

LEARN



AUNG MYINT (ME, AUSTRALIA)

မာတိတာ

စာမျက်နှာ

နံပါန်:

?

အနေ: (၁) C အကြခံများ

၁.၁	Structure of a C Program	၉
၁.၂	C Character Set	၁၁
၁.၃	Identifiers and Keywords	၁၂
၁.၄	Constants	၁၃
၁.၅	Variables and Arrays	၁၄
၁.၆	Data Types	၂၀
၁.၇	Symbolic Constants	၂၂
၁.၈	Operators and Expressions	၂၃
၁.၉	Library Functions	၂၅

အနေ: (၂) DATA INPUT နှင့် OUTPUT

J.၁	getchar Function	၃၈
J.၂	putchar Function	၄၀
J.၃	scanf Function	၄၁
J.၄	printf Function	၄၂
J.၅	gets and puts Functions	၄၂

အနေ: (၃) CONTROL STATEMENTS

၃.၁	if Statement	၅၇
၃.၂	goto Statement	၅၉
၃.၃	if-else Statement	၆၄

၃.၆	if-else if-else Statement	၆၇
၃.၇	while Statement	၇၀
၃.၈	do-while Statement	၈၁
၃.၉	for Statement	၈၂
၃.၁၀	Nested Loops	၈၅
၃.၁၁	switch Statement	၁၀၀
၃.၁၂	break Statement	၁၀၃
၃.၁၃	continue Statement	၁၀၆
၃.၁၄	comma Statement	၁၁၁

အနေ: (၄) FUNCTIONS

၄.၁	Accessing a Function	၁၂၂
၄.၂	Defining Own Constants	၁၂၄
၄.၃	Creating Own Header Files	၁၂၀
၄.၄	Passing Argument by Value	၁၂၁
၄.၅	Recursion	၁၂၄

အနေ: (၅) PROGRAM STRUCTURE

၅.၁	Automatic Variables	၁၄၀
၅.၂	External Variables	၁၄၁
၅.၃	Static Variables	၁၄၃
၅.၄	Multifile Programs	၁၄၇

፩፻፭፻

C language සිතා general purpose ලයුද්‍ය:ස්ංඝ්‍රා structured ලයුද්‍ය:ගුදු programming language ඩීපි|| Pascal තී. FORTRAN 77 තී.ලිඩ් structured ගුදු high-level programming language ප්‍රක්‍රියා තියෙන්|| මිලේමයුද් C ආ වාටාත්ත්‍රක C අන්. low-level programming රෙඛ්‍ය රාමාව පිබා|| මිලේගාද්‍ය ලයුද්‍ය: C භා උග්‍රීත්‍රා තුළ ප්‍රේරණ:රත්තා application programming තුළ මා ඇත්: යැන්වයින් operating system තුළ රෙඛ්‍ය:programming language ප්‍රක්‍රියා තියෙන්|| මිහා C අ. දූ: මිශ්‍රා ප්‍රාග්‍රැම්පි||

