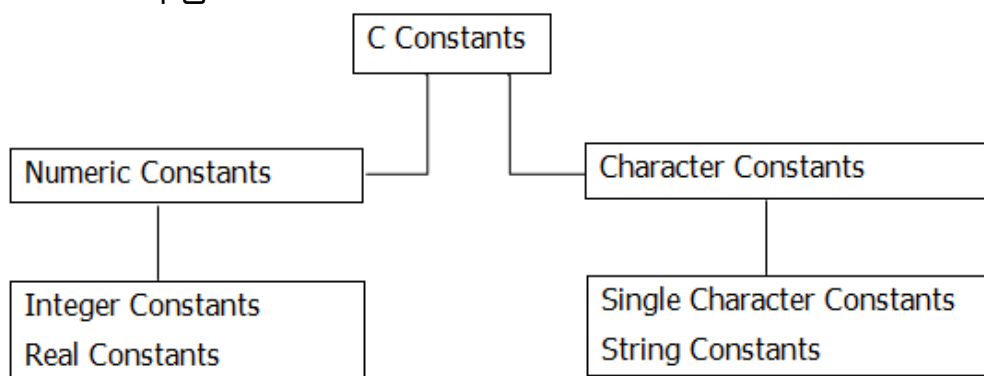
a) #define N 10

b) # define a 15

ဆိုပါတော့ဗျာ။ အဲ့ဒီမှာ N နဲ့ a ဟာ user သတ်မှတ်လိုက်တဲ့ identifiers တွေပဲဖြစ်ပါတယ်။ အဲလောက်ဆို သဘောပေါက်မယ်လို့ထင်ပါတယ်ဗျာ။

4. Constants ဆိုတာဘာလဲ

Constants ဆိုတာ program ကို execute လုပ်တဲ့အခါမှာ မပြောင်းလဲဘဲ ရှိနေတဲ့ ကိန်းသေအဖြစ် သတ်မှတ်ထားတဲ့ တန်ဖိုးတွေကို ဆိုလိုခြင်းဖြစ်ပါတယ်။ ဆိုလိုတာကတော့ဗျာ a ရဲ့ တန်ဖိုးကို 3 လို့ ကိန်းသေ အဖြစ် သတ်မှတ်ထားလိုက်ရင် program ကို execute လုပ်တဲ့အခါမှာ a ရဲ့ တန်ဖိုးဟာ 3 ပဲဖြစ် နေ မယ်လို့ ဆိုလို တာပါ။ ဒါကို constants value လို့ခေါ်ဆိုခြင်းဖြစ်ပါတယ်။ C language မှာ ကိန်းသေ သတ်မှတ်လို့ ရတဲ့ အမျိုး အစားတွေ အများကြီးရှိပါတယ်။ ဘာတွေလဲ ဆိုတာကို Table လေးနဲ့ပြပေးပါမယ်။



1. Numeric Constants

a) Integer Constants

Integer Constants ဆိုတာကတော့ ဒသမကိန်းမဟုတ်တဲ့ ကိန်းပြည့်တန်ဖိုး တွေကို ခေါ်ဆိုခြင်းဖြစ်ပါတယ်။ ကိန်းပြည့်ဆိုတော့ အပေါင်းရော၊အနှုတ်ရော ပါတာပေါ့။ လက္ခဏာမပါဘူး ဆိုရင် တော့ အပေါင်းပေါ့။ သူတို့တွေ ကိုလည်း C language မှာ Constants တွေအဖြစ် သတ်မှတ်လို့ရပါတယ်။

ဥပမာ- 10, 20, +30, -5 etc.

b) Real Constants

Real constants ဆိုတာကတော့ ကိန်းစစ်တန်ဖိုးကို ပြောခြင်းဖြစ်ပါတယ်။ တနည်းအားဖြင့် floatin point constants (ဒသမကိန်း) တွေကို ခေါ်ဆိုခြင်းဖြစ်ပါတယ်။

ဥပမာ- 2.5, 5.22, 3.14 etc. တွေဟာ real constants တွေဖြစ်ပါတယ်။

ဒသမကိန်းတွေကို နောက်တနည်းအားဖြင့် ထပ်ကိန်းတင်ပြီးရေးလို့ရပါသေးတယ်။

ဥပမာ- 2456.123 ကို 2.456123E3 လို့ exponent တင်ပြီးရေးလို့ရပါတယ်။ 'E' ဆိုတဲ့ letter လေးက mantissa နဲ့ exponent ကို ခွဲပေးထားပါတယ်။

2.456123E3 ← Exponent
 ↑
 Mantissa

Exponent တန်ဖိုးတွေဟာ အပေါင်းတန်ဖိုးတွေဖြစ်နိုင်သလို၊ အနှုတ်တန်ဖိုးတွေလည်း ဖြစ်နိုင်ပါတယ်။ Real constants တွေကို လည်း ကိန်းသေ(constant) တန်ဖိုးတွေအဖြစ် သတ်မှတ်လို့ရပါတယ်။

2. Character Constants

1) Single character constants

Character တစ်လုံးတည်းပါတဲ့ constant ကို Single character constant လို့ခေါ်ပါတယ်။ အဲဒီ constant တွေကို single quote (' ') လေးထဲမှာ ထည့်ပြီးရေးပေးရပါတယ်။

ဥပမာ- 'a' , '8' , ' ' etc.

Character constant တွေမှာ ASCII (American Standar Code For Information Interchange) ကနေ သတ်မှတ်ထားတာတွေရှိတယ်။ ဆိုလိုတာကတော့ဗျာ letter A to Z ကို integer တန်ဖိုး 65 ကနေ 90 ထိ အစဉ်လိုက်သတ်မှတ်ထားပါတယ်။ Small letter a to z ကိုတော့ 97 ကနေ 122 ထိ decimal တန်ဖိုးတွေနဲ့ သတ်မှတ်ပေးထားပါတယ်။ ပို ရှင်းသွားအောင် example လေးပြပါမယ်။

printf("%c %d", 65, 'B') ဆိုပြီး Output ထုတ်ရင် 'A' နဲ့ 66 ကို display လုပ်ပြပါမယ်။ ဘာလို့လည်း ဆိုတော့ A ရဲ့ ASCII value က 65 ပါ။ ဒါကြောင့် conversion character (%c) ကို သုံးပြီး character တန်ဖိုး Output ထုတ်လိုက်တဲ့ အတွက် 65 ကို မပြဘဲ သူ့ရဲ့ character တန်ဖိုး A ကို Output ထုတ်ပေးခြင်းဖြစ်ပါတယ်။

B လည်း ဒီအတိုင်းပါပဲ။ letter B ရဲ့ ASCII တန်ဖိုးက 66 ပါ။ conversion character (%d) ကိုသုံးပြီး သူ့ရဲ့ Decimal တန်ဖိုးကို output ထုတ်တဲ့အတွက် B ကို မပြဘဲ သူ့ရဲ့ decimal တန်ဖိုး 66 ကိုပြပေးထားခြင်းဖြစ် ပါတယ်။ နားလည်မယ်လို့ထင်ပါတယ်။ ASCII တန်ဖိုး 0 ကနေ 127 ထိရှိပါတယ်။ မိမိဘာသာ ဘယ် decimal တန်ဖိုးရဲ့ character ဟာ ဘာဖြစ်တယ်ဆိုတာ စမ်းသပ်ကြည့်ပါ။

2) String Constants

String Constants ဆိုတာကတော့ double quote (" ") ထဲမှာရေးပေးရတဲ့ စကားစုလေး ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ ကိန်းဂဏန်း အစုလေးလည်းဖြစ်နိုင်ပါတယ်။

ဥပမာပြမယ်ဗျာ..... "Hello" , "India" , "4444" , "a" ဆိုတာတွေကို ခေါ်ဆိုခြင်းဖြစ်ပါတယ်။ အလွယ် မှတ်ရရင် double quote(" ") ထဲမှာရေးတာတွေ ကို String constants တွေလို့ ခေါ်ပါတယ်။ နားလည်မယ် လို့ထင်ပါတယ်။

5. Variables ဆိုတာဘာလဲ

Variables ဆိုတာကို ရှင်းပြပါမယ်။ ကျွန်တော်တို့ program မှာ သုံးမယ့် data တွေကို ကွန်ပျူတာရဲ့ memory မှာ သိမ်းဆည်းဖို့အတွက် variable name တွေပေးပြီးကြေငြာပေးရတယ်။

အရှင်းဆုံးဥပမာပြရင်ဗျာ ဗိုလ်လောင်းအောင်အောင်ရဲ့ ပစ္စည်းတွေကို သိမ်းဖို့အတွက် အောင်အောင်မှာ သေတ္တာတစ်လုံးရှိရမယ်လေ။ အဲ့ဒီသေတ္တာဟာ အောင်အောင်ရဲ့ သေတ္တာပါလို့ သိအောင် နာမည်တပ်ပေးထားရပါမယ်။ အဲလို အောင်အောင် ရဲ့ သေတ္တာလို့ နာမည်ပေးတာကို variable name ကြေငြာပေးတာလို့ခေါ်ပါတယ်။ Variables ဆိုတဲ့အတိုင်း သူ့ရဲ့ တန်ဖိုးဟာ ကိန်းသေမရှိပါဘူး။ ဆိုလိုတာကတော့ဗျာ a ကို variable လို့သတ်မှတ်တယ်ဆိုပါစို့။ a ရဲ့ တန်ဖိုးဟာ 1 လည်းဖြစ်နိုင်သလို အခြားတန်ဖိုးတွေလည်းဖြစ်နိုင်ပါတယ်။ (a ကို constant အဖြစ်သတ်မှတ်မှတ် မယ်ဆိုရင်တော့ သူ့ရဲ့တန်ဖိုးက ပြောင်းလဲမသွားပါဘူး။) ဒါကို variable လို့ခေါ်ခြင်းဖြစ်ပါတယ်။

Variable ရဲ့ အမည်တွေပေးတဲ့အခါမှာ အဆင်ပြေသလိုပေးလို့ရပါတယ်။ ဒါပေမယ့် မိမိအဓိပ္ပာယ်သတ်မှတ် ပေးမယ့်အရာနဲ့ အဆင်ပြေတာကို ပေးစေချင်ပါတယ်။ ဆိုလိုတာက တော့ဗျာ..... Triangle တစ်ခုရဲ့ Area ကိုတွက်ချက်ဖို့ အလျားအနံ ကို variable name ပေးချင်တယ်ဆိုပါစို့။ length, width လို့ ပေးရင် ပိုအဆင်ပြေ ပါမယ်။ length, width လို့မပေးဘဲ a, b လို့ပေးလည်းရပါတယ်။ variable name ပေးတဲ့အခါမှာ အဓိပ္ပာယ်ရှိရှိ ပေးဖို့အကြံပေးချင်တာပါ။

6. Variables သတ်မှတ်ပေးရာမှာ လိုက်နာရမည့် စည်းကမ်းများ

Variables ဆိုတာ ဘာလဲသိပြီဆိုတော့ ခု variable တွေသတ်မှတ်တဲ့နေရာမှာ အမည်ပေးတဲ့ နေရာမှာ လိုက်နာရမယ့် စည်းကမ်းလေးတွေကို ပြောပြချင်ပါတယ်။

- ၁။ Variable name တွေရဲ့ အစ စာလုံးဟာ character တွေပဲဖြစ်ရပါမယ်။ ကိန်းဂဏန်းတွေနဲ့ စလို့မရပါဘူး။
ဥပမာ= 1height ဆိုမှားပါတယ်။ height လို့ရေးမှမှန်ပါမယ်။ နားလည်တယ်ဟုတ်။ ဒါပေမယ့် underscore တွေခံပြီးရေးလို့တော့ရပါတယ်။ ဒီလိုမျိုးပါ။ (F_name).
- ၂။ Variables တွေရဲ့ အမည်တွေဟာ C Keywords တွေရဲ့ အမည်တွေနဲ့ တူလို့မရပါဘူး။ အထက်မှာလည်း ပြောထားခဲ့ပြီးပါပြီ။ ဘာကြောင့်တူလို့ မရတာလဲဆိုတာကို။
- ၃။ Variable name တွေပေးတဲ့ အခါမှာ character တူပေမယ့်လည်း upper နဲ့ lower မတူဘူးဆိုရင် ဒီ variable နှစ်ခုဟာ မတူပါဘူး။ ဥပမာဗျာ..... SUM နဲ့ sum variable နှစ်ခုဟာ မတူပါဘူး။ နားလည်မယ်လို့ ထင်ပါတယ်။
/* ဒါတွေကိုတော့ သေချာလေး မှတ်ထားစေချင်ပါတယ်ဗျ။ */

7. Data Types အမျိုးအစားများ

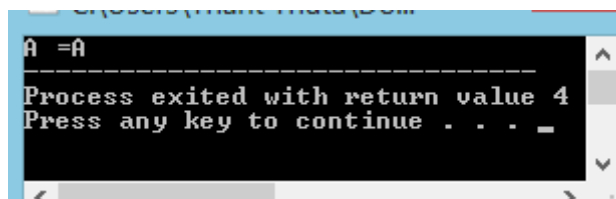
Data Types ဆိုတာကတော့ မိမိ program မှာ သုံးမယ့် data တွေရဲ့ အမျိုးအစားကို ဆိုလိုခြင်းဖြစ်ပါတယ်။ ဆိုလိုတာကတော့ဗျာ မိမိသုံးမယ့် data က integer လား၊ character လား၊ Floating point လားဆိုတာကို သတ်မှတ်ပေးဖို့ပြောတာပါ။ Data type သတ်မှတ်ပေးတဲ့နေရာမှာ သတိထားရပါမယ်။ မိမိသုံးမယ့် data က integer အမျိုးအစားဖြစ်တယ်ဆိုရင် integer data type ကို ကြေငြာပေးရပါမယ်။ Data type မှားပြီး ပေးမယ် ဆိုရင်တော့ မိမိလိုချင်တဲ့ output ကို မှန်ကန်အောင် ပြပေးမှာမဟုတ်တော့ပါဘူး။ ဟုတ်ပြီ....အခု ဘယ် data ဆိုရင် ဘယ်လို Data type တွေဖြစ်တယ်ဆိုတာ ဘယ်လို သတ်မှတ်ပေးရတယ်၊ရေးပေးရတယ်ဆိုတာကို Table လေးနဲ့ ပြပေးပါမယ်။

Data Type	Size/Bytes	Range	Format String
char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
short or int	2	-32768 to 32768	%d or %i
unsigned int	2	0 to 65535	%u
long	4	-214783648 to 214783648	%ld
unsigned long	4	0 to 4294967295	%lu
float	4	3.4e -38 to 3.4e +38	%f or %g
double	8	1.7e-308 to 1.7e+308	%lf
long double	10	3.4e-4932 to 1.1e+4932	%lf

အခု Table လေးကို ရှင်းပါမယ်။ char ဆိုတာ Character data တွေကို ကြေငြာမယ်ဆိုရင် ရေးပေးရမယ့် character data type name ပါ။ သူ့ရဲ့ Byte (memory allocation size) က 1 byte ဖြစ်ပါတယ်။ 1 byte ဖြစ် တဲ့အတွက် သူ့ရဲ့ သိမ်းဆည်းနိုင်မှု range က -128 to 127 ထိဖြစ်ပါတယ်။ format string(or) conversion character က %c ဖြစ်ပါတယ်။ အောက်က data type တွေကလည်း ဒီအတိုင်းပါပဲ။ အကုန်ရှင်းပြနေရင် အချိန် ကြာသွားမှာစိုးတဲ့ အတွက် ဒီလောက်ပါပဲ။ အပြည့်အစုံရှင်းလင်းတာကို တော့ video lesson မှာ နားထောင်ပေး ဖို့ ပြောချင်ပါတယ်။ (data type name နဲ့သူ့ရဲ့ format string တွဲလျက်ဖြစ်နေရပါမယ်။ဆိုလိုတာကဗျာ int ဆိုရင် format string က %d ဖြစ်ရမယ်လို့ပြောချင်တာပါ။ %c ဆိုပြီး ရေးလိုက်မယ်ဆိုရင်တော့ user လိုချင်တဲ့ output ကို မှန်ကန်အောင် ထုတ်ပေးမှာမဟုတ်တော့ပါဘူး။) ပိုပြီးနားလည်အောင် program လေးတပုဒ် နဲ့ ရှင်းပြပါမယ်။

```

1      #include<stdio.h>
2      main()
3      {
4          int a=65;
5          printf("A =%c",a);
6      }
```



ဒီ Program မှာဆိုရင် a ဟာ integer data type ပါဆိုပြီး (int) ဆိုတဲ့ data type ကိုသုံးပြီး ကြေငြာထားတယ်။ ဒါပေမယ့် integer data type ရဲ့ format string(conversion character) က

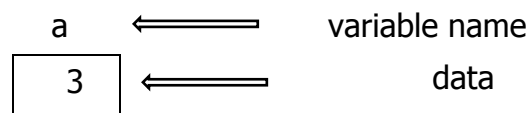
%d လေ။ ဒါပေမယ့် %d ကို မသုံးဘဲ %c ဆိုပြီး character data type ရဲ့ format string ကိုသုံးလိုက်တယ်။ အဲ့တော့ မိမိ Output ထုတ်ချင် တဲ့ a ရဲ့ integer တန်ဖိုး 65 ကို ထုတ်ပေးဘဲ character တန်ဖိုး A ကို output ပေးတယ်။ အဲ့တော့ data type နဲ့ format string တွဲလျက်မဖြစ်ဘူးဆိုရင် မိမိ output ထုတ်ချင်တဲ့ တန်ဖိုးတွေကို ထုတ်ပေးမှာ မဟုတ် တော့ဘူးပေါ့။ ဒါကြောင့် ဘယ် data type ဆိုရင် ဘယ် format string သုံးရမယ်ဆိုတာကို တွဲလျက်မှတ်ထား ပေးဖို့လိုပါတယ်။ အဲလောက်ဆို နားလည်မယ်လို့ထင်ပါတယ်။ program တွေရေးတဲ့ အခါမှာကောင်းကောင်း နားလည်လာပါမယ်။

8. Declaring Variables (Variables ကြေငြာခြင်း)

ဒီအပိုင်းမှာတော့ Variables တွေဘယ်လိုကြေငြာရလဲ၊ ဘာကြောင့် ကြေငြာပေးရတယ် ဆိုတာကို ရှင်းပြပေးပါ မယ်။ အပေါ်မှာ variables ကို ရှင်းပြတုန်းကလည်း အတော်များများပြောပြပြီးပါပြီ။ Variables တွေကို ကြေငြာ တဲ့အခါမှာ C program ရဲ့ structure အတိုင်းပါပဲ။ Declaration part မှာ ကြေငြာပေးရပါမယ်။ ဆိုလိုတာက တော့ဗျာ program ရဲ့ အစ main function ရဲ့အောက်မှာ ကြေငြာပေးရပါမယ်။ Global အနေနဲ့ သုံးချင်တယ် ဆိုရင်တော့ main function ရဲ့ အပြင်ဘက်မှာ ကြေငြာပေးရပါမယ်။ ဘာကြောင့်ကြေငြာပေးရလဲဆိုတာကို အထက်မှာလည်း ပြောပြခဲ့ပါသေးပါတယ်။ ဗိုလ်လောင်းအောင်အောင်ရဲ့ ပစ္စည်းတွေထည့်ဖို့ သေတ္တာမှာ ဗိုလ်လောင်းအောင်အောင် ဆိုပြီး နာမည်ကပ်ထားပေးရတယ်ဆိုတာ။ ဒါကို ရှင်းသွား အောင် ဥပမာပြပါမယ်။

\int int b=3; \leftarrow (3=အောင်အောင် ပစ္စည်း)
 ဗိုလ်လောင်း
 အောင်အောင် \leftarrow သေတ္တာ

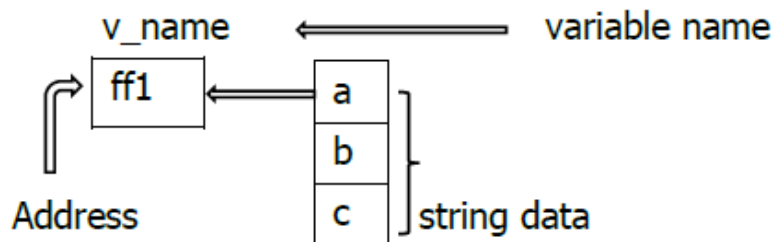
int ဆိုတာက data type ပါ။ 3 ဆိုတဲ့ အောင်အောင်ရဲ့ ပစ္စည်းတွေကို ဘယ်မှာသိမ်းမလဲ။ bဆိုတဲ့ variable name သေတ္တာထဲမှာ သိမ်းမယ်။ ဒီ သေတ္တာက ဘယ်လိုသေတ္တာလဲ integer ဆိုတဲ့ ဗိုလ်လောင်းအောင်အောင် ပိုင်တဲ့သေတ္တာ။ ဒါကြောင့် မိမိသိမ်းမယ့် data ကို variable name ကြေငြာပြီးတော့သိမ်းမယ်။ ပြီးရင် အဲ့ဒီ variable က ဘာ data အမျိုးအစားလဲဆိုပြီး data type ကြေငြာပေးမယ်။ ကွန်ပျူတာရဲ့ memory မှာ သိမ်းဆည်းထားပုံလေးကို ပုံလေးနဲ့ပြပေးပါမယ်။



ကဲရှင်းလားတော့မသိဘူးဗျာ။ ကျွန်တော်စာအရေးအသား ညံ့တယ်ဆိုလည်း နားလည်ပေးပါ။ အဲလိုမျိုး ကြေငြာပေးမှ Compiler က variable name ကိုသိပါမယ်။ အထက်မှာပြောခဲ့တာ မှတ်မိသေးလားမသိဘူးဗျာ။ ကွန်ပျူတာရဲ့ memory မှာ data တွေသိမ်းဆည်းဖို့ variable name တွေပေးပြီး ကြေငြာပေးရတယ်ဆိုတာ။ အဲ့တော့ ကျွန်တော်တို့ ကြေငြာမယ့် variable မှာ data တွေဘယ်လောက်သိမ်း ထားမယ်ဆိုတာ လိုတယ်ဗျာ။ လိုရင်းအချက်က အဲ့ဒါပဲဗျာ။ data type

အမျိုးအစားလိုက် memory နေရာယူမှု၊ အနဲ အများရှိတယ်။ ဥပမာ integer ဆိုရင် 2 byte နေရာယူမယ်။ char ဆိုရင် 1 byte နေရာယူမယ်။ ဒါကြောင့် program ရဲ့ အစမှာ ဒီ variable က ဘာ data type အမျိုးအစားဖြစ်ပါတယ်။ memory space ဘယ်လောက် ယူပါမယ်ဆိုတာ Compilerက သိမှ သူ့ရဲ့ အလုပ်ကို ဆက်လုပ်မှာပါ။ integer data တွေဟာ value type အမျိုးအစားတွေဖြစ်ပါတယ်။ String ဆိုလည်း ဒီအတိုင်းပါဘဲခင်ဗျ။ ဒါပေမယ့် string ကျတော့ data သိမ်းဆည်း တာမတူဘူး။ ဘာလို့လဲဆိုတော့ သူက reference data type ဖြစ်လို့ပါ။ ပုံလေးနဲ့ပြပါမယ်။

```
char* v_name="abc"
```



သူ့ရဲ့ memory သိမ်းဆည်းထားပုံလေးကို ရှင်းပြပါမယ်။ `abc` ဆိုတဲ့ string ကို `v_name` ဆိုတဲ့ variable name ထဲမှာ သိမ်းမထားပါဘူး။ `v_name` ဆိုတဲ့ variable name မှာ `abc` ဆိုတဲ့ string ရှိနေတဲ့ Address ကို သိမ်းထားပါတယ်။ ဒါကြောင့်မို့ String ဟာ reference type လို့ပြောတာပါ။ ကဲဒီလောက်ဆို variable name တွေမှာ data တွေ သိမ်းဆည်းပုံကို နားလည်မယ်လို့ထင်ပါတယ်။

9. Variables သတ်မှတ်ခြင်း

Initializing ဆိုတာက variable တွေကို တန်ဖိုးတခု အစားသွင်းသတ်မှတ်ပေးလိုက်ခြင်းကို ဆိုလိုခြင်းဖြစ် ပါတယ်။ ဒါကို assign လုပ်တယ်လို့လည်းခေါ်သေးပါတယ်။ အဲလို assign လုပ်မယ်ဆိုရင်တော့ assignment operator ဖြစ်တဲ့ `" = "` ကို သုံးရပါမယ်။ Operator အကြောင်းကို နောက် သင်ခန်းစာတွေမှာ ရှင်းပြပေးပါမယ်။ ခုတော့ assignment operator လေးကို ဘာကြောင့်သုံးတာလဲဆိုတာကိုပဲ မှတ်ထားပေးပါ။ နောက်တချက်... initializing နဲ့ declaring ကို တစ်ကြောင်းတည်းမှာပဲ ကြေငြာပေးလို့ရပါတယ်။ ဥပမာပြပါမယ်။

```
int a;
```

ဒါကတော့ `a` ဆိုတဲ့ variable name ဟာ integer data type ဖြစ်ပါတယ်လို့ကြေငြာခြင်းဖြစ်ပါတယ်။

နောက်တစ်ကြောင်းမှာတော့....

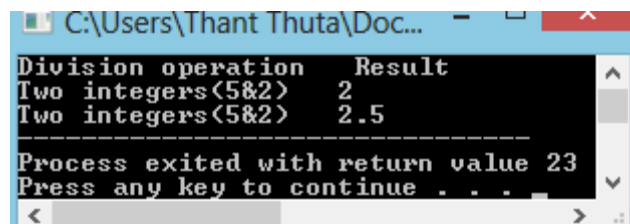
`a=3;` ဆိုပြီး `a` ရဲ့ တန်ဖိုးဟာ 3 ဖြစ်ပါတယ်လို့ assign လုပ်လိုက်ခြင်းဖြစ်ပါတယ်။ အဲ့ဒီမှာဆိုရင် declaring နဲ့ initializing ကို တစ်ကြောင်းဆီ ကြေငြာခြင်းဖြစ်ပါတယ်။ အဲလိုခွဲမရေးဘဲ...

`int a=3;` ဆိုပြီး တပြိုင်နက် ကြေငြာပေးလို့လည်းရပါတယ်။ ရှင်းမယ်လို့ထင်ပါတယ်ခင်ဗျ။

10. data type ပြောင်းလဲခြင်း

Type Conversion ဆိုတာကတော့ data type တမျိုးကနေ မိမိလိုချင်တဲ့ data type ကို ပြောင်းလဲယူ တာကို ခေါ်ဆိုခြင်းဖြစ်ပါတယ်။ ဒါကိုပိုရှင်းသွားအောင် program လေးရေးပြီး လေ့လာကြည့်ရအောင်။

```
1 #include<stdio.h>
2 main()
3 {
4     printf("Division operation   Result\n");
5     printf("Two integers(5&2)    %d \n",5/2);
5     printf("Two integers(5&2)    %g", (float)5/2);
7 }
```



ဒီ program ကို တစ်ကြောင်းချင်းစီမရှင်းတော့ဘူးပျ။ အဓိကအချက်ကို ပဲပြောသွားပါမယ်။

- ◆ ပထမတစ်ကြောင်း printf() function ကို ရေးပေးတာတော့ output မှာမြင်တဲ့အတိုင်း Division operation Result ဆိုပြီးပေါ် စေချင်လို့ရေးပေးထားတာပါ။ printf() ရဲ့ အလုပ်လုပ်ပုံကိုလည်း ရှေ့ပိုင်းမှာပြောပြပြီးသားပါ။ ဒုတိယ printf() ကိုတော့ 5 ကို 2 နဲ့ စားတဲ့အခါရလာမယ့် စားလဒ်တန်ဖိုးကို Screen မှာပြစေချင်တဲ့အတွက် သုံးခြင်းဖြစ်ပါတယ်။
- ◆ Printf() function သုံးပြီး output ထုတ်လို့ရတဲ့ နည်း ၂ခုရှိပါတယ်။တစ်ခုကတော့ ပထမတစ်ကြောင်းမှာ သုံးတဲ့ နည်းပါ။ နောက်တနည်းက ဒုတိယကြောင်းမှာရေးတဲ့ ပုံစံပါ။ တွက်ချက်လို့ရတဲ့ တန်ဖိုးတွေကို output ထုတ်ချင် တဲ့ အခါမျိုးပါ။အဲလို output ထုတ်တဲ့အခါမှာ တွက်ချက်လို့ရတဲ့ တန်ဖိုးဟာ ဘာ data type အမျိုးအစားလည်း ဆိုတာကို ကြည့်ပြီး format string သုံးပြီး ရေးပေးရပါမယ်။ ခု program မှာဆိုရင် 5 ဆိုတဲ့ integer ကို 2 ဆိုတဲ့ integer နဲ့ စား(Division) တဲ့အခါမှာ အဖြေကဘာထွက်မလဲ ။ ဒသမကိန်းနဲ့ ထွက်မယ်ဆိုတာ သိတယ်ဟုတ်။ ဒါပေမယ့် ကျွန်တော်တို့က ဒသမကိန်းနဲ့ မလိုချင်ဘူး။ integer တန်ဖိုးနဲ့ပဲလိုချင်တာဖြစ်တဲ့အတွက် integer data type ရဲ့ format string (%d) ကို double quote(" ")အတွင်းမှာရေးပြီး output ထုတ်လိုက်တဲ့အတွက် အမှန်တကယ် စားလို့ရတဲ့ တန်ဖိုး 2.5 ကို မပြဘဲ 2 လို့ပဲပြပါတယ်။ဘာကြောင့်လဲဆိုတော့ ကျွန်တော်တို့က %d ဆိုပြီး integer data type ရဲ့ f.s ကို သုံးထားလို့ဖြစ်ပါတယ်။ အဲလိုရေးတဲ့အခါမှာ သူ့ရဲ့ syntax လေးကိုတော့ မှတ်ထားပေးပါ။ double quote အတွင်းမှာ output return ပြန်မယ့်တန်ဖိုးရဲ့ f.s ကိုရေးပေးရမယ်။ ပြီးရင် comma(,) လေးခံမယ်။ တန်ဖိုးတွေ တွက်ချက်ထားတဲ့ statement ဒါမှမဟုတ် တွက်ချက်ထားတဲ့ တန်ဖိုး တွေ သိမ်းထားပေးတဲ့

variable name ကို ရေးပေးရမည်။ပြီးမှ semicolon ပိတ်ပေးလိုက်ပါ။ နားလည်မယ်လို့ ထင်ပါတယ်ဗျ။

- ◆ ခု Type Conversion လုပ်တာလေးကို ရှင်းပါမယ်။ အထက်ပါ program မှာ တွက်ချက်လို့ ရတဲ့ result ကို 2.5 ဆိုပြီး အတိအကျ အဖြေထွက်စေချင်တယ်ဗျ။အဲ့တော့ type ပြောင်းပေးဖို့လိုလာပါတယ်။ ဘာလို့လဲ ဆိုတော့ integer ကို integer နဲ့ စားသည်ဖြစ်စေ၊ပေါင်း၊နှုတ်၊မြှောက် လုပ်မယ်ဆိုရင် output ကလည်း integer data type ကို ထုတ်ပေးမှာပါ။ နားလည်တယ်ဟုတ်။ program ကို ကြည့်ပါ။ တတိယကြောင်း မှာ ရေးထားတဲ့ statement မှာ type conversion လုပ်ထားတာကို ရှင်းပြပါမယ်။ (float)5/2 လို့ရေးပြီး type ပြောင်း လဲ လိုက်ပါ တယ်။ ဘယ်လိုပြောင်းလဲဆိုတော့ 5 ကို 2 နဲ့ Division လုပ်တဲ့အခါမှာ ရလာမယ့် တန်ဖိုးဟာ integer ကို integer နဲ့ စားတာဆိုတော့ ရလာမယ့်တန်ဖိုးရဲ့ data type ကိုလည်း integer data type အဖြစ် အဖြေထုတ်ပေးပါလိမ့်မယ်။ ဒါကို တကယ်ရလာမယ့်တန်ဖိုးအတိအကျ ဒသမကိန်းကို အဖြေထုတ်ဖို့ integer ကို float အဖြစ်ပြောင်းမယ်။ဒါကြောင့် 5/2 ရဲ့ ရှေ့မှာ float ဆိုပြီးရေးပေးလိုက်တာပါ။ အဲလိုကြေငြာပေးလိုက်ရင် integer ကို integer နဲ့စားတဲ့ result ရဲ့ return integer data type ဟာ float data type ဖြစ်သွားပြီ။ နားလည် မယ်ထင်ပါတယ်ဗျ။ ဒါကို output ထုတ်ပေးဖို့အတွက် double quote နှစ်ခုကြားမှာ %g ဆိုပြီး float data type ရဲ့ format string နဲ့ output ထုတ်ပေးလိုက်တာဖြစ်ပါတယ်။ %g လို့မရေးဘဲ %f လို့ရေးလိုက်လည်းအတူတူ ပါပဲ။ နှစ်ခုစလုံးက float data type ရဲ့ F.S တွေပဲဖြစ်ပါတယ်။ ဒါကိုနားလည်သွားပြီလို့ထင်ပါတယ်ခင်ဗျ။

11. Constant And Volatile Variables

a) Constant Variables

Constant Variables ဆိုတာကတော့ အရှင်းဆုံးပြောရရင် program တပုဒ်လုံးမှာ constant အဖြစ် ကြေငြာထားတဲ့ variable ရဲ့ တန်ဖိုးဟာ လုံးဝပြောင်းလဲမသွားဘူးလို့ဆိုလိုတာပါ။ အဲလိုတန်ဖိုးတွေကို constant အဖြစ် ထားချင်တယ်ဆိုရင်တော့ ကြေငြာပေးရမယ့် ပုံစံကို ပြောပြပါမယ်။

const int a=4; အဲလိုမျိုးကြေငြာပေးရပါမယ်။ အဓိပ္ပာယ်ကတော့ဗျာ variable a ရဲ့ တန်ဖိုး 4 (integer data type) အမျိုးအစားက ဒီ program မှာ ကိန်းသေဖြစ်ပါမယ်လို့ Compiler ကိုပြောလိုက်ခြင်းဖြစ်ပါတယ်။ ဒီလိုမျိုး constant အဖြစ်ထားချင်တယ်ဆိုရင်တော့ const ဆိုတဲ့ keyword လေးကို data type ရဲ့ ရှေ့က နေပြီး ရေးပေးရပါမယ်။

b) Volatile variable

Volatile variable ကတော့ constant variable နဲ့ပြောင်းပြန်ပါ။သူ့ရဲ့ တန်ဖိုးကပြောင်းလဲမှု ရှိပါတယ်။ ပြောင်းလဲမှုဆိုတာကတော့ဗျာ volatile အဖြစ်သတ်မှတ်ထားတဲ့ variable ရဲ့ တန်ဖိုးကတော့ user က 3 လို့ input ထည့်ရင် 3 ဖြစ်မယ်။ ဆိုလိုတာကတော့ user input

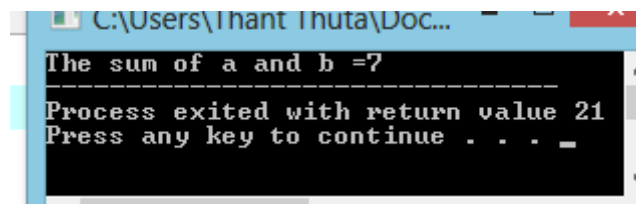
ထည့်လိုက်တဲ့တန်ဖိုးတိုင်းဟာ ဒီ variable ရဲ့ တန်ဖိုးပဲ ဖြစ်တယ်။နားလည်ပြီနော်...။ အဲလို volatile အဖြစ်ထားချင်တဲ့ variable တွေကိုကြောငြာတဲ့ syntax လေးကို ပြပါမယ်။

volatile int d; ဟုတ်ပြီ။ဒါကို ဘာသာပြန်မယ်ဆိုရင်တော့ဗျာ။ d ဆိုတဲ့ variable လေးက integer data type လေးဖြစ်ပြီး တန်ဖိုးပြောင်းလဲမှုရှိတဲ့ volatile variable လေးဖြစ်ပါတယ်။အာယူ...အိုကေ(ဟဲ)။သူက လည်း Constant variable လို့ပဲ။ volatile ဆိုတဲ့ keyword လေးကြောငြာပေးရမယ်။ သတိပေးချင်တာလေး ရှိတယ်ဗျာ။ keyword တွေကြောငြာတဲ့အခါ small letter ကိုသုံးပြီးကြောငြာပေးရမယ်နော်။

ကဲ..ဒါဆိုရင် program လေးတပုဒ်လောက် ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  void main()
3  {
4      int a;
5      a=3;
6      int b=4;
7      printf("The sum of a and b =%d",a+b);
8  }
```



ကဲဒီ program လေးကို line by line trace(စဉ်းစား)ကြည့်ရအောင် ။

- ◆ ပထမဆုံး header file ကိုကြောငြာမယ်။ဘာကြောင့်ကြောငြာတာလဲ။ သူ့ထဲမှာရှိတဲ့ function တွေကို ယူသုံးချင်လို့ပါ။ data တွေကို output ထုတ်ဖို့ printf() function ကို သုံးချင်တဲ့အတွက် standard input /output (stdio.h) ဆိုတဲ့ header file ကို include လုပ်ထားလိုက်တယ်။
- ◆ ပြီးရင် main() function ကို ဖြတ်ပြီး တော့ သူ့ထဲမှာရှိတဲ့ statement တွေကို run မယ်။ တခု ပြောဖို့ မှေး ခဲ့တယ်ဗျာ။ program ကို လေ့လာတဲ့ အခါမှာ line by line စဉ်းစားပါလို့ပြောခဲ့တယ်နော်။ဘာလို့လည်းဆိုတော့ compiler က programကို run တဲ့ အခါမှာ တစ်ကြောင်းချင်းစီ run တာဖြစ်တယ်ဗျာ။ဒါကြောင့်မို့ တစ်ကြောင်း ချင်းစီမှာ ရှိတဲ့ statement တွေကို နားလည်သဘောပေါက်စေချင်လို့ပါ။
- ◆ main () function ပြီးရင် programရဲ့ အစနဲ့ အဆုံးကို သတ်မှတ်ပေးတဲ့ curly braces({ }) ကိုရေးမယ်။ ပထမတစ်ကြောင်းမှာ ရေးထားတာက Declaration part ၊ variable တွေကို ကြောငြာပေးတဲ့အပိုင်းပါ။ int a; ဆိုပြီး variable a မှာ integer data type တွေသိမ်းပါမယ်လို့ ကြောငြာလိုက်တာပါ။

- ◆ နောက်တစ်ကြောင်း a=3;၊ ဒါကတော့ initializing or assigningလုပ်လိုက်တာပါ။ 3 ကို variable a ရဲ့ တန်ဖိုးဖြစ်ပါတယ်လို့ သတ်မှတ်လိုက်တာပါ။ အရှင်းဆုံးပြောရရင် a လို့ပြောလိုက်တိုင်း 3 ကို ပြောခြင်း ဖြစ်ပါတယ်။ အိုကေ ...ဒီအထိနားလည်ပြီနော်။
- ◆ တတိယကြောင်းမှာ Declaration နဲ့ Initializing ကို တစ်ကြောင်းတည်းမှာပဲ ရေးထားတာကို သတိထားမိလားဗျ။ int b=4; ဟုတ်ပြီနော်။ b ဟာ integer data ဖြစ်တယ်လို့ကြေငြာတဲ့အပိုင်းရယ်၊ b တန်ဖိုးဟာ 4 ဖြစ်ပါတယ်လို့ initializing လုပ်တာရယ်တကြောင်း ထဲ မှာပဲနော်။ဒါကို နားလည်ပြီနော်။ ခုဆိုရင် program မှာ တန်ဖိုးတွေလည်း ထည့်ပြီးပြီ။ variable name တွေလည်း ကြေငြာပြီးပြီ။ဒါဆိုရင် variable တွေမှာ သိမ်းထားတဲ့ တန်ဖိုးတွေကို ပေါင်းတော့မယ်။ a+b ဆိုပြီးပေါ့။
- ◆ နောက်ဆုံး ကြောင်းမှာ ရေးထားတဲ့ statement အတိုင်းပဲပေါ့။ printf() function ကိုသုံးပြီး a+b ကို output ထုတ်လိုက်တယ်။ a နဲ့ b မှာ သိမ်းထားတဲ့ တန်ဖိုးတွေရဲ့ ပေါင်းလဒ်က integer data type ဖြစ်တဲ့အတွက် %d ကို သုံးပြီး output ထုတ်လိုက်ခြင်းဖြစ်ပါတယ်။ ဒီ program ကို ရှင်းမယ်လို့ထင်ပါတယ်ခင်ဗျ။ ပေါင်းနုတ်၊ မြှောက်၊ စား ဒီအတိုင်းပါပဲ။program လေးရေးပေးမယ်နော်။

```

1 #include<stdio.h>
2 main()
3 {
4     int a=6;
5     int b=4;
6     printf("Sum of a and b=%d\n",a+b);
7     printf("Subtraction of a and b=%d\n",a-b);
8     printf("Multiplication of a and b=%d\n",a*b);
9     printf("Division of a and b=%d",a/b);
10 }
```

```

Sum of a and b=10
Subtraction of a and b=2
Multiplication of a and b=24
Division of a and b=1
-----
Process exited with return value 21
```

ဒီ program ကို ရှင်းမပြော့ပါဘူး။မိမိဘာသာ line by line trace ကြည့်ပါ။ \n လေး သုံးထားတာလေး ကိုတော့ရှင်းပြပါမယ်။ \n က Escape Sequence ပါ။Escape Sequence တွေအကြောင်းကိုတော့ နောက် lesson မှာရှင်းပြပါမယ်။ခု ဒီ programသုံးထားတဲ့ \n ကိုပဲရှင်းပြပါမယ်။ \n ကို သုံးတာက new line (နောက်တစ်ကြောင်း) ဆင်းစေလိုတဲ့အတွက်ပါ။ Output မှာကြည့်ပါ။ a နဲ့ b ရဲ့ ပေါင်း၊နုတ်၊မြှောက်၊စား ကို တကြောင်းစီပြထားတာ ကိုတွေ့တယ်ဟုတ်။ဒါက \n သုံးပြီးရေးထားတဲ့ အတွက်ကြောင့်ပါ။ နားလည်တယ်နော်။ \n မပါဘဲရေးမယ်ဆိုရင် ဘလို output ထွက်မလဲ။အောက်က အတိုင်းထွက်မှာပါ။

Sum of a and b=10 Subtraction of a and b=2 Multiplication of a and b=24 Division of a and b=1

ကဲ ဒီလိုဆိုရင် ကြည့်လို့ကောင်းမလားဗျ။မကောင်းဘူးနော် ။ဒါကြောင့် \n ကို သုံးထားခြင်း ဖြစ်ပါတယ်။ အိုကေ..

3. Operators and Expressions

1. Introduction

ဒီအခန်းမှာတော့ Operators တွေနဲ့ပတ်သတ်ပြီး ရှင်းပြပါမယ်။Operator တွေကို ဘာကြောင့် သုံးတာလဲ။ Operators တွေ programမှာ ဘယ်လိုအလုပ်လုပ်တာလဲဆိုတာကို ရှင်းပြမှာဖြစ်ပါတယ်။ ခု Operators သုံးရတဲ့အကြောင်းကိုပြောပြပါမယ်။ ကျွန်တော်တို့ ကိန်းဂဏန်းတွေ တစ်ခုနဲ့ တစ်ခုပေါင်းမယ်၊ မြှောက်မယ်၊ လုပ်ချင်တဲ့အခါမျိုးမှာ operators တွေကို သုံးပါတယ်။ရှေ့ခန်းမှာ a နဲ့ b ပေါင်းနှုတ်၊မြှောက်၊ စား လုပ်တာကို ရှင်းပြပြီးနော်။ အဲလိုမျိုး operation တွေလုပ်ဆောင်ဖို့အတွက် operators တွေကို သုံးခြင်းဖြစ်ပါတယ်။ C program မှာ operation တွေအမျိုးမျိုးလုပ်ဖို့အတွက် Operators တွေအများ ကြီး ရှိပါတယ်။ Operators တွေ ပေါကြွယ်ဝ တယ်လို့တောင် ပြောစမှတ်ပြုရတယ်ဗျ(ဟဲ့..). ခု Operators အမျိုးအစားတွေကို Table လေးနဲ့ ပြပေးပါမယ်။

Type of Operators	Symbolic Representation
Arithmetic operators	+, -, *, / and %
Relational operators	>, <, ==, >=, <= and !=
Logical operators	&&, and !
Increment and decrement operator	++ and --
Assignment operators	=
Bitwise operators	&, , ^, >>, << and ~
Comma operators	,
Conditional operators	? :

ဒီ operators တွေထဲက အဓိကလိုအပ်တဲ့ Operators ကိုရှင်းပြပါမယ်။ တစ်ခုတော့ ပြောမယ်ဗျ။ ခုကျွန်တော် ရေးတဲ့ စာအုပ်ကို ဖတ်ပြီး၊လေ့လာပြီးရင် software /application programတွေရေးတတ်သွားမှာမဟုတ်ဘူး နော်။ဒီစာအုပ်ကိုရေးတဲ့ရည်ရွယ်ချက်က programming flow ကို သိစေချင်လို့။ programming sense ရသွား စေဖို့အတွက်ဖြစ်ပါတယ်။ software /application programတွေရေးချင်တယ်ဆိုရင်တော့ ခု C language ကနေ programming sense ကို ရအောင်ယူသွားပါ။ရသွားအောင်ထိလည်း ရှင်းပြပေးပါမယ်။ C language နဲ့ software program တွေရေးဖို့ဆိုတာ အရမ်းကို ခက်ခဲလွန်းပါတယ်။ဒါကြောင့် C language ကနေ sense ကို ရပြီဆိုရင် ဒီထက် ပိုလွယ်ကူတဲ့ Language တွေဘက်ကို ပြောင်းလဲဖို့အကြံပြုချင်ပါတယ်။ C အခြေခံကို ပိုင်ပြီ

ဆိုရင်တော့ ဘယ် language မဆို သင့်အတွက် မခက်ခဲနိုင်တော့ပါဘူး။ ဒါကြောင့် C program မှာအဓိကထား ပြီးလိုအပ်တဲ့ Operators 3 မျိုးကိုပြောပြပါမယ်။ အခြား operators တွေက အရေးမကြီးလို့တော့ မဟုတ်ဘူနော်။ programတွေရေးတဲ့အချိန်မှာ ထပ်ရှင်းပြပေးပါမယ်။ ပထမတော့ ဒီ ဥပမာကိုပဲ အရင်ဆုံးနားလည်အောင် ရှင်းပြ ပေးပါမယ်။

- 1.Arithmetic operators
- 2.Relational operators
- 3.Logical operators

1. Arithmetic operators

Arithmetic operators မှာ နှစ်မျိုးရှိပါတယ်။ အဲ့နှစ်မျိုးကတော့.....

- 1) Binary Operator နဲ့
- 2) Unary Operator ဖြစ်ပါတယ်။ တစ်ခုချင်းစီခွဲပြီး ရှင်းပြပါမယ်။

1) Binary Operators

ဒီ Operator ကိုတော့ Computer Language အတော်များများမှာ သုံးကြပါတယ်ခင်ဗျ။ ဒီ arithmetic operators တွေကို ဘယ်လိုနေရာတွေမှာ သုံးလဲဆိုတော့ ကိန်းသေတန်ဖိုး နှစ်ခုကြားက သင်္ချာဆိုင်ရာတွက်ချက်ချင်းတွေ လုပ်ဆောင်ဖို့အတွက်ပါ။ အရှင်းဆုံးပြောရရင် ကိန်းနှစ်ခု ပေါင်းချင်တယ်ဆိုရင် Addition Operator ကို သုံးမယ်ပေါ့ဗျာ။ ဒါကြောင့် သင်္ချာဆိုင်ရာ တွက်ချက်မှု တွေလုပ်ဆောင်ဖို့အတွက် သုံးတာပါလို့ပြောခြင်းပါ။ Arithmetic operator လေးတွေကို Table လေးနဲ့ပြပေးပါမယ်။

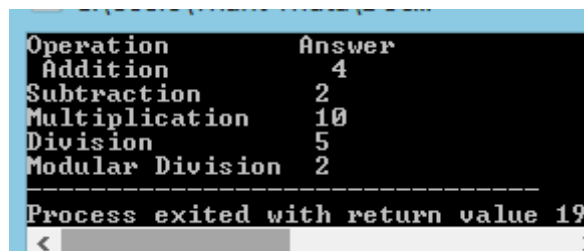
Arithmetic Operators	Operator Explanation	Example
+	Addition	2+2=4
-	Subtraction	5-3=2
*	Multiplication	2*5=10
/	Division	10/2 =5
%	Modular division	11%3=2(Remainder 2)

Table မှာပြထားတာကို နားလည်မယ်လို့ထင်ပါတယ်ခင်ဗျ။ ပိုရှင်းသွားအောင် Program လေးရေး ကြည့် ရအောင်။

```

1  # include<stdio.h>
2  main()
3  {
4  printf("Operation      Answer\n ");
5  printf("Addition        %d\n",2+2);
6  printf("Subtraction      %d\n",5-3);
7  printf("Multiplication    %d\n",2*5);
8  printf("Division          %d\n",10/2);
9  printf("Modular Division  %d",11%3);
0  }

```



```

Operation      Answer
Addition        4
Subtraction      2
Multiplication   10
Division         5
Modular Division 2
Process exited with return value 19

```

- ◆ ဒီ program မှာဆိုရင် Table ထဲမှာရေးထားတဲ့ Arithmetic operator တွေကို သုံးပြီး ကိန်းနှစ်ခုကို တွက်ချက် ထားခြင်းဖြစ်ပါတယ်။ ပထမ printf () ကို ဘာကြောင့် ရေးရတယ်ဆိုတာ သိတယ်ဟုတ်။ Program ကို run တဲ့ အခါမှာ ပေါ်တဲ့ run box လေးမှာ Operation Answer လို့ ပေါ်စေချင်တဲ့အတွက် ဖြစ်ပါတယ်။ Printf() က သူ့ရဲ့ double quote နှစ်ခုကြားမှာ ရေးသမျှကို display ပြပေးတဲ့ Output function ဖြစ်လို့ပါ။ သိပြီးသားနော်။
- ◆ ဒုတိယ printf ကတော့ 2 နဲ့ 2 ရဲ့ ပေါင်းလို့ရတဲ့ အဖြေကို ပြစေချင်တဲ့အတွက်ပါ။ ဒါကြောင့် double quote အတွင်းမှာ ရေးထားတဲ့ Addition ကို ပြပေးတယ်။ double quote အတွင်းမှာ ရေးသမျှကို display လုပ်ပေးတယ်ပြောသေး။ %d ကို ကျ ဘာလို့မပြတာလဲ။ မေးစရာဖြစ်နေပီ။ ပြောပြပါမယ်။ %d က Compiler က သိပြီးသားဖြစ်တဲ့ Data type တွေရဲ့ Conversion character (format string) လေ။ 2 နဲ့ 2 ကို ပေါင်းတဲ့အခါမှာ ရလာမယ့် အဖြေ 4 က ကိန်းပြည့်ဖြစ်တဲ့အတွက် integer data type ပါ။ ဒါပေမယ့် ကိန်းနှစ်ခု ပေါင်းလို့ရတဲ့ အဖြေက ဘာဖြစ်မယ်ဆိုတာကို မသိဘူးလေ။ဒါကြောင့် ပေါင်းလို့ရလာမယ့် အဖြေရဲ့ data type ကိုတော့ ကြိုတင်ခန့်မှန်းတတ်ရပါမယ်။ integer နှစ်ခုကို ပေါင်းတာဆိုတော့ ရလာမယ့် တန်ဖိုးကလည်း integer data ပဲဖြစ်မှာပေါ့။ဒါကြောင့် integer data တွေကို Output ထုတ်မယ်ဆိုရင် %d ကို သုံးပြီး ရေးရမယ်။ %d က place order အနေနဲ့ အလုပ်လုပ်ပါတယ်။ program မှာ မြှားပြထားတာကို နားလည်မယ်လို့ထင်ပါ တယ်။ float data type တွေကို ထုတ်ချင်တာဆိုရင် %f ကို double quote ထဲမှာရေးပြီး အဖြေထုတ်မယ်။ ဒီ Syntax လေးကို တော့ သေချာလေးမှတ်ထားပေးပါနော်။ double quote ရေးပြီးရင် comma ခံပေး ရမယ်နော်။ဘာကြောင့် ခံပေးရတာလဲဆိုတော့ Expression နှစ်ခုကို ခြားနား

ဖို့ဖြစ်ပါတယ်။ ဒီ program မှာဆိုရင် double quote ထဲမှာ ရေးထားတဲ့ data တွေကို display ပြဖို့အတွက် ရေးတဲ့ Expression နဲ့ $2+2$ ကိန်းနှစ်ခု ပေါင်းမယ့် expression ကို ခြားဖို့အတွက်သုံးခြင်းဖြစ်ပါတယ်။ C program မှာ ရေးထားတဲ့ Code တွေအားလုံးမှာ အဓိပ္ပာယ်တွေ ရှိ ပါတယ်။ ခု comma ကို ဘာကြောင့်ရေးပေးရတာလဲဆိုတာကို သိပြီနော်။ comma ရဲ့ နောက်မှာတော့ သင်တွက် ချက်လိုတဲ့ expression တစ်ခုရေးပါ။ ဒီအကြောင်းမှာတော့ ပေါင်းချင်တာဖြစ်တဲ့အတွက် Additional operator (+) ကို သုံးပြီး ကိန်းနှစ်ခုကို ပေါင်းတဲ့ expression ကိုရေးထားခြင်းဖြစ်ပါတယ်။နောက်အကြောင်း တွေကလည်း ဒီအတိုင်းပါပဲ။ ဒီလို တွက်ချက်လို့ ရလာတဲ့ အဖြေကိုတော့ double quote ထဲမှာ ရေးထားတဲ့ %d ကပဲ ထုတ်ပေးမှာဖြစ်ပါတယ်။ ပြီးသွားပြီဆိုရင် semicolon(;) ရေးပေးရမယ်။ အလကားရေးတာမဟုတ်ဘူးနော်။ English စာမှာဆို ရင် Sentence တစ်ကြောင်းဆုံး full stop(.) ရေးရသလိုပါပဲ။ C language မှာလည်း Statement တစ်ကြောင်းပြီးသွားပြီဆိုတာ Compiler ကို သိစေချင်တဲ့အတွက် ရေးပေးရခြင်းဖြစ်ပါတယ်။ နားလည်တယ်ဟုတ်။

- ◆ ဒီနေရာမှာ နည်းနည်းရှင်းပြစရာလေးရှိတယ်ဗျ။ တချို့က Division နဲ့ Modular division ကို သိပ်မကွဲကြဘူးဗျ။နှစ်ခုလုံးက စားတာပဲ ။ဒါပေမယ့် ဘာလို့မတူတာလဲ။ပြောပြပါမယ်။ Division ဆိုရင် စားလဒ်ကို အဖြေထုတ်ပေးမယ်။ Modular division ဆိုရင် အကြွင်း ကို အဖြေထုတ်ပေးမယ်။အိုကေ...။ ခု programကို ကြည့်ပါ။ 10 ကို 2 နဲ့ division လုပ်တဲ့အခါမှာ Output ကို စားလဒ် 5 ကို ထုတ်ပေးတယ်။ 11 ကို 3 နဲ့ modular division လုပ်တဲ့အခါမှာ output ကို အကြွင်း 2 ကို ထုတ်ပေးတယ်။ ရှင်းပြီနော်.... Division နဲ့ Modular division ရဲ့ အလုပ်လုပ်ပုံကို.....။

b) Unary Operator

Operator	Description or Action
-	Minus
++	Increment
--	Decrement
&	Address Operator
Size of	Gives the size of variable

Table ထဲက Operator လေးတွေရဲ့ အသုံးပြုပုံကို ရှင်းပြပါမယ်။ Unary Minus (-) ကို ဘယ်နေရာမှာ သုံးလဲ။ တန်ဖိုးတွေရဲ့ လက္ခဏာပြောင်းလဲတဲ့ နေရာမှာသုံးပါတယ်။ ဘယ်လိုပြောင်းလဲ ပေးတာလဲဆိုတာကို ဥပမာပြပါ မယ်။

```
int x=-50;
```

```
int y=-x;
```

ဒီ ဥပမာ ကိုရှင်းပြမယ်နော်။ $x=-50$ ဆိုပြီးတော့ x ရဲ့တန်ဖိုးက -50 ဖြစ်ပါတယ်။ နောက်တစ်ကြောင်းမှာကျတော့ $y=-x$ လို့ပြန်ပြီး assign လေးပေးလိုက်တယ်။ x ထဲမှာ သိမ်းထားတဲ့ -50 ကို y ထဲကို ပြောင်းထည့်ပေးလိုက်တာ ပေါ့ဗျာ။ ဒါပေမယ့် y ရဲ့ တန်ဖိုးက -50

ဖြစ်သွားပြီလားဆိုတော့ မဖြစ်ပါဘူး။ ဘာလို့လဲဆိုတော့ x ရဲ့ ရှေ့မှာ - ကို ထည့်ရေးထားတယ်လေ။ ဒီတော့ $-(-50)$ ဖြစ်သွားပြီလေ။ - x - ဆိုတော့ + ဖြစ်သွားပြီပေါ့။ ဒါကြောင့် y ရဲ့ တန်ဖိုးက 50 ဖြစ်ပါတယ်။ အဲလို မျိုးပြောင်းလဲဖို့ အတွက် Unary minus ကို သုံးခြင်းဖြစ်ပါတယ်။

Increment /Decrement Operators တွေကို C မှာ loop statement တွေနဲ့ အသုံးများပါတယ်။ ဒီ operator ကို သုံးတာက ရည်ရွယ်ချက်ရှိပါတယ်။ ဘာလို့လည်းဆို တော့ ကိန်းဂဏန်းတွေကို တိုးတိုးပြီး တွက်စေချင် တဲ့အခါ၊ တိုးတိုးပြီး လျော့သွားစေချင်တဲ့ အခါမျိုးမှာ ခဏ ခဏ Assign လုပ်နေစရာမလိုရန် အတွက် ဖြစ်ပါတယ်။ $x = x + 1$ လို့ assign လုပ်ထားတာက $x++$ လို့ increment operator ကို သုံးပြီးရေးတာနဲ့ အလုပ်လုပ်တာ အတူတူပဲဖြစ်ပါတယ်။ $x = x - 1$ ဆိုရင် တော့ decrement operator သုံးပြီး $x--$ လို့ရေးတာပေါ့။

Increment နဲ့ Decrement ကို သုံးတဲ့ နေရာမှာ နှစ်မျိုးရှိတယ်ဗျ။ ဒါကတော့..

- 1) Suffix and
- 2) Prefix ဆိုပြီးပေါ့။

1) Suffix

Suffix ဆိုရင် Compiler ကနေ ဒုတိယတစ်ကြိမ် run တဲ့ အခါမှသာ အတိုးအလျှော့ လုပ်ပေးတာပါ။

Suffix အနေနဲ့ လုပ်စေချင်ရင် ရေးပေးရမယ့် ပုံစံက $(x++)$ or $(x--)$ ပါ။ ပိုနားလည်အောင် example လေးနဲ့ပြပေးပါမယ်။

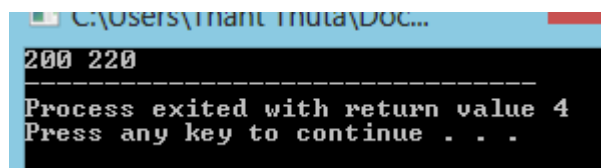
$x=20;$

$y=10;$

$z=x*y++;$ မှာဆိုရင် x ရဲ့ တန်ဖိုးက 20 y တန်ဖိုးကလည်း 10 ပဲရှိဦးမှာပါ။ ဒုတိယတစ်ကြိမ် y ကို run တဲ့ အခါမှ 11 ဆိုပြီး တစ်တိုးမှာပါ။ program လေးရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4      int x=20;
5      int y=10;
6      int z,b;
7      z=x*y++;
8      b=x*y;
9      printf("%d ",z);
10     printf("%d ",b);
11 }
```



C:\Users\mant\ntut\DOC...

```

200 220
-----
Process exited with return value 4
Press any key to continue . . .
```


Program ကို ရှင်းရအောင်။ အရင်ဆုံး ကိန်းပြည့်ဖြစ်တဲ့ 20 နဲ့ 10 ကို သိမ်းဖို့ variable x နဲ့ y ကို int ဆိုပြီး integer data type တွေသိမ်းထားတာပါဆိုပြီး assignment operator (=) ကို သုံးပြီး တန်ဖိုးတွေကို သိမ်းထား လိုက်တာပါ။ နောက် variable z နဲ့ b ကတော့ suffix လုပ်ပြီး မြှောက်တဲ့ အခါမှာ ရလာမယ့် တန်ဖိုးတွေကို သိမ်းဖို့အတွက် ကြိုကြေငြာပေးထားတာပါ။

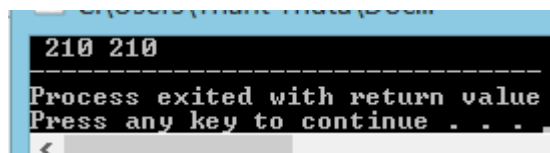
- ◆ ပထမ statement $z=x*y++$; မှာ x တန်ဖိုး 20 နဲ့ y တန်ဖိုး 10 ကို မြှောက်ခိုင်းတာပါ။ အဲ့ချိန်မှာ y ကို တိုးထားပေးမယ့် suffix လုပ်ထားတာဖြစ်တဲ့အတွက် ဒီ တစ်ကြိမ် run တဲ့ အခါမှာ တစ် မတိုးသေးပါဘူး။ ဒါကြောင့် အဲ့ဒီ ကိန်းနှစ်ခုရဲ့ မြှောက်လဒ်က 200 ဖြစ်ပါတယ်။ အဲ့ဒီ 200 ကို z ထဲမှာ သိမ်းထားတာဖြစ်တဲ့အတွက် z ကို output ထုတ်တဲ့အချိန်မှာ 200 ကို ထုတ်ပေးခြင်းဖြစ်ပါတယ်။
- ◆ ဒုတိယကြောင်း မှာတော့ $b=x*y$; လို့ရေးပေးထားပါတယ်။ x တန်ဖိုးကတော့ ဘာမလုပ်ထားတဲ့အတွက် တန်ဖိုးက မပြောင်း လဲပါဘူး 20 ပဲဖြစ်ပါတယ်။ y တန်ဖိုးကျတော့ ပြောင်းသွားပြီ။ ပြောခဲ့ပြီးပြီနော် y ကို နောက်တစ်ကြိမ် run တဲ့ အခါမှာတော့ တစ်တိုးပေးသွားတယ်ဆိုတော့။ ဒီတော့ y တန်ဖိုးက 11 ဖြစ်သွားပြီ။ မြှောက်လိုက်တော့ 220 ရတယ်။ အဲ့ဒီတန်ဖိုးကို b ထဲမှာ သိမ်းထားလိုက်တယ်။ ဒါကြောင့် b ကို output ထုတ်တော့ 220 ကို Monitor မှာ display လုပ်ပေးတယ်။ ကဲ....ဒါဆို suffix ကို နားမလည်တာ မရှိတော့ဘူးနော်။

2) Prefix

Prefix မှာကျတော့ prefix လုပ်ထားတဲ့ variable ကို ပထမတစ်ခါ run တဲ့ ချိန်မှာပဲ အတိုးအလျှော့ လုပ်ပေးပါတယ်။ prefix လုပ်မယ်ဆိုရင် ရေးပေးရမှာက $...(++x)$ or $(--x)$ ဆိုပြီးရေးပေးရမှာပါ။ program လေး ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int a,z,x=10,y=20;
5  z=x*++y;
6  a=x*y;
7  printf(" %d %d",z,a);
8  }
```



```

210 210
Process exited with return value
Press any key to continue
```

Program ကို ရှင်းပါမယ်။ အကုန်လုံးကိုတော့ ရှင်းပြဖို့တော့ လိုမယ်မထင်တော့ဘူးနော်။ ရှေ့မှာ လည်း ရှင်းခဲ့ပြီးပြီဆိုတော့။ အတိုးအလျှော့ လုပ်တာလေးကို ပဲရှင်းပြပါမယ်။ prefix မလုပ်ခင်မှာ x ရဲ့ တန်ဖိုးက 10 , y ရဲ့ တန်ဖိုးက 20 ဖြစ်ပါတယ်။ ဒါပေမယ့် z ရဲ့ တန်ဖိုးကို ရှာတဲ့ အခါမှာ y ကို

prefix လုပ်ထားပါတယ်။ ဒါကြောင့် y တန်ဖိုးက 21 ဖြစ်နေပါပြီ။ ပထမ တစ်ခါ y ကို run တွဲ အခါမှာပဲ တစ်တိုးပေးလိုက်တာပါ။ suffix ဆိုရင်တော့ မတိုးသေးဘူးနော်။ ဒီတော့ x နဲ့ y ကို မြှောက်လိုက်တဲ့ အခါမှာ 210 ကို ရပါတယ်။ ရှင်းတယ်နော်။

- ◆ နောက် တစ်ကြောင်း a ကို တွက်တဲ့အခါမှာ x တန်ဖိုးကတော့ 10 ပါ။ y တန်ဖိုးကတော့ 21 ပါ။ အပေါ်မှာ y တန်ဖိုးကို 20 လို့ကြေငြာထားတာလေ။ $a=x*y$ မှာ y ကို $(++y)$ လို့မရေးဘဲနဲ့ ဘာလို့ 21 ဖြစ်နေရတာလဲ။ သေချာမှတ် ထားပါ။ y က volatile variable ဖြစ်နေလို့ပါ။ volatile variable ကို ရှေ့မှာ ပြောပြခဲ့ သေး တယ်နော်။ ပထမ တစ် ကြောင်းမှာ y ကို 20 လို့ကြေငြာထားလိုက်တယ်။ ဒါပေမယ့် ဒုတိယတစ်ကြောင်းမှာ y ကို prefix လုပ်လိုက်တဲ့ အတွက် y တန်ဖိုးက 21 ဖြစ်သွားပြီ။
- ◆ ဒါကြောင့် y ထဲမှာ 20 ရှိနေတော့ဘဲ 21 ပဲရှိနေပါတယ်။ a ကို တွက်တဲ့အခါ မှာလည်း y က သူ့ထဲမှာ သိမ်းထားတဲ့ 21 ကိုပဲထုတ်ပေးတော့ output တန်ဖိုးက 210 ဖြစ်တာပါ။ နားလည်မယ် လို့ထင်ပါတယ်။
ဒါဆို Suffix ရော Prefix ကို နားလည်မယ်နော်။

c) Size of () and '&' Operator

'&' (Address) Operator ကတော့ computer ရဲ့ memory မှာ variable တွေရဲ့ တန်ဖိုး တွေထိန်းသိမ်းထားတာကို ပြဖို့အတွက်။ သုံးပါတယ်။ program လေးနဲ့ ရှင်းပြပါမယ်။

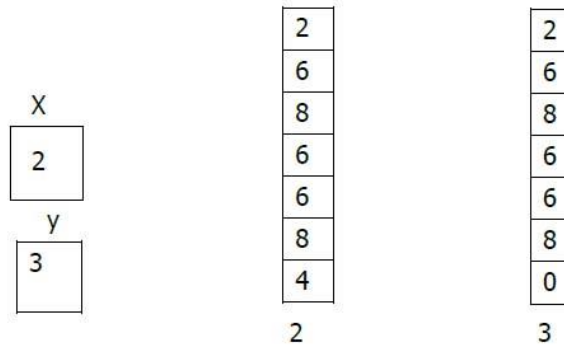
```

1  #include<stdio.h>
2  main()
3  {
4      int x=2;
5      int y =3;
6      printf(" Address of x=%d and y=%d",&x,&y);
7  }
```

```

Address of x=2686684 and y=2686680
Process exited with return value 37
Press any key to continue . . .
```

ကဲ...ဒီ program မှာဆိုရင် x နဲ့ y variable နှစ်ခုမှာ သိမ်းထားတဲ့ တန်ဖိုးနှစ်ခုကို အဖြေမထုတ်ဘဲ အဲ့ဒီတန်ဖိုး တွေရှိတဲ့ Address (နေရာ) ကို ပဲအဖြေထုတ်ပေးပါတယ်။ ပုံလေးနဲ့ပြပေးပါမယ်။



ပုံလေးကို နားလည်လားမသိဘူးဗျ။ x နဲ့ y ထဲမှာ သိမ်းထားတာ က 2 နဲ့ 3 ပါ။ program မှာ x နဲ့ y တန်ဖိုးကို မထုတ်ဘဲ အဲဒီ တန်ဖိုးတွေရှိနေတဲ့ နေရာကို ပြစေချင်တဲ့အတွက် & operator ကို သုံးပြီး display လုပ်ခိုင်းတာ ပါ။&x ဆိုပြီး x ရဲ့ ရှေ့မှာ & ကို ရေးပေးရုံပါပဲ။ & မပါဘူးဆိုရင်တော့ x ထဲမှာ သိမ်းထားတဲ့ 2 ကို display လုပ်မှာ ပါ။ &y လို့ရေးတာလည်း ဒီသဘောတရားပါပဲ။ ဒီ Program ကို ပဲ output ထုတ်တဲ့အခါမှာ & ကို ရှေ့ကနေ မရေးဘူးဆိုရင် ဘယ်လို output ထုတ်မလဲ ကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int x=2;
5  int y =3;
6  printf(" Address of x=%d and y=%d",x,y);
7  }

```

ဒီ program နှစ်ခုကို ယှဉ်တွဲကြည့်ပြီး & operator ကို ဘာကြောင့်သုံးတာလဲ ဆိုတာကို ရှင်းပြီလို့ ထင်ပါတယ်။

Size of () Operator

sizeof() Operator ကို ဘယ်လိုနေရာတွေမှာ သုံးလဲဆိုတာပြောပြပါမယ်။ sizeof() Operator ကို ကျွန်တော်တို့ သုံးမယ့် data ဟာ memory ပေါ်မှာ ဘယ်လောက်နေရာယူလဲဆိုတာကို သိချင်တဲ့ အခါမှာ သုံးနိုင်ပါတယ်။ program လေးရေးပြပါမယ်။ Program တွေ ခဏ ခဏ ရေးပေးတယ်ဆိုတာ programming sense ရစေချင်လို့ပါ။

```

1  #include<stdio.h>
2  main()
3  {
4  int x=2;
5  float y=2;
6  printf("The size of x=%d bytes\n",sizeof(x));
7  printf("The size of y=%d bytes",sizeof(y));
8  }

```

```
The size of x=4 bytes
The size of y=4 bytes
```

Program ကို ရှင်းရအောင်။ Program ရှင်းတာကို အရင်ဆုံး မကြည့်ပါနဲ့။ မိမိဘာသာ program ကို အရင်ဆုံး line by line trace ကြည့်ပါ။ နားမလည်ဘူးဆိုမှ ရှင်းလင်းချက်ကို ကြည့်ပါ။ မရှင်းတာဆိုလို့ size of () သုံးတာ ပဲရှိမယ်လို့ ထင်ပါတယ်။ sizeof () operator ရဲ့ လက်သညးကွင်း () ထဲမှာ ရေးတဲ့ variable ရဲ့ byte တန်ဖိုးကို Output ထုတ်ပေးပါတယ်။ Programming terms နဲ့ ပြောရရင်တော့ sizeof () operator ရဲ့ parenthesis ထဲက argument ရဲ့ byte တန်ဖိုးကို တွက်ထုတ်ပေးတာပါ။ ဒီ program မှာ ဆိုရင် sizeof () ရဲ့ လက်သညးကွင်းထဲက argument x ရဲ့ byte တန်ဖိုးကို တွက်ပေးတာပါ။ variable x ရဲ့ data type က integer data type ဖြစ်တဲ့အတွက် 4 byte လို့ Output ထုတ်ပေးတာပါ။ sizeof(y) လည်း ဒီအတိုင်းပါပဲ။ သူကတော့ float data type ဖြစ်လို့ float data တွေရဲ့ memory နေရာယူတဲ့ byte တန်ဖိုး 4 ဆိုပြီး output ထုတ်ပေးလိုက်တာဖြစ်ပါတယ်။ နားလည်မယ်လို့ထင်ပါတယ်ခင်ဗျ။

2.Relational operators

Relational operators တွေကို သုံးရတဲ့ရည်ရွယ်ချက်က values တွေ တစ်ခုနဲ့ တစ်ခု နှိုင်းယှဉ်ဖို့အတွက် သုံးခြင်းဖြစ်ပါတယ်။ ဒါကြောင့် relational operators လေးတွေရဲ့ အဓိပ္ပာယ်လေးကို အရင်ပြောပြပါမယ်။

- > ကိုတော့ greater than လို့ခေါ်ပါတယ်။
- < ကိုတော့ less than လို့ခေါ်ပါတယ်။
- == ကိုတော့ equal to လို့ခေါ်ပါတယ်။
- >= ကိုတော့ greater than or equal လို့ခေါ်ပါတယ်။
- <= ကိုတော့ less than or equal လို့ခေါ်ပါတယ်။
- != ကိုတော့ not equal လို့ခေါ်ပါတယ်။

ဒီ operator တွေရဲ့ အဓိပ္ပာယ်ကို သိပြီးဆိုတော့ သူတို့ရဲ့ အလုပ်လုပ်ပုံလေးကို ကြည့်ရအောင်။

Expression	Result	Value
4>3	True	1
4<3	False	0
5>=5	True	1
4<=3	False	0
3!=4	True	1
5==4	False	0

ဒီပုံလေးကိုရှင်းပြမယ်နော်။ 4 ဟာ 3 ထက်ကြီးတယ်ဆိုတော့ မှန်တာပေါ့။အဲ့တော့ result က true ဖြစ်ပါတယ်။ value အနေနဲ့ ပြောရင်တော့ 1 ပေါ့။ Less than operator ကလည်း

ဒီအတိုင်းပါပဲ။ 4 ဟာ 3 ထက်ကြီးတယ် ဆိုတာ မှားတယ်လေ။ ဒါကြောင့် result က false ဖြစ်ပြီး value က 0 ဖြစ်ပါတယ်။ နောက်တကြောင်းကတော့ 5 က 5 ထက်ကြီးတယ်၊ ဒါမှမဟုတ် 5 နဲ့ ညီတယ် ဆိုတော့ true ဖြစ်တယ်။ နောက်တကြောင်းကလည်း ဒီအတိုင်းပါပဲ။ တန်ဖိုးနှစ်ခုကို နှိုင်းယှဉ်တဲ့အခါမှာ မှန်ရင် true ဖြစ်မယ်။ result က 1 ဖြစ်မယ်။ အဲလောက်တော့ သိမယ်ထင်ပါတယ်။ program လေးတပုဒ်လောက်ရေးပြမယ်နော်။

```

1  #include<stdio.h>
2  main()
3  {
4      printf("Expression    Value\n");
5      printf("4>3           %d\n",4>3);
6      printf("4<3           %d\n",4<3);
7      printf("5>=5          %d\n",5>=5);
8      printf("4<=3          %d\n",4<=3);
9      printf("3!=4           %d\n",4>3);
10     printf("5==4           %d",4>3);
11 }

```

```

Expression    Value
4>3            1
4<3            0
5>=5           1
4<=3           0
3!=4           1
5==4           1
-----
Process exited with return

```

ဒီ program လေးကို ကြည့်ပြီး relational operator ရဲ့ သဘောကို နားလည်မယ်လို့ ထင်ပါတယ်ဗျ။ ဒါကို ပိုနားလည်အောင် Conditional operator လေးသုံးပြီး ရေးကြည့်ရအောင်။ Sorry...ဗျာ။ Conditional operator ကဘာလဲဆိုတာကိုပြောမပြရသေးဘူး။ Conditional ဆိုတာ ဘာလဲဗျ။ အခြေအနေ မဟုတ်ဘူးလား ဗျ။ ဟုတ်ပါတယ်ဗျ။ ဒီ operator က အခြေအနေတခု နဲ့ တခုကို နှိုင်းယှဉ်ပြီး result ထုတ်ပေးတဲ့သူပါ။ ကဲ program လေးနဲ့ ယှဉ်ပြီးကြည့်ရအောင်ဗျာ။

```

1  include <stdio.h>
2  main()
3  {
4      printf("Result=%d\n",4>3? 1:0);
5      printf("Result=%d\n",4<3? 1:0);
6      printf("Result=%d\n",5>=5? 1:0);
7      printf("Result=%d\n",4<=3? 1:0);
8      printf("Result=%d\n",3!=4? 1:0);
9      printf("Result=%d",5==4? 1:0);
10 }

```

```

Result=1
Result=0
Result=1
Result=0
Result=1
Result=0
-----
Process exited with

```

- ◆ Program ကတော့ပြီးပြီဗျ။ ရှင်းမယ်နော်။ Conditional operator ရဲ့ Symbol ကိုတော့ သိတယ်ဟုတ်။ (? :) ဒီလိုမျိုးပါ။ အဓိပ္ပာယ်ကတော့ဗျာ၊ရှင်းရှင်းလေးပါ။ Question mark ရဲ့ ရှေ့က နှိုင်းယှဉ်ချက်ကို မှန်လား၊ မှားလားစစ်မယ်။မှန်ရင် Question mark နောက်က တန်ဖိုးကို အဖြေထုတ်ပေးမယ်။မှားတယ်ဆိုရင်တော့ is to (:) ရဲ့ နောက်က တန်ဖိုးကို အဖြေထုတ်ပေးမယ်။အိုကေတယ်နော်။ဒါကို နားလည်ပြီဆိုရင် program ကို ကြည့်ရအောင်။
4>3?1:0; ဆိုပြီးရေးထားတယ်။ Question mark ရဲ့ရှေ့မှာ 4 ဟာ 3 ထက် ကြီးလားမကြီးဘူးလားဆိုတဲ့ အခြေအနေကိုစစ်တယ်။ ကြီးတယ်ဆိုတော့ Question mark ရဲ့နောက်က 1 ကို အဖြေထုတ်ပေး တယ်။ Output မှာ Result = 1 ဆိုပြီး ရေးထားတာတွေ့တယ်ဟုတ်။
- ◆ ဒုတိယတစ်ကြောင်းမှာတော့ 4 ဟာ 3 ထက် ငယ်တယ်ဆိုတော့ ဒီအခြေအနေက မှားနေတယ်။ဒါကြောင့် is to (:) ရဲ့နောက်က တန်ဖိုး 0 ကို အဖြေထုတ် ပေးတာဖြစ်ပါတယ်။ အောက်က statement တွေဟာလည်း ဒီအတိုင်းပဲဖြစ်ပါတယ်။ မိမိဘာသာပဲ trace ကြည့်ပါနော်။ဒါဆိုရင် ဒီ program လေးရဲ့ flow ကို သိပြီနော်။

3. Logical Operators

Logical operators ကတော့ expression နှစ်ခုကို နှိုင်းယှဉ်ပြီးမှ တန်ဖိုးတွေကိုထုတ်စေချင်တဲ့ အခါမျိုးမှာ သုံးပါတယ်။ program နဲ့ပြရင်တော့ နားလည်မယ်လို့ထင်ပါတယ်။အရင်ဆုံး logical operator တွေရဲ့ symbol ဧ တွနဲ့ အလုပ်လုပ်ပုံကို table လေးနဲ့ပြပေးပါမယ်။

Operator	Description or Action	Example	Return value
&&	Logical AND	5>3 && 5<10	1
	Logical OR	8>5 8<2	1
!	Logical NOT	8!=8	0

- ◆ အိုကေ Table လေးကို ရှင်းမယ်။သေချာလေး မှတ်ထားပေးပါ။ ပထမ operator && ဆိုတာက logical AND ဖြစ်ပါတယ်။သူက expression နှစ်ခုကို ဘယ်လိုနှိုင်းယှဉ်ပြီး value ကိုထုတ်ပေးလဲဆိုတာကို ပြောပြပါမယ်။ && ရဲ့ ရှေ့နဲ့ နောက် expression နှစ်ခုလုံး မှန်မှ value 1 ကို ထုတ်ပေးမှာ။ တစ်ခုက မှန်ပြီး တစ်ခုက မှားတယ်ဆို ရင် တော့ return value က 0 ပါ။ Example လေးရေးထားတာကို တချက်ကြည့်ပါ။ && ရဲ့ရှေ့က expression မှာ 5 ဟာ 3 ထက် ကြီးတယ်လို့ပြောတယ်။ နောက် က expression မှာကျတော့ 5 က 10 ထက် ငယ်တယ်ဆိုတော့ ဒီ expression နှစ်ခုလုံးက မှန်နေတယ်။ ဒါကြောင့် return value ဟာ 1 ဖြစ်ပါတယ်။ ရှင်းပြီနော်။

- ◆ || operator ကတော့ Logical OR ဖြစ်ပါတယ်။ OR ရဲ့ အဓိပ္ပာယ်အတိုင်းပါပဲ တခု မဟုတ်တစ်ခုပေါ့။ || ရဲ့ ရှေ့နဲ့ နောက် expression နှစ်ခုထဲက တစ်ခုမှန်တယ်ဆိုရင် ကို return value 1 ကိုအဖြေထုတ်ပေးပါတယ်။ အိုကေတယ်ဟုတ်။
- ◆ ! ဆိုတဲ့ operator ကတော့ logical NOT ဖြစ်ပါတယ်။ Example မှာကြည့်ပါ။ 8 က 8 နဲ့ မညီပါဘူးဆိုတော့ ဒီ expression က မှားနေတာပေါ့။ ဒါကြောင့် return value 0 ကို အဖြေထုတ်ပေးခြင်းဖြစ် ပါတယ်။ ပိုနားလည်သွားအောင် program လေးတပုဒ်လောက် ရေးပြမယ်နော်။

```

1  #include <stdio.h>
2  main()
3  {
4      printf(" \nCondition      : Return Values\n");
5      printf("\n5>3 && 5<10    :%d\n",5>3 && 5<10);
5      printf("\n 8>5 || 8<2    :%d\n", 8>5 || 8<2 );
7      printf("\n!(8==8)       :%d",!(8==8) );
3  }
```

```

Condition      : Return Values
5>3 && 5<10    :1
 8>5 || 8<2    :1
!(8==8)       :0
-----
Process exited with return value
Press any key to continue . . .
```

- ◆ ကဲ program လေးကတော့ ရေးပြီးပြီဆိုတော့ output ထုတ်ပုံလေးကို အတူတူ trace ကြည့်ကြရအောင်။ပထမ ဆုံး printf () function မှာ Condition: Return Values ဆိုပြီး ပေါ်အောင် double quote နှစ်ခုကြားမှာ ရေးလိုက်ခြင်းဖြစ်တယ်။ သင်ခန်းစာတွေ ပြန်နွေးပေးနေတာနော်။ ဟုတ်ပြီ နော်။ ပြီးတော့ နောက် statement ကို နောက်တစ်ကြောင်းမှာပဲ ပြချင်တဲ့အတွက် \n ထည့်ပြီး နောက်တကြောင်း ဆင်းပြီး ဘဲပြတော့လို့ ခိုင်းလိုက်တာ ပါ။
- ◆ နောက် printf ကတော့ Expression နှစ်ခု ကို နှိုင်းယှဉ်ပြီး return value ပြဖို့အတွက် ပါ။ return type က 1 နဲ့ 0 ဖြစ်တဲ့အတွက် integer data type ဖြစ်ပါတယ်။ ဒါကြောင့် %d ကို သုံးထားခြင်းပါ။ ခု ပုံစံက printf() function ရဲ့ output display လုပ်တဲ့ ဒုတိယ Syntax တခုနော် ။ရှေ့ပိုင်းမှာလည်း ပြောခဲ့ပြီးပါပြီ။ comma(,) ရဲ့ နောက်မှာ output value ကို သိမ်းထားတဲ့ variable ၊ ဒါမှမဟုတ် output တန်ဖိုးရအောင် တွက်ချက်တဲ့ statement ကို ရေးပေးရမယ်ဆိုတာ သိပြီးသားနော်။ခု program မှာ ဆိုရင်တော့ expression နှစ်ခုကို နှိုင်းယှဉ် ပြီး ရလာမယ့် return value ကို ဖော်ပြတဲ့ statement ကို ရေးထားတာပါ။ ဒီ programကို တော့ ရှင်းပြစရာ လိုမယ်မထင်တော့တဲ့အတွက် မရှင်းပြတော့ဘူးနော်။

4. Priority of Operators and their Clubbing

ဒီ အခန်းမှာဘာကို ပြောပြမလဲဆိုတော့..... Operator တွေရဲ့ Priority နဲ့ Clubbing လုပ်ပုံ ကို ရှင်းပြမှာဖြစ်ပါတယ်။ Operator တွေက အလုပ်လုပ်တဲ့အခါမှာ အစီအစဉ်တကျ လုပ်တာဖြစ်ပါတယ်။ Priority ဆိုတာက Statement တစ်ကြောင်းမှာ Operator ၅ ခုလောက်သုံးတယ်ဆိုပါစို့။ အဲလိုအခါမျိုးမှာ ဘယ် operator တွေက priority ဘယ်နှစ်ခုမြောက် အလုပ်လုပ်လဲဆိုတာကို ပြောတာပါ။ ရှင်းရှင်းပြောရရင် စီနီယာ နဲ့ ဂျူနီယာ system လိုဖြစ်ပါတယ်။ Clubbing ဆိုတာကတော့ Statement တစ်ကြောင်းတည်းမှာ Operators ၅ခုလောက် ကို ပေါင်း၊ နှုတ်၊ မြှောက်၊ စား ဘယ်ညာ ရေးထားတယ်ဆိုပါစို့။ အဲလိုမျိုးဆိုရင် ဘယ်ကနေညာကို အလုပ်လုပ်မှာလား၊ ညာက နေဘယ်ကို အလုပ်လုပ်မှာလားဆိုတာကို ပြောချင်တာပါ။ပိုရှင်းသွားအောင် ဘယ် operator တွေဆိုရင် ဘယ်ကနေ ညာကို လုပ်တယ်၊ညာကနေ ဘယ်ကို အလုပ်လုပ်တယ်ဆိုတာ Table လေးနဲ့ပြပေးပါမယ်။


Operators	Operation	Clubbing	Priority
() [] -> .	Function call Array expression or square bracket Structure Operator Structure Operator	Left to Right	1 st
+ - ++ -- ! ~ * & Size of type	Unary plus Unary minus Increment Decrement Not Operator Ones complement Pointer Operator Address Operator Size of object Type cast	Right to Left	2 nd
* / %	Multiplication Division Modular division	Left to Right	3 rd
+ -	Addition Subtraction	Left to Right	4 th
<< >>	Left shift Right shift	Left to Right	5 th
<	Less than	Left to Right	6 th

<=	Less than or equal to		
>	Greater than		
>=	Greater than or equal to		
==	Equality	Left to Right	7 th
!=	Inequality		
&	Bitwise AND	Left to Right	8 th
^	Bitwise XOR	Left to Right	9 th
	Bitwise OR	Left to Right	10 th
&&	Logical AND	Left to Right	11 th
	Logical OR	Left to Right	12 th
? :	Conditional operator	Right to Left	13 th
=,*,-=, &=,+, =,^=, =, <<=,>>=	Assignment operator	Right to Left	14 th
,	Comma operator	Left to Right	15 th

ကဲ...ခုဆို table မှာပြထားတဲ့အတိုင်း ဘယ် operator တွေက ဘယ်နှစ်ခုမြောက်၊ ဘယ်ဘက်ကနေ ဘယ်ဘက် ကို လုပ်တာလဲဆိုတာကို သိပြီနော်။ ပိုရှင်းသွားအောင် Example လေးနဲ့ပြပေးပါမယ်။

Example

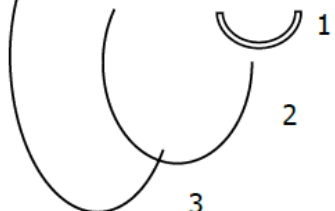
x= 5 * 4 + 8 / 2;



ဒီ example လေးမှာ ဆိုရင် Table ထဲက အတိုင်းပါပဲ။ multiplication (*) က အရင် စ အလုပ်လုပ်မယ်။ ပြီးရင် ဒုတိယ Division(/) လုပ်မယ်။ နောက်ဆုံးမှ Addition Operator(+) အလုပ်လုပ်မယ်။ သင်္ချာတွက်သလိုပါပဲနော်။ အိုကေ။ နောက်တစ်ခု ထပ်ကြည့်ရအောင်။

Example.

(8 / (2 * (2 * 2)));



ဒီ Example လေးမှာဆိုရင် priority အစဉ်လိုက် အလုပ်လုပ်တာလေးကြည့်ရအောင်။ function call () ,multiplication(*) ,Division (/) ဆိုပြီး Operators ၃ ခုပါတယ်။ ဒါဆိုရင် Table ထဲက အတိုင်းပဲ function call() operator ကို အရင်ဆုံးစပြီး အလုပ်လုပ်မယ်။ ဒါပေမယ့် function

call က ဥရုတောင် ပါတာဆိုတော့ ဘယ် တစ်ခုစပြီးလုပ်မလဲ။ သင်္ချာတွက်သလိုပါပဲ။ အတွင်းဆုံးက () ကို အရင်ရှင်းပါမယ်။ အတွင်းဆုံး () က စလုပ်မယ်ဆိုတော့ သူ့ထဲမှာရှိတဲ့ $2 * 2$ ကို အရင် စပြီး မြှောက်မယ်။ပြီးမှ ဒုတိယ () ကွင်းကိုလုပ်မယ်။ နောက်ဆုံးမှ Division operator ကို အလုပ်လုပ်မယ်။ နားလည်ပြီနော်။ ဘာကြောင့် ဒါကို ရှင်းပြနေတာလဲ ဆိုတော့ Program တပုဒ်ရေးတော့မယ်ဆိုရင် မိမိလိုချင်တဲ့ Output ကို မှန်ကန်အောင်ရဖို့ပါ။ ပိုနားလည်အောင် program လေးတပုဒ်ရေးပြပါမယ်။

```

1  #include<stdio.h>
2  main()
3  {
4      int a=2;
5      int b=3;
6      int c=4;
7      printf("The average of three integer numbers= %d", (a+b+c)/3);
8  }
```

```

The average of three integer numbers= 3
-----
Process exited with return value 39
```

ဒီ programမှာဆိုရင် ကိန်း ၃လုံးရဲ့ average ကို လိုချင်တာဖြစ်တဲ့ အတွက်။ အဲ့ဒီကိန်း ၃ လုံး သိမ်းထားတဲ့ variable a,b,c ကိုပေါင်းပြီး၊ ၃ နဲ့စားပေးလိုက်တယ်။ average ရှာတာကို သိတယ်ဟုတ်။ average လိုချင်တဲ့ ကိန်းအားလုံးပေါင်းပြီး၊ ကိန်းအရေအတွက်နဲ့ စားပေးရတယ်။ အိုကေ။ ဒီ program ရဲ့ အလုပ်လုပ်တာကို တော့ တစ်ကြောင်းချင်းစီ မရှင်းတော့ဘူးနော်။ ရှေ့ပိုင်းမှာလည်း ရှင်းပြပြီး ပြီဆိုတော့။ $(a+b+c)/3$ ဆိုတဲ့ Statement ရေးတာကို ပြောပြချင်တာပါ။ ရှေ့မှာပြောခဲ့တဲ့ operator တွေရဲ့ စီနီယာလိုက် အလုပ်လုပ်တာကို မသိဘူးဆိုရင် ဒီ Statement ကို ရေးတတ်မှာမဟုတ်ပါဘူး။ ဘာလို့လဲဆိုတော့ ကိန်း ၃ လုံးပေါင်းပြီးရင် ၃ နဲ့စားရမယ်။ဒါကိုတော့ သိတယ်။အဲ့တော့ အလွယ်ပဲ $a+b+c/3$ ဆိုပြီးရေးလိုက်မယ်။ ကဲဒါဆိုရင် program က ဘယ်လို အဖြေထုတ်ပေးမလဲ။ program လေးရေးကြည့်ရအောင်။

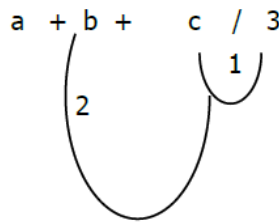
```

1  #include<stdio.h>
2  main()
3  {
4      int a=2;
5      int b=3;
6      int c=4;
7      printf("The average of three integer numbers= %d", a+b+c/3);
8  }
```

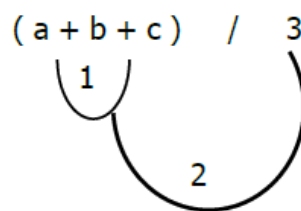
```

The average of three integer numbers= 6
-----
Process exited with return value 39
```

Program က 6 ကို output ထုတ်ပေးပါမယ်။ ဒါ Computer ရဲ့ အလုပ်လုပ်တာ မှားယွင်းမှုမဟုတ်ပါဘူး။ operator တွေရဲ့ priority အစဉ်လိုက် အလုပ်လုပ်တာကို နားမလည်လို့ ဖြစ်ပါတယ်။ခု ရေးပေးလိုက်တဲ့ Syntax ကို ဘယ်လို အလုပ်လုပ်သွားလဲ ပြောပြမယ်။



ခု ပုံမှာပြထားတဲ့အတိုင်း အစဉ်လိုက် လုပ်သွားတာဖြစ်ပါတယ်။ အမှန်က a,b,c သုံးခု ပေါင်းမယ်။ပြီးမှ 3 နဲ့ စားချင်တာလေ။ဒါပေမယ့် Compiler က လက်မခံပါဘူး။ သူ့သတ်မှတ်ထားတဲ့ operator တွေရဲ့ priority အတိုင်းပဲတွက်ထုတ်ပေးပါတယ်။ ဒါကြောင့် Division (/) operator ကို အရင်စပြီးလုပ်ပါတယ်။ variable c ကို 3 နဲ့စား ။ c ထဲမှာ သိမ်းထားတဲ့ တန်ဖိုးက 4 ဆိုတော့ စားလိုက်တဲ့အခါ 1.3333 ရပါတယ်။ဒါပေမယ့် integer data type လို့ပြောထားတဲ့အတွက် integer တန်ဖိုးကို ထုတ်ပေးပါတယ်။ 1 ဆိုပြီးပေါ့။ ပြီးမှ a , b နဲ့ 1 ကို ပေါင်းလိုက်တယ်။ ဒါကြောင့် 6 ကို output ထုတ်ပေးတာဖြစ်ပါတယ်။ ကဲ.ဒါဆိုရင် ကျွန်တော်တို့ လိုချင် တဲ့ average အမှန်ကို မရတော့ဘူးပေါ့။ ဒါကြောင့် ပထမ program မှာ ရေးထားတဲ့ Syntax အတိုင်းရေးရပါမယ်။ (a+b+c)/3 ဆိုပြီးပေါ့။ အဲလိုရေးလိုက်တော့ ဘာထူးသွားလဲဆိုတာကို ပြမယ်နော်။



ဒီ Syntax အတိုင်းဆိုရင် Operator 3 ခုပါပါမယ်။ function call (), Addition(+),Division(/) ဆိုပြီးပေါ့။ ဒီ operator 3 ခုမှာ priority စီနီယာအကျ ဆုံးက function call ဖြစ်တဲ့အတွက် သူ့ထဲမှာ ရှိတဲ့ Addition ပိုင်းကို အရင်လုပ်တယ်။အဲ့တော့ variables a,b and c ထဲမှာ သိမ်းထားတဲ့ 2,3,4 ကိုပေါင်းလိုက်တယ် ။ 9 ရမယ်။ ပြီးမှ Division လုပ်မယ်။ 9 ကို 3 နဲ့စားတော့ 3 ပေါ့။ဒါကြောင့် Average တန်ဖိုး 3 ဆိုပြီး output ထုတ်ပေးတယ်။ ကဲ.ဒါဆို Operators တွေရဲ့ Priority လိုက် အလုပ်လုပ်တာဟာ ဘာကြောင့်အရေးကြီးလဲဆိုတာကို နားလည်ပြီဟုတ်။ တော်တော်လေး ရှင်းလိုက်ရတယ်နော်။ Dummies တွေအတွက် ရည်ရွယ်ထားတာ ဆိုတော့ အဲလောက်တော့ ရှင်းပြရမှာပေါ့နော်။ Dummies (နလပိန်းတုန်း) ဆိုပြီး အားငယ် မသွားပါနဲ့ဗျ။ ဒီစာအုပ်ဖတ်ပြီး ရင်တော့ Real Programmer ဖြစ်လာမှာပေါ့နော်(ဟဲ.....)။