C နဲ့ ပတ်သက်လို့ စိတ်ဝင်စားစရာ နောက်ကြောင်းလေး ရှိခဲ့ပါတယ်။ စာရေးသူ ကြားဖူးနားဝရီ သလောက် တစ်ဆင့်ဖောက်သည်ချမှမယ်ဆုံးရင် C ရဲ့ မွေးဖားရာဇ်တိဟာ Bell Telephone Laboratories, Inc. ဖြစ်ပါတယ်။ အခြေတော့ AT & T Bell Laboratories လို့ ခေါ်ပါတယ်။ အဲဒီမှာ အလုပ်လုပ်ခဲ့တဲ့ Dennis Ritchie ဆိုသူကနေ C ကို ၁၉၇၀-ခုနှစ်မှာ စတင် develop လုပ်ခဲ့တာပါ။ develop လုပ်တယ်လို့ ပြောရခြင်းကတော့ C ဟာ Bell Laboratories မှာ အဲဒီအချိန်ခါတုန်းက အသုံးပြုနေခဲ့ဖြစ်တဲ့ BCPL နဲ့ B ဆိုတဲ့ language J-ခုကနေ idea ယူပြီး တစ်ဆင့်ပေါက်ဖားလာတာကို။ ဒါကြောင့်မို့ ထင်ပါရဲ့။ C ဟာ အပြင်လောကကို ထွက်လွှာခွင့်မရဘဲ Bell Laboratories မှာ အကျယ်ချုပ်ဘဝနဲ့ သောင်တင်နေရရှုရာတယ်၏ ၁၉၇၈-ခုနှစ်ကျေမှ Ritchie နဲ့ Brian Kernighan တို့နှစ်ဖောက် ပူးပေါင်းရေးသားတဲ့ "The C Programming Language" ဆိုတဲ့စာအုပ်ကို Prentice Hall စာအုပ်တိုက်ကနေ ထုတ်ဝေခွင့်ရဲ့ပါတယ်။ နောင်ခါမှာ ဒီစာအုပ်မှာပါတဲ့ C ကို "K & R C" လို့ ခေါ်ပေါ်ကြပါတယ်။

တကယ်တော့ C ဟာ ဝါဘာရကြွယ်ကြွယ်၊ လွှတ်လွှတ်လပ်လပ်နဲ့၊ စတင့်အမျိုးမျိုး ရေးလို့ရတဲ့ language တစ်မျိုးဖြစ်လို့မို့၊ ဒီနေ့ခေတ်မှာ လူကြိုက်များလာခြင်းပါ။ ဒါပေမယ့် C ရဲ့ သိပ်ကောင်းလွန်ပါတယ် ဆိတဲ့ လုပ်ကွက်တွေဟာ C ကို မကျမ်းကျင့်တဲ့ novice တစ်ယောက်အဖို့၊ သေကွက်တွေ ဖြစ်နေနိုင်ပါတယ်။ compact ဖြစ်အောင်လို့ဆိုပြီး အကျဉ်းရုံးရေးထားတဲ့ C program တစ်ခုဟာ C မကျမ်းကျင့်တဲ့ လေ့လာသူ တစ်ယောက်အတွက် အဓိပ္ပာယ်ဖော်ရခေက်တဲ့ problem တစ်ခုဖြစ်နေမှာပါ။ ဒါကြောင့်မို့လို့ စာရေးသူအနေနဲ့၊ အကြံပေးလိုတာကတော့ C ကို လေ့လာမယ်ဆိုရင် အနည်းဆုံး တစ်ခြား programming language တစ်ခုကို ကျမ်းကျမ်းကျင့်ကျင့်တတ်ပြီးသား ဖြစ်နေရင် အကောင်းဆုံးပါ။ ကားအကောင်းစားကို မောင်းမယ့်လူဟာ ကားမောင်းကျမ်းကျင့်ပို့ ပိုလိုသလိုပေါ့။ ကောင်းတဲ့ ကားဟာ အမှားမခံပါဘူး။ အကိုင်အတွယ် နည်းနည်းမှားတာနဲ့ တိုက်ပြီ၊ ခိုက်ပြီပဲ။ ဒီဥပမာလိုပဲ C ဟာ programmer တစ်ယောက် user-friendly မဖြစ်ဘူး ဆိတဲ့ ကြိုသိနေရင် C နဲ့၊ အလုပ်လုပ်တဲ့ အခါမှာ အမှားနည်းပါလိမ့်မယ်။