4. Input and Output in C

1. Introduction

ဒီအခန်းမှာတော့ c program မှာရှိတဲ့ input /output function အကြောင်းတွေကို ပြောပြပေးပါမယ်။ Programming Language တိုင်းမှာ Input/Output function တွေရှိပါတယ်။ ဒီ

function တွေရဲ့ အလုပ်လုပ်ပုံ ကိုတော့ သေချာနားလည် သဘောပေါက် ထားရပါမယ်။ ဘာလို့လဲဆိုတော့ User ဆီကနေ Input တွေတောင်း မယ်။ data တွေကို user မြင်အောင် ပြပေးဖို့အတွက် ဒီ function တွေကို သုံးရမှာမို့ပါ။

C မှာ I/O function အမျိုးအစားနှစ်မျိုးရှိပါတယ်။ I/O လုပ်မယ့် data type ကို မူတည်ပြီးတော့ပေါ့။ ဒါကတော့

- 1) Formatted functions and
- 2) Unformatted functions ဖြစ်ပါတယ်။

1) Formatted functions

ခု Formatted functions အကြောင်းကို ရှင်းပြပါမယ်။ ဒီ functions ကတော့ ဘယ် data type အမျိုးအစားမဆို Input/Output လုပ်လို့ရပါတယ်။ Formatted functions တွေက User ဆီကနေ data တွေကို တောင်းမယ်။ data တွေကို Output ထုတ်ပေးမယ်။ User ဆီကနေ Input တောင်းတဲ့ အခါမှာ conversion symbol တွေသုံးရပါမယ်။ မိမိတောင်းမယ့် data ရဲ့ data type အမျိုးအစားကို ကြည့်ပြီး သုံးပေးရ မှာပါ။ ဆိုလိုတာကတော့ဗျာ integer data type ကို Input value တောင်းမယ်ဆိုရင် integer data type ရဲ့ conversion symbol %d ကို သုံးရမယ်။ ရှေ့ပိုင်းမှာ printf () function သုံးပြီး data တွေကို conversion symbol သုံးပြီး Output ထုတ်ပေးတာကို ရှင်းပြပြီးနော်။ ဒီသဘောတရားပါပဲ။

3) Unformatted functions

Unformatted functions တွေကိုလည်း data တွေ Input/Output လုပ်ဖို့အတွက် သုံးပါတယ်။ ဒါပေမယ့် character data type တွေ၊ string type တောင်းဖို့။ Output ထုတ်ဖို့အတွက် အသုံးများပါတယ်။ ဒီ function တွေကို သုံးမယ် ဆိုရင်တော့ data တွေ I/o လုပ်ဖို့ conversion symbol တွေမလိုပါဘူး။

Formatted and Unformatted functions တွေမှာ ပါတဲ့ functionတွေကို table လေးဆွဲပြီး ပြပေးပါမယ်။

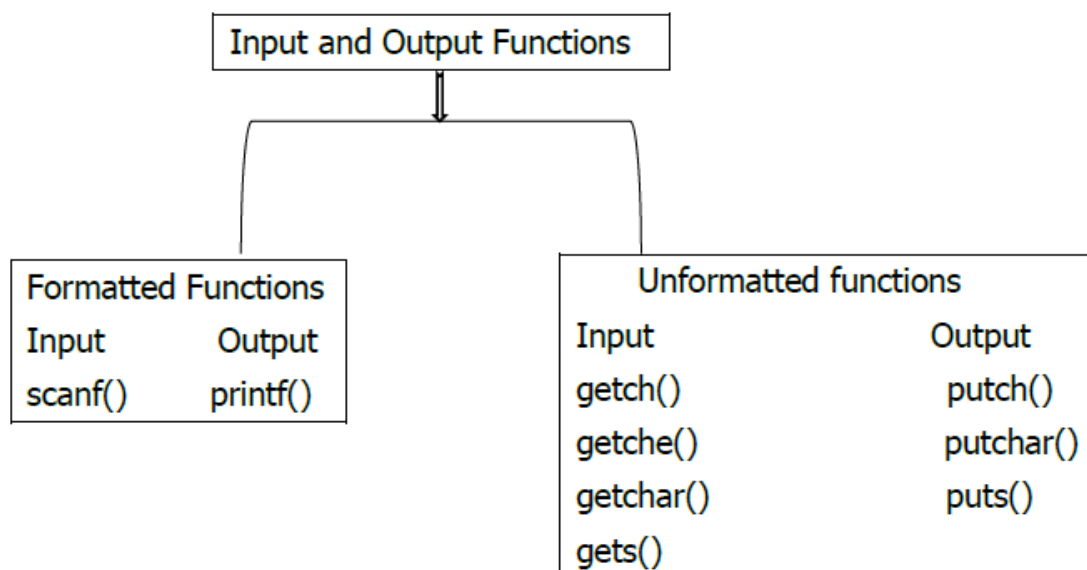


Table လေးတော့ဆွဲပြီးပြီ။ အဲဒီ function တွေရဲ့ အလုပ်လုပ်ပုံကို Program တွေနဲ့ တွဲပြီးရှင်းပြမယ်နော်။

1) Use of printf()

printf() ရဲ့ အလုပ်လုပ်ပုံကို ရှင်းပြပေးခဲ့ပြီးပြီနော်။ မှတ်မိအောင် ထပ်ရှင်းပြပေးပါမယ်။ printf() function က data values အားလုံးကို display လုပ်ပေးပါတယ်။အဲလို Output ထုတ်ဖို့အတွက် conversion symbol ရယ် variable names တွေရယ်လိုပါတယ်။conversion symbol နဲ့ variable names ဟာ format တူညီမျှ ရှိရပါမယ်။ example လေးရေးရအောင်။

```

1 #include<stdio.h>
2 main()
3 {
4     int x=2;
5     float y=2.2;
6     char z='C';
7     printf("%d    %f    %c",x,y,z);
8 }
```

```

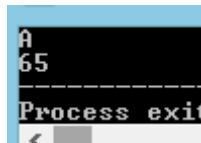
2    2.200000    C
-----
Process exited with ret
Press any key to contin
```

- ◆ ဒီ example program ဆိုရင် x,y,z ထဲမှာ သိမ်းထားတဲ့ data တွေကို printf () function သုံးပြီး output ထုတ် ပေးတာဖြစ်ပါတယ်။ x က integer data ဖြစ်လို့ conversion symbol %d ကို သုံးပေးရပါတယ်။ ပြီးတော့ printf () function ထဲမှာ output ထုတ်ချင်တဲ့ data ရဲ့ variable ကို ရေးပေးရပါမယ်။ ခု program မှာ ရေးပေး ထားသလိုပါ။ y က ကျတော့ float data type ဖြစ်လို့ conversion symbol %f ကို သုံးပေးရပါတယ်။ z က character data type ဖြစ်လို့ conversion symbol %c ကို သုံးပေးရပါတယ်။ ဘယ် data type ရဲ့ conversion symbol က ဘာဖြစ်တယ်ဆိုတာ ရှေ့ပိုင်းမှာ ရေးပေးထားပါတယ်။မမှတ်မိရင် ပြန်ကြည့်ပါ။ z ကို ကြေငြာတဲ့ အခါမှာ C က character အမျိုးအစားဖြစ်လို့ single quote(' ') ထဲမှာ ရေးပြီး initializing လုပ် ပေး ရပါမယ်။ program မှာ ရေးထားသလိုပါ။ printf() ကို ရင်းနှီးပြီးသားဖြစ်တဲ့ အတွက် သိပ်ရှင်းစရာ လိုမယ် မထင် ပါဘူးခင်ဗျ။ ဒီစာအုပ်ကို ဖတ်မှတ်သူတွေ အတွက် C ကို လေ့လာပြီး ASCII value ကို မသိဘူးဆိုတာမဖြစ်ရလေအောင် ASCII value တွေကို Table ဆွဲပြီးပြပေးပါမယ်။

ASCII value	Symbol	ASCII value	Symbol	ASCII value	Symbol	ASCII value	Symbol
0	NUL	20	DC4	40	(60	<
1	SOH	21	NAK	41)	61	=
2	STX	22	SYN	42	*	62	>
3	ETX	23	ETB	43	+	63	?
4	EOT	24	CAN	44	`	64	@
5	ENQ	25	EM	45	-	65	A
6	ACK	26	SUB	46	.	66	B
7	BEL	27	ESC	47	/	67	C
8	BS	28	FS	48	0	68	D
9	HT	29	GS	49	1	69	E
10	LF	30	RS	50	2	70	F
11	VT	31	US	51	3	71	G
12	FF	32	BLANK	52	4	72	H
13	CR	33	!	53	5	73	I
14	SO	34	"	54	6	74	J
15	SI	35	#	55	7	75	K
16	DLE	36	\$	56	8	76	L
17	DC1	37	%	57	9	77	M
18	DC2	38	&	58	:	78	N
19	DC3	39	,	59	;	79	O
80	P	92	\	104	H	116	t
81	Q	93]	105	i	117	u
82	R	94	^	106	j	118	v
83	S	95	_	107	k	119	w
84	T	96	`	108	l	120	x
85	U	97	a	109	m	121	y
86	V	98	b	110	n	123	z
87	W	99	c	111	o	124	{
88	X	100	d	112	p	125	
89	Y	101	e	113	q	126	}
90	Z	102	f	114	r	127	~
91	[103	g	115	s		

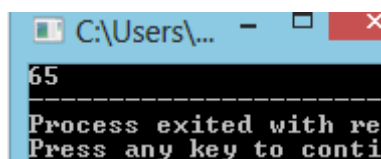
Table လေးတော့ ရေးပေးပြီးပြီနော်။ program လေးရေးပြီး ASCII value တွေနဲ့ Symbol တွေ display လုပ်တာကို ရေးကြည့်ရအောင်။

```
# include<stdio.h>
main()
{
    int y =65;
    printf("%c\n",y);
    printf("%d",y);
}
```



- ◆ ဒီ program မှာ y ကို 65 လို့ assign လုပ်ပေးထားပါတယ်။ y ရဲ့ တန်ဖိုး ကို Screen မှာ display လုပ်ဖို့အတွက် printf() function ကိုသုံးပါမယ်။ printf() function က data တွေကို Output ထုတ်ပေးတဲ့ function ကိုး။
- ◆ ပထမ statement မှာ y ကို conversion symbol %c ကိုသုံးပြီး Output ထုတ်လိုက်ပါတယ်။ %c ဆိုတာ character data တွေကို output ထုတ်တဲ့ ခါမှာသုံးတဲ့ conversion symbol ဖြစ်ပါတယ်။ဒါကြောင့် Output value က 65 ကို display မပြဘဲ 65 ရဲ့ ASCII Symbol A ကို display ပြပေးခြင်းဖြစ်ပါတယ်။
- ◆ ဒုတိယ Statement မှာ y ကို %d ကို သုံးပြီး output ထုတ်တော့ integer တန်ဖိုး 65 ကိုပဲ display ပြပေးပါတယ်။နားလည်မယ်နော်။.....
ဒီတစ်ခါ ASCII symbol ကို assign လုပ်ပေးပြီး ASCII value ထုတ်ပေးတာကို ရေးကြည့်ရအောင်။

```
# include<stdio.h>
main()
{
    char y ='A';
    printf("%d",y);
}
```



ဒီ program မှာ A က Character data ဖြစ်တဲ့အတွက် single quote ထဲမှာ ရေးပြီး ကြေငြာပေးရပါတယ်။ Data type ကိုတော့ Character data ဖြစ်တဲ့အတွက် char လို့ကြေငြာပေးရပါတယ်။ ဒါပေမယ့် character data type y ကို output ထုတ်တဲ့အခါမှာ %c နဲ့ မထုတ်ပေးဘဲ %d နဲ့ ထုတ်ထားပါတယ်။ ဒါကြောင့် character A ရဲ့ ASCII value 65 ကို Output ထုတ်ပေးခြင်းဖြစ်ပါတယ်။ ကျန်တဲ့ Symbol တွေနဲ့ value တွေကိုတော့ မိမိ ဘာသာစမ်းသပ်ကြည့်ပါ။ ASCII ရဲ့ Value ကို output ထုတ်ချင်ရင် %d ကိုသုံးရပါမယ်။ Symbol ကို ထုတ်ချင် ရင်တော့ %c ကိုသုံးပြီး ရေးပေးရပါမယ်။ အိုကေ.....!

b) scanf() function

scanf() function က တော့ Input function ပါ။ User ဆီကနေ value တွေကို တောင်းမယ်။ ပြီးရင် မိမိသိမ်းထား ချင်တဲ့ variable ထဲမှာ သိမ်းထားပေးမယ်။ အဲလို သိမ်းထားဖို့အတွက် input value ရဲ့ data type ကို ကြည့်ပြီး conversion symbol ကိုသုံးရပါမယ်။ ပြီးရင် မိမိသိမ်းထားမယ့် variable name ရဲ့ ရှေ့မှာ Address operator (&) ကို ရေးပေးရပါမယ်။ ဘာလို့လဲဆိုတော့ input data တွေကို variable name ရဲ့ Address မှာ သိမ်းထားတာ ဖြစ်လို့ပါ။ ရှေ့မှာ variable တွေ data သိမ်းဆည်းထားပုံကို ပြပေးပြီးပြီနော်။ ပိုနားလည်အောင် program လေးတခုဒီလောက် ရေးပြပါမယ်။ scanf() function ကတော့ user က 3 လို့ ရိုက်ရင် 3 ကို လက်ခံပေးပြီး user မြင်အောင် ပြပေးပါတယ်။

```
#include<stdio.h>
main()
{
    int a;
    printf("Enter a number");
    scanf("%d",&a);
    printf(" The number is %d",a);
}
```

```
Enter a number3
The number is 3
Process exited with retu
```

- ◆ ဒီ program မှာဆိုရင် ပထမဆုံး variable name ကို a လို့ပေးမယ်။ ပြီးတော့ a ဟာ integer data type တွေ သိမ်းထားပါမယ်လို့ Declare လုပ်လိုက်ပါတယ်။
- ◆ ပထမ printf() ကတော့ User ဆီကနေ input တောင်းဖို့ အတွက် User ကို ဘာလုပ်ရမယ်ဆိုတာကို သိအောင်လို့ ရေးပေးတာဖြစ်ပါတယ်။ Enter a number ဆိုတဲ့ စာသားကို monitor မှာ ပေါ်စေချင်တဲ့ အတွက်ပါ။ ဒါမှ User က number တစ်ခု ရိုက်ပေးရမယ်ဆိုတာကို သိမှာ ပေါ့။
- ◆ ဒုတိယ statement ကတော့ user ရိုက်လိုက်တဲ့ တန်ဖိုးကို လက်ခံပေးဖို့အတွက်ပါ။ ဘယ်ထဲမှာ လက်ခံ မလဲဆိုတော့ variable a ထဲမှာ ။ဒါကြောင့် &a လို့ရေးထားတာပါ။ & ကို

သုံးရတာကတော့ input data ကို သိမ်းမယ့် memory location ကို ပြဖို့အတွက်ပါ။ ခု program အရ User က 3 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆို input data 3 ကို variable a ထဲမှာ သိမ်းထားလိုက်ပြီ။

- ◆ ဒါကြောင့် တတိယ statement မှာ a ကို output ထုတ်လိုက်တဲ့အခါမှာ User ရိုက်လိုက်တဲ့ value 3 ကို output ထုတ်ပေးတာဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။ နောက်ထပ် float data type ကို တောင်းပြီး display ပြခိုင်းတာကို ကြည့်ရအောင်။

```

1 #include<stdio.h>
2 main()
3 {
4     float a;
5     printf("Enter a value ");
6     scanf("%f",&a);
7     printf(" The value you enter is %f",a);
8 }
```

```

Enter a value 2.5
The value you enter is 2.500000
Process exited with return value 3
```

ဒီ program မှာ ဆိုရင် foat data type ကို တောင်းပြီး display ပြခိုင်းမှာ ဖြစ်တဲ့အတွက် a ကို float လို့ ကြေငြာပေးထားတာ ဖြစ်ပါတယ်။ ဒါကြောင့် scanf() function နဲ့ data လက်ခံ သိမ်းဆည်းတဲ့အခါမှာ လည်း %f ကို သုံးထားခြင်းဖြစ်ပါတယ်။ printf() နဲ့ output display ပြခိုင်းတဲ့အခါမှာလည်း %f ကိုပဲ သုံးပြီး ရေးပေးရမှာ ဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။ ဒီ program မှာ user က 2.5 လို့ input value ရိုက်ထည့်ပေးမယ့် output display ပြတဲ့ အခါမှာ 2.500000 လို့ပြပေးပါတယ်။ အဲ့ဒီလိုမျိုး 2.5 ဆိုရင် 2.5 လို့ ပြစေချင်ရင် limit ကန့်သတ်ပြီး ရေးပေးရပါမယ်။

```

1 #include<stdio.h>
2 main()
3 {
4     float a;
5     printf("Enter a value ");
6     scanf("%f",&a);
7     printf(" The value you enter is %0.1f",a);
8 }
```

```

Enter a value 2.5
The value you enter is 2.5
```

ခုလိုမျိုး %0.1f လို့ရေးပေးလိုက်ရုံပါပဲ။ ဒသမ(.) ရှေ့ limint ကတော့ အပြည့်ကိန်းတန်ဖိုး ကန့်သတ်ဖို့အတွက် ဖြစ်ပါတယ်။နောက်က .1 (point 1) ဆိုတာကတော့ ဒသမ နောက်က တန်ဖိုး ကန့်သတ်ဖို့ဖြစ်ပါတယ်။

printf() function နဲ့ scanf () function တွေသုံးတဲ့အခါမှာ ဘယ် data type ဆိုရင် ဘယ် conversion symbol ကို သုံးရလဲဆိုတာကို Table နဲ့ပြပေးပါမယ်။

	Data type	Conversion symbol
integer	short integer	%d or %i
	short unsigned	%u
	long signed	%ld
	long unsigned	%lu
	unsigned hexadecimal	%x
	unsigned octal	%o
real	float	%f or %g
	double	%lf
character	signed character	%c
	unsigned character	%c
	string	%s

ခု Table ကို ဆွဲပြီးပြီဆိုတော့ ဘယ် data type ဆိုရင် ဘယ် conversion symbol သုံးရမယ်ဆိုတာ ကို သိပြီနော်။ ဒီ Table ထဲက နားမလည်ဘူးလို့ထင်တဲ့ data type တွေကို ပြောပြပါမယ်။ short integer ဆိုတာက integer ပါပဲ။ ဆိုလိုတာက int a=2;

short int b=2; လို့ရေးတာဟာ အတူတူပါပဲ။ short ကို ထည့်မရေးဘဲ int ကိုဘဲ ကြေငြာတယ်ဆိုရင် default အားဖြင့် short data type လို့ပဲ Compiler က သိမှာဖြစ်ပါတယ်။ နားလည်တယ် ဟုတ်။သူ့ရဲ့ range ... -32,768 to 32,768 ထိဖြစ်ပါတယ်။

short unsigned ဆိုရင်တော့ 0 to 65535 ဖြစ်ပါတယ်။ signed ဆိုရင် အပေါင်းဘက်နဲ့ အနှုတ်ဘက် တဝက်စီ ခွဲပြီးနေရာယူမယ်။ unsigned ဆိုရင်တော့ အပေါင်း ဘက်ကိုပဲအပြည့်ယူတယ်။ အောက်က data type တွေလည်းဒီ အတိုင်းပါပဲ။long, unsigned long အတူတူပါပဲ။ ဟုတ်ပြီ integer data type တွေတူရဲ့သားနဲ့ ဘာလို့ data type ပေးတဲ့နေရာမှာ အမျိုးမျိုး ရှိနေရ တာလဲ။ long တွေ short တွေကွဲသွားတာလဲ၊ ရှင်းပြပါမယ်။ Dr.Aung WinHtutt ပြောပြထားတဲ့ ဥပမာလေးနဲ့ ပြောပြပါမယ်။ သူ့ဝန်နဲ့ သူ့ဆိုင် ကိုက်အောင် ထားချင်တဲ့အခါမှာ short တွေ long တွေ ခွဲသုံးရတာပါ။ ဥပမာ လက်တဆုပ်စာလောက်ပဲ ရှိတဲ့ သဲတွေကို အိတ်ထဲထည့်မယ်ပေါ့။ ဒါဆိုရင် အိတ်သေးသေးလေးပဲလိုမယ်ပေါ့။ အဲလို အခါမျိုးဆိုရင် short လို့ ကြေငြာပေးရမှာပေါ့။ ငှပေါင် လောက်ရှိမယ့် သဲတွေထည့်ဖို့ ဆိုရင် အိတ်ကြီး ကြီးလိုမယ်။အဲလို အခါမျိုးမှာ long လို့ကြေငြာမယ်။ ဒီနေရာမှာ short လို့ကြေငြာမယ်ဆိုရင်တော့ memory overfull ဖြစ်နေမှာပါ။ ခုနက short

လို့ကြေငြာတဲ့နေရာမှာ long လို့ကြေငြာလည်းရတယ်နော်။ ဒါပေမယ့် ဝန်က လက်တစ်ဆုပ်စာ ဆိုဒ်က လိုတာ ထက်ကြီးနေတယ်။ ဒါဆိုရင် memory waste ဖြစ်နေမှာပါ။ ဒါကြောင့် သူ့ဝန်နဲ့ သူ့ဆိုဒ် ကိုက်အောင် သိမ်း ချင်တဲ့အခါမှာ သုံးတာကွဲပြားတာဖြစ်ပါတယ်။ real ဆိုလည်း ဒီအတိုင်းပါပဲ။ character ဆိုလည်းဒီအတိုင်းပါပဲ။ data type နဲ့ conversion symbol အတွဲကိုတော့ သေချာလေးဂရုစိုက် ပေးပါ။.....

ခု C language မှာသုံးတဲ့ Escape sequence တွေကိုလေ့လာရအောင်။

Escape Sequence	Use
\n	New line
\b	Backspace
\f	Form feed
\'	Single quote
\\	Backslash
\0	Null
\t	Horizontal tab
\r	Carriage return
\a	Alert
\"	Double quote
\V	Vertical tab
?	Question mark

Escape Sequence တွေနဲ့ သူတို့ရဲ့ အသုံးပြုပုံကို တွေ့ပြီဟုတ်။ program မှာ ဒါတွေကို ဘယ်လိုသုံးလဲ ကြည့်ရအောင်။

```
#include<stdio.h>
main()
{
    int a=2,b=3,c=4;
    printf("The value of a=%d\t b=%d and\n value of c=%d",a,b,c);
}
```

```
The value of a=2          b=3 and
value of c=4
-----
Process exited with return value
Press any key to continue . . .
```

ဒီ program မှာဆိုရင် Escape sequence \t နဲ့ \n ကိုသုံးထားတာကို လေ့လာရအောင်။ The value of a=2 ကို output ထုတ်ပေးပြီး tab ခြားသွားတာကို တွေ့တယ်ဟုတ်။ ဘာလို့ tab ခုန်သွားတာလဲဆိုတော့ \t ကို ရေးလိုက်တဲ့အတွက် ဖြစ်ပါတယ်။နားလည်တယ်ဟုတ်။ output မှာ value of c=4 နောက်တစ်ကြောင်းဆင်း သွားတာကို တွေ့တယ်ဟုတ်။ ဒါက \n ကို သုံးပြီး new line ဆင်းလိုက်တဲ့အတွက်ဖြစ်ပါတယ်။Escape sequence တွေရဲ့ အလုပ်လုပ်ပုံကို နားလည်ပြီလို့ ထင်ပါတယ်။ အခြား sequence တွေကို တော့ မိမိဘာသာ စမ်းသပ်ကြည့်ပါ။ အဆင်ပြေပါစေ။.....

ဒီတစ်ခါ 2 ရဲ့ 3 ထပ်က ဘယ်လောက်လဲ ဆိုတာကို အဖြေထုတ်ဖို့ program လေး ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  #include<math.h>
3
4  main()
5  {
6      float x=2,y=3;
7      printf("The power of 2 is %0.0f",pow(x,y));
8  }

```

```

The power of 2 is 8
Process exited with r

```

- ◆ ဒီ program မှာ ထူးခြားတာဘာတွေလဲဗျ။ pow(argument1,argument2) function သုံးထားတာကို တွေ့တယ်ဟုတ်။ဒီ function ကို ဒီအတိုင်းခေါ်သုံးလိုက်တာလား။ မဟုတ်ပါဘူး။ ဒီ function ရဲ့ library က <stdio.h> ထဲမှာ မရှိဘူးလေ။ ဒါကြောင့် <math.h> ကို ကြေငြာပေးရတာပါ။ pow() function လို့မျိုး Arithmetic function တွေကို သုံးချင်တယ်ဆိုရင် <math.h> ဆိုတဲ့ header file ကို ကြေငြာပေးရပါတယ်ဗျ။ ဒါဆို math.h ကြေငြာပေးရတာကို နားလည်မယ်ထင်ပါတယ်။

- ◆ x တန်ဖိုး 2 , y တန်ဖိုး 3 ကို pow(x,y) လို့ရေး လိုက်ရုံနဲ့ 8 ဘယ်လိုဖြစ်သွားတာလဲ။ Compiler က x^y လို့ နားလည်လိုက်တာဖြစ်ပါတယ်။ ဒါကြောင့် 2^3 ဆိုပြီး 8 ကို Output ထုတ်ပေးတာဖြစ်ပါတယ်။ pow(argument1,argument2) function မှာ argument1 မှာ base ထားမယ့် variable ကို ရေးပေးပြီး argument2 မှာ exponent လုပ်မယ့် variable ကို ရေးပေးရမယ်။ ဒီ program မှာ နားမလည်တာ မရှိတော့ ဘူးထင်ပါတယ်။.....။

ဒီတစ်ခါ Relational operator ရယ် Logical Operator ရယ် Conditional operator သုံးပြီး input value တောင်းမယ်။ input value က မိမိတောင်းတဲ့ value ဟုတ်မဟုတ် စစ်မယ်။ဟုတ်ရင် True လို့ output ထုတ်ပေးမယ်။မှားရင် False လို့ output ထုတ်တာကို ရေးကြည့်ရအောင်။

```
#include<stdio.h>
main()
{
    int a,b;
    printf("Enter vale of a and b");
    scanf(" %d %d",&a,&b);
    (a<3 && b<3? printf("True"):printf("False"));
}
```

```
Enter vale of a and b2 1
True
-----
Process exited with return value
Press any key to continue . . .
```

(or)

```
Enter vale of a and b4 1
False
-----
Process exited with return val
Press any key to continue . . .
```

- ◆ ဒီ program မှာ variable a နဲ့ b ကို input value တွေသိမ်းဖို့အတွက် ကြေငြာထားမယ်။ ပြီးရင် user ကို ဘာလုပ်ရမယ်ဆိုတာသိအောင် Enter value of a and b ဆိုတာလေး user ကို value တွေ ရေးပေးပါလို့ တောင်းတဲ့ စသားလေးပေါ်အောင် ရေးပေးပါတယ်။ ပြီးရင် variable a နဲ့ b ထဲမှာ သိမ်းထားမယ်။ ဥပမာ user က a ထဲကို 2 သိမ်းပြီး b ထဲကို 1 သိမ်းလိုက်တယ်ဆိုပါစို့။ နောက်ဆုံး Statement မှာ a နဲ့ b ရဲ့ တန်ဖိုးနှစ်ခုလုံး က 3 ထက်ငယ်လားဆိုပြီး Relational operator (a<3,b<3)ကိုသုံးပြီးစစ်ပါမယ်။ နှစ်ခုလုံးက 3 ထက်ငယ်မှ True လို့ output ထုတ်ချင်တာဖြစ်တဲ့အတွက် Logical operator ကို သုံးပြီး (a<3 && b<3) လို့ရေးပြီး နှစ် ခုလုံးငယ်လားဆိုပြီး စစ်လိုက်တာပါ။ ဒါကို Conditional operator သုံးပြီးရေးလိုက်တော့ (a<3 && b<3) ဆိုတဲ့ condition မှန်ရင် Question mark နောက်က Expression ကို display ပြပေးမယ်။ မှားရင် is to(:) နောက်က Expression ကို display ပြပေးမယ်။ user က 2 နဲ့ 1 ကို ထည့်လိုက်တာဖြစ်တဲ့အတွက် Condition စစ်တော့ နှစ်ခုလုံးက 3 ထက်ငယ်နေတာ ဖြစ်တယ်။ ဒါကြောင့် (?) နောက်က expression printf("True") ကို display လုပ်နိုင်တယ်။ ဒီတော့ printf က True ကို Output ထုတ်ပေးလိုက်ပါတယ်။ နားလည်တယ်နော်။ ဒု တိယတစ်ခါရေးထားတာက User က variable a ထဲကို 4 သိမ်းပြီး b ထဲကို 1 သိမ်းထားလိုက်တယ်ဆိုပါစို့။ ဒါကို Condition စစ်တော့ b ကတော့မှန်တယ်။ 3 ထက်ငယ်တာဆိုတော့။ a ကတော့မှားနေတယ်။ 4 ဆိုတော့ 3 ထက်ကြီးတယ်လေ။ && operator က နှစ်ခုလုံး မှန်မှ True ကို display ပြပေးမှာလေ။ ဒီတော့ တစ်ခုမှားနေ တာဖြစ်တဲ့အတွက် is to(:) နောက်က expression ကို output ထုတ်ပေးပါတယ်။ဒါကြောင့် False ကို output ထုတ်လိုက်ခြင်းဖြစ်ပါတယ်။ ဒီ program ကို နားလည်မယ်လို့ ထင်ပါတယ်ခင်ဗျ။
- ဒီတစ်ခါ user ဆီက တန်ဖိုးနှစ်ခု တောင်းမယ် ပြီးရင် အဲ့ဒီတန်ဖိုးနှစ်ခုကို ပေါင်းမယ်။ ဒါကို Addition operator သုံးပြီးရေးရအောင်။

```
#include<stdio.h>
main()
{
    int a,b,sum;
    printf("Enter value of a and b");
    scanf("%d , %d",&a,&b);
    sum=a+b;
    printf("The addition of a and b =%d",sum);
}
```

```
Enter value of a and b2,4
The addition of a and b =6
Process exited with return val
< >
```

ဒီ program မှာ အားလုံးကိုတော့ မရှင်းတော့ဘူးနော်။ရှေ့မှာလည်း ရှင်းထားခဲ့ပြီးပြီဆိုတော့။ User က input value ကို 2 နဲ့ 4 ကို ရိုက်လိုက်တယ်ဆိုပါစို့။ a နဲ့ b ထဲမှာ 2 နဲ့ 4 ကို သိမ်းထားလိုက်ပြီ။ Input တောင်းတဲ့ နေရာမှာ %d,%d လို့ရှေးလိုက်တာကို သတိထားမိလိုက်ဗျ။ ဘာလို့လဲဆိုတော့ user ကို a နဲ့ b တန်ဖိုးကို comma ခံပြီးရေးစေချင်တဲ့အတွက်ပါ။ comma မရေးမချင်း မိမိရိုက်လိုက်တဲ့ တန်ဖိုးကို a ထဲမှာ သိမ်းပေးမှာပါ။ comma ရေးပြီးရင်တော့ comma နောက်က တန်ဖိုးကို b ထဲမှာ သိမ်းထားပေးမှာပါ။ comma ရှေ့က တန်ဖိုးက a အတွက် comma နောက်က b အတွက် ဆိုပြီးကွဲပြားအောင် အတွက်ပါ။ အိုကေ.....။ sum=a+b; ဆိုတဲ့ statement က တော့ additional operator (+) ကို သုံးပြီး a နဲ့ b ကို ပေါင်းလိုက်တာဖြစ်ပါတယ်။ ပေါင်းလို့ ရတဲ့ တန်ဖိုးကို sum ထဲကို ပြောင်းပြီး ထည့်လိုက်တာပါ။ ဒီတော့ sum ရဲ့ တန်ဖိုးက 6 ဖြစ်သွားပြီပေါ့။ ဒါကို monitor မှာ display ပြဖို့အတွက် printf() function ကို သုံးပြီး Output ထုတ်လိုက်ခြင်းဖြစ်ပါတယ်။ ဒါကြောင့်

The addition of a and b =6 ဆိုပြီး Screen မှာမြင်ရတာဖြစ်ပါတယ်။ နားလည်သွားပြီလို့ထင်ပါတယ်။.....။

ဒီ program နဲ့ အလားတူပဲ Additonal operator နေရာမှာ Multiplication operator သုံးပြီးရေးရအောင်။

```
#include<stdio.h>
main()
{
    int a,b,multi;
    printf("Enter value of a and b");
    scanf("%d , %d",&a,&b);
    multi=a*b;
    printf("The multiplication of a and b =%d",multi);
}
```

```

Enter value of a and b2,4
The multiplication of a and b =8
-----
Process exited with return value 32
Press any key to continue . . .

```

ဒီ program ကိုတော့ မရှင်းတော့ဘူးနော်။ မိမိဘာသာ trace ကြည့်ပါ။ Operator အမျိုးမျိုးသုံးပြီး စမ်းသပ်ကြည့် ပါ။ပိုပြီး နားလည်သဘောပေါက်လာပါလိမ့်မယ်။ ဒီတစ်ခါတော့ Character တန်ဖိုးကို ရိုက်ထည့်ပြီး Character တန်ဖိုးကို output ထုတ်တာကို ရေးကြည့် ရအောင်။

```

#include<stdio.h>
main()
{
char a;
printf("Enter a character");
scanf("%c",&a);
printf("The character you enter =%c",a);
}

```

```

Enter a characterM
The character you enter =M
-----
Process exited with return val

```

ဒီ program ကိုလည်း အသေးစိတ်ရှင်းမနေတော့ဘူးနော်။ အားလုံးက အတူတူပဲလေ။ a ကို character data type လို့ကြေငြာပြီးတော့ %c ကို သုံးပြီး input data ကို a ထဲမှာ သိမ်းထားလိုက်တယ်။ပြီးတော့ printf () functionနဲ့ output ထုတ်ပေးလိုက်တယ်။ မိမိဘာသာ trace ကြည့်ပါ။ ဒါမှ programming သဘောသဘာဝကို နားလည်မှာပါ။ line by line စဉ်းစားပြီးကြည့်ပါ။ အဆင်ပြေပါစေချင်.....။

Programming နဲ့ ပတ်သက်ပြီး sence လေးရှိနေလောက်ပြီလို့ ထင်ပါတယ်။.....ဒီတစ်ခါ variable သုံးခု မကြေငြာဘဲ နှစ်ခုထဲ သုံးပြီး တန်ဖိုးနှစ်ခု အလဲအလှယ်(swap) လုပ်တာကို စမ်းသပ်ကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int a=7,b=4;
5  printf("  A=%d and B=%d\n",a,b);
6  a=a+b;
7  b=a-b;
8  a=a-b;
9  printf("Now A=%d and B=%d",a,b);
10 }

```

```
A=7 and B=4
Now A=4 and B=7
```

- ◆ ဒီ program မှာ A နဲ့ B တန်ဖိုးတွေကို ဘယ်လို လဲလိုက်တယ်ဆိုတာကို ကြည့်ရအောင်။ int a=7,b=4 ဆိုပြီး a နဲ့ b ထဲကို တန်ဖိုးတွေ သိမ်းထားလိုက်တယ်။
- ◆ ပြီးတော့ printf(" A=%d and B=%d",a,b); ဆိုပြီး a နဲ့ b တန်ဖိုးတွေကို output ထုတ်လိုက်တယ်။ ဒီထဲနားလည်မယ်နော်။
- ◆ ပထမ statement မှာ a=a+b; ဆိုပြီး a နဲ့ b တန်ဖိုးနှစ်ခုပေါင်းလဒ်ကို a ထဲမှာ သိမ်းထားလိုက်တယ်။ဒီ တော့ a တန်ဖိုး 7 နဲ့ b တန်ဖိုး 4 ကိုပေါင်းလိုက်တယ်။ 11 ရပါတယ်။ အဲ့ဒီတန်ဖိုးကို a ထဲမှာ သိမ်းလိုက်တယ်။ ဒီအချိန်မှာ a ထဲမှာ ပထမသိမ်းထားတဲ့ တန်ဖိုး 7 ရှိမနေ တော့ပါဘူး။ 11 ပဲရှိနေတော့ တာဖြစ်ပါတယ်။ ဒါကို နားလည်တယ်နော်။
- ◆ ဒုတိယ statement b=a-b; မှာ a ထဲကနေ b တန်ဖိုးကို နှုတ်ခိုင်းလိုက်တယ်။ a က 11ဆိုတော့ 7 ရတယ်။ အဲ့ဒီတန်ဖိုးကို b ထဲမှာ သိမ်းလိုက်တယ်။ ဒီတော့ b ထဲမှာ 4 ရှိမနေတော့ဘဲ၊ 7 ရှိနေပါတယ်။
- ◆ တတိယ statement a=a-b; လို့ကြေငြာတဲ့ အခါမှာ a ထဲမှာ ရှိနေတဲ့ 11 ထဲကနေ b ထဲမှာ ရှိတဲ့ 7 ကို နှုတ်လိုက်တယ်။ 4 ရတယ်။ဒီ 4 ကို a နဲ့ assign လုပ်လိုက်တာ ဖြစ်တဲ့အတွက် အခု a ထဲမှာ 11 ရှိမနေတော့ဘဲ 4 ရှိနေတာပါ။ ဒါကြောင့် printf("Now A=%d and B=%d",a,b); လို့ရေးတဲ့အခါမှာ a ထဲမှာ နောက်ဆုံး ရှိနေတဲ့ တန်ဖိုး 4 နဲ့ b ထဲမှာ နောက်ဆုံးရှိနေတဲ့ တန်ဖိုး 7 ကို Output ထုတ်ပေးတာဖြစ်ပါတယ်။ ခုဆိုရင် a နဲ့ b တန်ဖိုးဟာ မူလ ရှိနေတဲ့ တန်ဖိုးတွေနဲ့ ပြောင်းပြန်ဖြစ် နေပါတယ်။ဒါကို swapping လုပ်တာလို့ခေါ် ပါတယ်ခင်ဗျ။ နားလည် သဘောပေါက်သွားပြီလို့ထင်ပါတယ်.....။

b) Unformatted functions

ခု Unformatted functions ဖြစ်တဲ့ getch() function နဲ့ getche() function အလုပ်လုပ်တာကို လေ့လာ ကြရ အောင်။ program လေးရေးပြီး လေ့လာတာပေါ့။

```
1 #include<stdio.h>
2 #include<conio.h>
3 main()
4 {
5     printf("Enter any two alphabets");
6     getche();
7     getch();
8 }
```

```
Enter any two alphabetsA
Process exited with return
```

- ◆ ဒီ program မှာ #include<conio.h> ဆိုတဲ့ header file ကို ကြေငြာတာကို ပြောပြပါမယ်။ ဒီ header file ကို ကြေငြာပေးရတာကတော့ current program မှာ getche(),getch() ဆိုတဲ့

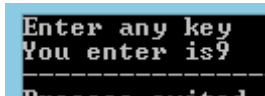
Console input /output function သုံးချင်တဲ့အတွက်ဖြစ်ပါတယ်။ ဒီ header file ကို မကြေငြာပေးရင်တော့ သုံးရမှာ မဟုတ်ပါဘူး။ ဒါကြောင့် program မှာ header file တွေဟာ အရေးကြီးတယ်လို့ပြောခဲ့တာဖြစ်ပါတယ်။

- ◆ ပထမ statement မှာ printf() function နဲ့ Enter any two alphabets မိမိနှစ်သက်တဲ့ alphabets နှစ်ခု ရိုက်ပါလို့ user သိအောင်ပြပေးဖို့ ရေးလိုက်ပါတယ်။ အဲ့တော့ user က alphabet နှစ်လုံး ရိုက်ပေးရမှာပါ။ ဥပမာ user က A နဲ့ B နှစ်လုံးကို ရိုက်လိုက်တယ်ဆိုပါစို့။ ပထမ ရိုက်လိုက်တဲ့ alphabet Aကို getch() function ကလက်ခံ ပေးမယ်ပြီးရင် user မြင်အောင် display လုပ်ပေးပါတယ်။ ဒုတိယ ရိုက်လိုက်တဲ့ alphabet B ကိုတော့ getch() function က လက်ခံပေးပါတယ်။ဒါပေမယ့် သူက user မြင်အောင် display ပြမပေးပါဘူး။ ဒါကြောင့် ဒီ program ကို run တဲ့အခါမှာ A ကိုပဲ မြင်ရပြီး B ကို မမြင်ရခြင်းဖြစ်ပါတယ်။ ရှင်းမယ်လို့ ထင်ပါ တယ်ဗျ။ ဒီမှာ ထူးခြားတာရှိတယ်ဗျ။ ဘာလဲဆိုတော့ data တွေကို input /output လုပ်ဖို့အတွက် printf(), scanf()function တွေလို conversion symbol တွေ မလိုခြင်းဖြစ်ပါတယ်။.... ဒီ program ကို ရှင်းသွားပြီလို့ ထင်ပါတယ်ဗျ။.....

ခုဒီ unformatted function ထဲက getch() နဲ့ putchar() functions အလုပ်လုပ်တာကို ကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  char ch;
5  printf("Enter any key \n");
6  ch=getch();
7  printf("You enter is");
8  putchar(ch);
9  }
```



- ◆ ဒီ program မှာ variable ch ကို character data type ပါလို့ ပထမ declare လုပ်လိုက်ပါတယ်။ ဒုတိယ statement မှာ Enter any key ဆိုတဲ့ စာတန်းလေး monitor မှာ display ပြခိုင်းလိုက်ပါတယ်။ ဒီတော့ user က key တစ်ခု ရိုက်ရမယ်ဆိုတာကို သိသွားပြီ။ ဥပမာ 9 ကို ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒီ user ရိုက်လိုက်တဲ့ တန်ဖိုးကို getch() function က လက်ခံပေးပါတယ်။ ဒါပေမယ့် user မြင်အောင် ပြမပေးပါဘူး။ scanf () function မှာဆို user က 3 လို့ရိုက်လိုက်တယ်ဆို 3 ကို ပြပေးပါတယ်။ ခု program မှာဆို Enter any key နောက်မှာ 9 ဆိုတာကို ပြမပေးဘူး။ ဒါပေမယ့် 9 ကိုတော့ getch() function က လက်ခံထားပေးတယ်။
- ◆ ပြီးတော့ ch=getch(); ဆိုပြီး သူလက်ခံထားတဲ့တန်ဖိုးကို variable ch ထဲကို ထည့်လိုက်တယ်။ ဒီတော့ ch ထဲမှာ 9 ရောက်သွားပြီ။ ခု user မြင်အောင် display ပြခိုင်းတော့မယ်။ display ပြဖို့အတွက် putchar(argument) function ကို သုံးပြီး display

ပြခိုင်းလိုက်တယ်။ ဒီနေရာမှာ printf() နဲ့လည်းပြလည်းရတယ်နော်။ putchar() function ရဲ့ အလုပ်လုပ်ပုံကို သိစေချင်လို့ ပါ။ putchar(argument) function က output function ဖြစ်ပါတယ်။ ဒါပေမယ့် conversion symbol သုံးစရာမလိုပါဘူး။ printf() နဲ့ပြမယ်ဆိုရင်တော့ လိုတာပေါ့။ သူက ဘယ်လို output ထုတ်ပေးလဲဆိုတော့ putchar() ရဲ့ လက်သညးကွင်း (parathesis) ထဲမှာ ရှိတဲ့ argument မှာ သိမ်းထားတဲ့ တန်ဖိုးကို display လုပ်ပေးတာ ဖြစ်ပါတယ်။ ဒီ program မှာ putchar(ch) လို့ရေးထားတဲ့ အတွက် argument ch ထဲမှာ သိမ်းထားတဲ့ 9 ကို display ပြပေးတာဖြစ်ပါတယ်။ ဒါဆို ဒီ program တစ်ခုလုံးကို နားလည်မယ်နော်။ ဘယ် statement တွေကို ဘာကြောင့်ရေးတာ။ ဘာလုပ်ဖို့အတွက် ရေးတာ။ ဒီမေးခွန်းတွေကို ဖြေနိုင်ရင်တော့ သင် ဒီ program ကို ကောင်းကောင်း နားလည်ပြီလို့ ယူဆလို့ရပါတယ်ခင်ဗျ။ နားလည် နိုင်ပါစေ.....။

အခြား unformatted function တွေကို တော့ pointer အခန်းမှ ရှင်းပြပေးပါမယ်။ ရှုပ်ထွေးသွားမှာ စိုးလို့ပါ။

5. Decision Statements

1. Introduction

ဒီ အခန်းမှာတော့ program ရဲ့ flow ကို ပြောင်းလဲသွားဖို့။ Logical operators တွေကို စစ်ဆေးဖို့ နဲ့ program ရဲ့ execution flow ကို ထိန်းချုပ်ဖို့ အတွက် Decision statements တွေအကြောင်းကို ရှင်းပြပါမယ်။

ဒီ Statement တွေရဲ့ အသုံးပြုပုံအားလုံးကို ရှင်းပြပေးပါမယ်။

1) The if statement

2) The if-else statement

3) The if-else-if statement

4) The switch () statement

1) The if statement

If statement ကို ဘာကြောင့်သုံးတာလဲဆိုတာကို ပြောပြပါမယ်။ ကျွန်တော်တို့ ရှေ့အခန်းမှာ Logical operator အကြောင်းကို လေ့လာခဲ့ပြီးပြီဟုတ်။ logical operator ကနေ expression နှစ်ခုကို နှိုင်းယှဉ် ပါမယ်။ဒါကို if statement သုံးပြီး စစ်ဆေးပါမယ်။ မှန်တယ်ဆိုရင် true ဖြစ်မယ် မှားရင် false ဖြစ်မယ်။ အရင်ဆုံး if statement ရဲ့ syntax ကိုပြောပြပါမယ်။

if (condition)

{

Statement;

}

ဒီ syntax ကို ရှင်းပါမယ်။ if က သူ့ရဲ့ လက်သညးကွင်း (parenthesis) ထဲက condition ကို စစ်မယ်။ အဲဒီ condition က မှန်ရင် if block ထဲက statement ကို အလုပ်လုပ်မယ်။ မှားရင်တော့ if block အပြင် ဘက်က statement ကို အလုပ်လုပ်ပါမယ်။ statement မရှိရင်တော့ program က ဆက်အလုပ် မလုပ် တော့ပါဘူး။ { } ကတော့ if block ရဲ့ scope ဖြစ်ပါတယ်။ { } ရဲ့ အပြင်ဘက်က Statement တွေက တော့ if block ထဲမှာ မရှိတဲ့ အတွက် if နဲ့ မဆိုင်ပါဘူး။ ပိုနားလည်သွားအောင် program ရေးပြီး ဥပမာပြပေးပါမယ်။

```

1  #include<stdio.h>
2  main()
3  {
4      int v;
5      printf("Enter the number");
6      scanf("%d",&v);
7      if(v<10)
8      {
9          printf("The number is less than 10");
10     }
11 }
```

```

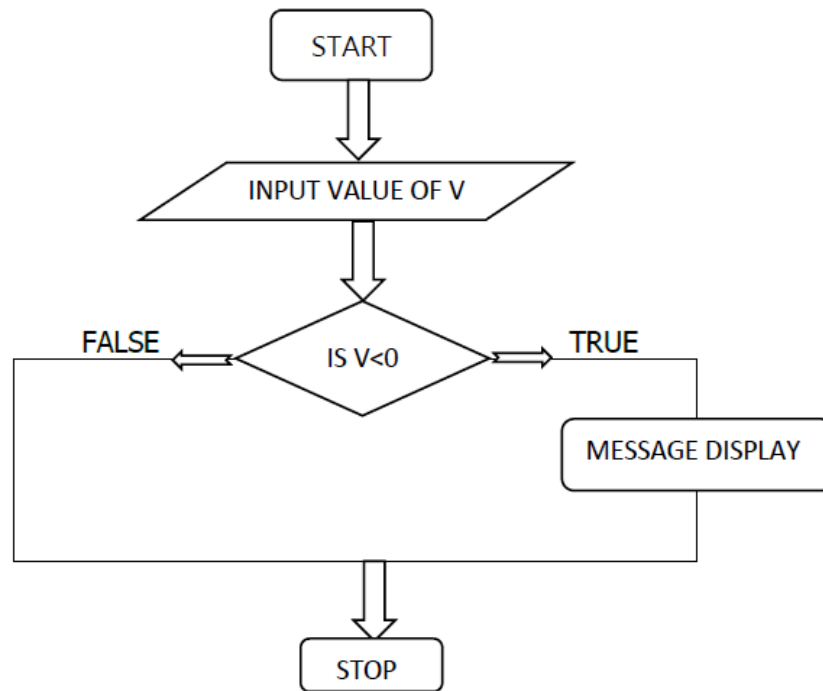
Enter the number 9
The number is less than 10
Process exited with return
```

(or)