စာရေးသူရဲ့ LEARN C စာအုပ်မှာ အခန်း (၅) ခန်း ပါပါတယ်။ ပထမအခန်းက C အခြေခံများ။ ဒုတိယအခန်းက DATA INPUT နှင့် OUTPUT တတိယအခန်းကတော့ CONTROL STATEMENTS ၁တဗ္ဗာအခန်းက FUNCTIONS နဲ့ ပုံမအခန်းက PROGRAM STRUCTURE တို့ ဖြစ်ပါတယ်။ အခန်းတိုင်းကို ပြည့်စုံအောင် ရေးပြထားလို့မို့၊ အပိုင်း (၂) ပိုင်း ခွဲထုတ်ရမှာ ဖြစ်ပါတယ်။ အခန်းတိုင်းမှာ တွက်ပြီးသား example တော်တော်များများကို ဖော်ပြထားပါတယ်။ သာမန်တွေ့နေကျ C grammar တွေကိုပါ ရှင်းလင်းပြထားသလို numerical problem တွေ၊ engineering problem တွေကိုပါ စုံလင်အောင် ဖော်ပြထားပါတယ်။ ဒါပေမယ့် ဒီစာအုပ်ဟာ C language အတွက် perfect မဖြစ်ဘူး ဆိတဲ့တော့ ဝန်ခံပါရမေး။ ဘာပဲဖြစ်ဖြစ် စာရေးသူအနေနဲ့ လက်တွေ့ခံယူထားတာကတော့ ဒီစာအုပ်အပေါ် စာဖတ်သူရဲ့ ရွှေမြင်သုံးသပ်ချက်ဟာ ကောင်းသည်ဖြစ်စေ၊ ဆိုးသည်ဖြစ်စေ စာရေးသူအတွက် အထိက်တန်ဆုံး ဆုလာဘ်တစ်ခု ရတာပဲလို့ ယုံကြည်ပါတယ်။

စာဖတ်သူအပေါင်း ပညာဗဟိုတုတေသနများ တိုးမှားနိုင်ကြပါစေ။

C အခြေခံများ

၁.၁ Structure of a C Program

C program ဆိတ် function တွေနဲ့ တည်ဆောက်ထားတဲ့ program အဖြစ်ပါပဲ။ ဒီအစိတ်မှာ function တစ်ခု သို့မဟုတ် တစ်ခုထက်ပိုပြီး ပါနိုင်ပါတယ်။ အနည်းဆုံးတစ်ခုတော့ ပါကိုပါရမှာပါ။ အဲဒီ ပေါ်မဖြစ်ပါရမယ့် function ကို main လို့ ခေါ်ပါတယ်။ main ရဲ့သဘောကတော့ C program တစ်ခုကို run လိုက်မယ်ဆိုလို့ရှိရင် main ကို ကန်ခြီးထားပြီး အလုပ်စလုပ်ရတယ်လော့။ ပြီးတော့မ ဘူးလဲမှာ ရေးထားတဲ့ instruction တွေကိုကြည့်ပြီး တစ်ခြား function တွေဆီ သွားလို့ရတာကိုး။ ဒါကြောင့်မို့လို့ main ဟာ C program ရဲ့ အဓိက ကျောရှိးလို့ပြောရင် မမှားပါဘူး။ ကောင်းပြီ၊ ကျွန်ုတ်တော်တို့ C programming ကို အလွယ်ဆုံးနဲ့ အမြန်ဆုံးသိချင်ပါတယ်။ ရမလား၊ ရပါတယ်။ လွယ်တာကို အရင်ပြောရင် မလွယ်စရာ မရှိပါဘူး။ အလွယ်ဆုံး သချ်သပ္ပါယောလေးတစ်ပုဒ်ကို C language နဲ့ ရေးကြည့်မယ်လော့ program ကတော့ စက်ရှိပါတယ်။

EXAMPLE 1.1: This C program reads the radius of a circle, calculates its area and then displays the calculated result on the screen.

```
# include < stdio.h >

/* program to calculate the area of a circle */

main ()
{
    float radius, area;
    printf ("Enter the radius :");
    scanf ("%f", &radius);
    area = 3.141593 * radius * radius;
    printf ("Area = %f", area);
}
```