```

Enter the number 11
Process exited with r
```

ခု if statement ရဲ့ အလုပ်လုပ်ပုံကို ကြည့်ရအောင်။ user ဆီကနေ တန်ဖိုးတစ်ခုတောင်းမယ်။ ဥပမာ user က 9 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ အဲဒီ 9 ကို variable v ထဲမှာ သိမ်းထားမယ်။ ပြီးရင် if နဲ့ စစ်ဆေးမယ်။ if(v<10) လို့ရေးထားတာကို နားလည်အောင်ပြောပြပါမယ်။ variable v ထဲမှာရှိတဲ့ တန်ဖိုးက 10 ထက်ငယ်တာမှန်ရင် လို့ဆိုလိုတာပါ။ အကြောင်းရှိရင် အကျိုးလည်းရှိမှာပဲလေ။ ဒီတော့ မှန်ရင် ဘာဖြစ်မလဲ။ မှန်ရင် if block ထဲမှာ ရှိတဲ့ statement ကို run မှာပါ။ if block ထဲက statement ကို run တော့ ဘာတွေ့လဲဆိုတော့ printf() function ကို compiler က တွေ့သွားပြီ။ print() function က output function လေ။ ဒါကြောင့် double quote ထဲမှာ ရှိတဲ့ စာသားကို display ပြပေး တာဖြစ်ပါတယ်။ နားလည်မယ်လို့ထင်ပါတယ်။ တကယ်လို့ user က 11 လို့ ရိုက်လိုက်တယ် ဆိုပါစို့။ if statement က ဘယ်လိုအလုပ်လုပ်မလဲ။ ကြည့်ရအောင်။ if(v<10) ဆိုပြီး condition ကို စစ်လိုက် တော့ v ထဲမှာသိမ်းထားတဲ့ တန်ဖိုး 11 က 10 ထက်ကြီးနေတယ်လေ။ ဒီတော့ condition က မှားနေပြီ။ မှားတဲ့ အတွက် if statement က သူ့ block ထဲမှာရှိတဲ့ statement ကို run ခွင့်မပေးတော့ဘူး။ ဒါကြောင့် program က ဘာမှ မပြတော့ဘဲ ရပ်သွားမှာပါ။ ဒါကို ပိုနားလည်အောင် ပုံလေးဆွဲပြီး ရှင်းပြ ပေးပါမယ်။



ဒီ ပုံကို ကြည့်ပြီး နားလည်မယ်လို့ ထင်ပါတယ်။ START ဆိုတဲ့ နေရာမှာ Programကို စရေးမယ်ပေါ့ ဗျာ။ INPUT VALUE OF V မှာ Variable v ထဲကို သိမ်းမယ့် တန်ဖိုး တစ်ခု ရိုက်လိုက်မယ်။ IS V<10 ဆိုတာကတော့ condition စစ်တာပါ။ variable v ထဲက တန်ဖိုးက 10 ထက်ငယ်လားပေါ့။ TRUE ဖြစ်ရင် မိမိပြချင်တဲ့ MESSAGE ကို DISPLAY ပြပေးမှာဖြစ်ပါတယ်။ FALSE ဆိုရင်တော့ MESSAGE မပြဘဲ program က ရပ်သွားမှာဖြစ်ပါတယ်။ ဒီလောက်ဆို သဘောပေါက် မယ်လို့ထင်ပါတယ်။

If statement ကို ပိုနားလည်သွားအောင် နောက် program တစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  int main()
3  {
4  int m,n;
5  printf("Enter two numbers");
6  scanf("%d %d",&m,&n);
7  if(m-n==0){
8  printf("Two numbers are equal");}
9  getch();
10 return 0;
11 }

```

```

Enter two numbers 5 5
Two numbers are equal

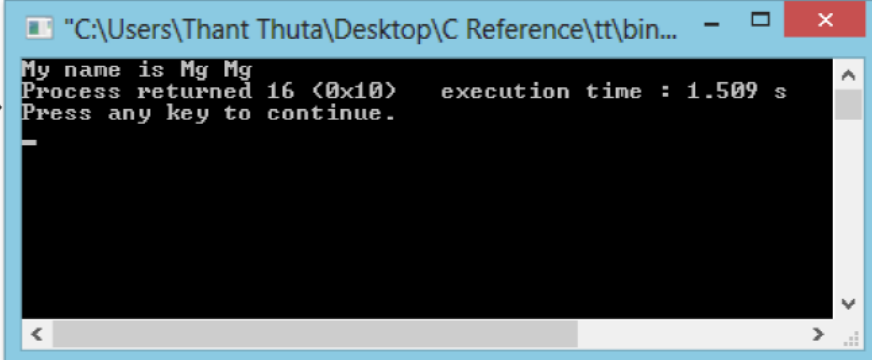
```

ဒီ program ကို ကြည့်မယ်ဆိုရင် ထူးခြားတာဘာရှိလဲ။ မရှိဘူးနော်။

- ◆ Line_2 `int main()` လို့ ရေးထားတာကတော့ program ကို run တဲ့အခါမှာ `main()` function ဆီ `return` ပြန်လာမယ့် တန်ဖိုးရဲ့ data ဟာ `integer data type` အမျိုးအစားဖြစ်ပါတယ်လို့ ကြေငြာတာဖြစ်ပါတယ်။ program ရဲ့ အဆုံးမှာ `return 0` လို့ ရေးထားတာကို တွေ့တယ်ဟုတ်။ အဓိပ္ပာယ်ကတော့ `main ()` function ဆီကို `0` `return` ပြန်ပါမယ်လို့ ဆိုလိုတာပါ။ ဒီတော့မ compiler က program ပြီးဆုံးသွားပါပြီ ဆိုတာကို သိသွားအောင် ရေးပေးတာဖြစ်ပါတယ်။ မရေးပေးလည်းရပါတယ်။ program ရဲ့ အလုပ်ကို စနစ်တကျ လုပ်ဆောင်ဖို့အတွက် ရေးပေးတာပါ။ `printf()` နဲ့ `scanf()` function ကိုသုံးပြီး data တွေတောင်းတာ လက်ခံတာကို တော့နားလည်မယ်လို့ ထင်ပါတယ်။
- ◆ Line_7 `if(m-n==0)` ဆိုတဲ့ statement ရေးတာကို ရှင်းပြပါမယ်။ variable `m` ထဲမှာ ရှိတဲ့ တန်ဖိုးထဲက `n` ထဲက တန်ဖိုးကို နှုတ်လို့ရတဲ့ တန်ဖိုးက `0` နဲ့ ညီမယ်ဆိုလျှင် လို့ Condition စစ်လိုက်တာဖြစ်ပါတယ်။ ဥပမာ `user` က `m` ထဲကို `5` `n` ထဲကို လည်း `5` လို့ ရေးလိုက်တယ်ဆိုပါစို့။ ဒါဆို `m-n` က `0` ဖြစ်သွားပြီ။ ဒီတော့ condition က မှန်သွားပြီလေ။ မှန်တော့ compiler က `if block` ထဲက statement ကို run မှာဖြစ်ပါတယ်။
- ◆ Line_8 `printf("Two numbers are equal");` ကို run မှာဖြစ်ပါတယ်။ ဒီတော့ `printf()` က သူ့ထဲက စာသားတွေကို output ထုတ်ပေးမှာ ဖြစ်ပါတယ်။ နားလည် မယ်ဟုတ်။ `m-n==0` ကို စစ်လိုက်တော့ မှားတယ်ဆိုရင်တော့ program က ဘာမ output ထုတ်မပေးဘဲ run box လေးပဲပြပေးမှာပါ။
- ◆ Line_9 `getche()` ဆိုတဲ့ console function ကို ရေးပေးရတာက အကြောင်းပြချက် နှစ်ခုရှိတယ် ဗျ။ ပထမ အကြောင်းပြချက်က program ကို run လိုက်တဲ့အခါမှာ အမဲရောင် run box လေးက ခဏလေးပဲ ပြပြီး ပိတ်သွားတဲ့အတွက်ကြောင့်ပါ။ အဲလိုပိတ်သွားတော့ `user` က ဘာ output ထုတ်ပေးလည်းဆိုတာကို မသိ တော့ဘူးလေ။ ဒါကြောင့် `getche()` ရေးပေးမယ်ဆိုရင် `user` က key တစ်ခုခုကို နှိပ်လိုက်မှသာ run box လေးက ပျောက်သွားမှာဖြစ်ပါတယ်။ `user key` မနှိပ်မချင်း run box လေးပေါ်နေမှာဖြစ်ပါတယ်။ Code block နဲ့ ရေးရင်တော့ အဲလိုမျိုး run box က ပျောက်သွားတာမျိုး မဖြစ်ပါဘူး။ Visual Studio သုံးပြီး ရေးတယ်ဆိုရင် ဒီပြဿနာကို ကြုံရမှာပါ။ ဒါကို သေချာလေးမှတ်ထားပေးပါ။ နောက်အကြောင်းပြချက်က run box လေးထဲမှာ process returned ဘာဆိုပြီး output data တွေရဲ့ အောက်မှာ ပြတဲ့ စာတန်းကို ဖျောက်ချင်တဲ့ အခါမှာ လည်းသုံးပါတယ်။ ပုံလေးနဲ့ တွဲပြီးပြပေးပါမယ်။

```
#include<stdio.h>
int main()
{
    char c;

    printf("My name is Mg Mg\n");
}
```



ခုနစ်ပြားတို့ စာသားလေးကို ဖျောက်ချင်တဲ့ အခါမှာလည်းရေးပါတယ်။ ရှင်းပြီဟုတ်။ ခုလို တစ်ကြောင်း ချင်းစီ ရှင်းပြတယ်ဆိုတာက programming sense ရစေဖို့အတွက်ပါ။ ဘာလို့လည်း ဆိုတော့ compiler က run တဲ့ အခါမှာ တစ်ကြောင်းချင်းစီ အစဉ်လိုက် run တာမို့ပါ။ ဒါကြောင့် compiler run တာကို မိမိကိုယ်တိုင် လိုက်ပြီး စဉ်းစားနိုင်တယ် နားလည်မယ်ဆိုရင် program ရဲ့ flow ကို နားလည်သဘောပေါက်မှာ ဖြစ်ပါတယ်။ ဒါကြောင့် program တိုင်းကို တစ်ကြောင်းချင်းစီ နားလည်အောင်လုပ်ဖို့ အကြံပေးချင်ပါတယ်ခင်ဗျ။....။

If statement ကို ပိုနားလည်သွားအောင် နောက်ထပ် program လေး တစ်ပုဒ်လောက် ရေးမယ်ဗျာ။ user ဆီကနေ မင်းအသက်ကို ရိုက်ပေးပါလို့ပြောမယ်။ ဒီတော့ user က အသက် တန်ဖိုးလေးရိုက်လိုက်မယ်။ အဲ့ဒီ အသက်က ၃၀ ထက်ငယ်ရင် you are young မင်းဟာ လူငယ်တစ်ယောက်ပါဆိုတဲ့ စာသားလေးပေါ်အောင် ရေးမယ်။ ၃၀ထက်ကြီးရင်တော့ program က ပြီးဆုံးသွားမှာပါ။ ရေးကြည့်ရအောင်။

```
1 #include<stdio.h>
2 main()
3 {
4     int age;
5     printf("Enter your age");
6     scanf("%d",&age);
7     if(age<30)
8     {
9         printf("You are young");
10    }
11 }
```

```
Enter your age 20
You are young
```

(or)

```
Enter your age 40
Process exited with
```

ဒီ program ကို နားလည်မယ်နော်။ တကယ်လို့ 40 လို့ရိုက်လိုက်မယ်ဆိုရင်တော့ 30 ထက် ကြီးတာ ဖြစ်တဲ့အတွက် program က ဘာမှာ display ပြပေးမှာ မဟုတ်ပါဘူး။ မိမိဘာသာ trace ပြီး programming sense ရအောင်လုပ်ပါ။.....။

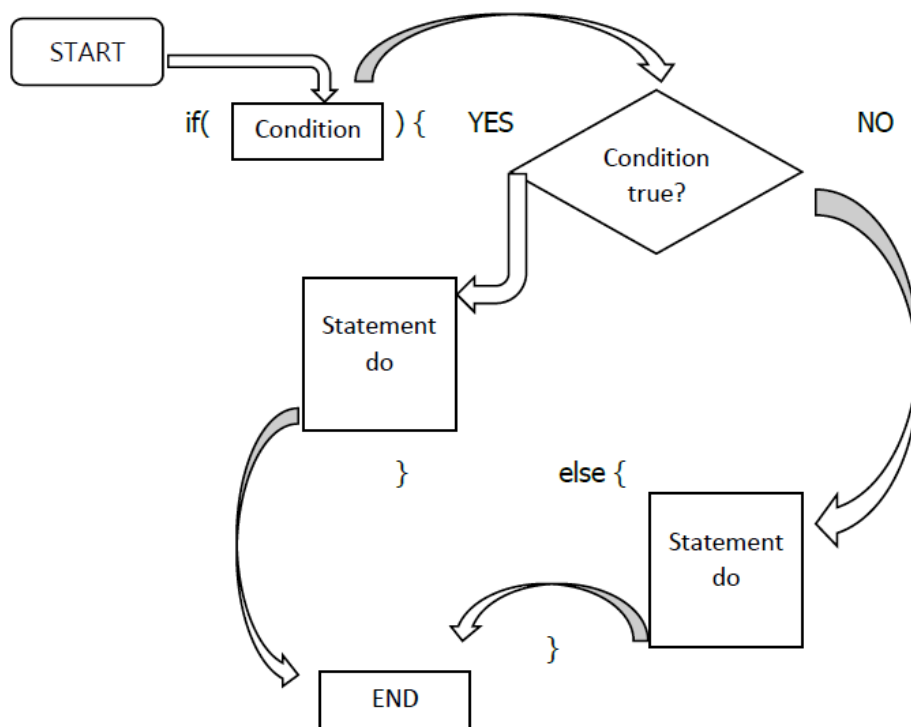
2) if ...else...statement

ဒီ control statement ကို ဘယ်လိုနေရာတွေမှာ သုံးလဲဆိုတာ ရှင်းပြပါမယ်။ if နဲ့ condition စစ်တာကို ရှေ့မှာ ပြောပြခဲ့ပြီးနော်။ အဲလိုစစ်လိုက်တဲ့အခါမှာ condition က မှန်ရင် if block ထဲက statement ကို အလုပ်လုပ်ပါမယ်။ မှားရင်ဘာဖြစ်မလဲ။ if statement မှာဆို မှားရင် program က ရပ်သွားမှာပါ။ if...else မှာတော့ အဲလိုမ ဟုတ်တော့ဘူး။ မှားရင် else block ထဲက statement ကို run ပြီး output ထုတ်ပေးမှာပါ။ သူ့ရဲ့ syntax လေးကို ကြည့်ရအောင်။

```
if(condition)
```

```
{
Statement;
}
else
{
Statement;
}
```

ဒါကတော့ သူ့ရဲ့ syntax ဖြစ်ပါတယ်။ သူ့ရဲ့ အလုပ်လုပ်ပုံကို ပိုနားလည်အောင် ပုံလေးဆွဲပြီး ပြပေးပါမယ်။



ပုံလေးကို ကြည့်ပြီး နားလည်မယ်လို့ ထင်ပါတယ်။ ခင်ဗျ။ program လေးရေးပြီး လေ့လာ ကြည့် ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int a;
5  printf("Enter number");
6  scanf("%d",&a);
7  if(a>9)
8  {
9  printf("The number is greater than 9");
10 }
11 else
12 {
13 printf("The number is less than 9");
14 }
15 }

```

```

Enter number 10
The number is greater than 9
Process exited with return value 0

```

(or)

```

Enter number 4
The number is less than 9

```

ဒီ program ကို ရှင်းပြပါမယ်။ တစ်ကြောင်းချင်းစီတော့ ရှင်းဖို့မလိုတော့ဘူးထင်ပါတယ်။ if statement ကနေပဲ ရှင်းပြပါမယ်။ ဥပမာ user က Enter number လို့ ပေါ်လာတဲ့အချိန်မှာ 10 လို့ ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆို a ထဲမှာ 10 ကို သိမ်းထားလိုက်ပြီ။ if (a>9) လို့ condition စစ်လိုက်တဲ့အခါ a က 10 ဆိုတော့ 9 ထက်ကြီးနေ တယ်။ ဒီတော့ if condition က မှန်နေတယ်။ ဒါကြောင့် if block ထဲက statement ကို run လိုက်ပါတယ်။ printf() ကနေပြီး The number is greater than 9 ဆိုတဲ့ စာသားလေးကို output ထုတ်ပေးတာဖြစ်ပါတယ်။ if statement က မှန်တာဖြစ်တဲ့အတွက် else ထဲက block ကို မ run တော့ပါဘူး။ ပုံထဲက အတိုင်းပါပဲ program က ပြီးဆုံးသွားပါမယ်။ နားလည်မယ်ထင်ပါတယ်။ တကယ်လို့ user က 4 လို့ ရိုက်လိုက်တယ်ဆိုပါစို့။ if နဲ့ စစ်လိုက်တဲ့အခါ a က 4 ဆိုတော့ 9 ထက်ငယ်နေတယ်။ ဒီတော့ condition ကမှားသွားပြီ။ ဒါကြောင့် program က if block ထဲက statement ကို မ run တော့ဘဲ else ထဲက statement ကို run ပေးလိုက်ပါတယ်။ ဒါကြောင့် else block ထဲက The number is less than 9 ဆိုတဲ့ စာသားလေးကို output ထုတ်ပေးလိုက်တာဖြစ်ပါတယ်။ ဒီ program ရဲ့ run flow ကို နားလည်မယ်လို့ထင်ပါတယ်။.....။

ရှေ့က program ကိုနားလည်မယ်လို့ထင်ပါတယ်။ ခု if-else statement နဲ့ logical AND operator ကို သုံးပြီး program လေးတစ်ပုဒ်လောက်ရေးကြည့်ရအောင်။


```

1  #include<stdio.h>
2  main()
3  {
4  int b;
5  printf("Enter number");
6  scanf("%d",&b);
7  if(b>10 && b<20)
8  {
9  printf("The number is between 10 and 20");
10 }
11 else
12 {
13 printf("Invalid number");
14 }
15 }

```

```

Enter number12
The number is between 10 and 20
Process exited with return value

```

```

Enter number9
Invalid number
Process exited with

```

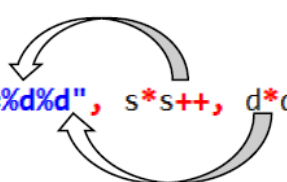
(or)

- ◆ Line_5 ဒီ program မှာဆို Enter number ဆိုပြီး ကိန်းတစ်ခုတောင်းမယ်။ input ထည့်လိုက်တဲ့ ကိန်းကို variable b ထဲမှာ သိမ်းထားမယ်။ ဒါကို နားလည်တယ်ဟုတ်။
 - ◆ Line_7 if(b>10 && b<20) ဆိုတဲ့ statement ကို ရှင်းရအောင်။ ဥပမာ ကျွန်တော်တို့က တန်ဖိုးကို 12 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ if condition နဲ့ စစ်လိုက်တော့ 12>10&& 12<20 ။ 12 က 10 ထက်ကြီးတယ်။ မှန်တယ်။ 12 က 20 ထက်လည်း ငယ်တယ်။ ဒီတော့ condition နှစ်ခုစလုံးက မှန်နေတယ်။ ဒါကြောင့် if statement က မှန်ပြီပေါ့ ။ ဒီတော့ သူ့ block ထဲ printf() statement ကို output ထုတ်ပေးလိုက်တယ်။ output ကိုကြည့်ပါ။ The number is between 10 and 20 ဆိုတဲ့ စာသားကို ဒါကြောင့် display ပြပေးတာဖြစ်ပါတယ်။ ဥပမာ user က တန်ဖိုးကို 9 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါကို If condition နဲ့စစ်မယ်။ 9>10 && 9<20 ဆိုတော့ 9 က 10 ထက်ကြီးလားစစ်တော့ မကြီးဘူးလေ။ ဒီတော့ ဒီ expression ကမှားနေပြီ။ 9 က 20 ထက်ငယ်လားစစ်တော့ မှန်နေတယ်။ ဒါပေမယ့် && (logical AND) operator က expression နှစ်ခုလုံးမှန်မှ conditionစစ်တာ မှန်ပါတယ်လို့ ထောက်ခံမယ်လေ။ တခုက မှားနေ တော့ if statement block ကို မ run တော့ဘဲ if block ကို ကျော်ပြီး else block ထဲက statement ကိုပဲ output ထုတ်ပေးတာဖြစ်ပါတယ်။ ဒါကြောင့် Invalid number ဆိုတဲ့ စာသားကို display လုပ်ပေးတာပါ။ ဒီ program မှာ နားမလည်တာ မရှိတော့ဘူးလို့ ထင်ပါတယ်။
- ခု နောက် program လေးတစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int s,d;
5  printf("Enter number");
6  scanf("%d",&s);
7  d=s%10;
8  if(d=5)
9  {
10 s=s/10;
11 printf("Square=%d%d", s*s++, d*d);
12 }
13 else
14 {
15 printf("Invalid number");
16 }
17 }

```



```

Enter number 25
Square=625
Process exited with return

```

Program လေးကနည်းနည်းတော့ရှည်တယ်ဗျ။ ဒါပေမယ့် သေချာနားလည်အောင်ကြည့်ပါ။ ပထမဆုံး user ဆီက တန်ဖိုး ခုတောင်းပြီး လက်ခံသိမ်းဆည်းပေးမယ့် variable name တွေ အရင် ကြေငြာပေးမယ်။ ပြီးရင် ဒီဒေတာတွေကို လက်ဆင့်ကမ်း လက်ခံပေးမယ့် variable d ကိုပါ ကြေငြာပေးထားမယ်။ ဥပမာ user က 25 လို့ ရိုက်လိုက်တယ်ဆိုပါစို့။ဒီ value ကို variable s ထဲမှာ သိမ်းထားမယ်။

- ◆ Line_7 ဒီ d= s%10 ဆိုတဲ့ statement ကို ရှင်းပြပါမယ်။s က 25 ဆိုတော့ module(%) စားလိုက်တော့ စားလဒ်က 2 အကြွင်း က 5။ ရှေ့မှာလည်း ပြောခဲ့ ပြီးပြီနော်။ module လုပ်တယ်ဆိုရင် အကြွင်းကို ယူတယ်ဆိုတာ။ ဒီတော့ အကြွင်း တန်ဖိုး 5 ကို d နဲ့ assign လုပ်ထားတာဖြစ်တဲ့အတွက် d ထဲမှာ သိမ်းထားလိုက်ပြီ။
- ◆ Line_8 နောက်တစ်ကြောင်း if(d==5) ဆိုပြီး d ထဲက တန်ဖိုးက 5 နဲ့ ညီလျှင်ဆိုတော့။ ညီတယ်လေ။ ဒီတော့ if block ထဲကို statement တွေကို run မယ်။ ပထမဆုံး statement က s=s/10; ဆိုတော့ s တန်ဖိုးက 25။ 25 ကို 10 နဲ့ Division လုပ်တော့ စားလဒ်က 2 အကြွင်း က 5 ဖြစ်ပါတယ်။ Division လုပ်ရင် စားလဒ်တန်ဖိုးကို ယူမယ်။ ဒါကြောင့် s ထဲမှာ မူလရှိနေတဲ့ တန်ဖိုး 25 မရှိတော့ ဘဲနောက်ဆုံး assign လုပ်လိုက်တဲ့စားလဒ်တန်ဖိုး 2 ပဲရှိတော့မယ်။
- ◆ Line_11 နောက် statement printf ("Square=%d%d", s*s++, d*d); ကို run တော့ s*s++ ဆိုတော့ s က 2 ဖြစ်မယ်။ s++ ကတော့ Output တခါတည်းထုတ်တာ ဖြစ်တဲ့အတွက် s ကို တစ်တိုးပြီး s တန်ဖိုး 2 ကိုတိုးတော့ 3 ဖြစ်ပါမယ်။ ဒါကြောင့် 2*3 ကို multiply(*) လုပ်တော့ 6 ရပါတယ်။ 6 ကို output ထုတ်ဖို့အတွက် conversion symbol %d ကို သုံးခြင်းဖြစ်ပါတယ်။ expression တစ်ခုပြီးသွားပြီဖြစ်တဲ့အတွက် comma(,) ခံပေးရပါမယ်။

နောက် expression တစ်ခုက $d*d$ ။ d ထဲမှာရှိတဲ့ တန်ဖိုးက 5 ဆိုတော့ $5*5$ ဖြစ်ပါမယ်။ 25 ရပါမယ်။ 25 ကို output ထုတ်ဖို့အတွက် ဒုတိယ $%d$ ကို သုံးခြင်းဖြစ်ပါတယ်။ (Expression 4 ခုရှိရင် conversion symbol လည်း ၄ ခုရှိရပါမယ်။) ဒါကို monitor မှာ display ပြပါမယ်။ $%d\%d$ လို့ရေးထားတဲ့ အတွက် 625 လို့ display ပြပါမယ်။ တကယ်လို့ $%d,\%d$ လို့ရေးမယ်ဆိုရင်တော့ 6,25 လို့ပေါ်ပါမယ်။ နားလည်မယ်နော်။ ဒီတော့ ကျွန်တော်တို့ ရိုက်လိုက်တဲ့ တန်ဖိုး 25 ရဲ့ နှစ်ထပ်ကိန်း 625 ဆိုပြီး output display ပြခြင်းဖြစ်ပါတယ်။

- ◆ တကယ်လို့ user က 20 လို့ရိုက် လိုက်တယ်ဆိုပါစို့။ s တန်ဖိုးက 20 ဖြစ်သွားမယ်။ $d=s\%10$; ဆိုတဲ့ statement ကို run တော့ $20\%10$ ဒါကို modular division လုပ်မယ်ဆို 0 ရပါမယ်။ module ဆိုတော့ အကြွင်း တန်ဖိုးကို ယူတာလေ။ ပြီးရင် နောက် statement $\text{if}(d==5)$ ဆိုတဲ့ if နဲ့ condition စစ်လိုက်တော့ မှားပြီလေ။ d က s ကို 10 နဲ့ module လုပ်ထားတဲ့ တန်ဖိုး 0 ဆိုတော့ 5 နဲ့ မညီတော့ဘူး။ ဒီနေရာမှာ တစ်ခုမှတ်ထားစေချင်တာက $\text{Equal}(==)$ နဲ့ $\text{Assign}(=)$ က မတူဘူးနော်။ $\text{Equal}(==)$ ဆိုတာက ဘယ်ဘက်နဲ့ညာဘက် တန်ဖိုးတူညီလား နှိုင်းယှဉ်တဲ့အခါမျိုးမှာသုံးတာ။ $\text{Assign}(=)$ ဆိုတာက ညာဘက်က တန်ဖိုးကို ဘယ်ဘက်ထဲ ကိုသိမ်းတာ။ ပိုနားလည်အောင်ပြောရရင် $a=4$ ဆိုပါစို့။ ဒါဆိုရင် a ကို 4 နဲ့ assign လုပ်လုပ်တာ။ ဒီတော့ ညာဘက်က တန်ဖိုး 4 ကို ဘယ်က variable a ထဲမှာ သိမ်းလိုက်တာ။ ဒီနှစ်ခုကို ရှင်းမယ်လို့ထင်ပါတယ်။ ပြန်သွားရအောင်။ အဲလို ညီလားလို့ စစ်လိုက်တော့ မညီဘူး။ ဒါကြောင့် compiler က if block ထဲက statement ကို မ run တော့ဘဲ else block ထဲက statement ကို သွားပြီး run ပါမယ်။ ဒီတော့ $\text{printf}()$ function ကနေပြီး Invalid number ဆိုတဲ့ စာသားကို monitor ပေါ် output display ပြပေးတာဖြစ်ပါတယ်။ ဒီ program ကို နားလည်မယ်လို့ထင်ပါတယ်။ နားမလည် ရင် နောက်တစ်ခေါက် ပြန်ဖတ်ပြီး line by line စဉ်းစားပြီး တွက်ကြည့်ပါ။အဆင်ပြေပါစေ....။

3. Nested if-else Statement

Nested ဆိုတဲ့ အတိုင်းပါပဲ if-else statement ကို ငုံထားတဲ့ statement မျိုးကို Nested if-else statement လို့ခေါ်ပါတယ်။ ခု Nested if-else statement ကို မရှင်းခင် $\text{if...else if...else}$ statement ကို ရှင်းပြပါမယ်။ ဒီ statement ကို သုံးရတဲ့အကြောင်းကတော့ Statements အမျိုးမျိုးကို output ထုတ်ချင်တဲ့ အခါမျိုးမှာ သုံးပါတယ်။ ဆိုလိုတာကတော့ဗျာ။ if နဲ့ စစ်မယ်။ မှားရင် else if ထဲက statement ကို run မယ်။အဲဒီမှာ ထပ်မှား ရင် else ထဲက statement ကိုပဲ execute (တွက်ထုတ်) ပေးမှာဖြစ်ပါတယ်။ သူ့ရဲ့ Syntax လေးကို ကြည့်ရအောင်။

```
if(condition)
{
statement;
}
else if(condition)
{
```

```

statement;
}
else
{
statement;
}

```

အိုကေ...ဒီ syntax ကို သိပြီးဆိုတော့ သူ့ရဲ့ flow ကို သိဖို့ program လေး ရေးကြည့်ပြီး ရှင်းတာပေါ့။ ဘာအကြောင်းနဲ့ ပတ်သတ်ပြီးရေးမလဲဆိုတာကို ပြောပြပါမယ်။ Energy bill ကို ရှာတဲ့ program လေး ရေးမှာပါ။ ပထမ user ဆီကနေ တန်ဖိုးနှစ်ခုတောင်းပါမယ်။ ပထမတန်ဖိုးကို initial တန်ဖိုးအဖြစ် သတ်မှတ်ပါမယ်။ ဒုတိယ တန်ဖိုးကိုတော့ final အဖြစ်သတ်မှတ်ပါမယ်။ ပြီးရင် consumed ကို ရှာဖို့အတွက် final ထဲက initial ကို နှုတ်ပါမယ်။ ပြီးရင် နှုတ်လို့ရတဲ့ consumed တန်ဖိုးက အောက်မှာ ပေးထားတဲ့ No.of unit consumed ထဲက ဘယ် တန်ဖိုးနှစ်ခုကြားမှာရှိလဲ ရှာပြီး Total bill ကို ရှာပါမယ်။ total bill ကို ရှာဖို့အတွက် consumed တန်ဖိုးကို No.of units consumed ထဲက ဘယ်တန်ဖိုးနှစ်ခုကြားမှာ ရှိလဲကြည့်ပြီး ရှိတဲ့နေရာက တန်ဖိုးရဲ့ Rate နဲ့ မြှောက်ပေး ရပါမယ်။ ဒါဆိုရင် energy bill ကို ရှာလို့ရပါပြီ။ ကဲ ...program ရေးတာလေးကို ကြည့်ပါ။

No.of units consumed	rates in (Rs)
200-500	3.5
100-200	2.50
Less than 100	1.50

```

1  #include<stdio.h>
2  main()
3  {
4      int initial,final,consumed;
5      float total;
6      printf("Enter Initial and final value");
7      scanf("%d %d",&initial,&final);
8      consumed=final-initial;
9      if(consumed>=200&& consumed<=500)
10     {
11         total=consumed*3.50;
12     }
13     else if(consumed>=100 && consumed<=200)
14     {
15         total=consumed*2.50;
16     }
17     else if(consumed<100)
18         total=consumed*1.500;
19     printf("Total bill for %d unit is %f",consumed,total);
20 }

```

```
Enter Initial and final value 800 850
Total bill for 50 unit is 75.000000
-----
Process exited with return value 35
```

- ◆ ဒီ program မှာဆိုရင် ပထမဆုံး USER ဆီက initial နဲ့ final ရဲ့ တန်ဖိုးတွေကို တောင်းမယ်။ ဥပမာ userက initial အတွက် 800 နဲ့ final အတွက် 850 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒီတော့ consumed ကို ရှာဖို့အတွက် final ထဲက initial ကို နှုတ်လိုက်တယ်။
- ◆ Line_8 consumed=final-initial; ဆိုပြီး နှုတ်ပေးလိုက်တာပါ။ 850-800 ဆိုတော့ consumed တန်ဖိုးက 50 ဖြစ်ပါ မယ်။ ရလာတဲ့ consumed တန်ဖိုးကို if နဲ့ စစ်ပါမယ်။
- ◆ Line_9 if (consumed>=200 && consumed <=500) ဖြစ်လားလို့ စစ်တာပါ။ ဘာလို့စစ်တာလဲ ဆိုတော့ No.of units consumed 200-500 ရဲ့ total energy bill ကို ရှာဖို့အတွက်ပါ။ အဲလိုစစ်လိုက်တော့ consumed က 200 ထက်ကြီးလားဆိုတာက မှားနေတယ်။ 500 ထက်ငယ်လားဆိုတော့ မှန်တယ်။ ဒါပေမယ့် logical AND ကို သုံးထားတာ ဖြစ်တာအတွက် တခုက မှားနေတာဆိုတော့ if block ကို run တော့ဘူး။ else if statement ကို run မယ်။
- ◆ Line_11 တကယ်လို့ မှန်တယ်ဆိုရင်တော့ total=consumed*3.5 ကို output ထုတ်ပေးမှာ ဖြစ်ပါတယ်။ ခုမှား တာဖြစ်တဲ့အတွက် ဒုတိယ else if နဲ့ 100-200 ရဲ့ total ကို ရှာဖို့ ထပ်စစ်မယ်။ if(consumed>=100 && <=200); ဆိုပြီး စစ်လိုက်တော့ consumed တန်ဖိုး 50 က 100 ထက်မကြီးပြန်ဘူး။ 200 ထက်တော့ ငယ်တာ မှန်ပါတယ်။ ဒါကြောင့် else if ကို လည်း မ run တော့ဘဲ နောက်ဆုံး else if ကို run ပါမယ်။
- ◆ Line_15 မှန်တယ်ဆို ရင်တော့ total =consumed*2.500 ဆိုတဲ့ statement ကို output ထုတ်ပေးမှာ ဖြစ်ပါတယ်။ ခုတော့ else if block ကို ကျော်ပြီး နောက်ဆုံး else if နဲ့ less than 100 ရဲ့ energy bill ကို စစ်မယ်။ else if (consumed<100) ဆိုတဲ့ အတွက် consumed တန်ဖိုးက 100 ထက်ငယ်တယ်။ ဒါကြောင့် နောက်ဆုံး else if block ထဲက statement ကို run ပါမယ်။
- ◆ Line_18 total =consumed*1.50; ဆိုတော့ total=50*1.5 ကို run တော့ total=75.00 လို့ရပါတယ်။ ဒါကို display ပြဖို့အတွက် printf () function ကို pass လုပ်ပြီး output ထုတ်ပါမယ်။ ဒါကြောင့် run လိုက်တော့ Total bill for 50 is %f",consumed,total ဆိုပြီး display ပြပေးတာဖြစ်ပါတယ်။ %f လို့ float data type ရဲ့ conversion symbol ကို သုံးတာဖြစ်တဲ့အတွက် total တန်ဖိုးကို display ပြတဲ့အခါ ဒသမကိန်းနဲ့ပြပေးရ တာပါ။ 75.00 ဆိုပြီးပေါ့။ ဒါဆို ဒီ program မှာ if-else if-else if ကို သုံးတာ နားလည်မယ်လို့ ထင်ပါတယ်။ programming sense ရအောင် လေ့လာဖို့ပြောချင်ပါတယ်။ အဆင်ပြေပါစေချူ။
- ◆ ဒီတစ်ခါ user ဆီက ကိန်း ၃လုံးတောင်းပြီး အကြီးဆုံးကိန်းကို output ထုတ်ပေးတဲ့ program လေးရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4      int a,b,c;
5      printf("Enter three numbers");
6      scanf("%d,%d,%d",&a,&b,&c);
7      if(a>b && a<c)
8      {
9          printf("A is the largest");
10     }
11     else if(b>a && b>c)
12     {
13         printf("B is the largest");
14     }
15     else
16     {
17         printf("C is the largest");
18     }
19 }

```

```

Enter three numbers3,4,5
C is the largest
-----
Process exited with return
Press any key to continue

```

- ◆ Program ကို ရှင်းမယ်ဗျာ။ ပထမဆုံး user ဆီက တန်ဖိုး ခုရတောင်းပြီး variable a,b,c မှာ သိမ်းထားမယ်။ သတိပေးမယ်နော်။ တန်ဖိုးတွေ ထည့်တဲ့ချိန်မှာ comma ခံပြီး ရိုက်ရမယ်နော်။ scanf() function မှာ conversion symbol တွေကို comma ခံပြီးရေးထားတာလေ။ အိုကေ....။
- ◆ ဥပမာ user က 3,4,5 လို့ ရိုက်လိုက် တယ်ဆိုပါစို့။ ဒါဆို a က 3, b က 4 နဲ့ c က 5 ဆိုပြီး တန်ဖိုးတွေကို တခုဆီ သိမ်းထားလိုက်ပါ။ ဒီ ကိန်း ခု ခုထဲ က အကြီးဆုံးကိန်းကို ရှာမှာဆိုတော့ ဘယ်ကိန်းက အကြီးဆုံးလဲဆိုတာကို Relational operator နဲ့ Logical operator ကို သုံးပြီး စစ်မယ်။ ကိန်းက ခုခုဆိုတော့ if else if else ကိုသုံးပြီး Condition စစ်လိုက်လို့ မှန်တဲ့ stage က statement ကို output ထုတ်ပေးမှာ ဖြစ်ပါတယ်။
- ◆ ပထမဆုံး if နဲ့ a ဟာ b ထက်ကြီးတယ်၊ a က c ထက်လည်းကြီးတယ်။ ဒီတော့ a က 3 ,b က 4 နဲ့ c က 5 ဆိုတော့ a ကကြီးတယ်ဆိုတာ မှားနေတယ်။ ဒါကြောင့် ပထမဆုံး if block ကို မ run တော့ဘဲ ဒုတိယ else if ကို run မယ်။
- ◆ တကယ်လို့ condition စစ်လိုက်လို့ မှန်တယ်ဆိုရင်တော့ A is the largest ဆိုတဲ့ စာသားကို display ပြပြီး အောက်က statement တွေကို မ run တော့ဘဲ program က ပြီးဆုံးသွားမှာ ဖြစ်ပါတယ်။ ဒါကိုတော့ ကောင်းကောင်းသိမယ်လို့ ထင်ပါတယ်။ ဟုတ်ပြီ။ else if ကို run တော့ if နဲ့ ထပ်စစ်မယ်။ b က a ထက်ကြီးတယ်။ b က c ထက်ကြီးတယ်ဆိုတော့။ b က 4

ဆိုတော့ a ထက်ကြီးတယ်ဆိုတာကမှန်ပါတယ်။ ဒါပေမယ့် c က 5 ဆိုတော့ b က c ထက်ကြီးတယ် ဆိုတာက မှားသွားပြီ။ logical AND ကို သုံးထားတာဖြစ်တဲ့အတွက် expression တစ်ခုက မှားနေတာဖြစ်တဲ့အတွက် ဒီ else if block ထဲက statement တွေကို မ run တော့ဘဲ နောက်ဆုံး else ကို run တော့မယ်။ ဒါကတော့ နောက်ဆုံး condition စစ်တာလေ။ အပေါ်က statement တွေအားလုံး မှားတယ်ဆိုရင်တော့ နောက်ဆုံး else block ထဲက statement ကိုပဲ display ပြပေးမှာဖြစ်ပါတယ်။ ဒါကြောင့် c is the largest ဆိုတဲ့ စာသားကို display ပြခြင်းဖြစ် ပါတယ်။ ဒါဆို ဒီ program ကို နားလည်မယ်လို့ ထင်ပါတယ်။ if else if else ရဲ့ အလုပ်လုပ်ပုံကိုလည်း ကောင်း ကောင်းနားလည်မယ်လို့ ထင်ပါတယ်။ ဒါကို နားလည်ပြီဆိုတော့ Nested if ကို လေ့လာရအောင်။ လွယ်လွယ်လေးပါ။ program လေး တစ်ပုဒ်လောက် ရေးပြမယ်နော်။

```

1  #include<stdio.h>
2  main()
3  {
4  int a,b,c;
5  printf("Enter three numbers");
6  scanf("%d %d %d",&a,&b,&c);
7  printf("Largest of three number:");
8  if(a>b)                // Outer if
9  {
10 if(a>c)                // Inner if
11 {
12 printf("a=%d",a);
13 }
14 else
15 {
16 printf("c=%d",c);
17 }
18 }
19 else                    // Outer else
20 {
21 if(c>b)
22 {
23 printf("c=%d",c);
24 }
25 else
26 {
27 printf("b=%d",b);      } } }
28

```

```

Enter three numbers10 20 30
Largest of three number:c=30
Process exited with return val

```

ဒီ program မှာဆို user ဆီက တန်ဖိုး ၃ ခုတောင်းမယ်။ပြီးရင် a,b,c ထဲမှာ သိမ်းမယ်။ printf("Largest of three number:"); လို့ရေးထားတဲ့အတွက် တန်ဖိုး ၃ခုတောင်းပြီးတာနဲ့ Largest of three number: ဆိုတဲ့ စာသားလေးပေါ်လာမယ်။ဒါပေမယ့် ဘယ်ကိန်းက အကြီးဆုံးလဲဆိုတာ မပြသေးဘူး။ဒါကြောင့် အကြီးဆုံးကိန်း ကို ရှာဖို့ if else ကို သုံးပြီးရှာမယ်။ဒါပေမယ့် neste if else အကြောင်းကို သိစေချင်တဲ့အတွက် if else ကို မသုံးဘဲ nested if else ကို သုံးပြီးရှာပါမယ်။

ပထမဆုံး if နဲ့ a က b ထက်ကြီးရင်လို့ outer if နဲ့ စစ်လိုက်မယ်။ တကယ်လို့ condition စစ်တာမှန်တယ်ဆိုရင် inner if နဲ့ a က c ထက်ကြီးလားလို့ ထပ်စစ်ပါမယ်။ ဥပမာ user က a တန်ဖိုးကို 10 b တန်ဖိုးကို 20 c တန်ဖိုးကို 30 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ outer if နဲ့စစ်တော့ 10 က 20 ထက်ကြီးတယ်ဆိုတော့ ။မှားတယ်။ဒါကြောင့် if block ကို မ run တော့ဘဲ outer else ကို run မှာ ဖြစ်ပါတယ်။ outer else ကို run တော့ else block ထဲက if နဲ့ထပ်စစ်ပါမယ်။ if(c>b) လို့စစ်တော့ 30 က 20 ထက်ကြီးတာဖြစ်တဲ့အတွက် inner if block ထဲက statement ကို output ထုတ်ပေးပါတယ်။ ဒါကြောင့် c=30 လို့ display ပြတာဖြစ်ပါတယ်။တကယ်လို့ c က b ထက်ကြီးတယ်ဆိုတာမှားရင်တော့ inner else ထဲက statement ကို output ထုတ်ပေးမှာဖြစ်ပါတယ်။ b=20 လို့ထုတ်ပေးမှာပါ။ နားလည်မယ်လို့ထင်ပါတယ်။ ပိုမှတ်မိအောင်ပြောရရင် Nested if ကို သုံးရင် ပထမဆုံး outer if နဲ့ စစ်မယ်။မှန်ရင် inner if နဲ့ ထပ်စစ်မယ်။ မှန်ရင် inner if block ထဲက statement ကို output ထုတ်ပေးမယ်။ဒီ program မှာဆို a=10 လို့ ထုတ်ပေး မှာဖြစ်ပါတယ်။ မှားတယ်ဆိုရင်တော့ inner else ထဲက statement ကို ထုတ်ပေးမှာပါ။ c=30 လို့ပေါ့။

ပထမ if နဲ့စစ်လိုက်လို့ မှန်တယ်ဆိုရင်တော့ outer else ထဲက statement ကို မ runတော့ဘဲ outer if ထဲက sstatementတွေကိုပဲ output ထုတ်ပြီး program က ပြီးသွားမှာဖြစ်ပါတယ်။ မှားတယ်ဆိုရင်တော့ outer else ထဲက if နဲ့ ထပ်စစ်ပါမယ်။မှန်ရင်တော့ c=30 လို့ထုတ်ပေးမှာဖြစ်ပါတယ်။ မှားတယ်ဆိုရင်တော့ inner else ထဲက statement ကို run ပါမယ်။ပြီးရင် b=20 ကို အကြီးဆုံးကိန်းအဖြစ် ထုတ်ပေးမှာဖြစ်ပါတယ်။ဒါဆို ကော င်းကောင်းနားလည်မယ်လို့ ထင်ပါတယ်။

ပိုနားလည်အောင် ကိန်း ၃ လုံးတောင်းပြီး အငယ်ဆုံးကိန်းရှာတာကို ရေးပြပါမယ်။


```

1  #include<stdio.h>
2  main()
3  {
4  int a,b,c;
5  printf("Enter three numbers");
6  scanf("%d %d %d",&a,&b,&c);
7  printf("Smallest of three number:");
8  if(a<b) // Outer if
9  {
10 if(a<c) // Inner if
11 {
12 printf("a=%d",a);
13 }
14 else
15 {
16 printf("c=%d",c);
17 }
18 }
19 else // Outer else
20 {
21 if(b<c)
22 {
23 printf("b=%d",b);
24 }
25 else
26 printf("c=%d",c); } }

```

```

Enter three numbers10 20 30
Smallest of three number:a=10
-----
Process exited with return value
Press any key to continue . . .

```

ဒီ program ကို တော့ရှင်းမပြတော့ပါဘူး။ မိမိဘာသာ Trace ကြည့်ပါ။ အဆင်ပြေပါစေ။

5.The break statement

Break သည် C keyword ဖြစ်ပါတယ်။ C keyword တွေကို ရှေ့မှာ ပြောခဲ့ပြီးပြီနော်။ break keyword ကို ဘယ်နေရာမှာ သုံးလဲဆိုတော့ switch statement နဲ့ တွဲပြီး အသုံးများပါတယ်။ break ရဲ့ အလုပ်ကတော့ program ရဲ့ current flow ကို ရပ်တန့်စေဖို့အတွက်ဖြစ်ပါတယ်။ ဆိုလိုတာကတော့ ဘာမှ ထမင်းစား ပြီးပြီဆိုတဲ့ statement နောက်မှာ break keyword ကို ရေးလိုက်မယ်ဆိုရင် program ကရပ်တန့်သွားပါမယ်။ အခြား ရေသောက်မယ် ဘာညာ ရေးထားတဲ့ instruction တွေကို မ run တော့ဘူး။ break ဆိုပြီး program ကို အဆုံးသတ်ပေးတာဖြစ်ပါတယ်။ break ရဲ့ အသုံးပြုပုံကို switch statement ခန်းရောက်မှ ထပ်ရှင်းပြပေးပါမယ်။

6. The continue statement

Continue ကလည်း C keyword ဖြစ်ပါတယ်။ သူကတော့ break နဲ့ ပြောင်းပြန်ပါ။ သူကတော့ statement တစ်ခု ကို အဆုံး မသတ်သေးဘဲ continue ရဲ့သဘောတရားအတိုင်း ဆက်ပြီးတော့ အခြား statement တွေကို run စေလိုသေးတဲ့ အခါမှာ သုံးပါတယ်။ ဒါကိုလည်း looping ခန်းမှာ ရှင်းပြ ပေးပါမယ်။

7. The goto statement


goto statement ဟာ C program မှာ အာဏာရှိတဲ့သူ တစ်ယောက်နဲ့ တူပါတယ်။ ဘာလို့လည်းဆိုတော့ သူက program ရဲ့ ကြိုက်တဲ့ နေရာကို condition တွေစစ်စရာ မလိုဘဲ။ သွားလို့ရပါတယ်။ အဲလို သွားမယ်ဆိုရင် ဒီအတိုင်းသွားလို့တော့မရပါဘူး။ သွားမယ်နေရာကိုတော့ goto ရဲ့ နောက်မှာရေးပေးရမှာပါ။ သူ့ရဲ့ syntax ကို ကြည့်ရအောင်။

goto label; label ဆိုတာကတော့ goto သွားမယ့်နေရာရဲ့ အမည်ကို ရေးပေးတာဖြစ်ပါတယ်။ label နေရာ မှာ မိမိသွားမယ့်နေရာရဲ့ အမည်ကို ရေးပြီးရင်တော့ program မှာ ကျော်လွှားသွားလို့ ရပါပြီ။ ပိုရှင်းသွားအောင် program လေးတပုဒ်လောက်ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4      int x;
5      printf("Enter number");
6      scanf("%d",&x);
7      if(x%2==0)
8      {
9          goto even;
10     }
11     else
12     {
13         goto odd;
14     }
15     even:
16     printf(" %d is Even",x);
17     return;
18     odd:
19     printf("%d is odd",x);
20 }

```



```

Enter number 3
3 is odd
Process exited with

```

- ◆ ဒီ program ကိုရှင်းပြီးရင်တော့ goto statement ကိုကောင်းကောင်း နားလည်မယ်လို့ ထင်ပါတယ်။ ပထမဆုံး User ဆီကနေ Enter number လို့ ကိန်းတစ်ခုကို တောင်းမယ့် စာသားလေးပြဖို့ ရေးမယ်။ ဒါတွေကိုတော့ ရှင်းနေ စရာမလိုတော့ဘူးထင်ပါတယ်။ ဥပမာ user က 3 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါကို အပေါင်းလားအနှုတ်လား စစ်ဖို့ if နဲ့ condition စစ်ဖို့လိုပါတယ်။
- ◆ ဒါကြောင့် if(x%2==0) လို့ရေးပေးရတာဖြစ်ပါတယ်။ ဆိုလိုတာကတော့ ဗျာ 2 နဲ့စားလို့ အကြွင်း က 0 နဲ့ညီတယ်ဆိုရင် user ရိုက်လိုက်တဲ့ input value က စုံကိန်းဖြစ်မယ်။ အကြွင်း 0 နဲ့ မညီရင် မ ကိန်းဖြစ်မယ်။ ဒီသဘောတရားကိုတော့ သိမယ်လို့ ထင်ပါတယ်။ စုံကိန်း အားလုံးကို 2 နဲ့စားရင် အကြွင်း က 0 ပဲရတယ်လေ။ အိုကေ။ အခု user က 3 လို့ရိုက်လိုက်တာဖြစ်တဲ့အတွက် 2 နဲ့ စားတော့ မပြတ်ဘူး။ ဒါကြောင့် if block ထဲက statement ကို မ run တော့ဘဲ else block ထဲက statement ကို run ပြီး output ထုတ်ပေးမယ်။ မှန်တယ်ဆိုရင်တော့ if block ထဲက goto statement ကို run ပြီး output ထုတ်ပေးမှာဖြစ်ပါတယ်။
- ◆ ခု မှားတဲ့အတွက် else block ထဲက goto statement ကို run မှာဖြစ်ပါတယ်။ goto odd; ဆိုတဲ့ statement ကို ဘယ်လို run ပြီး 3 is odd ဆိုတဲ့ စာသားကို display လုပ်ပေးလဲဆိုတာပြောပြပါမယ်။ အထက်မှာ ပြောပြပြီးသားနော်။ goto နောက်က မိမိသွားလိုတဲ့ statement မှာ သတ်မှတ်ထားတဲ့ အမည်ကို ရေးပေးရတယ်ဆိုတာ goto odd ဆိုတော့ odd ဆိုတဲ့ အမည်သတ်မှတ်ထားတဲ့ statement ဆီကို သွားပြီး run ပါမယ်။ program မှာ မြှားပြထားတာကို တွေ့တယ်ဟုတ်။ အမည်ပေးမယ်ဆိုရင် ဒီအတိုင်းပေးလို့ မရဘူးဗျ။ အမည်ပေးတဲ့ syntax လေးကိုမှတ်ထားပါ။ ဒီ program မှာ odd လို့ရေးပြီး နောက်က is to(:) လေးရေးပေး တာကို သတိထားမိတယ်ဟုတ်။ အဲလိုမျိုး သတ်မှတ်ပေး ရမှာပါ။ odd: လို့ အမည်ပေးထားတဲ့ statement က ဘာလုပ်ခိုင်းလဲဆိုတော့ %d is odd ဆိုပြီး user ရိုက်လိုက်တဲ့ value က မကိန်းဖြစ်ပါတယ်လို့ display လုပ်ခိုင်း တာပါ။ ဒါကြောင့် 3 is odd လို့ display ပြခြင်းဖြစ်ပါတယ်။ goto even ကလည်းဒီအတိုင်းပဲ ဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

goto ကို ပိုနားလည်သွားအောင် user ဆီကနေ ခုနှစ်တစ်ခုကို ရိုက်ထည့်ခိုင်းမယ်။ ပြီးရင် ဒီခုနှစ်က ရက်ထပ်နှစ် ဟုတ်မဟုတ်စစ်ဆေးပြီး ဟုတ်ရင်ဟုတ်ကြောင်း၊ မဟုတ်ရင်မဟုတ်ကြောင်း ပြပေးမယ့် program လေးရေးကြည့် ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4      int leap;
5      printf("Enter year");
6      scanf("%d",&leap);
7      if(leap%4==0)
8      {
9          goto leap;
10     }
11     else
12     {
13         goto noleap;
14     }
15     leap:
16     printf("%d is leap year",leap);
17     return 0;
18     noleap:
19     printf("%d is a not a leap year",leap);
20 }
21

```

```

Enter year 2000
2000 is leap year
-----

```

ဒီ program ကိုတော့ ရှင်းမပြောဘူးပါဘူး။ ရှေ့က program နဲ့ အတူတူပဲဆိုတော့။ 4 နဲ့ စားလို့ အကြွင်း 0 နဲ့ ညီတယ်ဆိုရင် ရက်ထပ်နှစ်ဖြစ်မယ်။ မညီရင်တော့ ရက်ထပ်နှစ်မဟုတ်ဘူးဆိုတဲ့ စာသားလေးကို display ပြပေးမယ်။ ဘာလို့ 4 နဲ့စားလဲဆိုတော့ 4 နှစ်ပြည့်ပြီးလာတဲ့ နှစ်က ရက်ထပ်နှစ်ဖြစ်လို့စားပေးတာဖြစ်ပါတယ်။ မိမိဘာသာ trace ကြည့်ပါ။ ဒါမှ programming flow ကိုနားလည်ပြီး programming sense ရမှာဖြစ်ပါတယ်။ အဆင်ပြေပါစေ။ ဒါဆိုရင် goto statement ကို နားလည်ပြီလို့ထင်ပါတယ်။.....။

6. The switch statement

Switch statement ကို ဘယ်လိုနေရာတွေမှာ သုံးလဲဆိုတာကို ရှင်းပြပါမယ်။ အရှင်းဆုံး ဥပမာ ပြရရင် smart home control တစ်ခုလို့ပဲ အလုပ်လုပ်ပါတယ်။ မီးဖွင့်ချင်ရင် 1 ကို နှိပ်မယ်။ ရေဖွင့်ချင်ရင် 2 ကို နှိပ်မယ်။ TV ဖွင့် မယ်ဆို 3 ကို နှိပ်မယ်ပေါ့ဗျာ။ အဲလို option တွေကိုလုပ်ဖို့အတွက် user ဆီကနေ တန်ဖိုးတစ်ခုတောင်းမယ်။ integer တန်ဖိုးတခုပေါ့ဗျာ။ ပြီးရင် variable name တခုပေးပြီး သိမ်းမယ်ပေါ့။ အဲ့ဒီ variable name ကို switch() ရဲ့ parenthesis () လက်သည်းကွင်းထဲမှာထည့်ပြီး ဒီ variable name ထဲမှာရှိတဲ့ တန်ဖိုးက case constant: ထဲက ဘယ်

constant နဲ့ တူလဲစစ်မယ်။ တူတဲ့ case constant မှာသတ်မှတ်ထားတဲ့ statement ကို အလုပ်လုပ်မယ်။ ပိုနားလည်အောင် သူ့ရဲ့ syntax လေးကို အရင်ပြောပြပါမယ်။

```
switch(variable or expression)
```

```
{
```

```
case constant A:
```

```
Statement;
```

```
break;
```

```
case constant B:`1
```

```
statement;
```

```
break;
```

```
default:
```

```
Statement;
```

```
}
```

ပိုနားလည်အောင် program လေးရေးပြီးလေ့လာရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int ch;
5  printf("Enter your choice between 1 to 3:");
6  scanf("%d",&ch);
7  switch(ch)
8  {
9  case 1:
0  printf("Your enter 1");
1  break;
2  case 2:
3  printf("Your enter 2");
4  break;
5  case 3:
6  printf("You enter 3");
7  break;
8  default:
9  printf("Invalid chioce");
0  }
1

```

```

Enter your choice between 1 to 3:2
Your enter 2
-----
Process exited with return value 12

```

(or)

```
Enter your choice between 1 to 3:4
Invalid choice
-----
Process exited with return value 14
```

- ◆ ဒီ program မှာဆိုရင် case constant options တွေထဲက ဘယ် statement ကို display ပြချင်တာလဲ။ option 3 ခု ထဲက user display ပြချင်တဲ့ case constant ရဲ့ number ကို ရိုက်ခိုင်းပါမယ်။ case constant က 1,2 and 3 ဆိုတော့ 1 to 3 ထဲက တခုကို ရိုက်ပေးရပါမယ်။ တကယ်လို့ 1 to 4 ထဲက number ကို ရိုက်ချင်တယ်ဆိုရင်တော့ case statement option ၄ ခုရှိရပါမယ်။ အိုကေ...။ ရှင်းရအောင်။
 - ◆ user က 1 to 3 ထဲက 2 ကို ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒီတော့ 2 ကို variable ch ထဲမှာ သိမ်းထားလိုက်ပြီ။ ခု switch(ch) လို့ရေးပြီး user display ပြချင်တဲ့ option ကို ရွေးဖို့ switch ရဲ့ parenthesis ထဲမှာရှိတဲ့ variable ရဲ့ number နဲ့တိုက်စစ်မယ်။ number က 2 ဆိုတော့ case 2: option ကို display ပြပေးတာဖြစ်ပါတယ်။ ဒါကြောင့် You enter 2 ဆိုတဲ့ စာသားကို display ပြခြင်းဖြစ်ပါတယ်။ number 3 ကို ရိုက်မယ်ဆိုရင်လည်း case 3: option က statement ကို display ပြပေးမှာ ဖြစ်ပါတယ်။ ဒါကို နားလည်မယ်နော်။
 - ◆ ဥပမာ user က 4 လို့ ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆို switch (ch) ထဲက variable 4 နဲ့ case constant option နဲ့ တိုက်စစ်မယ်။ case constatement မှာ 4 အတွက် display ပြချင်တဲ့ statement မရှိဘူးလေ။ ဒါကြောင့် case 1:,case 2:,case 3: options ၃ ခုထဲက ဘယ် constant နဲ့မှ မတူတဲ့အတွက် default statement ကိုပဲ display ပြပေးမယ် ဖြစ်ပါတယ်။ Invalid choice ဆိုပြီးပေါ့။ တကယ်လို့ 6 လို့ ရိုက်မယ်ဆိုလည်း case 6: ဆိုပြီး display ပြဖို့ statement မရှိတဲ့ အတွက် default statement ကိုပဲ display ပြပေးမှာဖြစ်ပါတယ်။ switch () ရဲ့ အလုပ်လုပ်ပုံကို နားလည်မယ်လို့ ထင်ပါတယ်။
- ဒီတစ်ခါ calculator program လေး တစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။

```
#include<stdio.h>
main()
{
    int ch,a,b;
    printf("1]Addition\n");
    printf("2]Subtraction\n");
    printf("3]Multiplication\n");
    printf("4]Division\n");
    printf("Enter Your choice");
    scanf("%d",&ch);
    switch(ch)
    {
        case 1:
            printf("1]Addition\n");
            printf("Enter two numbers");
            scanf("%d %d",&a,&b);
            printf("\n Addition of a and b=%d",a+b);
            break;
```

```

case 2:
printf("2]Subtraction\n");
printf("Enter two numbers");
scanf("%d %d",&a,&b);
printf("\n Subtraction of a and b=%d",a-b);
break;
case 3:
printf("3]Multiplication\n");
printf("Enter two numbers");
scanf("%d %d",&a,&b);
printf("\n Multiplication of a and b=%d",a*b);
break;
case 4:
printf("4] Division\n");
printf("Enter two numbers");
scanf("%d %d",&a,&b);
printf("\n Division of a and b=%d",a/b);
break;
default:
printf("Exit"); }

```

```

1]Addition
2]Subtraction
3]Multiplication
4]Division
Enter Your choice2
2]Subtraction
Enter two numbers3 6

Subtraction of a and b=-3

```

- ◆ ဒီတစ်ခါ ရေးထားတဲ့ program ကို မရှင်းဘဲ။ output ကို ကြည့်ပြီး ကိုယ်တိုင်ဘယ်လိုရေးမလဲဆိုတာ စဉ်းစားတတ်အောင် လေ့ကျင့်တဲ့ အနေနဲ့ ရှင်းပြပေးပါမယ်။ ဒီ program လေးရဲ့ output လေးကို ကြည့်ပါ။ Menu title လေးတွေနဲ့ပြထားတာ။ Menu 1 to 4 ထိ။ အဲလိုမျိုး display ပြဖို့ဘယ်လိုရေးမလဲ။ လွယ်လွယ်လေးပေါ့။ printf() function သုံးပြီးရေးပေးရုံပဲ။ program မှာ ရေးထားတဲ့အတိုင်း။ လွယ်တယ်ဟုတ်။
- ◆ ပြီးရင် user ကို ဒီ Menu တွေထဲက ဘယ် title ကိုရွေးမလဲဆိုပြီး Enter Your choice ဆိုတဲ့ စာတန်းလေးပေါ်အောင် ရေးမယ်။ program မှာပြထားတဲ့အတိုင်း printf() function လေးသုံးပြီးရေးမယ်လေ။ အိုကေ။ ပြီးရင် user ရိုက်လိုက်တဲ့ တန်ဖိုးကို memory မှာ လက်ခံဖို့၊ သိမ်းဆည်းဖို့အတွက် scanf() function ကို ရေးမယ်။ ဒီအထိ ရေးတတ်မယ်ဟုတ်။ တန်ဖိုးကို တွေကို သိမ်းဆည်းဖို့အတွက် variable name ကိုတော့ အစမှာ ကြေငြာပေးထားရမယ်နော်။ program မှာဆိုရင် ch လို့ပေးထားတယ်။
- ◆ ဥပမာ user က 2 လို့ choice လိုက်တယ်ဆိုပါစို့။ ဒါဆိုရင် title 2 ရဲ့ Subtraction statement တွေကို အလုပ်လုပ်မယ်။ အဲလိုမျိုး 1 လို့ ရိုက်ရင် 1 နဲ့ ဆိုင်တဲ့ Addition တွေကို လုပ် ။အဲလိုမျိုး Calculator တစ်လုံးရဲ့ features တွေနဲ့ တူအောင် ရေးဖို့အတွက် ဘာကိုသုံးမလဲ။

switch() statement ကို သုံးပြီးရေးရင် ပိုအဆင်ပြေတယ်။ ဒါကြောင့် program မှာ switch() statement ကို သုံးပြီးရေးပြ ထားပါတယ်။ နားလည်မယ် နော်။ ဒီတော့ ခု user ရိုက်လိုက်တဲ့ တန်ဖိုး 2 ကို switch() ရဲ့ parenthesis ထဲမှာ ထည့်လိုက်ပြီ။ ပြီးရင် အဲ့တန်ဖိုးနဲ့ တူညီတဲ့ case constant ကို တိုက်ကြည့်မယ်။

- ◆ ဒါကြောင့် case 2: ဆိုတဲ့ option ကိုရေးပေးမယ်။ title အရ ကိန်းနှစ်ခုကို နှုတ်ချင်တာ ဖြစ်တဲ့အတွက် subtraction statement ကို ရေးပေးရမှာပေါ့။ ကိန်းနှစ်ခုကို နှုတ်မှာဆိုတော့ user ဆီက အနှုတ်ခိုင်းစေချင်တဲ့ တန်ဖိုးနှစ်ခုကို တောင်းမယ်။ ဒါကြောင့် program မှာ Enter two numbers ဆိုပြီး user ကို တန်ဖိုးနှစ်ခု တောင်းမယ့် စာတန်းလေး display လုပ်ခိုင်းတာ ဖြစ်ပါတယ်။ ဒီလိုတောင်းမယ့် စာတန်းကို တော့ ရေးတတ်မယ်နော်။ တန်ဖိုးတွေကို သိမ်းမယ့် variable name တွေကိုတော့ program အစမှာပဲကြေငြာ ပေးခဲ့ရမယ်နော်။ ပြီးရင် user ရိုက်လိုက်တဲ့ တန်ဖိုးကို လက်ခံမယ်။ သိမ်းဆည်းမယ်။ သိမ်းဆည်းဖို့အတွက် နေရာသတ်မှတ် ပေးရမယ်။ ဒါကြောင့် scanf() function ကို ရေးပေးရမယ်။ အိုကေ...။
- ◆ ဒီ program မှာဆို တန်ဖိုးနှစ်ခုကို a နဲ့ b ထဲမှာ သိမ်းထားလိုက်ပြီ။ တန်ဖိုးနှစ်ခုကို နှုတ်စေချင်တာ ဖြစ်တဲ့အတွက် a-b ဆိုပြီး နှုတ်ပေးလိုက်မယ်။ နှုတ်လို့ရတဲ့ တန်ဖိုးကို display ပြခိုင်းမယ်။ ဒီတော့ printf() function ကိုပဲ သုံးမယ်။ program မှာ ကြည့်ပါ။ printf("\n Subtraction of a and b=%d",a-b); လို့ရေး ပေးလိုက်ရုံပဲ။ ဒါဆို Subtraction of a and b =1 ဆိုပြီး display ပြပါမယ်။အိုကေ....။ဒါဆို မိမိ ကိုယ်တိုင် စဉ်းစားပြီးရေးကြည့်လို့ အဆင်ပြေမယ်ထင်ပါ တယ်။.....။

ဒီ တစ်ခါ switch() statement ကိုပဲသုံးပြီး နှစ်ကို minute,hour,day,month,second ပြောင်းတဲ့ program လေးရေးကြည့်ရအောင်။ လွယ်လွယ်လေးပါ။ program တပုဒ်ရေးတော့မယ်ဆို အဓိက က logic သိဖို့ပဲ။ ခု year ကို days တွေ month တွေပြောင်းမယ်ဆိုတော့ logic ကိုစဉ်းစားရမယ်။ ဒီတော့ တစ်နှစ်မှာ ဘယ်နှစ်လ ရှိလဲ။ ၁၂ လ ရှိတယ်။ ဒီတော့ month ကို သိချင်ရင် year ကို 12 နှစ်နဲ့မြှောက်မယ်။ 2 year ဆိုရင် 12 နှစ်နဲ့မြှောက်တော့ 24 month။ အိုကေ....။ month ကို သိပြီဆိုရင် day ကိုပြောင်းလို့ရပြီ။ တစ်လမှာ ဘယ်နှစ်ရက်လဲ။ ၃၆၅ ရက်။ ဒီ တော့။ အရင်ဆုံး လကို 30 နဲ့မြှောက်။ ဒါဆို 360 ရက်ရမယ်။ 365 ရစေချင်တာဖြစ်တဲ့အတွက် 360 ကို year*5 နဲ့ပေါင်းမယ်။ ဒါဆိုရင် တစ်နှစ်မှာ ရက်ပေါင်း 365 ရက်ကိုတွက်လို့ရပြီ။ ပြီးရင် hour ကိုပြောင်းမယ်။ တစ်နေ့မှာ နာရီ ပေါင်း ဘယ်လောက်ရှိလဲ။ 24 နာရီရှိတယ်။ ဒီတော့ 365 ရက်ကို 24 နဲ့မြှောက်လိုက်။ ဒါဆို တနှစ်မှာ နာရီပေါင်း 8760 နာရီကို ရမယ်။ ပြီးရင် minuteကိုပြောင်းမယ်။ တစ်နာရီမှာ minute 60 ရှိတယ်ဆိုတော့ 60 ကို 8760 နဲ့ မြှောက်လိုက်။ ဒါဆို တစ်နှစ်မှာ မိနစ်ပေါင်း 525600 ရမယ်။ အိုကေ...။ ပြီးရင် second ကိုပြောင်းမယ် တစ်မိနစ်မှာ second 60 ရှိတယ်။ ဒါကြောင့် မိနစ် 525600 ကို 60 နဲ့မြှောက်လိုက်။ ဒါဆိုရင် တနှစ်မှာ second ပေါင်း 3153600 second ရမယ်။ ဒီတော့ user က 3 နှစ်ဆို လ၊ ဂုတ်၊ မိနစ် ပေါင်းဘယ်လောက်လဲ သိချင်ရင် year နေရာမှာ 3 နဲ့မြှောက်မယ်။ဒါဆို 3 နှစ်အတွက် ရက်၊ လ ပေါင်းကိုရမယ်။ အိုကေ။ ဒီတော့ program လေးရေးရ အောင်။


```
1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5      long ch,min,hrs,ds,mon,yrs,sec;
6      printf("\n 1]MINUTES");
7      printf("\n 2]HOURS");
8      printf("\n 3]DAYS");
9      printf("\n 4]MONTHS");
10     printf("\n 5]SECONDS");
11     printf("\n 0]EXIT");
12     printf("Enter your choice");
13     scanf("%ld",&ch);
14     if(ch>=0 && ch<6)
15     {
16         printf("Enter years");
17         scanf("%ld",&yrs);
18     }
19     mon=yrs*12;
20     ds=mon*30;
21     ds=ds+yrs*5;
22     hrs=ds*24;
23     min=hrs*60;
24     sec=min*60;
25     switch(ch)
26     {
27         case 1:
28             printf("\n Minutes=%ld",min);
29             break;
30         case 2:
31             printf("\n Hours=%ld",hrs);
32             break;
33         case 3:
34             printf("\n Days=%ld",ds);
35             break;
36         case 4:
37             printf("\n Months=%ld",mon);
38             break;
39         case 5:
40             printf("\n Seconds=%ld",sec);
41             break;
42         case 0:
43             printf("\nTerminated by choice");
44             break;
45         default:
46             printf("\n Invalid number");
47     }
48 }
```

```

1 MINUTES
2 HOURS
3 DAYS
4 MONTHS
5 SECONDS
0 EXIT Enter your choice4
Enter years 2

Months=24

```

ဒီ program ရဲ့ အသေးစိတ်ကိုတော့ ပြောပြပြီးနော်။ ပထမဆုံး နှစ်၊ ရက်၊ လ စတာတွေကို display မပြခင် သိမ်းဆည်းဖို့အတွက် variable name တွေကို ကြေငြာပေးရမယ်။ အဲလို ကြေငြာတဲ့အခါမှာ data type ကိုတော့ long လို့ကြေငြာပေးရမယ်။ ဘာလို့လည်းဆိုတော့ သူ့ပန်နဲ့ သူ့ဆိုဒ် တိကျမှန်ရှိအောင်လို့ပါ။ ပြီးရင် user ရိုက်လိုက်တဲ့ နှစ်ရဲ့ လ၊ရက် စတာတွေကို ရွေးချယ်ဖို့ အတွက် Menu လေးရေးပေးပါမယ်။ program မှာရေးထား တဲ့အတိုင်း 1 to 5 နဲ့ Exit ထွက်မယ် ဆိုတဲ့ရွေးချယ်တော့ဘူးဆိုပြီး title တွေကို display ပြခိုင်းမယ်။ program မှာ if statement နဲ့ condition စစ်ထားတာကို ပြောပြပါမယ်။ user က 0 to 5 ထဲက ကိန်းတခုကို ရိုက်မယ်ဆိုမှ Enter years ဆိုပြီး နှစ်ကို ရိုက်ဖို့ စာသားလေးပြပေးမယ်။ ပြီးရင် သက်ဆိုင်ရာ case constant ရဲ့ statement တွေကို အလုပ်လုပ်ခိုင်းမယ်။ 0 to 5 မဟုတ်ဘဲ အခြားကိန်းတွေ ရိုက်လိုက်မယ် ဆိုရင်တော့ program က ပြီးဆုံးသွားမှာပါ။ ဘာလို့လည်းဆိုတော့ program မှာ 0 to 5 အတွက်ဘဲ case constant ရှိလို့ပါ။ နားလည်တယ်ဟုတ်။

ဥပမာ user က နှစ်တစ်နှစ်ရဲ့ လပေါင်းကို သိချင်တယ်ဆိုပါစို့။ ဒါဆို Menu မှာကြည့်ပြီး 4 လို့ ရိုက်ပေးရပါမယ်။ ဒီ choice လိုက်တဲ့ တန်ဖိုး 4 ကို ch ထဲမှာ သိမ်းထားလိုက်ပြီ။ ဒီတော့ if statement နဲ့ စစ်တာမှန်တာ ဖြစ်တဲ့အတွက် Enter years နှစ်ဘယ်ခုရဲ့ လကို သိချင်တာလဲဆိုပြီး သိချင်တဲ့ နှစ်ပေါင်းကို ရိုက်ပေးရမယ်။ ဥပမာ user က ၂နှစ်မှာ လပေါင်းဘယ်လောက်လဲ သိချင်တယ်ဆိုပါစို့။ ဒါဆို 2 လို့ ရိုက်လိုက်မယ်။ အဲ့ဒီ တန်ဖိုးကိုတော့ yrs variable ထဲမှာ သိမ်းထားလိုက်ပြီ။ ပြီးရင် ရှေ့မှာပြောခဲ့တဲ့အတိုင်း year ကို month,day,hour ပြောင်းမယ့် ဆက်သွယ်ချက်လေးတွေ ရေးပေးထားရပါမယ်။ program မှာပြထားတဲ့ အတိုင်းပါပဲ။ $mon = yrs * 12$; ဆိုတဲ့ statement လေးရေးပေးထားတာပါ။ အဲ့တော့ $mon = 2 * 12$ ဖြစ်သွားပြီပေါ့။ ဒီတော့ $mon = 24$ ဖြစ်သွားပြီ။ 24 ကို variable mon ထဲမှာ သိမ်းထားလိုက်ပြီ။

user က 4 လို့ ရိုက်လိုက်တာ ဖြစ်တဲ့အတွက် switch(ch) ထဲမှာ 4 ကို လက်ခံလိုက်ပြီ။ ပြီးရင် case constant နဲ့ တိုက်စစ်ကြည့်မယ်။ 4 ဖြစ်တဲ့အတွက် case 4: ရဲ့ statement ကို run ခိုင်းပါမယ်။ `printf("\n Months=%ld",mon);` ဆိုတဲ့ statement ကို run မယ်။ ဒါကြောင့် Month=24 ဆိုပြီး display ပြလိုက်တာဖြစ်ပါတယ်။ mon က 24 လေ။ အပေါ်မှာ တွက်ချက်လို့ရတဲ့ တန်ဖိုး။ နားလည် မယ်နော်။ ရှေ့က program မှာ မရှင်းပြခဲ့မိဘူး။ break; ဆိုတဲ့ statement ကို ဘာကြောင့်ရေးပေးလဲ ဆိုတာ။ အိုကေ...ပြောပြပါမယ်။ break ရဲ့ အဓိပ္ပာယ်အတိုင်းပါပဲ။ Month=24 လို့ display ပြပြီးတာနဲ့ program ကို ရပ်ခိုင်းချင်တဲ့အတွက်ပါ။ ဒါကြောင့် အောက်က statement တွေကို မ run တော့ဘဲ program ကပြီးဆုံး သွားမှာပါ။ တကယ်လို့ 3 လို့ ရိုက်လိုက်မယ်ဆိုပါစို့။ ဒါဆိုရင် 2 year မှာရှိတဲ့ နာရီပေါင်းကို ပြပေးမှာ ဖြစ်ပါတယ်။ 2 year မှာရှိတဲ့ နာရီပေါင်းကိုတော့ $hrs = ds * 24$; ဆိုတဲ့

statement ကနေ တွက်ချက်ပြီး အဖြေထုတ် ပေးမှာဖြစ်ပါတယ်။ နားလည်မယ် ထင်ပါတယ်။ တကယ်လို့ 0 ကို ရိုက်မယ်ဆိုရင်တော့ case 0 ကို run ပေးမှာပါ။ ပြီးရင် Terminate by choice ဆိုတဲ့ စာသားလေးပေါ်ပြီး program ကို break လုပ်ခိုင်းတာဖြစ်တဲ့အတွက် program ပြီးဆုံးသွားပါမယ်။ ဒီ program က အရမ်းရှည်ပေမယ့် သူ့သဘောတရားကို နားလည်မယ်ဆိုရင်တော့ အရမ်း လွယ်ကူမယ်လို့ ထင်ပါတယ်။.....။

ဒီတစ်ခါ switch() statement ကိုပဲသုံးပြီး user input ထည့်လိုက်တဲ့ value ဟာ စုံကိန်းလား၊ မကိန်းလား ဆိုတာ display ပြပေးမယ့် program လေးတစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။

```
#include<stdio.h>
#include<conio.h>
main()
{
    int x,y;
    printf("\nEnter a number");
    scanf("%d",&x);
    switch(x)
    {
        case 0:
            printf("\n Number is Even");
            break;
        case 1:
            printf("\n Number is Odd");
            break;
        default:
            y=x%2;
            switch(y)
            {
                case 0:
                    printf("Number is even");
                    break;
                default:
                    printf("Number is odd");
            } } }
}
```

```
Enter a number5
Number is odd
-----
Process exited with re
```

ဒီ program မှာဆို ရှင်းပြစရာတော့ သိပ်မရှိပါဘူး။ ရေးထားတာလေးတွေ ဘယ်လို အလုပ် လုပ်သွားလဲ ကြည့်ရအောင်။ ပထမ user ဆီက input value တစ်ခုတောင်းမယ်။ ဥပမာ user က 5

လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆို 5 ကို variable x ထဲမှာ သိမ်းထားလိုက်ပြီ။ user က input value ကို နှစ်သက်ရာ ကိန်းတစ်ခုကို ရိုက်နိုင် တယ်လေ။ ဒီတော့ 0 လို့ရိုက်တယ်ဆိုရင်တော့ စုံကိန်းဖြစ်တယ်လို့ display ပြမယ်။ 1 ကို ရိုက်တယ်ဆိုရင်တော့ မကိန်းလို့ display ပြခိုင်းပါမယ်။ ဘာလို့လည်း ဆိုတော့ 0 နဲ့ 1 ကလွဲလို့ ကျန်တဲ့ ကိန်းတွေက 2 နဲ့ စားလို့ အကြွင်း 0 ရရင် စုံကိန်းတွေ ဖြစ်မယ်။ 1 ကြွင်းတယ်ဆိုရင်တော့ မကိန်းဆိုတာသိနိုင်တယ်လေ။ ဒါကြောင့် ပထမ switch() ရဲ့ case constant မှာ 0 နဲ့ 1 ကို ရိုက်တယ်ဆိုရင် display ပြဖို့ရေးတာဖြစ်ပါတယ်။ ခု user က 5 လို့ ရိုက်လိုက်တာဖြစ်တဲ့အတွက် outer switch(x) ထဲက x တန်ဖိုး 5 ကို case constant နဲ့ တိုက်စစ်မယ်။ program မှာ case constant 5 အတွက် ရေးပေးထားတာမရှိဘူးလေ။ ဒီတော့ case constant နဲ့ တိုက်စစ် လို့ မတွေ့တာဖြစ်တဲ့ အတွက် default statement ကို run မယ်။ default statement မှာ $y = x \% 2$ လို့ ရေးထားတာကို ရှင်းပြပါမယ်။ user ရိုက်လိုက်တဲ့ value ကို 2 နဲ့စားမယ်။ စားလို့ရတဲ့ အကြွင်း ကို y နဲ့ assign လုပ်တာဖြစ်တဲ့ အတွက် y ထဲမှာ သိမ်းထားမယ်။ ပြီးရင် inner switch(y) နဲ့ ထပ်စစ်မယ်။ 5 ကို 2 နဲ့ စားတော့ အကြွင်း က 1 လေ။ ဒီတော့ switch(1) ရဲ့ case constant နဲ့တိုက်စစ်မယ်။ program မှာရေးထားတာတွေ တယ်ဟုတ်။ case 0: နဲ့ default: ။ case constant 1 အတွက်ရေးထားတဲ့ statement တော့မရှိဘူး။ case 0: အတွက်ပဲ ရှိတယ်။ ဘယ်ကိန်းမဆို 2 နဲ့ စားရင် အကြွင်း က 0 မဟုတ်ရင် ၁ ပဲဖြစ်မယ်လေ။ ဒီတော့ 0 မဟုတ်တဲ့ အတွက် default statement ကို ပဲ run မယ်။ ဒါကြောင့် Number is odd ဆိုပြီး display ပြတာဖြစ်ပါတယ်။ တကယ်လို့ အကြွင်း က 0 ဆိုရင်တော့ case 0: ရဲ့ statement ကိုပဲ display ပြမှာ ဖြစ်ပါတယ်။ ပြီးရင်တော့ break ဆိုပြီး program ကို အဆုံးသတ်သွားမှာဖြစ်ပါတယ်။ switch() နဲ့ break statement က တွဲပြီးတော့ အသုံးပြုရပါတယ်။ ဒါဆို ဒီ program မှာ နားမလည်တာ မရှိဘူးလို့ ထင်ပါတယ်။

6. Loop control Statements

1. Introduction

Loop control statement ကို ဘာကြောင့်အသုံးပြုရလဲ။ program မှာ ဘယ်လို အလုပ်လုပ်ပေးလဲ ဆိုတာကို ရှင်းပြပါမယ်။ loop ပတ်တယ်ဆိုတာက current program မှာ value တွေကို တိုးမယ်၊ လျှော့မယ်၊ တန်ဖိုးတွေ ပြောင်းလဲပြီး တွက်ချက်တဲ့အခါမှာ ထပ်ခါထပ်ခါ assign လုပ်စရာ မလိုဘဲ statement တစ်ခုထဲကိုပဲ loop ပတ်ပြီး အတိုးအလျှော့လုပ်တာကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။ program တွေရေးပြီး ရှင်းပြတဲ့အခါမှာ ပိုပြီးနားလည်လာပါ လိမ့်မယ်။ Loop control statement ကို အသုံးပြုရာမှာ နည်းလမ်း ၃မျိုးနဲ့ အသုံးပြုပြီး ရေးလို့ရပါတယ်။ အောက်မှာ ဖော်ပြ ထားတဲ့ နည်းတွေရဲ့ syntax ကိုသုံးပြီး ရေးလို့ရပါတယ်။

- 1) The for loop
- 2) The while loop
- 3) The do while loop

1) The for loop

ပထမဆုံး The for loop statement ရဲ့ syntax ကိုပြောပြပါမယ်။

```

for(start;stop;step)
{
Statement;
}

```

for loop statement ရဲ့ parenthesis () ထဲမှာ start;stop;step တွေကို ရေးပေးရပါမယ်။ start ဆိုတာ ဘာကိုဆိုလိုတာလဲဆိုတာ ပြောပြပါမယ်။ ဥပမာ user က 1 ကနေ 5 အထိ ကိန်းတွေကိုပေါင်း မယ်ဆိုပါစို့။ ဒါဆိုရင် ၁ ကနေစမယ်။ အဲလို condition တစ်ခုကို မိမိစချင်တဲ့ နေရာကစပြီး တွက်ချက်တဲ့ အခါမှာ ရေးရတဲ့ expression ကို start လို့ခေါ်ပါတယ်။ stop ဆိုတာကတော့ အစရှိရင် အဆုံးလည်း ရှိရမယ်လေ။ ဒါကြောင့် ၁ ကနေ စပြီး ၅ မှာ ဆုံးစေချင်တာ ဖြစ်တဲ့အတွက် stop expression ကိုရေးရတာ ဖြစ်ပါတယ်။ ဘယ်မှာ stop မလဲဆိုတဲ့ expression ဖြစ်ပါတယ်။ step ကတော့ 1 ကနေ 5 အထိ တန်ဖိုး တွေကို ပေါင်းမယ်ဆိုရင် တစ်တိုးပြီးပေါင်းမလား၊ နှစ်တိုးပြီးပေါင်းမလား။ အဲလိုတိုးပြီးပေါင်းဖို့အတွက် increment,decrement operator တွေသုံးပြီး ရေးရတဲ့ expression ကို ဆိုလိုခြင်း ဖြစ်ပါတယ်။

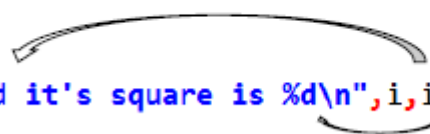
ဒီ start,stop,step expressions တွေကိုရေးတဲ့အခါမှာ semicolon(;) ခံပြီးရေးပေးရပါမယ်။ နောက်ထပ် သိထားသင့်တဲ့ for loop statement အချို့ကို ပြောပြပါမယ်။

Syntax	Output	Remarks
1)for(; ;)	Infinite loop	No arguments
2)for(a=0;a<=20)	Infinite loop	'a' is neither increased nor decreased
3)for(a=0;a<=10;a++) printf("%d",a);	Displays value From 1 to 10	'a' is increased from 0 to 10. Curly braces are not necessary. Default scope of for loop is one statement after for loop.
4)for(a=10;a>=0;a--) printf("%d",a);	Displays value From 10 to 0	'a' is decreased from 10 to 0. kdd

ဒီ table လေးကို ရှင်းပြပါမယ်။ ပထမ 1) for(; ;) ဆိုတဲ့ statement ကတော့ infinite loop ဖြစ်ပါတယ်။ ဘာလို့လဲဆိုတော့ start,stop,step expressions တစ်ခုမှ သတ်မှတ် မထားတဲ့အတွက် ဖြစ်ပါတယ်။ ဘယ်ကနေစပြီး ဘယ်မှာဆုံးမလဲဆိုတာ မရှိတဲ့ အတွက် infinite loop လို့ဆိုလိုခြင်းဖြစ်ပါတယ်။ ဒုတိယ 2) for(a=0;a<=20) ကလည်း infinite loop ဖြစ်ပါတယ်။ ဘာလို့လည်းဆိုတော့ ဒီ statement မှာ အစ၊အဆုံးရှိပေမယ့် step မရှိတဲ့အတွက် loop ပတ်တဲ့ချိန်မှာ a ရဲ့ တန်ဖိုးက အတိုး အလျှော့ မရှိတဲ့အတွက် stop expression ကို မရောက်တဲ့အတွက် infinite loop လို့ခေါ်ဆိုတာဖြစ်ပါတယ်။ တတိယ 3) for(a=0;a<=10;a++) printf("%d",a); မှာတော့ start,stop,step expressions တွေအားလုံးပါတယ်။ ဒီတော့ a ရဲ့တန်ဖိုး 0 ကနေ 10 ထိကို display ပြခိုင်းမယ်။ ဘယ်လို display ပြလဲဆိုတော့ ပထမ start မှာ a ကို 0 နဲ့ assign

လုပ်တာဖြစ်တဲ့အတွက် printf() function နဲ့ a ကို output ထုတ်တဲ့အခါမှာ 0 ကို display ပြခိုင်းမယ်။ for loop statement ကို သုံးထားတာဖြစ်တဲ့ အတွက် program က ရပ်မသွားဘဲ for loop statement ကို ထပ်ပြီး run ပါမယ်။ step မှာ a ကို increment operator သုံးပြီး တစ်တိုးခိုင်းတာ ဖြစ်တဲ့အတွက် 0 ကို တစ်တိုးပြီး a ကို 1 နဲ့ assign လုပ်တာ ဖြစ်ပါတယ်။ ပြီးရင် 1 ကို display ပြမယ်။ အဲ့တိုင်းပဲ a ကို display ပြပြီး for loop statement ကို ပဲ stop 10 နဲ့ assign မဖြစ်မချင်း တစ်တိုးပြီး a ကို display ပြမှာဖြစ်ပါတယ်။ နောက်တစ်ခု က 4) for(a=10; a>=0; a--) printf("%d", a); ဒါကတော့ နံပါတ် ၃ နဲ့ အတူတူပါပဲ start ကို 10 နဲ့ assign လုပ်တာဖြစ်တဲ့အတွက် 10 ကို display ပြမယ်။ step မှာ တစ်လျှော့ခိုင်းတာဖြစ်တဲ့ အတွက် 10-1=9 ကို a နဲ့ assign လုပ်ပြီး 9 ကို display ပြမယ်။ ဒီအတိုင်းဘဲ a=0 မဖြစ်မချင်း တစ်လျှော့ပြီး for loop statement ကို ထပ်ထပ်ခါ run ပြီး a ကို display ပြခိုင်းမှာ ဖြစ်ပါတယ်။ နားလည်မယ်လို့ထင်ပါတယ်။ ပိုနားလည်အောင် program လေးရေးပြီး လေ့လာကြတာပေါ့။

```
#include<stdio.h>
main()
{
    int i;
    for(i=1;i<=5;i++)
        printf("Number of %d it's square is %d\n",i,i*i);
}
```



```
Number of 1 it's square is 1
Number of 2 it's square is 4
Number of 3 it's square is 9
Number of 4 it's square is 16
Number of 5 it's square is 25
-----
Process exited with return value
```

ဒီ program ကို ရှင်းလင်းပါမယ်။ for(i=1;i<=5;i++) ဆိုပြီး start 1 ကနေစမယ်။ stop 5 ဆိုရင် ဆုံးမယ်။ ဆိုလိုတာကတော့မှာ i=1 i က 1 ကနေစမယ်။ ပြီးရင် ဘယ်ထိလည်း ဆိုတော့ i less than or equal to 5 ။ 5 နဲ့ assign မဖြစ်မချင်း step i++ တစ်တိုးတိုးသွားမယ်။

အိုကေ...။ ခု ပထမ for loop ကို စ run တဲ့အခါမှာ i=1 ။ ဒီတော့ printf() function သုံးပြီး output ထုတ်တဲ့အခါမှာ Number of i it's square is 1 ဆိုပြီး display ပြပေးတယ်။ ပထမ conversion symbol %d က i အတွက်။ ဒုတိယ %d က i*i expression အတွက် မြှားထိုးထားတာကို နားလည်တယ်ဟုတ်။ ဟုတ်ပြီ။ 1ရဲ့ square ကို display ပြပြီးတော့ program က ရပ်မသွားဘူး။ stop 5 ဆိုတဲ့အတွက် i=5 မဖြစ်မချင်း loop ပတ်ပါမယ်။

ဒုတိယ တစ်ခါ for loop ကို runတော့ step (i++) i ကို တစ်တိုးတာဖြစ်တဲ့အတွက် ခု i က 2 ဖြစ်သွားပြီ။ ပြီးရင် 2 ရဲ့ square ကို display ပြပေးမယ်။ ဒီအတိုင်းဘဲ i=3,i=4,i=5 ဆိုပြီး loop ပတ်ပြီး တစ်တိုးတိုးသွားမယ်။ နောက်ဆုံး i=5 ဖြစ်တဲ့အချိန်မှာ program က ရပ်သွားပါမယ်။ ဘာလို့လည်း ဆိုတော့ i က ထပ်ပြီးတစ်တိုးမယ်ဆို i=6 ဖြစ်သွားမယ်။ ဒီတော့ stop မှာက 5

နဲ့ညီရမယ်(ဒါမှမဟုတ်) 5 ထက်ငယ် ရမယ်လို့ condition test လုပ်ထားတာဖြစ်လို့ပါ။ အိုကေ ဒီ program ကို နား လည်မယ်လို့ထင်ပါ တယ်။ ဒီ program မှာ for loop statement ကို သုံးထားတဲ့အတွက်ဘာထူးခြား လဲဆိုတော့ user က 1 ကနေ 5 အထိ တန်ဖိုးတွေရဲ့ နှစ်ထပ်ကိန်းကို သိချင်တယ်ဆိုပါစို့။ 1 ကနေ 5 အထိ ခဏ ခဏ ဂိုက်ပြီး output ထုတ်နေစရာမလိုဘဲ for loop statement မှာ start,stop,step ရေး ပေးရုံနဲ့ တစ်ခါတည်း 1 to 5 ရဲ့ နှစ်ထပ်ကိန်းကို display ပြပေးမှာဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

ဒီတစ်ခါ 1 ကနေ 15 ထိတန်ဖိုးတွေကို display ပြဖို့ for loop statement သုံးပြီးရေးကြည့်ရအောင်။

```
#include<stdio.h>
main()
{
    int i;
    for(i=1;i<16;i++)
    printf("%d ",i);
}
```

```
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
-----
Process exited with return value 3
Press any key to continue
```

ဒီ program ကို ရှင်းပြပါမယ်။ I ကို ပထမ start i=1 လို့ ရေးထားတဲ့အတွက် I ရဲ့တန်ဖိုးက 1 ဖြစ်မယ်။ ပြီးရင် printf() function သုံးပြီး output ထုတ်ခိုင်းမယ်။ ဒါကြောင့် ပထမတစ်ခါ for loop statementကို run တဲ့အခါမှာ 1 ကို display ပြပေးမယ်။ display ပြပြီးတာနဲ့ program ကမပြီးသေး ပါဘူး။for loop statement ကို ထပ်ပြီး run ပါမယ်။ ဘာလို့လည်းဆိုတော့ i<16 ဖြစ်မှ program က ရပ်သွားမှာ ဖြစ်ပါတယ်။ ဒုတိယ တစ်ခါ run တော့ step (i++) ဖြစ်တဲ့အတွက် i ကို တစ်တိုးပါမယ်။ ဒီတော့ i=2 ဖြစ်သွားပြီ။ ပြီးရင် display ပြခိုင်းမယ်။ အဲ့ဒီအတိုင်းပဲ 1 to 15 ထိကို loop ပတ်ပြီး တစ်တိုး ကာ display ပြပေးမှာဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

ဒီတစ်ခါ user ရဲ့ အမည်ကို ၅ ကြောင်း display ပြချင်တယ်ဆို ဘယ်လို ရေးမလဲ။ ကြည့်ရအောင်။

```
#include<stdio.h>
main()
{
    int i;
    for(i=0;i<5;i++)
    printf("Mg Mg\n");
}
```



```

Mg Mg
Mg Mg
Mg Mg
Mg Mg
Mg Mg
-----
Process exited
Press any key

```

ကဲ ဒါဆိုရင် ဘယ်လိုရေးရလဲ သိပြီထင်ပါတယ်။ ဒီ program လေး ဘယ်လ အလုပ် လုပ်သွားလဲ ကြည့်ရအောင်။ ပထမ for loop ကို စပြီး run တဲ့အခါမှာ $i=0$ ဖြစ်ပါမယ်။ ပြီးရင် printf() function သုံး ပြီး i ကို display ပြခိုင်းမယ်။ ဒါပေမယ့် program မှာ i ကို display မပြချင်တဲ့အတွက် သူ့ရဲ့ conversion symbol %d နဲ့ i ကို ရေးမထားဘူးလေ။ ဒါကြောင့် i ရဲ့ တန်ဖိုးကို display ပြမပေးဘဲ double quote ထဲမှာ ရှိတဲ့ Mg Mg ဆိုတဲ့ စာသားကို ပဲ display ပြပေးတာဖြစ်ပါ တယ်။ နားလည်မ ယ်လို့ ထင်ပါတယ်။

ဒီတစ်ခါ logic ကျတဲ့ program လေးတစ်ပုဒ်လောက်ရေးကြည့်ရအောင်။ ရေးမယ့် program လေးကို ပြောပြပါမယ်။ 7 to 100 ကြားက 4 နဲ့ စားလျှင် အကြွင်း 0 နဲ့ ညီရမယ်။ ပြီးရင် 5 နဲ့ 6 နဲ့စားလျှင်လည်း အကြွင်း 4 ဖြစ်ရမယ့် ကိန်းကို သိချင်တယ်။ user ဆိုရင်ရော ခုမေးတာနဲ့ ချက်ချင်းဖြေနိုင်လား။ အိုကေ computer ကိုပဲ program လေးရေးပြီး တွက်ခိုင်း လိုက်မယ်နော်။

```

#include<stdio.h>
main()
{
    int i;
    for(i=7;i<100;i++)
    {
        if(i%4==0 && i%5==4 && i%6==4)
        {
            printf("Answer :%d",i);} } }