ဒီ program ကို run လိုက်မယ်ဆိုလိုရှိရင် computer မှာ အခုလိုပေါ်လာမှာပါ။

Enter the radius : 5
Area = 78.539825

(5) ဂဏန်းကို မျဉ်းတားထားတဲ့ အဓိပ္ပာယ်ကတော့ ဒီ data ကို ကျွန်တော်တို့ ကိုယ်တိုင် keyboard ကနေ ရှိက်ထည့်ပေးရမယ်လို့ ဆိုလိုပါတယ်။ မျဉ်းတားထားတာကို တွေ့ရင် input data တွေလို့ မှတ်လိုက်ပါ။ တွက်လို့ချက်လို့ ရတဲ့ အဖြေတွေဆိုရင် မျဉ်းမတားပါဘူး၊ ရှင်းပါတယ်နော်။ အကာင်းပြီ၊ EXAMPLE 1.1 က C program ကို တစ်ကြောင်းချင်းအဓိပ္ပာယ် ဖော်ကြည့်ရအောင်။

- ၁။ C program ကို အရေးတဲ့ အခါးမှာ အများအားဖြင့် စာလုံးအသေး (Lowercase letter) နဲ့ ပေးလေ့ ရှိပါတယ်။ C programmer တွေရဲ့ စတိုင်ပဲခေါ်မလားမသိဘူး။ စာလုံးအကြီးနဲ့ ရေးလည်း ရတော့ရတာပါပဲ။ ဒါပေမယ့် တစ်ခုသတိထားရမှာက C language မှာ Lowercase letter နဲ့ Uppercase letter တို့ဟာ အဓိပ္ပာယ်မထူးရော်။ ဘာပြုလို့လဲဆိုတော့ AUNG, Aung, aung ဆိုတာတွေဟာ အမျိုးမတူတဲ့ variable name တွေပါပဲ။ အသုံးပြုတဲ့ အခါးမှာ သတိထားဖို့ လိုပါတယ်။ လိုင်းနံပါတ် (၁) မှာ stdio.h ဆိုတဲ့ header ဖိုင်တစ်ခု ပါပါတယ်။ အခုပုစ္စာမှာ သူမပါလည်း program က run ပါတယ်။
- ၂။ လိုင်းနံပါတ် (၂) မှာ ရေးထားတဲ့ statement ကတော့ C program ရဲ့ ရည်ရွယ်ချက်ကိုသိအောင် တော်မှတ်ထားတဲ့ မှတ်ချက်တစ်ခုပါပဲ။ comment လို့လည်း ခေါ်ပါတယ်။ ဒီစာတွေကို ဒီသက်တနှစ်ခု /* */ နဲ့ ရေးနောက် ပိတ်ထားမယ်ဆိုလို့ရှိရင် computer က မှတ်ချက်မှန်း သိသွားပါပြီ။ C program ကို run တဲ့ အခါးမှာ ဒီစာကြောင်းတွေကို ကျော်သွားပါလိမ့်မယ်။ BASIC language က REM နဲ့ သဘောအတူတူပါပဲ။
- ၃။ လိုင်းနံပါတ် (၃) ကတော့ main ဆိုတဲ့ function ကို ရေးထားတာပါ။ main နောက်မှာ ဖော်နှင့် နေတဲ့ ကွင်းနှစ်ခုရဲ့ အဓိပ္ပာယ်ဟာဆိုရင် ဒီ function အတွက် argument မရှိဘူးလို့ ဆိုလိုပါတယ်။
- ၄။ လိုင်းနံပါတ် (၄) မှာ brace ဆိုတဲ့ သက်တတ်ခု { ကို ရေးထားပါတယ်။ open brace ရဲ့ အဓိပ္ပာယ်ကတော့ main program စပြီလို့ ပြောလိုက်တာပါပဲ။
- ၅။ လိုင်းနံပါတ် (၅) ကတော့ C program မှာ အသုံးပြုမယ့် variable တွေရဲ့ type ကို declare လုပ်ပေးတာဖြစ်ပါတယ်။ type declaration section လို့ မှတ်ထားလိုက်ပါ။ radius နဲ့ area တို့ဟာဆိုရင် ဒေသမကိန်းအမျိုးအစားတွေ (floating-point number) လို့ ကြေညာလိုက်တာပဲ ဖြစ်ပါတယ်။
- ၆။ လိုင်းနံပါတ် (၆) ကတော့ printf ဆိုတဲ့ command ကို အသုံးပြုပြီး computer screen မှာ Enter the radius ဆိုတဲ့ message ပေါ်လာအောင် လုပ်ခိုင်းတာပါ။ အဲလို့ လုပ်တာကို computer စကားနဲ့ ပြောမယ်ရှိရင် information ကို prompt လုပ်ခိုင်းတယ်လို့ ခေါ်ပါတယ်။