```

```

Answer :64
-----
Process exited
Press any key

```

ကဲ... computer ကို တွက်ခိုင်းလိုက်တာ မမြန်ဘူးလား။ user က သူဖြစ်စေချင်တဲ့ အတိုင်း logic လေးရေးလိုက်ရုံနဲ့ computer က သူ့ဘာသာတွက်ချက်ပြီး အဖြေ ထုတ်ပေးတာ။ မှန်လားဆိုတာ တွက် ကြည့်လိုက်ပါ။ 64 ကို 4 နဲ့ စားတော့ အကြွင်း က 0 ရတယ်။ 5 နဲ့စားတော့လည်း အကြွင်း 4 ရတယ်။ 6 နဲ့စားပြန်တော့လည်း 4 ရတယ်။ ကဲ computer တွက်ချက်တာမမှန်ဘူးလား။ ဒါပါပဲ programmer ဆိုတာ 1 နဲ့ 0 နှစ်လုံးထဲသာ နားလည်တဲ့ computer ကို Logic ကျကျ ရေးပြီး ခိုင်းစေနိုင်တဲ့သူ။ ကဲ program အလုပ်လုပ်သွားပုံလေးရှင်း ကြရအောင်။ ပထမ $i=7$ လို့ ကြေငြာထားတဲ့အတွက် for loop ကို စ run တဲ့အခါမှာ i က 7 ဖြစ်ပါမယ်။ ပြီးရင် for loop ရဲ့ scope ထဲက statement တွေကို run မယ်။ ပထမ statement က if

statement နဲ့ condition စစ်တာဖြစ်ပါတယ်။ ဘာလို့လည်း ဆိုတာတော့ သိတယ်ဟုတ်။ ဒီတော့ 7 ကို 4 နဲ့ စားတော့ အကြွင်း က 3 ရတယ်။ ဒီ expression ကတော့ မှားသွားပြီ။ 5 နဲ့ စားတော့လည်း အကြွင်း 2 ရတယ်။ 6 နဲ့ စားတော့လည်း 1 ရတယ်။ ဒီတော့ ဒီ condition test က fail ဖြစ်သွားပြီ။ ဒါကြောင့် if block ထဲက statement တွေကို မ run တော့ဘူး။ for loop ကို သုံးထားတာ ဖြစ်တဲ့အတွက် နောက်တစ်ကြိမ် for loop statement ကို ထပ်ပြီး run မယ်။ တစ်တိုးခိုင်း ထားတာ ဖြစ်တဲ့အတွက် ဒုတိယ run တဲ့အခါမှာ i က 8 ဖြစ်သွားပြီ။ ဒီအတိုင်းပဲ 7 ကနေ 99 ထိ loop ပတ်ပြီး run မယ်။ တစ်တိုးမယ်။ ပြီးရင် if နဲ့ စစ်မယ်။ expression ခု ခုလုံး မှန်မှ if block ထဲက statement ကို run မှာ ဖြစ်ပါတယ်။ အဲလိုမျိုး loop ပတ်ပြီး တစ်တိုးလိုက် if နဲ့ စစ်လိုက်နဲ့ run တဲ့ အခါမှာ i=64 ဖြစ်တဲ့အချိန်မှာ condition က မှန်သွားပြီ။ ဒါကြောင့် Answer:64 ဆိုပြီး display ပြပေး တာဖြစ်ပါတယ်။ ဒီ program ကို ကောင်းကောင်း နားလည်ပြီလို့ ထင်ပါတယ်။

ဒီတစ်ခါ 0 to 15 ကြားက စုံကိန်းတွေကို display ပြခိုင်းမယ့် program လေး ရေးကြည့် ရအောင်။

```
#include<stdio.h>
main()
{
    int i;
    for(i=0;i<=15;)
    {
        printf("%d ",i);
        i=i+2;
    }
}
```

```
0 2 4 6 8 10 12 14
Process exited with re
Press any key to conti
```

ဒီ program မှာ for loop stament ရေးထားတာကို တွေ့တယ်ဟုတ်။ start,stop ကို for loop ရဲ့ parenthesis ထဲမှာရေးပြီး step ကို တော့ for loop statement ရဲ့ scope ထဲမှာ statement အနေနဲ့ ရေးထားတာ။ for ရဲ့ parenthesis ထဲမှာ ရေးလည်း အတူတူပါပဲ။ ရေးလို့ရတယ်ဆိုတာ သိစေချင်တဲ့ အတွက် ရေးပြထားတာဖြစ်ပါတယ်။ အိုကေ...program အလုပ်လုပ်သွားတာလေး ရှင်းရအောင်။ ပထမ i=0 ဆိုတဲ့အတွက် ပထမ စ run တဲ့အခါမှာ i က 0 ဖြစ်ပါမယ်။ ပြီးရင် printf() function ကို run မယ်။ 0 ကို display ပြပေးမယ်။ printf() function ကို run ပြီးရင် i=i+2 ဆိုတဲ့ statement ကို run မယ်။ ဒီတော့ i=0+2 ဖြစ်သွားမယ်။ နောက်တစ်ခေါက် for loop ကို run တော့ i=2 က 15 ထက်ငယ်တာ ဖြစ်တဲ့ အတွက် printf() function ကို run မယ်။ 2 ကို display ပြမယ်။ 2 ကို 2 ထပ်ပေါင်းမယ်။ ပြီးရင် display ပြမယ်။ အဲ့တိုင်းပဲ i ရဲ့ တန်ဖိုးတွေကို 2 တိုးပြီး for loop ကို run မယ်။ 15 ထက်ငယ်လား၊ညီ လားစစ်မယ်။ ပြီးရင် display ပြပေးမယ်။ i=12 ကို display ပြပြီးတော့ i

ကို 2 တိုးတော့ 16 ဖြစ်သွားမယ်။ $i \leq 15$ ဆိုပြီး စစ်လိုက်တော့ fail ဖြစ်သွားပြီ။ ဒီတော့ for loop statement ရဲ့ scope ထဲက statement တွေကို မ run တော့ဘဲ program က ပြီးဆုံးသွားပါမယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

for loop ကို ပိုနားလည်အောင် ဒီတစ်ခါ Ascending order နဲ့ Descending order ကို ရေးကြည့်ရအောင်။ ဆိုလိုတာကတော့ ငါ့ ကြီးစဉ်ငယ်လိုက်နဲ့၊ ငယ်စဉ်ကြီးလိုက် ရေးခိုင်းတာကို ပြောတာပါ။ အိုကေ ...ရေးကြည့်ရအောင်။

```
#include<stdio.h>
main()
{
    int i;
    printf("Number in ascending order");
    for(i=0;i<=5;i++)
    {
        printf("\t\n%d\t",i);
        printf("\n");
    }
    printf("Number in descending order");
    for(i=5;i>=0;i--)
    {
        printf("\t\n%d\n",i);
    }
}
```

```
Number in ascending order
0
1
2
3
4
5
Number in descending order
5
4
3
2
1
0
```

ဒီ program ရဲ့ အလုပ်လုပ်ပုံကို တော့ သိပ်ရှင်းပြစရာမလိုတော့ဘူးထင်ပါတယ်။ ascending လုပ်သွားတာကိုတော့ ရှင်းမပြတော့ပါဘူး။ program က ပထမ for loop statement ကို အရင် run ပါတယ်။ 0 to 5 ထိ loop ပတ်ပြီး တစ်တိုးတိုးသွားမှာပါ။ I က 16 ဖြစ်သွားတဲ့ချိန်မှာ stop

$i \leq 15$ လို့ test condition လုပ်ထားတာဖြစ်တဲ့အတွက် ပထမ for loop ထဲက statement တွေကို မ run တော့ဘဲ ဒုတိယ for loop ကို run မှာဖြစ်ပါတယ်။ ဒုတိယ for loop မှာ i ကို 15 နဲ့ assign လုပ်ထားတာဖြစ်တဲ့ အတွက် i ရဲ့ တန်ဖိုးက 15 ဖြစ်သွားပါမယ်။ ပြီးရင် display ပြခိုင်းမယ်။ နောက်တစ်ကြိမ် for loop ကို run တော့ i ကို တစ်လျှော့ခိုင်းထားတာ ဖြစ်တဲ့အတွက် i က 14 ဖြစ်သွားပါမယ်။ ပြီးရင် display ပြမယ်။ ဒီအတိုင်းဘဲ loop ပတ်ပြီး i ရဲ့တန်ဖိုးကို တစ်လျှော့လျှော့သွားပါမယ်။ နောက်ဆုံး i က 0 ဖြစ်တဲ့ထိ လျှော့သွားပါမယ်။ i ရဲ့ တန်ဖိုး 0 ကို display ပြပြီး for loop ကို run တော့ 0-1 ဖြစ်သွားပါမယ်။ ဒီတော့ i ရဲ့တန်ဖိုးက -1 ဆိုတော့ $i \geq 0$ လို့ test condition လုပ်ထားတာဖြစ်တဲ့အတွက် program က ပြီးဆုံး သွားမှာ ဖြစ်ပါတယ်။ ဒါဆို နားလည် သဘောပေါက်မယ်လို့ ထင်ပါတယ်။

ဒီတစ်ခါ infinite for loop နဲ့ goto statement ကို သုံးပြီး 1 to 5 ကို display ပြပေးမယ့် program လေး တစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။

```
#include<stdio.h>
#include<conio.h>
main()
{
    int i=1;
    for(;;)
    {
        printf("%d ",i);
        i++;
        if(i==6)
            goto stop;
    }
    stop;;
}
```

```
1 2 3 4 5
-----
Process exited with
Press any key to con
```

အိုကေ program လေးကို ရှင်းရအောင်။ ပထမ statement မှာ i ကို 1 လို့ assign လုပ်ထားတာဖြစ် တဲ့အတွက် i တန်ဖိုးက 1 ဖြစ်ပါမယ်။ $\text{for}(;;)$ ဆိုတာကတော့ infinite loop ဖြစ်ပါတယ်။ start ,stop,step တွေကို ဖော်ပြထားပါဘူး။ ဒါပေမယ့် for loop ကို သုံးရင်တော့ program က loop ပတ်ပြီး run မှာပါ။ $\text{Printf}("%d",i);$ ဆိုပြီး i ကို display ပြခိုင်းမယ်။ ဒီတော့ 1 ကို display ပြမှာပါ။ ပြီးရင် $i++$ ဆိုတဲ့ statement ကို တွေ့တော့ i ကို တစ်တိုးပါမယ်။ ဒီတော့ $i=2$ ဖြစ်သွားပါပြီ။ ပြီးရင် i က 6 နဲ့ ညီလာစစ်ပါမယ်။ မညီတဲ့အတွက် for loop ကို ထပ်ပတ်မယ်။ i တန်ဖိုးကို display ပြခိုင်းမယ်။ အဲ့ဒီအ တိုင်းပဲ တစ်တိုးလိုက် i ကို display ပြလိုက်နဲ့ နောက်ဆုံး i က 6 ဖြစ်သွားတဲ့ အချိန်မှာ $\text{if}(i==6)$ နဲ့စစ်လိုက်တာ မှန်သွားပြီ။ ဒီတော့ if block ထဲက statement ကို run ပါမယ်။ ဒီတော့ goto stop ဆိုတဲ့ statement ကို တွေ့ပါမယ်။ goto ရဲ့ အလုပ်လုပ်ပုံကို ရှေ့မှာ

ပြောခဲ့ပြီးပြီဟုတ်။ ဒီတော့ stop ဆိုတဲ့ label ရှိတဲ့နေရာကို သွားပြီး run ပါမယ်။ ဒါပေမယ့် stop label မှာ ဘာ statement မှာ ရေးမထားတဲ့ အတွက် program က ပြီးဆုံးသွားပါမယ်။ ဒီ program ကို နားလည်မယ်လို့ ထင်ပါတယ်။...

ဒီတစ်ခါ switch () statement နဲ့ for loop statement ကို သုံးပြီး ရက်သတ္တပတ်ရဲ့ နေ့တွေကို ဖော် ပြခိုင်းမယ့် program လေး ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int day ,i;
5  printf("Enter a number between 1 to 7:");
6  scanf("%d",&day);
7  for(i=1;i<=day;i++)
8  switch(i)
9  {
10 case 1:
11 printf("\n 1st day of week is Sunday");
12 break;
13 case 2:
14 printf("\n 2nd  day of week is Monday");
15 break;
16 case 3:
17 printf("\n 3rd  day of week is Tuesday");
18 break;
19 case 4:
20 printf("\n 4th day of week is  Wednesday");
21 break;
22 case 5:
23 printf("\n 5th day of week is Thursday");
24 break;
25 case 6:
26 printf("\n 6th  day of week is Friday");
27 break;
28 case 7:
29 printf("\n 7th  day of week is Saturday");
30 break;
31 default:
32 printf("Invalid day");
33 } }

```

Enter a number between 1 to 7: 4

```

1st day of week is Sunday
2nd  day of week is Monday
3rd  day of week is Tuesday
4th day of week is  Wednesday

```

Process exited with return value 4

ဒီ program မှာ သုံးထားတဲ့ control statement တွေကို တော့ သိပြီးသားလို့ထင်ပါတယ်။ ဒီ program ရဲ့ အလုပ်လုပ်သွားပုံကို ပြောပြပါမယ်။ ပထမ user ကို day အတွက် တန်ဖိုးရိုက်ခိုင်းပါမယ်။ ဥပမာ user က 2 လို့ ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆိုရင် 1st day နဲ့ 2nd day ကို display ပြပေးမှာဖြစ်ပါတယ်။ 4 လို့ ရိုက်မယ်ဆိုရင် လည်း 1st to 4th ရဲ့ day တွေကို ပြပေးမှာဖြစ်ပါတယ်။ ဘာလို့ အဲလိုပြပေးလဲဆိုတာကတော့ for loop statement ကို သုံးထားတဲ့အတွက် ဖြစ်ပါတယ်။ for loop statement ကို သုံးမထားဘူးဆိုရင်တော့ 2 လို့ရိုက် ရင် case 2: ရဲ့ statement တစ်ခုထဲကိုပဲ display ပြပေးမှာဖြစ်ပါတယ်။

ဒီ program မှာ for loop ရဲ့ အလုပ် လုပ်သွားပုံလေးကို ပြောပြပါမယ်။ user ကို day ကို ရိုက်ပါလို့ ပြောတဲ့အချိန်မှာ 4 လို့ ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆိုရင် day=4 ဖြစ်သွားပြီ။ ပြီးရင် for loop statement နဲ့ loop ပတ်ခိုင်းပါမယ်။ for(i=0;i<=4;i++) ဖြစ် သွားမယ်။ ဒီတော့ i က 1 ဖြစ်တဲ့အချိန်မှာ switch(i) က ဖြစ်မယ်။ ပြီးရင် case 1: statement ကို display ပြခိုင်းမယ်။ for loop ကို သုံးထားတာဖြစ်တဲ့အတွက် for loop ကို ထပ်ပြီး run မယ်။ i ကိုလည်း တစ်တိုးမယ်။ အဲ့ဒီ အတိုင်းပဲ i ကို တစ်တိုးပြီး switch (i) တန်ဖိုးနဲ့ case constant ကို တိုက်စစ်ပြီး statement တွေကို display ပြပေးမှာ ဖြစ်ပါတယ်။ နောက်ဆုံး i=5 ဖြစ်တဲ့အချိန်မှာ for loop statement မှာ i<=4 လို့ condition test လုပ် ထားတာ ဖြစ်တဲ့အတွက် program က ရပ်သွားမှာ ဖြစ်ပါတယ်။ အသေးစိတ်တော့ ရှင်းမပြတော့ပါဘူး။ နားလည် နိုင်မယ်လို့ထင်ပါတယ်။

For loop statement ကို ကောင်းကောင်းနားလည်မယ်လို့ထင်ပါတယ်။ဒီတော့ နောက်ဆုံး for loop ကို သုံးပြီး user ရိုက်လိုက်တဲ့ ကိန်းတွေထဲက အကြီးဆုံးကိန်းကို ရှာတဲ့ program လေးရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  int a,b,c,i,sum;
5  printf("Enter three numbers");
6  scanf("%d %d %d",&a,&b,&c);
7  sum=a+b+c;
8  for(i=sum;i<=sum;i--)
9  {
10 if(i==a || i==b || i==c)
11 {
12 printf("The largest number is %d",i);
13 break;
14 }
15 }
16 }
```

```

Enter three numbers1 2 3
The largest number is 3
Process exited with return v
```

ဒီ program လေးကို ရှင်းပြပါမယ်။ ပထမ user ဆီကနေ ကိန်း ဂုဏ်ရိုက်ပါဆိုတဲ့ စာသားလေးပြပြီး ကိန်းဂုဏ် ရိုက်ခိုင်းမယ်။ ပြီးရင် ပထမ ရိုက်လိုက်တဲ့ ကိန်းကို a ထဲမှာ သိမ်းမယ်။ ဒုတိယ b ထဲ တတိယ c ထဲမှာ သိမ်းထား လိုက်မယ်။ $sum=a+b+c$; ဆိုတဲ့ statement ကို run တော့ a,b,c ထဲမှာ သိမ်းထားတဲ့ တန်ဖိုးတွေကို ပေါင်းပြီး sum ထဲမှာ သိမ်းထားခိုင်းလိုက်ပါမယ်။

ဥပမာ user က $a=1, b=2, c=3$ လို့ ရိုက်ပြီး သိမ်းထားတယ်ဆိုပါစို့။ ဒါဆိုရင် $sum=6$ ဖြစ်သွားပါမယ်။ ဒီထိ နားလည်မယ်နော်။ ပြီးရင် $for(i=sum; i \leq sum; i--)$ ဆိုတဲ့ statement ကို run မယ်။ sum က 6 ဖြစ်တဲ့အတွက် compiler က $for(i=6; i \leq 6; i--)$ လို့ နားလည်လိုက်ပါမယ်။ start i က 6 ဖြစ်မယ်။ ပြီးရင် for loop ရဲ့ scope ထဲက statement တွေကို run ပါမယ်။

$if(i==a \ || \ i==b \ || \ i==c)$ လို့ if နဲ့ စစ်ခိုင်းပါတယ်။ i က 6 ဆိုတော့ a,b,c ထဲမှာ သိမ်းထားတဲ့ တန်ဖိုးတွေထဲက တစ်ခုနဲ့မှ မတူဘူးလေ။ ဒါကြောင့် if block ထဲက statement ကို မ run ဘဲ for loop ကို ထပ် run မှာ ဖြစ်ပါတယ်။ step (i--) ဆိုတဲ့ အတွက် i တန်ဖိုးကို တစ်လျှော့ပေးပါတယ်။ ဒီတော့ $i=5$ ဖြစ်သွားပါမယ်။ ပြီးရင် if နဲ့ စစ်ပါမယ်။ a,b,c ဂုဏ်ထဲက ဘယ်တန်ဖိုးနဲ့ တူလဲစစ်ပါမယ်။ $||$ (logical OR) operator ကို သုံးထားတာ ဖြစ်တဲ့အတွက် 3 ခုထဲက တစ်ခုနဲ့ တူတယ်ဆိုရင်တောင် if block ထဲက statement ကို run မှာပါ။ ခုတစ်ခုနဲ့မှ မတူတဲ့အတွက် for loop ကို ထပ် ပတ်ပါမယ်။ နောက်ဆုံး $i=3$ ဖြစ်တဲ့အချိန်မှာ a,b,c ဂုဏ်ထဲက c နဲ့ညီသွားပြီ။ ဒါကြောင့် if block ထဲက statement ကို run ပါမယ်။ i ကို display ပြခိုင်းတာ ဖြစ်တဲ့အတွက် i တန်ဖိုး 3 ကို display ပြပေးတာ ဖြစ်ပါတယ်။ ကဲ...ဒါဆို ကြည့်လိုက်ဗျာ။ user ရိုက်လိုက်တဲ့ ကိန်း ဂုဏ်ထဲက 3 က အကြီးဆုံးဖြစ်လားဆိုတာ။ အိုကေ .မှန်တယ်နော်။ ဒါပါပဲ...program ဆိုတာ logic instruction လေး ရေးပေးလိုက်ရုံ နဲ့ မိမိသိချင်တဲ့ data တွေကို မှန်မှန်ကန်ကန် တွက်ချက်ပေးတဲ့အရာ။ စိတ်ဝင်စားစရာ ကောင်းတယ်ဟုတ်။

ဒီ program ကို နားလည်မယ်ဆိုရင် ကိန်း ဂုဏ်ထဲက အငယ်ဆုံးကိန်းကို ရှာတဲ့ program လေး ရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4      int a,b,c,sum,i;
5      printf("Enter three numbers");
5      scanf("%d %d %d",&a,&b,&c);
7      sum=a+b+c;
3      for(i=1;i<=sum;i++)
3      {
3          if(i==a || i==b || i==c)
1          {
2              printf("The smallest number is %d",i);
3              break;
4          }
5      }
5  }
```

```
Enter three numbers1 2 3
The smallest number is 1
-----
Process exited with return
```

ဒီ program ကိုတော့ ရှင်းမပြော့ပါဘူး။ မိမိကိုယ်တိုင် sense ရအောင် ကြိုးစားပြီး trace ကြည့်ပါ။ program တွေကို ရှင်းပြရင်တော့ အများကြီးပဲခင်ဗျ။ သိချင်တယ်ဆိုရင်တော့ ကျွန်တော့် ဆီကိုလာပြီး မေးလို့ရပါတယ်။ ခုတော့ အခြေခံလွယ်ကူတဲ့ program လေးတွေကိုပဲ ရေးပြီး ရှင်းပြထားတာပါ။ သဘောတရားကို သိရင်တော့ ဘယ် program မဆို နားလည်နိုင်မယ်လို့ ထင်ပါတယ်။

ဒီတစ်ခါ Nested if လိုပဲ for loop မှာလည်း Nested လုပ်ပြီး ရေးလို့ရပါတယ်။ ခု Nested for loop ရဲ့ syntax ကို ရေးပြပါမယ်။

```
for(start;stop;step)
```

```
{
```

```
    for(start;stop;step)
```

```
{
```

```
    statement; } }
```

အိုကေ ဒီ syntax ကိုတော့ ရှင်းမပြော့ပါဘူး။ program လေးရေးပြီး ရှင်းပြပါမယ်။

```
#include<stdio.h>
main()
{
    int r,c,sum;
    for(r=1;r<=2;r++) Outer loop
    {
        for(c=1;c<=2;c++) Inner loop
        {
            sum=r+c;
            printf("r=%d c=%d sum=%d\n",r,c,sum);
        }
    }
}
```

```
r=1 c=1 sum=2
r=1 c=2 sum=3
r=2 c=1 sum=3
r=2 c=2 sum=4
```

```
-----
Process exited
```

ဒီ program မှာဆိုရင် Nested for loop ကို သုံးထားတာဖြစ်ပါတယ်။ Nested loop ရဲ့ အလုပ်လုပ်ပုံကို အရင်ရှင်းပြပါမယ်။ outer for loop နဲ့ inner for loop နှစ်ခုမှာ အရင်ဆုံး outer for loop ကို စ run ပါမယ်။

ဒီ program မှာဆို outer for loop ကို တစ်ကြိမ် run ပြီးတာနဲ့ inner for loop ကို stop $i \leq 2$ ဆိုတဲ့ condition fail မဖြစ်မချင်း loop ပတ်ပြီး run မှာ ဖြစ်ပါတယ်။ ဒီ program ကို ရှင်းရင်း ပြောပြပါမယ်။ ပထမ for loop ကို စ run တဲ့ အချိန်မှာ $r=1$ ဆိုတဲ့အတွက် r ရဲ့တန်ဖိုးက 1 ဖြစ်ပါမယ်။ ပြီးရင် inner for loop ကို run ပါမယ်။

inner for loop မှာ $c=1$ လို့ initialize လုပ်ထားတဲ့အတွက် c ရဲ့ တန်ဖိုးက 1 ဖြစ်ပါမယ်။ ပြီးရင် $\text{sum}=r+c$ ဆိုတော့ statement ကို run ပြီး $1+1=2$ ကို sum ထဲမှာ သိမ်းထား လိုက်မယ်။ ဒီတော့ $\text{printf}()$ function ကို သုံးပြီး r, c , နဲ့ sum ရဲ့ တန်ဖိုးတွေကို display ပြခိုင်း တာဖြစ်တဲ့အတွက် r တန်ဖိုး 1, c တန်ဖိုး 1, $\text{sum}=2$ ကို display ပြပေးတာ ဖြစ်ပါတယ်။ ပြီးရင် for loop statement ကို သုံးထားတာဖြစ်တဲ့အတွက် loop ပတ်ပါမယ်။ ဘယ် loop ကို ပတ်မလဲဆိုတော့ inner loop ကိုပဲ ထပ်ပတ်မှာ ဖြစ်ပါတယ်။ ဘယ်ချိန်ထိ ပတ်မလဲဆိုတော့ $c \leq 2$ မဖြစ်မချင်း loop ပတ်ပါမယ်။ inner loop ကို ပတ်ပြီး stop ဖြစ်တဲ့အချိန်မှ outer loop ကို ထပ်ပတ်မှာ ဖြစ်ပါတယ်။

ဒီတော့ inner loop ကို run ပါမယ်။ $\text{step}(c++)$ ဆိုတဲ့အတွက် c ကို တစ်တိုးပါမယ်။ ဒီတော့ $c=2$ ဖြစ်သွားပါပြီ။ ပြီးရင် $\text{sum}=r+c$ ကို run ပါမယ်။ r တန်ဖိုးကတော့ outer loop ကို run တာမဟုတ်တဲ့အတွက် တစ်မတိုးပါဘူး။ r တန်ဖိုး က 1 ပဲဖြစ်ပါမယ်။ ဒီတော့ $\text{sum}=1+2=3$ ဖြစ်ပါ မယ်။ ဒါကြောင့် r, c, sum ကို display ပြတဲ့ အချိန်မှာ $r=1, c=2, \text{sum}=3$ ကို display ပြပေး တာဖြစ်ပါတယ်။ ပြီးရင် c ကို တစ်တိုးမယ်။

ဒီတော့ $c=3$ ဖြစ်သွားပြီ။ $\text{stop}(c \leq 2)$ လို့ condition test လုပ်ထားတာဖြစ်တဲ့အတွက် c က 3 ဆိုတော့ condition က fail ဖြစ်သွားပြီ။ ဒါကြောင့် outer for loop ကို သွားပြီး run ပါမယ်။ $\text{step}(r++)$ ဆိုတဲ့အတွက် r ကို တစ်တိုးပါမယ်။ ဒီတော့ $r=2$ ဖြစ်ပါမယ်။ ပြီးရင် inner for loop ကို run ပါမယ်။ inner for loop မှာ c ကို 1 လို့ start လုပ်ခိုင်းတာ ဖြစ်တဲ့အတွက် $r=2, c=1, \text{sum}=3$ ဖြစ်ပါ မယ်။ ပြီးရင် display ပြခိုင်းမယ်။ inner for loop မှာ c ကို တစ်တိုးခိုင်းတာ ဖြစ်တဲ့အတွက် $c=2$ ဖြစ်သွားပါမယ်။ ဒီတော့ $\text{sum}=2+2=4$ ဖြစ်သွား မယ်။ ပြီးရင် display ပြမယ်။ inner for loop မှာ c ကို တစ်တိုးခိုင်းတာ ဖြစ်တဲ့အတွက် တစ်တိုးမယ်။ $c=3$ ဖြစ်သွားပြီ။ condition ထပ်စစ်မယ်။ c က 3 ဖြစ်တဲ့ အတွက် condition က fail ဖြစ်သွားပြီ။

ဒါကြောင့် outer loop ကို run မယ်။ r ကို တစ်တိုး ခိုင်းတဲ့အတွက် $r=3$ ဖြစ်သွားပြီ။ condition စစ်တော့ $r \leq 2$ က မှားသွားပြီ။ ဒါကြောင့် program က inner loop ကို မ run တော့ဘဲ program ပြီး ဆုံးသွားမှာ ဖြစ်ပါတယ်။ ဒီ program ကို အတိုချုပ်ပြောရရင် outer loop မှာ r တန်ဖိုး 1 ဖြစ်တဲ့အချိန်မှာ inner loop က c တန်ဖိုးက 1 နဲ့ 2 ဖြစ်ပါမယ်။ Outer loop r တန်ဖိုးက 2 ဖြစ်တဲ့အချိန်မှာ inner loop c တန်ဖိုးက 1 နဲ့ 2 ဖြစ်ပါမယ်။ ဒီထက်ပိုရှင်းလင်းအောင်ပြောရရင် outer loop ကို တစ်ကြိမ် run ပြီးတာနဲ့ inner loop ကို နှစ်ကြိမ် loop ပတ်ပါမယ်။ outer loop ကို နောက်တစ်ကြိမ် run တော့လည်း inner loop ကို နှစ်ကြိမ် loop ပတ်မှာဖြစ်ပါတယ်။ ဒါဆို နားလည် သဘောပေါက်မယ်လို့ ထင်ပါတယ်။

ဒါဆိုရင် Nested for loop ကို နားလည်မယ်လို့ ထင်ပါတယ်။

2) The while loop

The while loop ရဲ့ အလုပ်လုပ်ပုံက လည်း for loop နဲ့အတူတူ ပါပဲ။ syntax ကွဲပြားတာပဲ ရှိတာပါ။ သူ့ရဲ့ syntax ကို ကြည့်ရအောင်။

```
start;
while(stop(test condition))
{
statement;
step;
}
```

အိုကေ သူ့ရဲ့ syntax ကို သိပြီး ဆိုတော့ program လေးတစ်ပုဒ်လောက်ရေးကြည့်ရအောင်။

```
1 #include<stdio.h>
2 main()
3 {
4     int i=1;
5     while(i<=3)
6     {
7         printf("%d ",i);
8         i++;
9     }
```

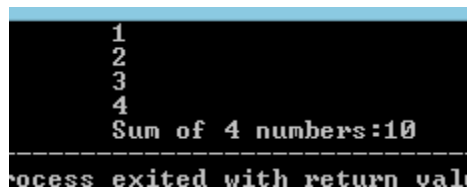
```
1 2 3
Process exited
```

အိုကေ ဒီ programလေးကို ရှင်းရအောင်။ ပထမ int i =1 လို့ i ကို 1 ကနေ စမယ်လို့ start သတ်မှတ်လိုက်တာ ဖြစ်ပါတယ်။ ပြီးရင် while(i<=3) ဆိုတဲ့ statement နဲ့ condition စစ်ပါမယ်။ while ရဲ့ အဓိပ္ပာယ်ကို မြန်မာလို ပြန်ကြည့်ရင် ကာလပတ်လုံး လို့ အဓိပ္ပာယ်ရပါတယ်။ ဒီတော့ ဆိုလိုတာက i<=3 ဖြစ်နေသမျှ ကာလ ပတ်လုံး while loop ရဲ့ scope ထဲက statementကို run မှာဖြစ်ပါတယ်။ အိုကေ.....။ ဒီတော့ i=1 ဖြစ်တဲ့ အတွက် condition က မှန်နေတယ်။ ဒါကြောင့် printf() function ကို run ပြီး i တန်ဖိုး 1 ကို display ပြပေးမှာ ဖြစ်ပါတယ်။ ပြီးရင် step(i++) လို့ရေးထားတဲ့အတွက် i ကိုတစ်တိုးပါမယ်။ ဒီတော့ i=2 ဖြစ်သွားပြီ။ while loop ကို သုံးထားတာဖြစ်တဲ့အတွက် for loop လိုပဲ loop ပတ်ပါမယ်။ while (i<=3) ဆိုတော့ i က 2 ဖြစ်တဲ့အတွက် test condition က မှန်နေပါသေးတယ်။ ဒီတော့ i=2 ကို display ပြပေးမှာဖြစ်ပါတယ်။ ပြီးရင် တစ်တိုးမယ်။ condition ထပ်စစ်မယ်။ i တန်ဖိုး 3 ကို display ပြမယ်။ i ကို တစ်တိုးခိုင်းတာဖြစ်တဲ့အတွက် တစ်တိုးတော့ i=4 ဖြစ်သွားပြီ။ ဒီတော့ condition test က မှားသွားပြီ။

ဒါကြောင့် I တန်ဖိုး 1 2 3 ကို display ပြပြီးတာနဲ့ program က ပြီးသွားမှာ ဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

နောက်ထပ် while loop ကို သုံးပြီး 1 to 4 ကို display ပြပြီး 1 to 4 ထိ ကိန်းအားလုံးပေါင်းလဒ်ကို ရှာတဲ့ program လေးရေးကြည့်ရအောင်။

```
#include<stdio.h>
main()
{
    int a=1,sum=0;
    while(a<=4)
    {
        printf("\t%d \n",a);
        sum=sum+a;
        a++;
    }
    printf("\tSum of 4 numbers:%d",sum);
}
```



```
1
2
3
4
Sum of 4 numbers:10
process exited with return value
```

ကဲ...ဒါဆို program ကိုရှင်းရအောင်။ ဒီ program မှာ a =1 လို့ declare ကြေငြာထားတာကတော့ a =1 က စပါမယ်ဆိုတဲ့သဘောပါ။ ပြီးရင် sum=0 လို့ ကြေငြာထားပါမယ်။ ဘာလို့လဲဆိုတော့ 1 to 4 ရဲ့ ကိန်းအားလုံးကို ပေါင်းပြီး sum ထဲမှာ သိမ်းထားမှာ ဖြစ်ပါတယ်။ အဲလိုပေါင်းဖို့ အတွက် while loop ကို တစ်ကြိမ် loop ပတ်တိုင်း ရလာတဲ့ a ရဲ့ တန်ဖိုးတွေကို တစ်ခုချင်းစီ ပေါင်းမှာ ဖြစ်ပါတယ်။ ပိုနားလည်သွားအောင် program ကိုရှင်းတာကို ကြည့်ပါ။

ပထမ a ကို 1 လို့ start သတ်မှတ်ထားတဲ့ အတွက် a တန်ဖိုးက 1 ဖြစ်ပါမယ်။ while(a<=4) လို့ test condition လုပ်ပြီး loop ပတ်ခိုင်းတဲ့အတွက် a က 1 ဖြစ်တဲ့ အတွက် condition ကမှန်ပါတယ်။ ပြီးရင် a တန်ဖိုးကို display ပြခိုင်းပါမယ်။ ဒါကြောင့် sum=sum+a; ကို run ပါမယ်။ sum ကို 0 လို့ ကြေငြာထားတဲ့အတွက် sum=0+1; ဖြစ်ပါမယ်။ ဒီတော့ sum=1 ဖြစ်သွားပါပြီ။ ကဲ...ဒါဆို sum ကို 0 လို့ ဘာကြောင့် ကြေငြာပေးတာလဲဆိုတာကို နားလည်ပြီလို့ထင်ပါတယ်။ ဘာတန်ဖိုးနဲ့မှ ပေါင်းထားတာ မဟုတ်တဲ့ အတွက် 0 လို့ ကြေငြာပေးထားခြင်း ဖြစ်ပါတယ်။ ပြီးရင် step မှာ a++ ဆိုတဲ့အတွက် a ကို တစ်တိုးပါမယ်။ ဒီတော့ a=2 ဖြစ်သွားပါပြီ။

ပြီးရင် while condition နဲ့ စစ်မယ်။ မှန်တဲ့အတွက် a တန်ဖိုးကို display ပြမယ်။ ပြီးရင် sum=sum+a ကို run မယ်။ sum က 1, a=2 ဖြစ်တဲ့အတွက် sum=1+2 ဖြစ်ပါမယ်။ ဒီအတိုင်းပဲ a ကို တစ်တိုးလိုက် condition စစ်လိုက်၊ a တန်ဖိုးကို display ပြလိုက် sum ကို ရှာလိုက်နဲ့ loop ပတ်နေမှာ ဖြစ်ပါတယ်။ နောက်ဆုံး sum=6+4 ဖြစ်တဲ့အချိန်မှာ a တန်ဖိုး 4 ကို တစ်တိုးပါမယ်။

ဒီတော့ $a=5$ ဖြစ်သွားပါမယ်။ ပြီးရင် condition test လုပ်တဲ့အခါမှာ $\text{while}(a \leq 4)$ ဆိုရင် stop ပါမယ်ဆိုတဲ့အတွက် while loop statement ရဲ့ scope ထဲက statement တွေကို မ run တော့ဘဲ while loop scope ရဲ့ အပြင်ဘက်ကို ရောက်သွားပါမယ်။ ဒီတော့ while loop scope ရဲ့ အပြင်ဘက်က $\text{printf}()$ function ကို run မှာ ဖြစ်ပါတယ်။ $\text{printf}()$ function ကို run တဲ့အခါမှာ sum ကို display ပြခိုင်းတာဖြစ်တဲ့အတွက် နောက်ဆုံးတွက်လို့ ရတဲ့ sum ရဲ့တန်ဖိုးက 10 ဖြစ်ပါတယ်။ ဒါကြောင့် Sum of 4 numbers:10 လို့ display ပြချင်းဖြစ်ပါတယ်။ ကဲ..ဒီလောက် ဆို while loop statement ရဲ့ သဘောတရားကို နားလည်မယ်လို့ ထင်ပါတယ်။

3) The do while loop

The do while loop ကတော့ for loop ,while loop နဲ့ အတူတူ ပါပဲ။ မတူတဲ့ အချက်လေး တခုတော့ ရှိပါတယ်။ အရင်ဆုံး သူ့ရဲ့ syntax ကို ကြည့်ရအောင်။

```
start;
do
{
statement;
step;
}
while(stop)
```

ကဲ...သူ့ရဲ့ syntax ကို သိပြီဆိုတော့ program လေးရေးပြီးလေ့လာကြရအောင်။

```
1  #include<stdio.h>
2  main()
3  {
4      int i=7;
5      do
6      {
7          printf("This is a program of do while loop");
8          i++;
9      }
10     while(i<=5);
11 }
```

```
This is a program of do while loop
-----
Process exited with return value 34
Press any key to continue . . . _
```

ဒီ program မှာဆိုရင် do while loop နဲ့ for loop, while loop နဲ့ မတူတဲ့အချက်လေးကို သိအောင် ရေးပြ ထားပါတယ်။ ပထမ start ကို i=1 လို့ကြေငြာထားပါတယ်။

Line_3. ပြီးရင် do ဆိုပြီး condition မစစ်ခိုင်းဘဲနဲ့ run ခိုင်းပါတယ်။ ဒီတော့ run တာပေါ့။ This is a program of do while loop ဆိုတဲ့ စာသားလေးကို display ပြပေးလိုက်ပါတယ်။

line_8 . ပြီးမှ i++ ဆိုတဲ့အတွက် i ကို တစ်တိုးပါမယ်။ ဒီတော့ i တန်ဖိုးက 8 ဖြစ်သွားပါမယ်။

line_10. ပြီးမှ while loop ကို run ပြီး condition စစ်ပါမယ်။ i<=5 ဆိုတော့ i က 8 ဆိုတော့ condition က မှားသွားပြီ။ ဒီတော့မှ do while loop ကို ထပ်ပြီး loop မပတ်တော့ဘဲ။ program က ပြီးဆုံးသွားမှာ ဖြစ်ပါတယ်။ do while loop ရဲ့ မတူ တဲ့ အချက်ကို နားလည်ပြီဟုတ်။ သူက condition ကို အရင်မစစ်ဘဲ do ဆိုပြီး statement တွေကို run ခိုင်းပါတယ်။ ပြီးမှ condition စစ်ပါတယ်။ condition မှန်ရင်တော့ do scope ထဲက statement ကို loop ပတ်ပြီး run မှာဖြစ်ပါတယ်။ မှားတယ်ဆိုရင်တော့ program က ပြီးဆုံးသွားပါမယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

နောက်ထပ် do while loop သုံးပြီး program လေးတစ်ပုဒ်လောက်ရေးကြည့်ရအောင်။ while loop သုံးပြီး ရေး ထားတဲ့ program လေးကို ဝဲ do while loop သုံးပြီးရေးတာပေါ့။

```

1  #include<stdio.h>
2  main()
3  {
4  int i,a=1;
5  int s=0;
6  printf("Enter a number");
7  scanf("%d",&i);
8  do
9  {
10 printf("%d+",a);
11 s=s+a;
12 a++;
13 }
14 while(a<=i);
15 printf("\nb=%d",s);
16 }

```

Enter a number5
1+2+3+4+5=15
Process exited with

<different>

Enter a number5
1+2+3+4+5=15
Process exited with

Line_4. ဒီ program မှာဆိုရင် start ကို a=1 လို့သတ်မှတ်ထားပါတယ်။

Line_5. s=0 ကတော့ user ရိုက်လိုက်တဲ့ ကိန်းထိ တန်ဖိုးတွေကိုပေါင်းပြီး သိမ်းဆည်းထားဖို့ပါ။ ဆိုလိုတာတော့မှာ ၃ လို့ရိုက်ရင် 1 to 3 ထိ ကိန်းတွေကို ပေါင်းပြီး သိမ်းထားဖို့ပါ။ ရှေ့မှာလည်း အလားတူ program ကို ရေးပြထားပါတယ်။

Line_6. ပထမ user ကို ကိန်းတစ်လုံး ရိုက်ပါလို့ တောင်းပါမယ်။

Line_7. ဥပမာ user က 5 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆို 1 to 5 ထိတန်ဖိုးတွေကို ပေါင်းပါမယ်။ အဲ့ဒီ 5 ကို တော့ variable I ထဲမှာ သိမ်းထားပါမယ်။

Line_8 . ပြီးရင် do ကို run တော့ do ရဲ့ scope ထဲက statement တွေကို run ပါမယ်။

Line_10. printf() ဆိုတဲ့အတွက် a ကို display ပြပါမယ်။ a ရဲ့တန်ဖိုးက 1 ဖြစ်တဲ့အတွက် 1+လို့ display ပြပါမယ်။+ လက္ခဏာဘာလို့ပါလာတယ်ဆိုတာကို တော့ သိတယ်ဟုတ်။ printf() ရဲ့ double quote ထဲမှာ ရေးထားတဲ့အတွက်ဖြစ်ပါတယ်။အိုကေ....။

Line_11. ပြီးရင် s=s+a ကို run ပြီး s=0+1 လို့တွက်ချက်လိုက်ပါမယ်။ ဒီတော့ s=1 ဖြစ်သွားပါမယ်။

Line_12. နောက် a++ ဆိုတဲ့ statement ကိုတွေ့တော့ a ကို တစ်တိုးလိုက်ပါမယ်။ ဒီတော့ a=2 ဖြစ်သွား ပါမယ်။

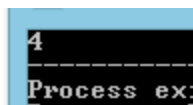
Line_14. ပြီးရင် while(a<=i) ဆိုပြီး condition စစ်ပါမယ်။ while(2<=5) လို့ နားလည်လိုက်ပါမယ်။ ဒီတော့ condition က မှန်သွားပြီဖြစ်တဲ့အတွက် do statement ဆီကို ပြန်ရောက်ပါမယ်။ ပြီးရင် ပထမ run တဲ့ အတိုင်းပဲ statement တွေကို run ပြီး တန်ဖိုးတွေ တွက်ထုတ်ပေးမှာဖြစ်ပါတယ်။ နောက် ဆုံး s=10+5 ဆိုပြီး s တန်ဖိုးကို ရှာပြီးတဲ့အခါမှာ a ကို တစ်တိုးခိုင်းတဲ့အတွက် a=6 ဖြစ်ပါမယ်။ ပြီးရင် while statement မှာ condition စစ်တော့ i က 5 ဖြစ်တဲ့အတွက် condition က fail ဖြစ်သွားပါမယ်။ ဒါကြောင့် do statement ကိုထပ်မံ run တော့ဘဲ program က ပြီးဆုံးသွားမှာ ဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

7.Arrays

C language မှာ Array ကို ဘာကြောင့်သုံးလဲဆိုတာကို ဥပမာပြပါမယ်။

```

1  #include<stdio.h>
2  main()
3  {
4      int a=2;
5      a=4;
6      printf("%d",a);
7  }
```



Line_4. ဒီ program မှာ ပထမ a ကို 2 လို့ assign လုပ်ပြီး သိမ်းထားပေးပါတယ်။

Line_5 ဒုတိယ statement မှာတော့ a ကို 4 လို့ ထပ်ပြီး assign လုပ်ထားပါတယ်။ ဒီတော့ a ထဲမှာ မူလ သိမ်းထားတဲ့ 2 ရှိမနေတော့ဘဲ။ ဒုတိယ assign လုပ် တဲ့ 4 ပဲရှိနေပါတယ်။

Line_6. ဒါကြောင့် printf() function သုံးပြီး a ကို output ထုတ်တဲ့အခါမှာ 4 ကို display ပြခြင်းဖြစ်ပါတယ်။

ဒီ program ကို ကြည့်ပြီး array ကို ဘာကြောင့်သုံးလဲဆိုတာကို ပြောပြပါမယ်။ ရိုးရိုး variable တစ်ခုဟာ တစ်ကြိမ်မှာ value တန်ဖိုး တစ်ခုထဲကိုပဲ သိမ်းဆည်းထားပေးနိုင်ပါတယ်။ နောက်ထပ်တန်ဖိုး တစ်ခု သိမ်းဆည်းမယ်ဆိုရင် မူလရှိနေတဲ့ value နေရာကို အစားထိုးသွားပါတယ်။ ဒါဆိုရင် value တစ်ခုစီအတွက် variable name တစ်ခုစီပေးပြီးကြေငြာပေးရပါမယ်။ အဲလိုမျိုး variable တွေအများကြီးမကြေငြာဘဲ variable တစ်ခုထဲမှာ တန်ဖိုးတွေ များများသိမ်းထားချင်တဲ့အခါမှာ Array ကို အသုံးပြုရေးသားကြပါတယ်။

One dimensional Array

Array တွေကို အသုံးပြုဖို့အတွက် program မှာကြေငြာပေးရမယ့် ပုံစံကို ပြောပြပါမယ်။

```
int a[20];
```

ဒီ ပုံစံအတိုင်းကြေငြာပေးရမှာပါ။ ဒါကတော့ single dimensional array အတွက် ဖြစ်ပါတယ်။ Array တွေကို ရိုးရိုး variable တွေလိုပဲ တန်ဖိုးတွေနဲ့ assign လုပ်ထားလို့ရပါတယ်။ တန်ဖိုးတွေနဲ့ assign လုပ်ပြီးကြေငြာမယ် ဆိုရင်တော့ ခုလိုမျိုးရေးပေးရမှာပါ။