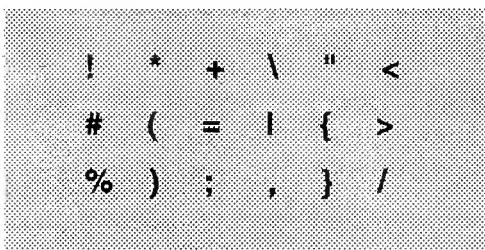
- ၇။ လိုင်းနံပါတ် (၇) မှာကျတော့ `scanf` ဆိုတဲ့ command ကို သုံးထားတာကြောင့် `radius` အတွက် data ကို `keyboard` ကနေ ကျွန်ုပ်တော်တို့ ရိုက်ထည့်ပေးရမှာဖြစ်ပါတယ်။

၈။ လိုင်းနံပါတ် (၈) ကတော့ `assignment statement` ပါ။ `area` ရဲ့ တန်ဖိုးကို ပုံသေနည်းသုံးပြီး ရွှာလိုက်တာပါပဲ။ ဒါ `statement` ကို `execute` လုပ်ပြီးတာနဲ့ `area` ရဲ့ တန်ဖိုးဟာ စက်တဲ့မှာ အဖြေရနေဖို့။

၉။ အခုခံရင် `radius` နဲ့ `area` တို့ရဲ့ တန်ဖိုးတွေကို `computer` က သိနေပြီးလေ။ ဒါပေမယ့် ကျွန်ုပ်တော်တို့က ဒီအဖြေကို မတောင်းဘဲနဲ့ `computer` က အလိုက်သိသီး သဘောကောင်းစွာနဲ့ အဖြေထဲတော်ပါဘူး။ တောင်းပုံတောင်းနည်းက လိုင်းနံပါတ် (၉) မှာ ပြထားတဲ့အတိုင်း `printf` command ကို သုံးပြီးတောင်းရပါတယ်။ အခုခံအခါကျွဲမှ `computer screen` ပေါ်မှာ $\text{Area} = 78.539825$ လို့ ပေါ်လာမှာပါ။

၁၀။ ကျွန်ုပ်တော်တို့ရေးထားတဲ့ C program ဟာ ဒီမှာတင် ဆုံးခန်းတိုင်သွားပါပြီ။ အဲဒါ `computer` ကို အသိပေးပို့အတွက် နောက်ဆုံး `statement` မှာ `closing brace }` သက်တနဲ့ ပိတ်ပေးလိုက်ပါ။ အခုမှ `program` တကယ်ပြီးသွားပါပြီ။ C program ဆိုတာ ဒါပါပဲ။ ဒါနဲ့ ကျွန်ုပ်တော်တို့ရေးထားသူမှာ အသိပေးပို့ ကျွန်ုပ်သွားပါတယ်။ ဘာလဲဆိုတော့ C program statement တွေကို ရေးပြီးတိုင်း `statement` တစ်ခုခုံးသွားရင် `semicolon` (;) သက်တနဲ့ အဆုံးသတ်ပေးဖို့ မမေ့ပါနဲ့။ အဲဒါ ကျွန်ုပ်သွားရင်တော့ `computer` က အားကြိုးစိတ်ဆိုးပြီး အမှားပေးပါလိမ့်မယ်။