```
int a[5]={1,2,3,4,5};
```

ဒီ declaration နဲ့ ပတ်သတ်ပြီးပြောပြပါမယ်။ a ဆိုတာကတော့ Array name ဖြစ်ပါတယ်။ The bracket([]) ကတော့ ဒီ variable declaration ဟာ Array declaration ပါလို့ Compiler ကို သိအောင်ပြောတဲ့ operator ဖြစ်ပါတယ်။ နောက်ထပ် curly brace({ }) ထဲမှာ ရေးထားတဲ့ value တွေကတော့ Array element တွေဖြစ်ပါတယ်။ ဒီ Array element တွေကို a[5] ထဲမှာ assign လုပ်ပြီးသိမ်းထားပါတယ်။ ဘယ်လိုသိမ်းထားလဲဆိုတာကို ပုံလေးနဲ့ပြပေးပါမယ်။

Element no	a[0]	a[1]	a[2]	a[3]	a[4]
Element	1	2	3	4	5
Address	2000	2002	2004	2006	2008

Array element တွေကို ဒီ table ထဲကအတိုင်း သိမ်းထားတာ ဖြစ်ပါတယ်။ ပိုနားလည်အောင် ရှင်းပြပါမယ်။

a[0] ကတော့ 1st array element 1 ကို ဆိုလိုတာပါ။

a[1] ကတော့ 2nd array element 2 ကိုဆိုလိုတာပါ။

a[2] ကတော့ 3rd array element 3 ကို ဆိုလိုတာပါ။

a[3] ကတော့ 4th array element 4 ကိုဆိုလိုတာပါ။

a[4] ကတော့ 5th array element 5 ကိုဆိုလိုတာပါ။

နားလည်မယ်လို့ထင်ပါတယ်။ element number (the bracket[]) ထဲက number ကို index လို့လည်း ခေါ်ပါတယ်။ index 0 ဆိုရင် 1st element ကို ကိုယ်စားပြုပါမယ်။ index 1 ဆိုရင် 2nd element စသည်ဖြင့်ပေါ့။ ဒါနဲ့ပတ်သက်ပြီး မှတ်ထားစေချင်တာက array element တွေကို

သိမ်းထားတဲ့အခါမှာ ဘယ်တော့ပဲဖြစ်ဖြစ် index 0 ကနေပဲ စပြီးသိမ်းပါတယ်။ index 1 ကနေမစပါဘူး။

**** Array သုံးပြီး ကြေငြာတဲ့အခါမှာ နောက်ထပ်သိထားသင့်တာလေးကို ပြောပြပါမယ်။ ဥပမာ int a[5]={1,2,3,4} လို့ assign လုပ်ထားတယ်ဆိုပါစို့။ ကျွန်တော် တို့က program ရေးသားနေရင်းနဲ့ array element 3 ကို 6 လို့ပြောင်းပြီး assign လုပ်ချင်တယ်ဆိုပါစို့။ ဒါဆိုရင် လွယ်လွယ်လေးပါ။ a[2]=6; လို့ assign လုပ်ပေးလိုက်ရုံပါပဲ။ index 2 က 3 ကို ကိုယ်စားပြုတာလေ။ ဒီတော့ index 2 ကို 6 လို့ assign လုပ်လိုက်တဲ့ အတွက် မူလရှိနေတဲ့ index 2 က 3 ကို ပယ်လိုက်ပြီး 6 ကို သိမ်းထားလိုက်မှာ ဖြစ်ပါတယ်။ ဒါဆို ကောင်း ကောင်းနားလည်မယ်လို့ ထင်ပါတယ်။

***** နောက်တစ်ခု Array ထဲမှာ input value တွေသိမ်းဆည်းတာကို ပြောပြပါမယ်။

```
for(i=0;i<=20;i++)
{
printf("Enter marks");
scanf("%d",&marks[i]);
}
```

ဒီ program မှာဆိုရင် Enter marks ဆိုပြီး user ကို value တွေတောင်းပါတယ်။ for loop statement ကိုသုံး ထားတာဖြစ်တဲ့အတွက် 0 to 20 ထိကိုပဲ လက်ခံပြီး သိမ်းပေးမှာ ဖြစ်ပါတယ်။ ဘယ်လိုသိမ်းထားလဲဆိုတော့ ရှေ့ကပြောခဲ့တဲ့အတိုင်းပါပဲ။

```
marks[0]=1
marks[1]=2
```

```
.....
```

```
.....
```

```
.....
```

```
marks[19]=20
```

ခုလိုမျိုးသိမ်းထားမှာ ဖြစ်ပါတယ်။ ပထမ user ရိုက်လိုက်တဲ့ value ကို index 0 အဖြစ်သိမ်းထားပေးမှာ ဖြစ်ပါတယ်။ တကယ်လို့ 20 လို့ user က ပထမရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆို marks[0]=20 ဆိုပြီး သိမ်းထား ပေးမှာဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

***** နောက်တစ်ခု သိထားသင့်တာကတော့ Array element ပမာဏကို သိမ်းဆည်းထားနိုင်မှုဟာ ဒီ array element တွေကို သိမ်းထားမယ့် variable ရဲ့ data type နဲ့ size အပေါ်မှီခိုနေပါတယ်။ ဆိုလိုတာကတော့ ဗျာ integer data type အမျိုးအစားဖြစ်မယ်ဆိုရင် memory မှာ 2 byte နေရာယူမယ်လို့ဆိုလိုတာပါ။ အဲလိုမျိုး array တွေရဲ့ size ကို သိချင်တယ်ဆိုရင်တော့ sizeof() operator ကိုသုံးပြီး ရှာလို့ရပါတယ်။

Example program လေးတစ်ပုဒ်လောက်ရေးပြပါမယ်။

```

1  #include<stdio.h>
2  main()
3  {
4  int i[10];
5  char c[10];
6  long l[10];
7  printf("The type int requires %d bytes\n",sizeof(int));
8  printf("The type char requires %d bytes\n",sizeof(char));
9  printf("The type long requires %d bytes\n",sizeof(long));
10 printf(" %d memory locations are reserved for tin int elements\n",sizeof(i));
11 printf(" %d memory locations are reserved for tin char elements\n",sizeof(c));
12 printf(" %d memory locations are reserved for tin long elements\n",sizeof(l));
13 }

```

```

The type int requires 4 bytes
The type char requires 1 bytes
The type long requires 4 bytes
40 memory locations are reserved for tin int elements
10 memory locations are reserved for tin char elements
40 memory locations are reserved for tin long elements

```

ဒီ program ဆိုရင် i ကို array integer type variable လို့ကြေငြာထားပါတယ်။ c ကိုတော့ char type array , l ကိုတော့ long type array လို့ကြေငြာထားပါတယ်။ ပြီးရင် 10 array elements လို့ သတ်မှတ်ထားပါတယ်။

ဒါကြောင့် sizeof() operator ကိုသုံးပြီး memory မှာ bytes ဘယ်လောက် နေရာယူမလဲဆိုတာကို တွက်ချက် ထားတာဖြစ်ပါတယ်။ ပထမ statement မှာ sizeof(int) လို့ ရေးထားတဲ့အတွက် integer data type ရဲ့ memory နေရာယူမှု byte ကို display ပြပေးခြင်းဖြစ်ပါတယ်။ ဘယ် data type ဆိုရင် byte ဘယ်လောက် နေရာယူမလဲဆိုတာကို ရှေ့မှာလည်း ပြောပြခဲ့ပြီးပါပြီ။ sizeof(char),sizeof(long) ဆိုလည်း ဒီအတိုင်းပါပဲ။ နောက်ထပ် sizeof(i) ဆိုပြီး output display ပြခိုင်းတဲ့အတွက် 40 memory locations နေရာယူပါမယ်လို့ တွက်ထုတ်ပေးတာ ဖြစ်ပါတယ်။ ဘယ်လိုတွက်ထုတ်ပေးလဲဆိုတော့ array name i ထဲမှာ သိမ်းထားတဲ့ integer data type array element တစ်ခုချင်းစီက 4 byte နေရာယူပါတယ်။ ဒါကြောင့် 10 ခုဖြစ်တဲ့အတွက် 10×4 ဆိုပြီး အားလုံးပေါင်း 40 bytes ဆိုပြီး တွက်ထုတ်ပေးတာ ဖြစ်ပါတယ်။ sizeof(c),sizeof(l) ကို ဒီအတိုင်းတွက်ထုတ်ပြီး display ပြပေးတာဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

နောက်ထပ် Character array (string) တွေသိမ်းဆည်းပြီး output ထုတ်တာကို လေ့လာ ကြည့်ရအောင်။


```

1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5      int i=0;
6      char name[10]={'A', 'R', 'R', 'A', 'Y'};
7      while(name[i]!='\0')
8      {
9          printf("%c",name[i]);
10         i++;
11     }
12 }
13

```



ဒီ program မှာဆိုရင် 'A' , 'R' , 'R' , 'A' , 'Y' ဆိုတဲ့ character တွေကို Array element အနေနဲ့ ကြေငြာထားတာ ဖြစ်ပါတယ်။ အဲဒီလို character အစုအဝေးကို string လို့ခေါ်ပါတယ်။ ဒီ string တွေကို memory မှာ ဘယ်လို သိမ်းထားလဲဆိုတာကို ပြောပြပါမယ်။

Array element	A	R	R	A	Y	\n
Array no.	name[0]	name[1]	name[2]	name[3]	name[4]	name[5]

ဒီ table လေးကို တွေ့ပြီးဆိုတော့ ဘယ်လိုသိမ်းထားလဲဆိုတာကို နားလည်မယ်လို့ ထင်ပါတယ်။ အိုကေ.. program ကို ရှင်းပြပါမယ်။ Array declaration နဲ့ပတ်သတ်ပြီး သိထားပြီး ဆိုတော့ while loop statement က စပြီး ရှင်းပြပါမယ်။

Line_7. while(name[i]!='\0') ရဲ့ အဓိပ္ပာယ်ကို ပြောပြပါမယ်။ name[i] က null (\0) နဲ့ မတူမချင်း scope ထဲက statement တွေကို run ပေးပါလို့ဆိုလိုတာပါ။ ဘာကြောင့် အဲလိုရေးလဲဆိုတာကို ပြောပြပါမယ်။ string တွေကို array element အဖြစ် ကြေငြာမယ်ဆိုရင် string element ရဲ့ အဆုံးကို \n(null character) နဲ့ auto ပိတ်ပေးပါတယ်။ ဒီ table မှာဆိုရင် string element ရဲ့ နောက်ဆုံး character ဖြစ်တဲ့ 'y' နဲ့ မဆုံးဘဲ \n နဲ့ ဆုံးထားပါတယ်။ နားလည်မယ်လို့ထင်ပါတယ်။ ကဲ..ဒီတော့ program မှာ i ကို 0 လို့ assign လုပ်ပေးထား ပါတယ်။ ဒီတော့ name[0] ဖြစ်ပါမယ်။ while statement နဲ့ စစ်တော့ name[0] ဟာ \n နဲ့ မတူ တာဖြစ်တဲ့ အတွက် printf() function ကို run ပြီး A ကို display ပြခိုင်းမှာဖြစ်ပါတယ်။ ပြီးရင် i ကို တစ်တိုးခိုင်း တာဖြစ်တဲ့အတွက် i=1 ဖြစ်ပါမယ်။ ဒီတော့ name[1]=R ဖြစ်တဲ့အတွက် R ကို display ပြပေးမှာဖြစ်ပါတယ်။ နောက်ဆုံး i တန်ဖိုး 5 , name[5] ဖြစ်တဲ့အချိန်မှာ \n character နဲ့ ညီသွားပြီ။ ဒါကြောင့် program က while statement ကို မ run တော့ဘဲ program က ပြီးဆုံးသွားပါမယ်။ နားလည်မယ်လို့ထင်ပါတယ်။

နောက်ထပ် 1 to 10 ထဲက စုံကိန်း၊ မကိန်း ခွဲပြီး စုံကိန်းအားလုံးပေါင်းလဒ်နဲ့ မကိန်းအားလုံးပေါင်းလဒ် ကိုရှာ တဲ့ program လေးရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  #include<conio.h>
3  main()
4  {
5      int sumo=0,sume=0; /* sumo=sum of odd , sume=sum of even*/
6      int i=0, odd[5],even[5];
7      int a=-1,b=-1;
8      for(i=1;i<=10;i++)
9      {
10         if(i%2==0)
11         {
12             even[++a]=i; //even[0]=2,even[1]=4.....even[4]=10
13         }
14         else
15         {
16             odd[++b]=i; //odd[0]=1,odd[1]=3,.....odd[4]=9
17         }
18     }

19     printf("\n\t Even\t\tOdd");
20
21     for(i=0;i<5;i++)
22     {
23         printf("\n\t %d\t\t %d",even[i], odd[i]);
24         sume=sume+even[i];
25         sumo=sumo+odd[i];
26     }
27     printf("\n\t=====n");
28     printf("Addition:%d \t\t %d ",sume,sumo);
29 }

```

```

      Even      Odd
      2         1
      4         3
      6         5
      8         7
     10         9
=====
Addition:30      25
=====
Process exited with return value

```

ဒီ program ကိုနားလည်အောင်သေချာကြည့်ပါ။ variable name ကြေငြာတဲ့အပိုင်းကိုတော့ ရှင်းမပြောဘူး။ for loop statement ကစပြီး ရှင်းပြပါမယ်။

Line_8. for(i=1;i<=10;i++) ဆိုတဲ့ statement ကတော့ i တန်ဖိုး 0 ကစပြီး loop ပတ်ပြီး တစ်တိုးတဲ့အခါ 10 နဲ့ မတူမချင်း for loop ရဲ့ scope ထဲက statement တွေကို run ပါမယ်လို့ ဆိုလိုတာပါ။

Line_10. ပထမဆုံး statement ကတော့ `if(i%2==0)` ဆိုပြီး `if` နဲ့ စစ်ခိုင်းတာပါ။ `i` ကို 2 နဲ့ စားတဲ့ အခါမှာ အကြွင်း သည် 0 နှင့်ညီလျှင် `if` block ကို run ပါမယ်။ မညီရင် `else` block ကို run ပေးမှာပါ။ ဒီတော့ `i` က 1 လို့ start လုပ်ထားတဲ့အတွက် 2 နဲ့စားတော့ 0 နဲ့မညီပါဘူး။ ဒါကြောင့် `else` block ထဲက statement ကို run ပါမယ်။

Line_16. `odd[++b]=i;` လို့ ဆိုတဲ့အတွက် `b` ကို တစ်တိုးပါမယ်။ ဘာလို့လည်းဆိုတော့ prefix operator ကို သုံးထားတဲ့အတွက်ဖြစ်ပါတယ်။ `b=-1` လို့ assign လုပ်ထားတာဖြစ်တဲ့အတွက် `odd[0]=1` ဖြစ်ပါမယ်။ နားလည်မယ်လို့ထင်ပါတယ်။ `for` loop သုံးထားတာဖြစ်တဲ့အတွက် loop ပတ်ပါမယ်။ `i` ကို တစ်တိုးခိုင်းတာ ဖြစ်တဲ့အတွက် `i=2` ဖြစ်သွားပါပြီ။ ပြီးရင် `if` နဲ့စစ်ပါမယ်။ ဒီတော့ `2%2=0` ဖြစ်တဲ့အတွက် `if` block ထဲက statement ကို run မှာဖြစ်ပါတယ်။

Line_12. `even[++a]=i` ဆိုတဲ့အတွက် `a` ကိုတစ်တိုးတဲ့အခါ `-1+1` ဖြစ်ပါမယ်။ ဒီတော့ `even[0]=2` ဖြစ်ပါမယ်။ နားလည်မယ်လို့ထင်ပါတယ်။ array element တွေကို သိမ်းတဲ့အခါမှာ 1st element ကို index 0 လို့သတ်မှတ်ပြီး သိမ်းပေးမှာဖြစ်ပါတယ်။ program ဘေးမှာလည်း ရေးပြထားပါတယ်။

ပြီးရင် `i` တန်ဖိုးကို တစ်တိုးမယ်။ဒီတော့ `i=3` ဖြစ်သွားမယ်။ 2 နဲ့စားမယ်။ 0 နဲ့ညီလားစစ်မယ်။ မညီတဲ့ အတွက် `else` block ထဲက statement ကို run မယ်။ဒီတော့ `b` ကို တစ်တိုးမယ်။ `b` က 0 ဖြစ်တဲ့အတွက် 1 ဖြစ်သွားမယ်။ ဒါကြောင့် `odd[1]=3` ဖြစ်မယ်။ နားလည်တယ်ဟုတ်။ ဒီအတိုင်းပဲ `i` ကို တစ်တိုးလိုက် အကြွင်း 0 နဲ့ညီလားစစ် မယ်။ညီရင် `if` block ထဲက statement ကို run မယ်။ မညီရင် `else` block ထဲက statement ကို run မယ်။ ဒီလို စစ်ပြီးတာနဲ့ `even` နဲ့ `odd` ထဲမှာ အောက်က table အတိုင်းသိမ်းသွားမယ်။

<i>even[0]</i>	<i>even[1]</i>	<i>even[2]</i>	<i>even[3]</i>	<i>even[4]</i>
2	4	6	8	10

<i>odd[0]</i>	<i>odd[1]</i>	<i>odd[2]</i>	<i>odd[3]</i>	<i>odd[4]</i>
1	3	5	7	9

ဒီ table ရလာတာကိုတော့ နားလည်မယ်လို့ထင်ပါတယ်။ `for` loop ကို run ပြီးတာနဲ့ program က နောက် ထပ် statement ကို run ပါမယ်။ `printf("\n\t Even\t\tOdd");` ဆိုတဲ့အတွက် Even နဲ့ Odd ဆိုတဲ့ စာသားကို display ပြပေးပါမယ်။ `\t` ကိုရေးထားတဲ့အတွက် Even နဲ့ Odd ကြားမှာ tab ခြားပေးမှာဖြစ်ပါတယ်။

ပြီးရင် နောက်ထပ် statement ကို run တော့ `for(i=0;i<5;i++)` ဆိုတဲ့အတွက် loop ထပ်ပတ်ပါမယ်။ `i` ကို 0 လို့ start လုပ်ထားတာဖြစ်တဲ့အတွက် `i` တန်ဖိုးက 0 ဖြစ်ပါမယ်။ ဒီတော့ `even[i]` က `even[0]` ဖြစ်ပါမယ်။ ပြီးရင် `odd[i]` ကလည်း `odd[0]` ဖြစ်ပါမယ်။ ဒါကြောင့် `even[0]` ရဲ့ element ဖြစ်တဲ့ 2 ကို ပြမယ်၊ `odd[0]` ရဲ့ element 1 ကို display ပြပေးပါမယ်။ နားလည်တယ်ဟုတ်။

ဒီအတိုင်းပဲ loop ပတ်ပြီး i ကို တစ်တိုးသွားပါမယ်။ ပြီးရင် element တွေကို display ပြပေးမှာဖြစ်ပါတယ်။

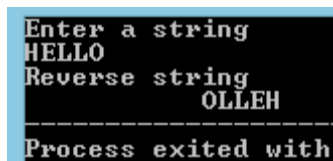
sume=sume+even[i]; sumo=sumo+odd[i]; ဒီ statement တွေကိုတော့ i ကို တစ်တိုးတာနဲ့ တစ်တည်းပေါင်း ပေးမှာ ဖြစ်ပါတယ်။ ပထမ loop ပတ်တဲ့အချိန်မှာ i=0 ဖြစ်တဲ့အတွက် even[0] ရဲ့ တန်ဖိုး 2 နဲ့ odd[0] ရဲ့တန်ဖိုးတွေကို ပေါင်းပေးပါမယ်။ ဒီတော့ sume=0+2 နဲ့ sumo=0+1 ဆိုပြီးပေါင်းပေးမှာ ဖြစ်ပါတယ်။ sume,sumo ကို 0 လို့ ကြေငြာထားတယ်လေ။ နားလည်တယ် ဟုတ်။ ဒီအတိုင်းပဲ even[i] ရဲ့ element တွေကို ပေါင်းပြီး sume ထဲမှာသိမ်းထားမယ်။ odd[i] ရဲ့ element တွေ ကိုလည်း ပေါင်းပြီး sumo ထဲမှာပေါင်းထားမယ်။ အဲလိုပေါင်းလိုက်တဲ့အခါမှာ sume=30 နဲ့ sumo=25 ဆိုပြီး ရပါမယ်။ ဒီတန်ဖိုးကို user မြင် အောင်ပြဖို့အတွက် Output ထုတ်ခိုင်းမယ်။

ဒါကြောင့် printf("Addition:%d \t\t %d ",sume,sumo); ဆိုပြီး printf() function နဲ့ display ပြခိုင်းလိုက်ပါတယ်။ ဒါဆိုရင် program ကို run တဲ့ အခါမှာ အပေါ်က ပြထားတဲ့အတိုင်း display ပြတာကို နားလည်မယ်လို့ထင်ပါတယ်။

ဒီတစ်ခါ user ဆီက စာသားတစ်ခုရိုက်ခိုင်းမယ်။ ပြီးရင် ဒီစာသားကို ပြောင်းပြန်ပေါ်အောင် display ပြခိုင်း တဲ့ program လေးရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  main()
3  {
4  static char s[15];
5  int i;
6  puts("Enter a string");
7  gets(s);
8  puts("Reverse string");
9  for(i=15;i>=0;i--)
10 printf("%c",s[i]);
11 }
```



```

Enter a string
HELLO
Reverse string
OLLEH
Process exited with
```

ဒီ program ဘယ်လိုအလုပ်လုပ်သွားလဲဆိုတာကို ရှင်းပြပါမယ်။ variable name declare လုပ်တဲ့နေရာမှာ static ဆိုတဲ့ keyword ကိုရေးပေးတာကို ပြောပြပါမယ်။ memory မှာ data တွေကို သိမ်းဆည်းတဲ့အခါ data type အမျိုးအစားကို ကြည့်ပြီး သိမ်းဆည်းပေးပါတယ်။

ဒီ program မှာ s[15] ဆိုတဲ့ array မှာ string data type တွေကို သိမ်းဆည်းဖို့ဖြစ်ပါတယ်။ string array type တွေကို သိမ်းဆည်းတဲ့အခါမှာ static storage class(or) external storage class

မှာ သိမ်းပေးပါတယ်။ ဒါကြောင့် static storage class မှာ သိမ်းမှာဖြစ်တဲ့အတွက် static ဆိုတဲ့ keyword ကိုရေးပေးတာဖြစ်ပါတယ်။ ဒီထိန်းလည်မယ်နော်။ ပြီးရင် Enter a string ဆိုတဲ့ စာသားကို display ပြဖို့ အတွက် puts() function ကို သုံးထားပါတယ်။

printf() သုံးလည်းအတူတူပါပဲ။ puts() ဆိုတာက put string လို့ဆိုလိုတာပါ။ puts ရဲ့ parenthesis() ထဲက string data တွေကို display ပြခိုင်းတာဖြစ်ပါတယ်။ ပြီးရင် user ရိုက်လိုက်တဲ့တန်ဖိုးတွေကို လက်ခံပေး တာကတော့ gets(s) function ဖြစ်ပါတယ်။ scanf () function နဲ့အတူတူပါပဲ။ gets(s) ဆိုတာကတော့ get string လို့ဆိုလိုပါတယ်။ ပြီးရင် gets ရဲ့ parenthesisထဲက variable (argument) ထဲမှာ သိမ်းထားပေးမှာ ဖြစ်ပါတယ်။ ဘယ်လိုသိမ်းထားလဲဆိုတော့ s ဆိုတာက array name ပါ။ array name ဆိုတာက Address ပဲဖြစ်ပါတယ်။ pointer ခန်းမှာပြောပြပါဦးမယ်။

ဒီတော့ user ရိုက်လိုက်တဲ့ string တွေရဲ့ address ကို သိမ်းထားလိုက်တာ ဖြစ်ပါတယ်။ သူကလည်း scanf() functionလိုပါပဲ။ user ရိုက်လိုက်တဲ့ data တွေကို သိမ်းဆည်းပေးရုံမဟုတ်ဘဲ user မြင်အောင် ပြန်ပြပေးပါ တယ်။ ဥပမာ user က HELLO လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆိုရင် အပေါ်မှာပြထားတဲ့ အတိုင်း HELLO ကို ပြန်ပြ ပေးပါတယ်။ နောက် statement တစ်ခုက puts function နဲ့ Reverse String ဆိုတဲ့ စာသားကို display ပြခိုင်း တာဖြစ်ပါတယ်။ ဒီထိန်းလည်မယ်လို့ ထင်ပါတယ်။ ဒါဆို string data ကို memory မှာ သိမ်းဆည်းထားပုံကို table လေးနဲ့ပြပေးပါမယ်။

s[0]	s[1]	s[2]	s[3]	s[4]
H	E	L	L	O

အိုကေ...ဒါဆို programကိုဆက်ရှင်းရအောင်။ for(i=15;i>=0;i--) ဆိုတဲ့အတွက် start i=15 ဖြစ်ပါမယ်။ ဒါဆိုရင် s[15] လို့ compiler က နားလည်လိုက်ပါမယ်။ ဒါပေမယ့် s[15] အတွက် သိမ်းဆည်းထားတဲ့ data မရှိ ပါဘူး။ ဒါကြောင့် ဘာတန်ဖိုးမှာ display ပြမပေးပါဘူး။ step မှာ တစ်လျှော့ခိုင်းတာဖြစ်တဲ့အတွက် i=14 ဖြစ်ပါ မယ်။ ဒါဆို s[14] ဖြစ်သွားမယ်။ s[14] မှာလည်း data display ပြဖို့မရှိပါဘူး။

ဒီအတိုင်းပဲ Loop ပတ်ပြီး i ကို တစ် လျှော့သွားပါမယ်။ နောက်ဆုံး i=4 ဖြစ်တဲ့အချိန်မှာ s[4] ဖြစ်ပါမယ်။ ဒီတော့ s[4] ရဲ့ array element ဖြစ်တဲ့ O ကို display ပြပေးမှာဖြစ်ပါတယ်။ ပြီးရင် i ကို တစ်လျှော့မယ်။ i=3 ဆိုတော့ s[3] ရဲ့ array element L ကို display ပြပေးမှာဖြစ်ပါတယ်။ ဒါကြောင့် i=0 ဖြစ်တဲ့ချိန်မှာ s[0] ရဲ့ element H ကို display ပြလိုက်တဲ့အတွက် ဒီ output element တွေက input element နဲ့ပြောင်းပြန် ဖြစ်နေတာပါ။ နားလည်မယ်လို့ထင်ပါတယ်။

ဒီလောက်ဆိုရင်တော့ One Dimensional Array ကို ကောင်းကောင်း နားလည်မယ်လို့ ထင်ပါတယ်။

Two Dimensional Array

One dimensional array ကို သိပြီးဆိုတော့ Two dimensional array လည်း မခက်တော့ပါဘူး။ နားလည် အောင် program တစ်ပုဒ်လောက်ရေးပြီး ရှင်းပြပါမယ်။

```

1  #include<stdio.h>
2  main()
3  {
4  int i,j;
5  int a[3][3]={ {1,2,3},{4,5,6},{7,8,9}};
6  printf("Array elements and address\n");
7  for(i=0;i<3;i++)
8
9      for(j=0;j<3;j++)
10 {
11     printf("%d \t\t %d\n",a[i][j],&a[i][j]);
12 }
13 }
14 }
```

```

Array elements and address
1      2686644
2      2686648
3      2686652
4      2686656
5      2686660
6      2686664
7      2686668
8      2686672
9      2686676

-----
Process exited with return value
Press any key to continue
```

ဒီ program မှာ ဆိုရင် Array element 1 to 9 ကို matrix ပုံစံနဲ့ သိမ်းထားတာဖြစ်ပါတယ်။ 3 row ,3 column ပုံစံနဲ့ ဖြစ်ပါတယ်။ ရှေ့က one dimensional array က row အတွက် ဖြစ်ပါတယ်။ နောက်က array ကတော့ column အတွက်ဖြစ်ပါတယ်။

	Column1	Column2	Column3
Row 1	1	2	3
Row2	4	5	6
Row3	7	8	9

ပုံကိုကြည့်ပြီးရှင်းရင် ပိုအဆင်ပြေမယ်ထင်ပါတယ်။ ပထမ {1,2,3} မှာဆိုရင် 1 row 3 column အတွက် ဖြစ်ပါတယ်။ ဒုတိယ {4,5,6} ဆိုတာကတော့ 2 row 3 column အတွက်ဖြစ်ပါတယ်။ တတိယ {7,8,9} ဆိုတာ ကတော့ 3 row 3column အတွက်ဖြစ်ပါတယ်။ ဒါက 3 row 3 column

အတွက်ဖြစ်ပါတယ်။ တကယ်လို့ 2row 3column အတွက်ဆိုရင်တော့ $a[2][3]=\{\{1,2,3\},\{4,5,6\}\}$ လို့ကြေငြာပေးရမှာ ဖြစ်ပါတယ်။ နားလည်မယ် လို့ထင်ပါတယ်။

	Column1	Column2	Column3
Row 1	$a[0][0]$	$a[0][1]$	$a[0][2]$
Row2	$a[1][0]$	$a[1][1]$	$a[1][2]$
Row3	$a[2][0]$	$a[2][1]$	$a[2][2]$

ဒါကတော့ ဒီ array element တွေကိုသိမ်းဆည်းထားပုံဖြစ်ပါတယ်။ program ကိုရှင်းရင် ပိုနားလည်မယ်လို့ ထင်ပါတယ်။ ဒီ Two dimensional array မှာသိမ်းထားတဲ့ 1 to 9 တန်ဖိုးတွေကို ဘယ်လို display ပြပေးလဲ ဆိုတာကို ရှင်းပြပါမယ်။ Two dimension array element တွေကို display ပြဖို့အတွက် variable 2 ခုသုံးရ ပါမယ်။ ပြီးရင် nested for loop ကိုလည်းသုံးရပါတယ်။

ဒီ program မှာဆိုရင် variable i နဲ့ j ကို သုံးထားပါတယ်။ for loop statement ကနေပြီး စတင်ပြပါမယ်။ Outer loop မှာ $\text{for}(i=0;i<=3;i++)$ ဆိုတဲ့အတွက် i တန်ဖိုးက 0 ဖြစ်ပါမယ်။ ပြီးရင် inner loop ကို run ပါမယ်။ inner loop မှာ $\text{for}(j=0;j<=3;j++)$ ဆိုတဲ့အတွက် j တန်ဖိုးက လည်း 0 ဖြစ်ပါမယ်။ ဒီတော့ $a[i][j]$ မှာ $a[0][0]$ လို့ compiler က နားလည်သွားမှာ ဖြစ်ပါတယ်။ ပြီးရင် $a[0][0]$ မှာ ပထမ array element 1 ကို သိမ်းပေးမှာဖြစ်ပါတယ်။

ဒါကြောင့် $a[i][j]$ ကို display ပြခိုင်းတဲ့အခါမှာ 1 ကို display ပြပေးတာဖြစ်ပါတယ်။ နားလည်တယ်ဟုတ်။ ပြီးရင် loop ပတ်ထားတာ ဖြစ်တဲ့ အတွက် inner loop ကို ပြန်ပတ်မှာ ဖြစ်ပါတယ်။ ဒါကိုတော့ သိမယ်လို့ထင်ပါတယ်။ inner loop မှာ stop ဖြစ်တဲ့ အချိန်မှ outer loop ကို ပြန်ပတ်မှာဖြစ်ပါတယ်။ ဒီတော့ j ကို တစ်တိုးခိုင်းတာဖြစ်တဲ့အတွက် $j=1$ ဖြစ်ပါမယ်။ ဒါကြောင့် $a[0][1]$ ဖြစ်သွားပါမယ်။

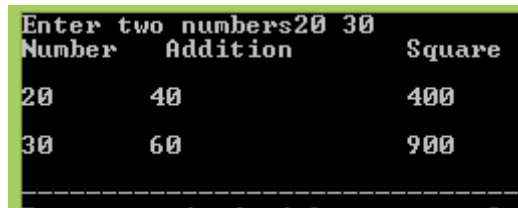
ပြီးရင် ဒုတိယ element 2 ကို သိမ်းထားပေးမှာဖြစ်ပါတယ်။ $a[i][j]$ ကို display ပြခိုင်းတဲ့အတွက် $a[0][1]$ မှာသိမ်းထားတဲ့ 2 ကို display ပြပေးတာဖြစ်ပါတယ်။ ပြီးရင် တစ်တိုးမယ်။ $i=2$ ဖြစ်မယ်။ $a[i][j]$ မှာလည်း $a[0][2]$ ဖြစ်သွားမယ်။ တတိယ element 3 ကို display ပြပေးမှာဖြစ်ပါတယ်။ i ကိုတစ်တိုးပါမယ်။ ဒီတော့ $=3$ ဖြစ်ပါမယ်။ $a<3$ ဆိုတဲ့အတွက် condition က မှားသွားပြီ။ ဒီတော့ outer loop ကို ပြန်သွားပြီး run ပါမယ်။ i ကို တစ်တိုးခိုင်းတာဖြစ်တဲ့အတွက် $i=1$ ဖြစ်ပါမယ်။ ပြီးရင် inner loop ကို သွားပြီး run ပါမယ်။ inner loop မှာ j ကို 0 လို့ start လုပ်ခိုင်းတာဖြစ်တဲ့အတွက် $j=0$ ဖြစ်ပါမယ်။

ဒီတော့ $a[i][j]$ က $a[1][0]$ ဖြစ်သွားပါမယ်။ ပြီးရင် တတိယ element 4 ကို သိမ်းပြီး display ပြပေးမှာဖြစ်ပါတယ်။ ဒီအတိုင်းဘဲ inner loop မှာ j ကို တစ်တိုးမယ် element တွေကို display ပြပေးပါမယ်။ $j=3$ ဖြစ်တဲ့အချိန်မှာ outer loop ကို ထပ်ပတ်မယ်။ i ကို တစ်တိုးမယ်။ inner loop ကို ပတ်မယ်။ element တွေကို display ပြမယ်။ j ကို တစ်တိုးမယ်။ loop ပတ်မယ်။ stop ဖြစ်သွားရင် outer loop ကို runပါမယ်။ နားလည်မယ်လို့ထင်ပါတယ်။ နောက်ဆုံး outer loop မှာ $i=3$ ဖြစ်တဲ့အချိန်မှာ stop ($i<3$) ဆိုတဲ့အတွက် program ကပြီးသွားပါမယ်။ ဒါဆို ဒီ program ကို နားလည်မယ်လို့ထင်ပါတယ်။ အဆင်ပြေပါစေ.....။

ဒီတစ်ခါ user ဆီက input value တခုတောင်းမယ်။ ပြီး ရင် အဲ့တန်ဖိုးရဲ့ ပေါင်းလဒ်နဲ့ နှစ်ထပ်ကိန်းကို display ပြပေးမယ့် program လေးရေးကြည့်ရအောင်။

```

1  #include<stdio.h>
2  #include<math.h>
3  main()
4  {
5      int i,j=0,a[2][2],b[2][2];
6      printf("Enter two numbers");
7      for(i=0;i<2;i++)
8          scanf("%d",b[i]);
9      for(i=0;i<2;i++)
10     {
11         a[i][j]=b[i][j]*2;
12         a[i][j+1]=pow(b[i][j],2);
13     }
14     printf("Number\t Addition\t Square\n");
15     for(i=0;i<2;i++)
16     {
17         printf("\n%d",*b[i]);
18         for(j=0;j<2;j++)
19             printf("\t%d\t",a[i][j]);
20         printf("\n");
21     }
22 }
```



Number	Addition	Square
20	40	400
30	60	900

ဒီ program ကိုရှင်းပြပါမယ်။ ပထမဆုံး variable declaration ပိုင်းကိုတော့ ရှင်းမပြော့ပါဘူး။ ပထမ USER ဆီ ကနေ တန်ဖိုးနှစ်ခုကို ရိုက်ခိုင်းပါမယ်။ နှစ်ခုထက်ပိုလို့ မရပါဘူး။ ဘာလို့လည်းဆိုတော့ for(i=0;i<2;i++)ဆိုပြီး scanf() function ရဲ့ရှေ့မှာ ကန့်သတ်ထားတဲ့အတွက် ဖြစ်ပါတယ်။ တန်ဖိုး နှစ်ခုထက်ပိုရင်တော့ variable b[i] ထဲမှာ သိမ်းထားပေးမှာ မဟုတ်ပါဘူး။ ဥပမာ user က 5 နဲ့ 4 လို့ရိုက်လိုက်တယ် ဆိုပါစို့။ ဒါဆိုရင် b[0]= 2686648=5 နဲ့ b[1]=2686656=4 ဆိုပြီး သိမ်းထားပေးမှာ ဖြစ်ပါတယ်။ b[0],b[1] ဆိုတာက Address နော်။ 5 နဲ့ 4 ရဲ့ Address ကို သိမ်းထားတာ။ ဆိုလိုတာက b[0],b[1] ကို display ပြခိုင်းရင် 5 နဲ့ 4 ကို ပြမပေးဘူး။ 5 နဲ့ 4 ကို သိမ်းထားတဲ့ Address ကိုပဲ ပြပေးမှာ။ နားလည်မယ်လို့ ထင်ပါတယ်။

တန်ဖိုးတွေကို သိမ်းပြီးသွားရင် နောက် statement ကို run ပါမယ်။ for(i=0;i<2;i++) ဆိုတဲ့အတွက် ပထမ စ run တဲ့ချိန်မှာ i တန်ဖိုးက 0 ဖြစ်ပါမယ်။ ပြီးရင် for loop ရဲ့ scope ထဲက statement တွေကို run ပါမယ်။ a[i][j]=b[i]*2 ဆိုတာကို a[0][0]=b[0]*2 လို့ နားလည်လိုက်ပါမယ်။ j ရဲ့တန်ဖိုးကို 0 လို့အပေါ်မှာ assign လုပ်ထားပါတယ်။ b[0][0] ရဲ့ တန်ဖိုးက 5

ဆိုတော့ 2 နဲ့ မြှောက်တဲ့အခါမှာ 10 ဖြစ်ပါမယ်။ အဲ့ဒီတန်ဖိုး 10 ကို $a[0][0]$ နဲ့ assign လုပ်ထားတာဖြစ်တဲ့အတွက် $a[0][0]$ ထဲမှာ သိမ်းထားပါမယ်။ ဒီထိန်းလည်တယ်ဟုတ်။

နောက် statement တစ်ကြောင်းက $a[i][j+1]=\text{pow}(b[i][j],2)$ ဆိုတဲ့အတွက် $a[0][0+1]=\text{pow}(5,2)$ လို့နားလည်လိုက်ပါမယ်။ pow function ရဲ့ အလုပ်လုပ်ပုံကိုတော့ နားလည်မယ် ထင်ပါတယ်။ ဒီတော့ $5^2 = 25$ ကိုရပါမယ်။ 25 ကိုတော့ $a[0][1]$ ထဲမှာ သိမ်းထားလိုက်ပါမယ်။ for loop ထဲမှာ statement ကုန်သွားပြီ ဖြစ်တဲ့အတွက် for loop ကို နောက်တစ်ကြိမ် loop ပတ်ပါမယ်။ i ကို တစ်တိုး ခိုင်းထားတာ ဖြစ်တဲ့အတွက် $i=1$ ဖြစ်ပါမယ်။ ပြီးရင် scope ထဲက statement တွေကို run ပါမယ်။ $a[i][j]=b[i][j]*2$ ဆိုတာကို $a[1][0]=b[1][0]*2$ လို့နားလည်လိုက်ပါမယ်။ i နေရာမှာ 1 j နေရာမှာ 0 ကို အစားသွင်းလိုက်တာ ဖြစ်ပါတယ်။ ဒီတော့ $b[1][0]$ က 4 ဖြစ်တဲ့အတွက် 2 နဲ့ မြှောက်တော့ 8 ရပါမယ်။ အဲ့ဒီ 8 ကို $a[1][0]$ ထဲမှာ သိမ်းထားပေးမှာဖြစ်ပါတယ်။

ပြီးရင်နောက် statement $a[i][j+1]=\text{pow}(b[i][j],2)$ ကိုလည်း $a[1][0+1] = \text{pow}(4,2)$ လို့နားလည်လိုက်ပါမယ်။ ဒီတော့ 16 ဖြစ်ပါမယ်။ 16 ကိုလည်း $a[1][1]$ ထဲမှာ သိမ်းထားလိုက်ပါမယ်။ နောက်ထပ် i တန်ဖိုးကို ထပ်တိုးတော့ $i=2$ ဖြစ်ပါမယ်။ ဒါပေမယ့် $i<2$ လို့ stop လုပ်ထားတာဖြစ်တဲ့အတွက်

for loop statementကို ထပ်ပြီးမပတ်တော့ပါဘူး။ ဒီတော့ memory မှာ သိမ်းဆည်း ထားတာလေးကို ပြပေးပါမယ်။

$b[0]$	$b[1]$
2686648	2686656

5	4
2686648	2686656

$a[0][0]$	$a[0][1]$	$a[1][0]$	$a[1][1]$
10	25	8	16

ဒါဆိုရင် program ဒီထိကို နားလည်မယ်လို့ထင်ပါတယ်။ နောက် statement ကို run တော့ Number Addition Square ဆိုတဲ့ စားသားလေးကို display ပြပေးပါမယ်။ ဒါကိုတော့ နားလည်မယ် ထင်ပါတယ်။ \t ကို သုံးထားတဲ့ အတွက် စာသားတစ်ခုနဲ့ တစ်ခုကြားမှာ tab ခြားပေးမှာ ဖြစ်ပါတယ်။ အိုကေ...။ နောက် statement တစ်ခုက Nested loop နဲ့ ရေးထားတာဖြစ်ပါတယ်။ outer loop မှာ $\text{for}(i=0; i<2; i++)$ ဆိုတဲ့အတွက် $i=0$ ဖြစ်ပါမယ်။ ပြီးရင် $* b[i]$ ကို output ထုတ်ခိုင်းပါမယ်။ i က 0 ဖြစ်တဲ့အတွက် $b[0]$ လို့နားလည်လိုက်ပါမယ်။

b[i] ရှေ့က (*) ကို တော့ pointer လို့ခေါ်ပါတယ်။ pointer အကြောင်းကို နောက်ဆုံးခန်းမှာ ရှင်းပြပေးထားပါတယ်။ ဒါကြောင့် b[0] မှာ သိမ်းထားတဲ့ Address ရဲ့ value ကို ရည်ညွှန်းတာ ဖြစ်ပါတယ်။ ဒါကြောင့် b[0] ထဲမှာ သိမ်းထားတဲ့ Address ရဲ့ value ဖြစ်တဲ့ 5 ကို display ပြပေးတာ ဖြစ်ပါတယ်။ ပြီးရင် Inner loop ကို run ပါမယ်။ for(j=0;j<2;j++) ဆိုတဲ့အတွက် j=0 ဖြစ်ပါမယ်။ ပြီးရင် printf() function နဲ့ a[i][j] ကို display ပြခိုင်းတဲ့အတွက် a[0][0] ထဲမှာ သိမ်းထားတဲ့ တန်ဖိုး 10 ကို ပြပေးမှာ ဖြစ်ပါတယ်။ပြီးရင် loop ပတ်ခိုင်းတာဖြစ်တဲ့အတွက် inner loop ကို ထပ် ပတ်ပါမယ်။ ဒီတော့ j=1 ဖြစ်သွားပါမယ်။ပြီးရင် a[i][j] ကို output ထုတ်ခိုင်းတဲ့အတွက် a[0][1] တန်ဖိုး 25 ကို display ပြပေးမှာ ဖြစ်ပါတယ်။ နားလည်တယ်ဟုတ်။ ဒါကြောင့် display ပြတဲ့အခါမှာ-
5 10 25 လို့ပြပေးတာဖြစ်ပါတယ်။

j ကို ထပ်ပြီး တစ်တိုးပါမယ်။ j=2 ဆိုတော့ condition က မှားသွားပြီ ။ဒါကြောင့် outer loop ကိုထပ်ပတ်ပါမယ်။i ကို တစ်တိုးခိုင်းတာဖြစ်တဲ့အတွက် i=1 ဖြစ်ပါမယ်။ ပြီးရင် b[1] ထဲမှာ သိမ်းထားတဲ့ Address ရဲ့တန်ဖိုး 4 ကို display ပြပေးပါမယ်။ inner loop ကိုထပ် run ပါမယ်။ ဒီတော့ j=0 ဖြစ်ပါမယ်။ ပြီးရင် a[1][0] တန်ဖိုး 8 ကို display ပြပါမယ်။ inner loop ကို ထပ်ပတ်ပါမယ်။ J ကို တစ်တိုးပါမယ်။ ဒီတော့ j=1 ဖြစ်ပါမယ်။ ပြီးရင် a[1][1] တန်ဖိုး 16 ကို display ပြပေးမှာဖြစ်ပါတယ်။ ဒါကြောင့် display ပြတဲ့အခါမှာ-

8 16 ဆိုပြီးပြပေးတာဖြစ်ပါတယ်။ inner loop မှာ j ကိုတစ်တိုးခိုင်းမယ်။ ဒီတော့ j=2 ဖြစ်သွား မယ်။ condition ကမှားတဲ့အတွက် outer loop ကို သွားပြီး run မယ်။ i ကို တစ်တိုးခိုင်းတဲ့အတွက် i=2 ဖြစ်မယ်။ ဒီတော့ condition false ဖြစ်ပြီး program ကပြီးဆုံးသွားမှာဖြစ်ပါတယ်။ ဒါဆို ဒီ program ကိုကောင်းကောင်းနား လည်မယ်လို့ ထင်ပါတယ်။

Two Dimensional array နဲ့ပတ်သက်ပြီး ဒီလောက်ပါပဲ။သူတို့ရဲ့ အလုပ်လုပ်ပုံကို သိပြီဆိုတော့ ဘယ် program လာလာမိမိကိုယ်တိုင် နားလည်နိုင်ပါတယ်။ program တွေကို လေ့လာပြီး စဉ်းစားတွက်ချက်ကြည့်ပါ။ အဆင်ပြေပါစေ...။

User defined Functions

ဒီအခန်းမှာတော့ functions တွေအကြောင်းကို ပြောပြပါမယ်။ function မှာ predefined functions (eg. printf(), scanf(),pow()..etc..) နဲ့ userdefined functions(eg.Myfun(), square(), sum()..etc) တို့ဖြစ်ပါတယ်။ ဒီအခန်းမှာ မိမိကိုယ်တိုင် functions တွေကို ဘယ်လိုတည်ဆောက်ရမလဲဆိုတာကို ပြောပြပါမယ်။

Why use userdefined functions

User defined functions တွေကို ဘာကြောင့်သုံးလဲဆိုတော့ program မှာ ဘယ် statement တွေကတော့ ဘာအတွက်ရေးထားလဲဆိုတာကို function တွေခွဲပြီး ရေးသားဖို့အတွက် အသုံးပြုကြ ပါတယ်။ နောက်တနည်း အားဖြင့် main() function ထဲမှာပဲ statement တွေကို စုပုံပြီး မရေးချင်တဲ့ အတွက် သုံးပါတယ်။

General form of functions

Functions တွေကို မိမိကိုယ်တိုင် တည်ဆောက်ပြီးရေးတဲ့အခါမှာ သူ့ရဲ့ syntax ကို သိထားဖို့လိုပါတယ်။ ဒါကြောင့် function တစ်ခုရဲ့ general form ကို ပြောပြပါမယ်။

```
function(argument1,argument2)
```

```
type argument1,argument2;
```

```
{
```

```
Statement;
```

```
Statement;
```

```
}
```

ဒါကတော့ function တစ်ခုရဲ့ general form ဖြစ်ပါတယ်။ argument1,argument2 ဆိုတာကတော့ variab name တွေကို ဆိုလိုတာပါ။ return variable တွေဖြစ်ပါတယ်။ arg1,arg2 လို့ပြထားလို့ 2 ခုထဲမဟုတ်ပါဘူး။ ဒါကတော့ syntax အနေနဲ့ ပြပေးတာဖြစ်ပါတယ်။ ပြီးရင်တော့ argument တွေရဲ့ data type ကိုတော့ ကြေငြာ ပေးရပါမယ်။ ဒီ argument တွေကတော့ return value ရှိမှ သာရေးပေးဖို့လိုတာပါ။ return value မရှိရင်တော့ ရေးပေးဖို့မလိုပါဘူး။ ခုရိုးရိုး user defined function လေးသုံးပြီး program တစ်ပုဒ်လောက်ရေးကြည့်ရအောင်။

```
1  #include <stdio.h>
2  main()
3  {
4      printf("This is main function\n");
5      Userdef_fun();
6  }
7  Userdef_fun()
8  {
9      printf("This is user defined function");
10 }
```

```
This is main function
This is user defined function
Process exited with return value
```

ဒီ program မှာဆိုရင် line_5. မှာ ရေးထားတဲ့ Userdef_fun() ဆိုတာကတော့ user သတ်မှတ်လိုက်တဲ့ function ဖြစ်ပါတယ်။ user သတ်မှတ်တဲ့ function လို့ပြောရတာက printf () function လိုမျိုး output display ပြဖို့အတွက် ကြိုတင်ရေး ထားတဲ့ function မဟုတ်တဲ့အတွက် ဆိုလိုခြင်း ဖြစ်ပါတယ်။ C language မှာ Userdef_fun() ဆိုတဲ့ function အတွက် ရေးသားထားတာ မရှိပါဘူး။

ဒီ program အလုပ်လုပ်သွားတာကို ပြောပြပါမယ်။ program တွေ run တဲ့အခါမှာ main() function ကနေပဲ စပြီး run တယ်လို့ပြောပြပြီးနော်။ ဒီ program မှာဆိုရင် main() function ကိုစ

run တဲ့အခါမှာ ပထမ Line_4 မှာရှိတဲ့ printf() function ကို run ပါမယ်။ ဒါကြောင့် run box မှာ This is main function လို့ display ပြပေးတာ ဖြစ်ပါတယ်။ program တွေကို run တဲ့အခါမှာ Top to down run တယ်လို့ပြောပြီးပြီနော်။ ဒီတော့ program က line_5 ကို run ပါမယ်။ line_5 ကို run တော့ Userdef_fun () ဆိုတဲ့ function ကိုတွေ့ပါမယ်။ ဒီတော့ program က Userdef_fun() ဆိုတဲ့ function ရှိတဲ့နေရာကိုသွားပြီး run ပါမယ်။

Line_7 မှာရှိတဲ့ Userdef_fun() function ကို run ပါမယ်။ ဒီတော့ Userdef_fun() function ထဲက statement ကို run ပေးမှာ ဖြစ်ပါတယ်။ ဒါကြောင့် run box မှာ This is user defined function လို့ display ပြပေးတာဖြစ် ပါတယ်။ ဒါဆိုရင် နားလည်မယ်လို့ထင်ပါတယ်။

ပိုနားလည်အောင်နောက်ထပ် user define function ဥပမာလောက်သုံးပြီးရေးကြည့်ရအောင်။