3.1 Character Set



၁.၃ Identifiers and Keywords

C program မှာ အသုံးပြုတဲ့ variable name တွက် Identifier လို့ ခေါ်ပါတယ်။ variable name တွေလို့ လွယ်လွယ်ခေါ်လည်း ရတာပါပဲ။ variable name တစ်ခုကို သတ်မှတ်တဲ့အပါမှာ အဓိက အကျဆုံးကတော့ နာမည်ရဲ့ထိပ်ဆုံး character ဟာ စာအကွဲရာတစ်ခု ဖြစ်ရပါမယ်။ ဂဏန်းတို့၊ special character တို့ဖြစ်လို့ မရပါဘူး။ သူ့နောက်က ဥစ္စတွေကျတော့ စာချည်းပဲဖြစ်ဖြစ်၊ ဂဏန်းချည်းပဲဖြစ်ဖြစ်၊ ဒါမှမဟုတ် သူတို့တွေကို ဤကိုယ်သလိုတွေပြီးတော့ ရေးလို့ ရပါတယ်။ letter တွေကို စာလုံးအကြွေးနဲ့ရေးရေး၊ အသေးနဲ့ပဲရေးရေး ဘယ်လိုဖြစ်ဖြစ် ရပါတယ်။ underscore (_) ကိုလည်း variable name မှာ ထည့်သုံးနိုင်ပါတယ်။ ဒါပေမယ့် အနှစ်လက္ခဏာကိုတော့ variable name ထဲမှာ သုံးခွင့်မပြုပါဘူး။

EXAMPLE 1.2: The following names are valid identifiers.

x	y4	area
TABLE	Sum1	tax_rate
4th		the first character must be a letter
"x"		illegal character (")
sum-1		illegal character (-)
sum 2		no blank space allowed

C language အတွက်ဆိုပြီး သီးသန်သတ်မှတ်ထားတဲ့ reserved word တွေကို C keyword တွေလို့ ခေါ်ပါတယ်။ ဒီဥစ္စတွေကို variable name အနေနဲ့ သုံးလို့ မရပါပါဘူး။ ကွက်တိမတူရင်တော့ သုံးလို့ ရပါတယ်။ အဲဒီ keyword တွေကတော့ –

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

၁.၄ Constants

C program တစ်ခုမှာ မပြောင်းလဲဘဲတည်ရှိနေတဲ့ fixed value တွေကို constants လို့ အော်ပါတယ်။ constant ဘယ်နှစ်မျိုးရှိလဲဆိုတော့ (၁) မျိုးပါ။ အဲဒါတွေက (၁) Integer constants (၂) Floating-point constants (၃) Character constants နဲ့ (၄) String constant တို့ ဖြစ်က ပါတယ်။

Integer Constants

Integer constant ဆိုတာ ကိန်းပြည့်ကဏ္ဍးတစ်ခုကို ဆိုလိုတာပါ။ Integer constant မှာပဲ (၁) မျိုး ထပ်ခွဲထားပါသေးတယ်။ (၁) Decimal constant (base 10) (၂) Octal constant (base 8) နဲ့ (၃) Hexadecimal constant (base 16) တို့ ဖြစ်ပါတယ်။

Decimal Integer Constants

Decimal integer constant တစ်ခုဟာဆိုရင် သူညောနေ့ ၉ ထဲက ဂဏန်းတွေနဲ့ စုပေါင်းဖြစ်နေတာပါ။ ဒီ constant တွေမှာ ဂဏန်းအလုံးအရေအတွက် (၂) ခ သို့မဟုတ် (၂) ခထက် ပိုမျိုးရင် ထိပ်ဆုံးဂဏန်းဟာ သူညာဖြစ်လို့ မရပါဘူး။ နောက်ပြီးတော့ အဲဒီ constant အတွက် သတ်မှတ်ထားတဲ့ lower bound နဲ့ upper bound ကြားက value တွေနဲ့ပဲ constant ဟာ သက်ဆိုင်ပါတယ်။

EXAMPLE 1.3: Here are the several valid decimal integer constants.