```

1  #include <stdio.h>
2  main()
3  {
4      printf("This is main function\n");
5      Userdef_fun();
6      SecUserdef_fun();
7      ThUserdef_fun();
8  }
9  Userdef_fun()
10 {
11     printf("This is first user defined function\n");
12 }
13 SecUserdef_fun()
14 {
15     printf("This is second user defined function\n");
16 }
17 ThUserdef_fun()
18 {
19     printf("This is third user defined function");
20 }
21

```

```

This is main function
This is first user defined function
This is second user defined function
This is third user defined function
-----
Process exited with return value 35

```

ဒီ program ရဲ့ အလုပ်လုပ်သွားပုံကလည်းရှေ့က program အတိုင်းပါပဲ။

Line_4 မှာ main() function ထဲက ပထမဆုံး printf() function ကို run ပြီး display ပြပေးပါတယ်။ ပြီးရင် ဒုတိယ statement ဖြစ်တဲ့ Line_5 က Userdef_fun() function ကို run ပါမယ်။ ဒါကြောင့် Userdef_fun() ရှိ တဲ့ Line_9 ကို သွားပြီး run ပါမယ်။ ပြီးရင် This is user defined function ဆိုတဲ့စာသားကို display ပြပေးမှာ ဖြစ်ပါတယ်။ ပြီးရင် program က main() function ထဲမှာရှိတဲ့ တတိယ statement Line_6 က SecUserdef_fun() function ကို သွားပြီး run ပါမယ်။ဒီတော့ SecUserdef_fun() function ရှိတဲ့ Line_13 ကို ရောက်သွားပါမယ်။ ပြီးရင် This is

second user defined function ကို display ပြပေးမှာ ဖြစ်ပါတယ်။ SecUserdef_fun() function မှာ Statement တွေအားလုံး run ပြီးတာနဲ့ main() function ကို ပြန်ရောက်သွားပါမယ်။ ဒီတော့ စတုတ္ထ Statement Line_7 က ThUsedef_fun() function ကို run မှာဖြစ်ပါတယ်။ ပြီးရင် Line_17 က ThUserdef_fun() function ကို run ပြီး This is third user defined function ကို display ပြပေးမှာဖြစ်ပါတယ်။ ဒါဆို ဒီ program ကိုလည်းကောင်းကောင်းနားလည်မယ်လို့ ထင်ပါတယ်။

ရှေ့မှာ ရေးခဲ့တဲ့ program နှစ်ပုဒ်ကတော့ User defined function ကို main() function ကနေ call ခေါ်ပြီး ရေးထားတာ ဖြစ်ပါတယ်။ အဲလို main() function ကနေ call ခေါ်ပြီးမရေးဘဲ User defined function ကို User defined function ထဲမှာပဲ call ခေါ်ပြီးရေးလို့ရပါတယ်။ program လေးတပုဒ်လောက် ရေးပြပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      printf("This is main function\n");
5      Userdef_fun();
6  }
7  Userdef_fun()
8  {
9      printf("This is first user defined function\n");
10     SecUserdef_fun();
11 }
12 SecUserdef_fun()
13 {
14     printf("This is second user defined function\n");
15     ThUserdef_fun();
16 }
17 ThUserdef_fun()
18 {
19     printf("This is third user defined function");
20 }
21

```

```

This is main function
This is first user defined function
This is second user defined function
This is third user defined function
-----
Process exited with return value 35

```

ဒီ program မှာဆိုရင် Userdef_fun() function ကိုပဲ main () function မှာ call ခေါ်ပြီး ရေးထားတာ။ SecUserdef_fun() function ကိုတော့ Userdef_fun () function ထဲမှာ call ခေါ်ပြီးရေးထားပါတယ်။ Line_10 မှာရေးထားတဲ့အတိုင်း ဖြစ်ပါတယ်။ နောက်တစ်ခု ThUserdef_fun() function ကို လည်း SecUserdef_fun() function မှာ call ခေါ်ထားတာ ဖြစ်ပါတယ်။ Lin_16 မှာရေးထားတဲ့အတိုင်းပါ။ program အလုပ်လုပ်သွားပုံကိုတော့ ရှင်းမပြတော့ ပါဘူး။ မိမိဘာသာ trace ကြည့်ပါ။ အဆင်ပြေပါစေ။

Program run တာကို ပိုနားလည်အောင် user defined function သုံးပြီးတော့ program ရေးပြီး ရှင်းပြပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      Userdef_fun();
5      printf("This is main function\n");
6  }
7  Userdef_fun()
8  {
9      SecUserdef_fun();
10     printf("This is first user defined function\n");
11 }
12 SecUserdef_fun()
13 {
14     ThUserdef_fun();
15     printf("This is second user defined function\n");
16 }
17 ThUserdef_fun()
18 {
19     printf("This is third user defined function\n");
20 }

```

```

This is third user defined function
This is second user defined function
This is first user defined function
This is main function
-----
Process exited with return value 0

```

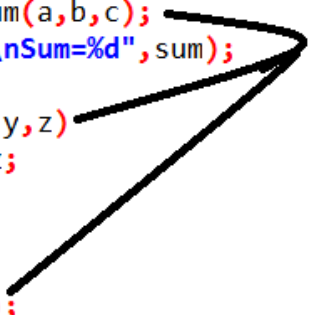
ဒီ program ကိုတော့ အသေးစိတ်ရှင်းပြဖို့ မလိုဘူးထင်ပါတယ်ခင်ဗျ။ program run သွားပုံလေးကိုပဲ ပြောပြပါ မယ်။ main () function ကို စ run တဲ့အချိန်မှာ ပထမဆုံး တွေ့တဲ့ statement Line_4 က Userdef_fun() function ကို စ run ပါမယ်။ ဒီတော့ Line_7 က Userdef_fun() function ထဲက ပထမဆုံး statement ကို run မယ်။ ဒီတော့ Line_9 က SecUserdef_fun() function ကို run မယ်။ ဒါကြောင့် program က Line_12 မှာရှိတဲ့ သူ့ရဲ့ function ကိုရောက်သွားမယ်။ အဲဒီမှာလည်း ပထမဆုံး statement ဖြစ်တဲ့ Line_14 က ThUserdef_fun() function ကို run မယ်။ ဒီတော့ Line_17 ကသူ့ရဲ့ function ကိုရောက်သွားမယ်။ ပြီးရင် This is third user defined function ကို display ပြပေးမယ်။ ThUserdef_fun() function မှာ statement ကုန်သွား ပြီဖြစ်တဲ့အတွက် ဒီ function ကို call ခေါ်ထားတဲ့ SecUserdef_fun() ကိုပြန်သွားပြီး printf() function ကို run မယ်။ display ပြမယ်။ ပြီးရင် SecUserdef_fun() ကို call ခေါ်ထားတဲ့ Userdef_fun() ဆီပြန်ရောက်သွားမယ်။ display ပြမယ်။ ပြီးရင် main() function ကို ပြန်သွားမယ်။ This is main function ဆိုတဲ့ စာသားကို display ပြမယ်။ ကဲ.....ဒါဆိုရင် ဒီ program မှာ နားမလည်တာမရှိဘူးလို့ ထင်ပါတယ်။ နားလည်နိုင်ပါစေ...။

ဒီတစ်ခါရှေ့မှာ General form of function မှာရေးပြခဲ့တဲ့ ပုံစံကို ရှင်းပြပါမယ်။ ဒါကို function with arguments လို့ခေါ်စဉ်ပေးပြီး ရှင်းပါမယ်။ arguments ဆိုတာဘာကိုခေါ်တာလဲဆိုတာကို အရင်ပြောပြပါမယ်။ ကျွန်တော် တို့ ရှေ့မှာ သုံးခဲ့တဲ့ printf() ,scanf() function တွေရဲ့ parenthesis () ထဲမှာရေးတဲ့ format string တွေကို arguments လို့ခေါ်ပါတယ်။ arguments တွေကို parameter လို့လည်း ခေါ်ပါသေးတယ်။ ဒါဆိုရင် program လေးတစ်ပုဒ်လောက်ရေးပြီး ရှင်းပြပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      int a,b,c,sum;
5      printf("Enter any three numbers");
6      scanf("%d %d %d",&a,&b,&c);
7      sum=calsum(a,b,c);
8      printf("\nSum=%d",sum);
9  }
10 calsum(x,y,z)
11 int x,y,z;
12 {
13     int d;
14     d=x+y+z;
15     return(d);
16 }
17

```



```

Enter any three numbers10 20 30
Sum=60
-----
Process exited with return value
Press any key to continue

```

ဒီ program ကိုရှင်းပြပါမယ်။ main() function မှာ variable a,b,c လို့ပေးပြီး user ရိုက်လိုက်တဲ့ တန်ဖိုးတွေကို သိမ်းထားပါမယ်။ ဥပမာ user က 10,20,30 တန်ဖိုး ခုရရိုက်လိုက်တယ် ဆိုပါစို့။ ဒါဆို a=10,b=20,c=30 ဖြစ်ပါမယ်။

ပြီးရင် Line_7 မှာ sum=calsum(a,b,c); ဆိုတဲ့ statement ကို run ပါမယ်။ calsum() ဆိုတာက user သတ်မှတ်လိုက်တဲ့ function ဖြစ်ပါတယ်။ ဒါကတော့ function with argmrnt ပုံစံဖြစ်ပါတယ်။ ဒီတော့ ဒီ function ရှိတဲ့ Line_10 ကိုသွား run ပါမယ်။ ဒီ function မှာကြေငြာပေးထားတဲ့ variable name နဲ့ main() function မှာ call ခေါ်ထားတဲ့ variable name ကမတူပါဘူး။ ဘယ်လိုသဘောတရားလဲဆိုတော့ variable a,b,c မှာ သိမ်းထားတဲ့ value တွေကို x,y,z ဆိုတဲ့ variable ထဲမှာ သိမ်းလိုက်တာဖြစ်ပါတယ်။ ဒါကြောင့် x=10, y=20, z=30 ဖြစ်သွားပါမယ်။ Variable a,b,c ကိုတော့ actual arguments လို့ခေါ်ပြီး variable x,y,z ကိုတော့ formal arguments လို့ခေါ်ပါတယ်။ သတိထားရမှာက actual arguments နဲ့ formal arguments အရေအတွက်ဟာ တူရပါမယ်။

ဒီ program မှာသုံးထားတဲ့ variable x,y,z နေရာမှာ a,b,c လို့ပေးလည်းရပါတယ်။ ဒါပေမယ့် ဒီ variable တွေက မတူပါဘူး။ ဘာလို့လည်းဆိုတော့ ဒီ variable တွေက function မတူတဲ့အတွက် ဖြစ်ပါတယ်။ အစမှာ ပြောပြခဲ့ပါသေးတယ်။ local အနေနဲ့ ကြေငြာမယ်ဆိုရင် ဒီ function မှာဘဲသုံးလို့ ရမယ်ဆိုတာကို။ program တစ်ခုလုံးမှာ သုံးချင်တယ်ဆိုရင်တော့ globl အနေနဲ့ ကြေငြာပေးရမယ်။

အိုကေ...ဒီတော့ ဒီ variable တွေက name တူတယ်ဆိုရင်တောင် local အနေနဲ့ ကြေငြာထားရင် function မတူရင် တူတယ်လို့ ယူဆလို့မရပါဘူး။

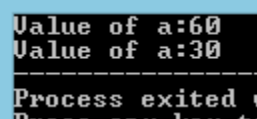
ပြီးရင် x,y,z ကို data type ကြေငြာပေးရပါမယ်။ general form မှာ ရေးပေးထားတဲ့ အတိုင်းပါပဲ။ ပြီးရင် ဒီ argument ၃ ခုပေါင်းလို့ရတဲ့ data ကို ကြေငြာပေးရပါမယ်။ ဒီ program မှာ int d; လို့ကြေငြာပေးရပါမယ်။ ဒီ variable d ကိုတော့ calsum() function မှာ local အနေနဲ့ကြေငြာပေးရပါမယ်။ ဘာလို့လည်းဆိုတော့ ဒီ variable က main() function ကနေ call လုပ်ထားတဲ့ parameter မဟုတ်တဲ့အတွက်ဖြစ်ပါတယ်။

Line_14 မှာ d=x+y+z; ဆိုတဲ့အတွက် d=60 ဖြစ်ပါမယ်။ ဒီပေါင်းလို့ရတဲ့တန်ဖိုး 60 ကို calling program ကို ပြန်သွားဖို့အတွက် return statement ကို သုံးရမှာ ဖြစ်ပါတယ်။ Line_15 မှာ return(d) လို့ရေးထားပါတယ်။ return statement က သူ့ရဲ့ parenthesis ထဲမှာရေးထားတဲ့ value ကို calling program ဖြစ်တဲ့ main() function ကို သွားပြီး return ပြန်မှာဖြစ်ပါတယ်။ ဒါကြောင့် calling program မှာ sum=60 ဖြစ်သွားပါမယ်။ ဒါကြောင့် line_8 မှာ sum ကို output ထုတ်ခိုင်းတဲ့အခါမှာ 60 ကို display ပြပေးတာဖြစ်ပါတယ်။ နားလည်မယ် လို့ထင်ပါတယ်။

ဒီ program ရှင်းတဲ့အချိန်မှာပြောခဲ့တဲ့ variable name တူပေမယ့်လည်း function မတူရင် ဒီ variable နှစ်ခု ဟာမတူဘူးဆိုတာကို ရှင်းသွားအောင် program တစ်ခုခွဲရေးပြီး ရှင်းပြပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      int a;
5      a=30;
6      fun(a);
7      printf("Value of a:%d",a);
8  }
9  fun(a)
10 int a;
11 {
12     a=60;
13     printf("Value of a:%d\n",a);
14 }
```



```

Value of a:60
Value of a:30
Process exited with code 0
Press any key to continue...
```

ဒီ program မှာဆိုရင် main () function မှာ a ကို 30 လို့ assign လုပ်ထားပါတယ်။ function call မှာလည်း variable name ကို a လို့ပေးပြီး 60 ကို assign လုပ်ထားပါတယ်။ a=60 လို့ main

function မှာသွားပြီး assign လုပ်မယ်ဆိုရင်တော့ မူလသိမ်းထားတဲ့ a=30 မှာ a တန်ဖိုး 30 ရှိမနေတော့ဘဲ နောက်ဆုံး assign လုပ် တဲ့ တန်ဖိုး 60 ကိုပဲ သိမ်းထားပေးမှာ ဖြစ်ပါတယ်။ ဒီ program မှာတော့ main function မှာ မဟုတ်ဘဲ အခြား function ဖြစ်တဲ့ fun() user defined function မှာ ကြေငြားထားပါတယ်။ ဒါကြောင့် main function ထဲက a နဲ့ user defined function က a နဲ့ မတူပါဘူး။ ဒါကြောင့် fun() က a ကို display ပြခိုင်းတဲ့အခါမှာ 60 ကို display ပြပြီး main function က a ကို display ပြတဲ့အခါမှာ 30 ကိုပြပေးတာဖြစ်ပါတယ်။ နားလည်မယ်လို့ ထင်ပါတယ်။

Return statement ကို ပိုနားလည်သွားအောင် program လေး တစ်ပုဒ်လောက် ရေးပြီးရှင်းပြပေးပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      char a,b;
5      b=fun(a);
6      printf("Alphabet= %c",b);
7  }
8      fun(ch)
9      char ch;
10 {
11     printf("Enter any alphabet");
12     scanf("%c",&ch);
13     if(ch>=65 && ch<=90)
14         return(ch);
15     else
16         return(ch+32);
17 }
```

```

Enter any alphabetB
Alphabet= B
-----
Process exited with r
```

(Or)

```

Enter any alphabet b
Alphabet= @
-----
Process exited with re
```

ဒီ program မှာဆိုရင် return statement ကို ပြောပြပါမယ်။ Line_11 မှာ user ကို alphabet တစ်ခုရိုက်ပါလို့ input value တောင်းမယ့် စာသားလေးပြမယ်။

Line_12 မှာ user ရိုက်လိုက်တဲ့ alphabet ကို သိမ်းထားလိုက်ပြီ။ ဥပမာ user က B လို့ Capital letter နဲ့ ရိုက် လိုက်တယ်ဆိုပါစို့။ ဒါဆိုရင် ch =B ဖြစ်သွားပါမယ်။

Line_13 မှာ if statement နဲ့ condition စစ်ပါမယ်။ if(ch>=65 && ch<=90) ဖြစ်လားဆိုတာကို စစ်ခိုင်းတာ ဖြစ်ပါတယ်။ ဖြစ်ရင် return(ch) ဆိုတဲ့ statement ကို run မယ်။

မဖြစ်ရင် else statement က return(ch+32) ကို run ပါမယ်။ user ရိုက်လိုက်တဲ့ capital B ရဲ့ ASCII value က 66 ဖြစ်ပါတယ်။ ဒီတော့ condition စစ်တာ မှန်သွားပြီလေ။ ဒါကြောင့် return(ch) ဆိုတဲ့ statement ကို run ပြီး B ကို calling program ကို သွားပြီး return ပြန်ပါမယ်။

Line_5 မှာ b=fun(ch); ဆိုတဲ့အတွက် b=B ဖြစ်သွားပါမယ်။ ပြီးရင် display ပြပါမယ်။ တကယ်လို့ user က small letter b လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ small letter b ရဲ့ ASCII value က 98 ဖြစ်ပါတယ်။ ဒါဆိုရင် if နဲ့ condition စစ်တော့ 65 နဲ့ 90 ကြားက မဟုတ်တဲ့အတွက် fail ဖြစ်ပါမယ်။ ဒါကြောင့် return(ch+32) ကို run ပါမယ်။ b ရဲ့ တန်ဖိုးက 98 ဆိုတော့ 98+32=130 ဖြစ်ပါမယ်။ 130 ရဲ့ ASCII symbol က @ ဖြစ်တဲ့အတွက် return ပြန်တဲ့အချိန်မှာ b=@ ဖြစ်သွားပါမယ်။ ဒါကြောင့် display ပြတဲ့အခါမှာ b ကို မပြဘဲ @ ကို ပြခြင်းဖြစ်ပါတယ်။ ဒါဆိုရင် return statement ရဲ့ အလုပ်လုပ်ပုံကို ကောင်းကောင်းနားလည်မယ်လို့ထင်ပါတယ်။

နောက်တစ်ခုပြောပြချင်တာကတော့ C language မှာ return statement က integer data type တွေကိုပဲ return ပြန်တယ်ဆိုတာ ပြောချင်ပါတယ်။ ဒါကိုပိုနားလည်သွားအောင် program လေးရေးပြပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      float a,b;
5      printf("Enter any number");
6      scanf("%f",&a);
7      b=square(a);
8      printf("\n Square of %f is %f",a,b);
9  }
10 square(x)
11 float x;
12 {
13     float y;
14     y=x*x;
15     return(y);
16 }
```

```

Enter any number2
Square of 2.000000 is 4.000000
-----
Process exited with return value 32
```

(or)

```

Enter any number1.5
Square of 1.500000 is 2.000000
-----
Process exited with return value 1
```

ဒီ program မှာဆိုရင် a,b ကို float data type အဖြစ် ကြေငြာပေးထားတယ်။ ပြီးရင် user ရိုက်လိုက်တဲ့ တန်ဖိုးကိုလည်း float data type အဖြစ်သိမ်းထားပေးပါတယ်။ ဥပမာ user က 2 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ ဒါဆို a=2 လို့သိမ်းထားလိုက်မယ်။ function call လုပ်တဲ့အခါမှာလည်းပဲ argument x ကို float အဖြစ်ကြေငြာ ပေးထားတယ်။

Line_14 မှာ $y=x*x$ ဆိုတဲ့အတွက် $y=2*2$ လို့တွက်ထုတ်လိုက်ပါမယ်။ ဒီတော့ $y=4$ ဖြစ်ပါမယ်။ ပြီးရင် called program ကို return ပြန်ပါမယ်။ return value 4 က integer data type ဖြစ်တဲ့အတွက် return ပြန်တဲ့အခါ မှာ 4 ကိုပဲ called program ကိုပြန်ပေးမှာ ဖြစ်ပါတယ်။

Line_7 မှာ $b=\text{square}(a)$; ဆိုတဲ့အတွက် $b=4$ ဖြစ်သွားပါမယ်။ ပြီးရင် display ပြခိုင်းပါမယ်။ display ပြခိုင်းတဲ့ အခါမှာ format string %f ဆိုတဲ့အတွက် 4.00000 လို့ display ပြပေးတာ ဖြစ်ပါတယ်။

ဥပမာ user က 1.5 လို့ရိုက်လိုက်တယ်ဆိုပါစို့။ဒါဆိုရင် float data type ဖြစ်ပါမယ်။

line_14 မှာ $y=x*x$ ကို run တဲ့အခါမှာ $y=1.5*1.5=2.25$ ရပါမယ်။ ပြီးရင် $\text{return}(y)$ ဆိုပြီး y တန်ဖိုးကို return ပြန်ခိုင်းမယ်။ ဒါပေမယ့် return statement က integer data type တွေကို return ပြန်ပေးတာ ဖြစ်တဲ့အတွက် floating point(ဒသမကိန်း) နောက်က တန်ဖိုးတွေကို မယူတော့ဘဲ ရှေ့ကအပြည့်ကိန်း 2 ကိုပဲ ယူပြီး return ပြန်ပေးတာ ဖြစ်ပါတယ်။ ဒါကြောင့် program ကို display ပြတဲ့အခါမှာ 2.00000 ကို ပြပေးတာ ဖြစ်ပါတယ်။ နားလည်မယ်လို့ထင်ပါတယ်။

ဒီလိုမျိုး အဖြေအတိအကျလိုချင်ရင်၊ ဒသမကိန်းတွေကို return ပြန်ခိုင်းချင်တယ်ဆိုရင် ဘယ်လိုရေးရလဲ ဆိုတာကို ပြောပြပေးပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      float square();
5      float a,b;
6      printf("Enter any number");
7      scanf("%f",&a);
8      b=square(a);
9      printf("\n Square of %f is %f",a,b);
10 }
11 float square(x)
12 float x;
13 {
14     float y;
15     y=x*x;
16     return(y);
17 }

```

```

Enter any number 1.5
Square of 1.500000 is 2.250000
Process exited with return value 1

```

ဒီလိုပြင်ရေးလိုက်တဲ့အခါမှာ 1.50000 =2.25000 ဆိုပြီး အဖြေအတိအကျရပါမယ်။ ဘာလို့လည်းဆိုတော့ line_4 မှာ float square() လို့ ကြေငြာ ပေးထားတဲ့အတွက် ဖြစ်ပါတယ်။ ဆိုလိုတာကတော့ square function ကနေပြီး float data type ကို return ပြန်ပါမယ်လို့ ပြောခြင်းဖြစ်ပါတယ်။

ဒီလောက်ဆိုရင်တော့ return statement အကြောင်းကို ကောင်းကောင်းနားလည်မယ်လို့ ထင်ပါတယ်။ ဒီတစ်ခါ called function ကနေ value တွေကို return မပြန်စေချင်ရင် ဘယ်လို ရေးရမလဲဆိုတာကို ပြောပြပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      void go();
5      go();
6  }
7  void go()
8  {
9      printf("University\n");
10     printf("Academy");
11 }

```

```

University
Academy
-----
Process exited

```

ဒီ program မှာဆိုရင် go() ဆိုတဲ့ user defined function ကနေပြီး ဘာ တန်ဖိုးကိုမှ return မပြန်ချင်တဲ့ အတွက် void ဆိုတဲ့ keyword ကိုသုံးပြီး main function မှာကြေငြာပေးထားဖြစ်ပါတယ်။ return value မရှိတဲ့ အတွက် arguments တွေရေးပေးစရာမလိုပါဘူး။ ဒါဆိုရင် void keyword ရဲ့အသုံးလေးကို နားလည်မယ်လို့ ထင်ပါတယ်။

ရှေ့မှာရေးခဲ့တဲ့ return statement တွေဟာဆိုရင် value တွေကိုပဲ called function ဆီကို return ပြန်တာဖြစ်ပါတယ်။ ဒါကို Call by value လို့ခေါ်ပါတယ်။ ဒီတော့ မေးစရာရှိပါမယ်။ return statement မှာ value တွေကို ပဲ return ပြန်လို့ရတာလား။ သူတို့ရဲ့ Address ကိုရော return ပြန်လို့မရဘူးလား။ ရပါတယ်။ အဲလိုမျိုး Address တွေကို return ပြန်တာကို Call by Reference လို့ခေါ်ပါတယ်။ Call by reference ကို မပြောခင် အရင်ဆုံးသိ ထားသင့်တဲ့ Pointer ကို အရင် ရှင်းပြပါမယ်။

9.Pointer

Pointer အသုံးပြုနည်း

Pointer ကို မပြောခင် အရင်ဆုံး variable တွေကို declare လုပ်လိုက်တဲ့အခါမှာ computer မှာ ဘယ်လိုအလုပ် လုပ်လဲဆိုတာကို ပြောပြပါမယ်။

```
int i=3 ;
```

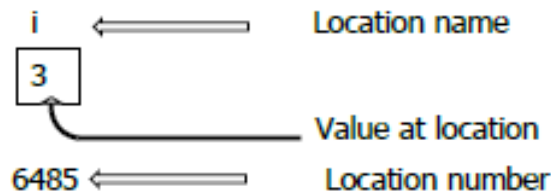
ခုလိုမျိုး variable i ဟာ integer data type ဖြစ်ပြီး value 3 ကို သိမ်းထားပါမယ်လို့ကြေငြာ လိုက်တာ ဖြစ်ပါတယ်။ ဒါကတော့ C Compiler ကို အချက် ခုချက် လုပ်ဆောင်ပေးဖို့ ကြေငြာလိုက်တာ ဖြစ်ပါတယ်။

၁။ integer value တွေ သိမ်းဖို့အတွက် memory space သတ်မှတ်ပေးဖို့။

၂။ variable name i ကို memory ကို သိမ်းမယ့် နာမည်ဖြစ်ပါတယ်ဆိုတာသိဖို့။

၃။ ဒီ variable name i ရှိတဲ့ memory မှာ 3 ကို သိမ်းပါမယ်လို့ သိအောင် ကြေငြာပေးခြင်း ဖြစ်ပါတယ်။

Memory မှာ သိမ်းဆည်းထားပုံကို ပုံလေးနဲ့ပြပေးပါမယ်။



Computer ရဲ့ memory မှာ ခုလိုမျိုး သိမ်းထားလိုက်မှာ ဖြစ်ပါတယ်။ 6485 ဆိုတာကတော့ value 3 ကို သိမ်းထားတဲ့ location number(Address) ဖြစ်ပါတယ်။ 6485 ဆိုရင် value 3 ရဲ့ Address ပါလို့ ပုံသေမှတ်လို့ မရဘူးနော်။ ဘာလို့လည်းဆိုတော့ computer ရဲ့ RAM(Random access Memory) မှာသိမ်းထားတာဖြစ်ပါတယ်။ ဒါကြောင့် ဘယ် data တွေကိုတော့ ဘယ် Address မှာပဲ သိမ်းမယ်လို့ အတိအကျမရှိပါဘူး။ data တွေကို သိမ်းဖို့အတွက် သူရွေးချယ်ချင်တဲ့ Address မှာပဲ ရွေးပြီး သိမ်းထားပေးတဲ့အတွက် ဖြစ်ပါတယ်။ ဒါကြောင့် program ရေးတဲ့အခါမှာ value 3 ရဲ့ address က 6485 လည်းမဟုတ်ပါဘူးလို့ ထင်နေမှာစိုးလို့ပါ။ ဒါဆို program လေး တစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။

```

1  #include <stdio.h>
2  main()
3  {
4      int i=3;
5      printf("Address of i= %d\n",&i);
6      printf(" Value of i= %d",i);
7  }
8

```

```

Address of i= 2686684
Value of i= 3
Process exited with va

```

ဒီ program မှာဆိုရင် printf() function ကို သုံးပြီး value 3 နဲ့ သူ့ရဲ့ Address ကို Output ထုတ်ထားတာ ဖြစ်ပါတယ်။ Line_5 မှာ ဆိုရင် value 3 ရဲ့ Address ကို display ပြဖို့အတွက် "&" ကိုသုံးပြီး ရေးပေးရပါမယ်။ ဒီ & ကိုသုံးပြီး ရှေ့ပိုင်းမှာ scanf() function နဲ့ တွဲသုံးတာကို သိခဲ့မယ်လို့ထင်ပါတယ်။ ဒီ & ကို သုံးရတာကတော့ value တွေသိမ်းထားတဲ့ Address ကို ပြဖို့ဖြစ်ပါတယ်။ ဒါကို Address of operator လို့ခေါ်ပါတယ်။ ဒါဆိုရင် pointer အကြောင်းကို ဆွေးနွေးလို့ရမယ်လို့ထင်ပါတယ်။ pointer operator ရဲ့ symbol ကတော့ "*" ဖြစ်ပါတယ်။ ဒါကို Value at address လို့ခေါ်ပါတယ်။ ဒီ pointer operator ကိုသုံးတဲ့အကြောင်းကတော့ Address မှာ

သိမ်းထားတဲ့ value ကို display ပြချင်တဲ့အခါမှာသုံးပါတယ်။ ဒီ operator ကို နောက်တနည်းအားဖြင့် Indirection operator လို့လည်းခေါ်ပါသေးတယ်။ ဒီ operator လေးရဲ့ အလုပ်လုပ်ပုံကို program ရေးပြီး ပြောပြပါမယ်။

```

1  #include <stdio.h>
2  main()
3  {
4      int i=3;
5      printf("Address of i= %d\n",&i);
6      printf(" Value of i= %d\n",i);
7      printf(" Value of i= %d",*(&i));
8
9  }
```

```

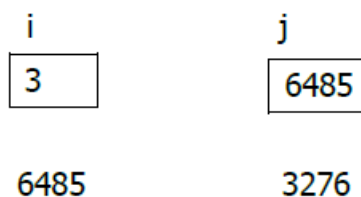
Address of i= 2686684
Value of i= 3
Value of i= 3
-----
Process exited with ret
```

ကဲ..ဒီ program မှာဆိုရင် ဘာထူးခြားသွားလဲဆိုတာကို တွေ့မယ်လို့ထင်ပါတယ်။ Line_5 မှာ &i လို့ output ထုတ်တဲ့အတွက် value 3 ကိုသိမ်းထားတဲ့ Address ကို display ပြပေးတယ်။ Line 6 မှာတော့ i ကို display ပြခိုင်းတဲ့အတွက် I ထဲမှာ သိမ်းထားတဲ့ value 3 ကို display ပြပေးတာဖြစ်ပါတယ်။ နောက်တစ်ခု Line_7 မှာတော့ *(&i) လို့ ရေးပေးတဲ့အတွက် 3 ရဲ့ address ကို display မပြဘဲ value 3 ကို display ပြပေးတာဖြစ်ပါတယ်။ ဒီ indirection operator (*) ရဲ့ အဓိပ္ပာယ်ကိုက value at address ဆိုတော့ &i မှာရှိတဲ့ value ကို display ပြပါလို့ဆိုလိုက်တာနဲ့ အတူတူပါပဲ။ i လို့ရေးပြီး display ပြတာနဲ့ *(&i) လို့ရေးပြီးပြတာ အတူတူပါပဲ။ value 3 ကိုပဲ display ပြပေးမှာဖြစ်ပါတယ်။

&i ဆိုရင် i ရဲ့ address ကို display ပြမယ်ဆိုတာ သိပြီနော်။ value တွေကိုမှာ variable name တွေပေးပြီး သိမ်းလို့ရတာမဟုတ်ပါဘူး။ address တွေကိုလည်း variable name ပေးပြီး သိမ်းလို့ရပါတယ်။ ဥပမာပြပါမယ်။

j=&i;

ဒီလိုကြေငြာလိုက်တယ်ဆိုရင် variable i ရဲ့ address ကို တခြား variable ဖြစ်တဲ့ variable j ထဲ မှာ သိမ်းထားလိုက်တာ ဖြစ်ပါတယ်။ ဒီမှာဆိုရင် j က အခြား variable တွေနဲ့မတူပါဘူး။ သူက value တွေသိမ်းထား တဲ့ variable မဟုတ်ပါဘူး။ အခြား variable ရဲ့ address ကိုသိမ်းထားတဲ့ variable ဖြစ်ပါတယ်။ ပိုနားလည် အောင် ပုံလေးနဲ့ပြပေးပါမယ်။



ဒါဆိုရင် နားလည်မယ်လို့ထင်ပါတယ်။ $j = \&i$ လို့ကြေငြာပေးတာနဲ့ ခုလိုမျိုး memory မှာ သိမ်းပေးမှာ ဖြစ်ပါတယ်။ j ကလည်း variable ဖြစ်တဲ့အတွက် compiler ကသူ့အတွက် memory location(address) ကို သတ်မှတ်ပေးပါတယ်။ ဒီပုံမှာ ပြထားတာဟာဆိုရင် variable i ရဲ့ address (6485) မှာ value 3 ကို သိမ်းထားပါတယ်။ ပြီးရင် variable j ရဲ့ address(3276) မှာ variable i ရဲ့ address ကို သိမ်းပေးထားပါတယ်လို့ ပြောခြင်းဖြစ်ပါတယ်။ တကယ်လို့ variable j မှာသိမ်းထားတဲ့ address ရဲ့ value ကို သိချင်တယ်ဆိုရင်တော့ $*j$ လို့ ကြေငြာပေးရမှာပါ။ ပိုနားလည်အောင် program တစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။

```

1  #include <stdio.h>
2  main()
3  {
4      int i=3;
5      int *j;
6      j=&i;
7      printf("Address of i= %d\n",&i);
8      printf("Address of i= %d\n",j);
9      printf("Address of j= %d\n",&j);
10     printf(" Value of i= %d\n",i);
11     printf(" Value of j= %d\n",j);
12     printf(" Value of i= %d\n",*(&i));
13     printf(" Value of i= %d\n",*j);
14
15 }
```

```

Address of i= 2686684
Address of i= 2686684
Address of j= 2686680
Value of i= 3
Value of j= 2686684
Value of i= 3
Value of i= 3
```

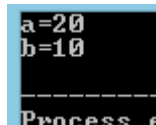
ဒါဆိုရင် ရှေ့ကပုံကို နားလည်မယ်လို့ထင်ပါတယ်။ Line_8 မှာ j ကို display ပြခိုင်းပါတယ်။ ဒီတော့ j ကို $\&i$ လို့ assign လုပ်ထားတဲ့အတွက် i ရဲ့ address ကို display ပြပေးတာ ဖြစ်ပါတယ်။ Line_9 မှာတော့ $\&j$ ကိုပြခိုင်း တဲ့အတွက် j ရဲ့ address ကို display ပြပေးတာ ဖြစ်ပါတယ်။ Line_13 မှာ $*j$ ကို display ပြခိုင်းတဲ့အတွက် value 3 ကို display ပြပေးတာ ဖြစ်ပါတယ်။ ဘယ်လိုပြပေးလဲဆိုတော့ $j = \&i$ လို့ assign လုပ်ထားတဲ့အတွက် $*(\&i)$ ဖြစ်သွားပါမယ်။ $*$ operator က value at address ဆိုတော့ address i မှာ ရှိတဲ့ value ကို display ပြ ခိုင်းတာ ဖြစ်ပါတယ်။

ဒါဆိုရင် pointer အကြောင်းကို တီးမိခေါက်မိရှိပြီလို့ ထင်ပါတယ်။ ဒါဆိုရင် call by reference ကိုသုံးပြီး variable နှစ်ခုရဲ့ တန်ဖိုးတွေကို ချိန်းမယ့် program လေးတစ်ပုဒ်လောက် ရေးကြည့်ရအောင်။


```

1  #include <stdio.h>
2  main()
3  {
4      int a=10;
5      int b=20;
6      swap(&a,&b);
7      printf("a=%d\n",a);
8      printf("b=%d\n",b);
9  }
10 swap(x,y)
11 int *x,*y;
12 {
13     int t;
14     t=*x;
15     *x=*y;
16     *y=t;
17 }

```



```

a=20
b=10
Process a

```

ဒီ program မှာဆိုရင် a=10 နဲ့ b=20 လို့ assign လုပ်ထားပါတယ်။ ဒါပေမယ့် output မှာ a=20 နဲ့ b=10 ဖြစ်သွားပါတယ်။ ဒါကို swapping (နေရာလဲလှယ်ခြင်း) လို့ခေါ်ပါတယ်။ ဘယ်လို နေရာလဲလိုက်လဲဆိုတော့ called by reference ပုံစံနဲ့ပါ။ Line_6 က swap() function မှာ argument a ,argument b ကို &a,&b လို့ declare လုပ်ထားပါတယ်။ ဒါကြောင့် a,b ထဲမှာ value ကို သိမ်းမထားဘဲ၊ reference(address) ကိုသိမ်းထား ပါတယ်။ ဒါကို line_10 မှာ x,y နဲ့ ပြန်ပြီး called လုပ်ထားပါတယ်။ ဒီတော့ x=&a,y=&b ဖြစ်ပါမယ်။

Line_11 မှာ int *x,*y လို့ကြေငြာထားတဲ့အတွက် *(&a),*(&b) နဲ့ သဘောတရားတူပါမယ်။ တကယ်တမ်းကတော့ x က main function မှာရှိတဲ့ swap(&a,&b) က a ရဲ့ address နဲ့ b ရဲ့ address ကို copy ကူးယူလိုက်တာ ဖြစ်ပါတယ်။ ဒါကြောင့် *x,*y လို့ရေးလိုက်တာဟာ value at address x ,y မှာ ရှိတဲ့တန်ဖိုးကို သိမ်း ပေး မှာဖြစ်ပါတယ်။ ဒါကြောင့်ခုဆိုရင် *x=10 နဲ့ *y=20 ဖြစ်ပါမယ်။ နားလည်တယ်ဟုတ်။

Line_14 မှာ t=*x; ဆိုတဲ့အတွက် t=10 ဖြစ်သွားပါမယ်။

Line_15 မှာ *x=*y; ဆိုတဲ့အတွက် *x=20 ဖြစ်သွားပါမယ်။ ဘာလို့လည်းဆိုတော့ *y=20 လေ။ ဒီတော့ *x ကမူလရှိနေတဲ့ 10 ကို မသိမ်းတော့ဘဲ နောက်ဆုံး assign လုပ်ထားတဲ့ value 20 ကိုပဲသိမ်းထားပေးမှာ ဖြစ်ပါတယ်။

Line_16 မှာ *y=t ဆိုတဲ့အတွက် *y=10 ဖြစ်ပါမယ်။ ဒါကိုတော့ နားလည်တယ်ဟုတ်။ ဒီတော့ called function ကိုပြန်သွားတဲ့အခါမှာ *x=20,*y=10 ဖြစ်ပါမယ်။ ဒါကြောင့် a ,b ကို display ပြခိုင်းတဲ့အခါမှာ a=20,b=10 ဖြစ်သွားတာပါ။ ဒါဆို call by reference နည်းနဲ့ swapping လုပ်တာကို နားလည်မယ်လို့ထင်ပါတယ်။

Exercise

C language ရဲ့ paradigm ကို သိပြီးလို့ထင်ပါတယ်။ ဒါဆိုရင် အောက်က program တွေကို မိမိဘာသာ trace ကြည့်ပါ။ Exercise အဖြစ် အနည်းငယ်ရေးပေးထားပါတယ်။ အဆင်ပြေပါစေ။

```

1  #include<stdio.h>
2  main()
3  {
4      char num;
5      int intake,cd_num,mark;
6      printf("Do you want to discard Training marks: Y or N\n ");
7      printf("Enter yes or no( enter Y or N: )");
8      scanf("%c",&num);
9      if(num=='Y')
10     {
11         printf(" 1]First year: \n");
12         printf(" 2]Second year:\n ");
13         printf(" 3]Third year: \n");
14         printf("Enter Intake(1 to 3):\n ");
15         scanf("%d",&intake);
16         switch(intake)
17         {
18             case 1:
19             {
20                 printf("1]First year: \n");
21                 printf("Enter cadet number (800 to 803) :\n");
22                 scanf("%d",&cd_num);
23                 switch(cd_num)
24                 {
25                     case 800:
26                         printf("He is Mg Mg\n ");
27                         printf("Enter discard mark:\n");
28                         scanf("%d",&mark);
29                         printf("Mg Mg is discard %d marks",mark);
30                         break;
31                     case 801:
32                         printf("He is Kyaw Kyaw\n");
33                         printf("Enter discard mark:\n");
34                         scanf("%d",&mark);
35                         printf("Kyaw Kyaw is discard %d marks",mark);
36                         break;

```

```
37         case 802:
38             printf("He is Mg Mya\n");
39             printf("Enter discard mark:\n");
40             scanf("%d",&mark);
41             printf("Mg Mya is discard %d marks",mark);
42             break;
43     }
44     break;
45 }
46 case 2:
47 {
48     printf("2]Second year: ");
49     printf("Enter cadet number (500 to 502)");
50     scanf("%d",&cd_num);
51     switch(cd_num)
52     {
53     case 500:
54         printf("He is Aung Aung\n");
55         printf("Enter discard mark:\n");
56         printf("Enter discard mark:\n");
57         scanf("%d",&mark);
58         printf("Mg Mg is discard %d marks",mark);
59         break;
60         case 501:
61             printf("He is Tun Tun\n");
62             printf("Enter discard mark:\n");
63             scanf("%d",&mark);
64             printf("Tun Tun is discard %d marks",mark);
65             break;
66         case 502:
67             printf("He is Sai Sai\n");
68             printf("Enter discard mark:\n");
69             scanf("%d",&mark);
70             printf("Sai Sai is discard %d marks",mark);
71             break;
72     }
73     break;
74 }
```

```

75     case 3:
76     {
77         printf("3]Third year: \n");
78         printf("Enter cadet number (400 to 402) :\n");
79         scanf("%d",&cd_num);
80         switch(cd_num)
81
82             case 400:
83                 printf("He is Mg Phyoe\n ");
84                 printf("Enter discard mark:\n");
85                 scanf("%d",&mark);
86                 printf("Mg Phyoe is discard %d marks",mark);
87                 break;
88             case 401:
89                 printf("He is Kyaw Myo\n");
90                 printf("Enter discard mark:\n");
91                 scanf("%d",&mark);
92                 printf("Kyaw Myo is discard %d marks",mark);
93                 break;
94             case 402:
95                 printf("He is Mg Kaung\n");
96                 printf("Enter discard mark:\n");
97                 scanf("%d",&mark);
98                 printf("Mg Kaung is discard %d marks",mark);
99                 break;
100
101         default:
102             printf("invalid");
103     }
104 }
105 }
106 else
107 {printf("invalid");
108 }
109 }

```

```

Do you want to discard Training marks: Y or N
Enter yes or no( enter Y or N: )Y
1]First year:
2]Second year:
3]Third year:
Enter Intake(1 to 3):
3
3]Third year:
Enter cadet number (400 to 402) :
402
Enter discard mark:
8
Mg Phyoe is discard 8 marks
-----
Process exited with return value 22

```

ဒီ Program ကိုတော့ ရှင်းမပြောဘူး။ Switch statement ကို သုံးပြီး ဗိုလ်လောင်း တစ်ယောက်ရဲ့ လေ့ကျင့် ရေးမှတ်ကို ဖြတ်တဲ့ program လေးဖြစ်ပါတယ်။ မိမိဘာသာ trace ပြီး programming sense ရအောင်ယူပါ။

ဒီတစ်ခါ ဗိုလ်လောင်းတစ်ယောက်ရဲ့ ကိုယ်ရေးဖိုင်ကို ကြည့်ဖို့အတွက် password တစ်ခုတောင်းပါမယ်။ ဒီ password မှန်တယ်ဆိုရင်တော့ ဒီဗိုလ်လောင်းရဲ့ ကိုယ်ရေးဖိုင်ကို ကြည့်လို့ရမယ်ပေါ့။ password ကိုတော့ program ကိုရေးတဲ့သူပဲ သိပါမယ်။

```

1  #include<stdio.h>
2  int main()
3  {
4      char num;
5      int password;
6      printf("\t Do you want to open the profile:");
7      scanf("%c",&num);
8      if(num=='Y' || num=='y')
9      {
10         printf("\nEnter password: ");
11         scanf("%d",&password);
12         if(password==7630)
13         {
14             printf("\nCadet no:\t7792\n");
15             printf(" Name:      \tKyaw Kyaw\n");
16             printf(" Intake:      \t22 \n");
17             printf(" Major:       \tMechatronic\n");
18         }
19     }
20     else
21     printf("Try again");
22 }
```

```

Do you want to open the profile:y
Enter password: 7630
Cadet no:      7792
Name:          Kyaw Kyaw
Intake:         22
Major:         Mechatronic
```

ဒီ program ကတော့ လွယ်မယ်လို့ထင်ပါတယ်။ password ကိုတော့ မိမိဘာသာ ပြင်ချင်သလိုပြင်ပြီး ရေးနိုင် ပါတယ်။ ဒါပေမယ့် program ကိုရေးတဲ့သူပဲ password ကို သိမှာနော်။ အဆင်ပြေပါစေ။

```

1  #include<stdio.h>
2  int main()
3  {
4      int i,j,row;
5      printf("Enter number of row");
6      scanf("%d",&row);
7      for(i=1;i<=row;i++)
8      {
9          for(j=1;j<=i;j++)
10         {
11             printf(" * ");
12         }
13         printf("\n");
14     }
15 }

```

```

Enter number of row 5
*
* *
* * *
* * * *
* * * * *

```

ဒီ program ကတော့ Nested for loop ကို သုံးပြီး star(*) လေးတွေကို display ပြဖို့ ရေးထားတာဖြစ်ပါတယ်။ Nested for loop အကြောင်းလေ့လာခဲ့ပြီးပြီဆိုတော့ နားလည်မယ်လို့ ထင်ပါတယ်။

```

1  #include<stdio.h>
2  int main()
3  {
4      int i,j,row;
5      printf("Enter number of row");
6      scanf("%d",&row);
7      for(i=1;i<=row;i++)
8      {
9          for(j=1;j<=i;j++)
10         {
11             printf(" %d ",j);
12         }
13         printf("\n");
14     }
15 }

```

```

Enter number of row 5
1
1 2
1 2 3
1 2 3 4
1 2 3 4 5

```

ဒီ program ကလည်း ဒီအတိုင်းပါပဲ။ မိမိဘာသာ နားလည်အောင် ကြည့်ပါ။အဆင်ပြေပါစေ။

ဒီစာအုပ်ကို နားလည်ပြီးဆိုရင်တော့ C language ရဲ့ အခြေခံ ကို ပိုင်ပြီလို့ ယူဆလို့ရသလို programming language အားလုံးရဲ့ အခြေခံ သဘောတရားကို ကောင်းစွာနားလည် သဘောပေါက်မှာ ဖြစ်ပါတယ်။ မိမိနှစ်သက်ရာ နယ်ပယ်တစ်ခုကို ရွေးချယ်နိုင်ပါစေ။

- 1) Arduino projects (Arduino ဖြင့် projects များကို နားလည်သဘောကပ်ပြီး လုပ်ဆောင်နိုင်ပါမယ်။)
- 2) Programmer ဖြစ်ချင်ရင်တော့ C#, Java ကဲ့သို့သော High level language များကို လေ့လာရာမှာ အဆင်ပြေ လွယ်ကူပါမယ်။
- 3) Wep Developer ဖြစ်ချင်ရင်တော့ HTML, CSS, PHP, Java Script, MYSQL စတာတွေကို လေ့လာဖို့ အကြံပြုချင်ပါတယ်။

(Meet at the top)