0 1547 7381 32767

The following decimal integer constants are not valid for the reasons stated.

12,567	illegal character (,)
36.54	illegal character (.)
12 34 56	no blank space allowed
123-456	illegal character (-)
0999	first character cannot be a zero
75678	maximum bound cannot be exceeded

Octal Integer Constants

Octal integer constant ඩීතා යුතුකළ ග තැන ගණන්: තෝක්. ඉපින්: ප්‍රතිඵලිතාවෙන් සියලුමය් Octal මූල්: විභාගයේ ලදී. තිබුණ්: ගණන්: කිහිපාවා යුතු මූල්: පෙ: රඩ් මය්.

EXAMPLE 1.4: Several valid octal integer constants are shown below.

0 01 0564 07777

The following octal integer constants are written incorrectly for the reasons stated.

456 does not begin with zero
0658 illegal digit (8)
0777.54 illegal character (.)

Hexadecimal Integer Constants

Hexadecimal integer constant ඩීතා ගණන් (0) තැන ගණන්: තෝක්. අතු ආගක් f තැන letter තෝක්. ඉපින්: ප්‍රතිඵලිතාවෙන් letter තෝක් uppercase ඩී. මහුත් lowercase ප්‍රතිඵලි. එහිතයේද මූල්: (J) ලු: කි. ox ඩී. මහුත් 0X ලදී. ප්‍රතිඵලිතයේද letter A ආගක් F නිස් තැන්: තෝක්. (20) ආගක් (2E) ගි. කියන්තා: පු ඩී: තා: තාවිධ්‍ය.

EXAMPLE 1.5: The following are valid hexadecimal integer constants.

0x 0X2 0X7FFA 0xabcd

The following hexadecimal integer constants are written incorrectly for the reasons stated.

0X12.34 illegal character (.)
OBE45 does not begin with ox or 0X
0x.97cd illegal character (.)
0XDEFG illegal character (G)

Floating-point Constants

Floating-point constant ဆိတ် base 10 ကို အခြေခံတဲ့ real number တွေကို ဆိုလိုပါတယ်။ သူတို့ကို အသမဂကန်းအနေနဲ့ဖြစ်ဖြစ်၊ exponent ထပ်ကိန်းပုံစံနဲ့ပြဖြစ်ဖြစ် ရေးပြလို့ရပါတယ်။

EXAMPLE 1.6: Several valid floating-point constants are shown below.

1.	0.75	765.98	50000.0
2E-8	0.007e-2	1.45E+7	.33333e12

The following are not valid floating-point constants for the reasons stated.

123	a decimal point or an exponent is missing
1,234.6	illegal character (,)
3E+2.5	the exponent must be an integer
4e 12	blank space in the exponent is illegal

Character Constants

Character constant ဆိတ်ကတော့ apostrophes (') တွေနဲ့ ပိတ်ပေးထားတဲ့ character တစ်ခုကို ဆိုလိုတာပါ။ character ဟာ ဂဏန်းဖြစ်ဖြစ်၊ စာပဲဖြစ်ဖြစ်၊ blank ကိုတောင် character လိုလက်ခံပါတယ်။

EXAMPLE 1.7: Several character constants are shown below.

'A'	'X'	'7'	' '
-----	-----	-----	-----

Character constant တွေမှာ numeric values ရှိကြပါတယ်။ ဒီ value တွေကို American Standard Code for Information (ASCII) တန်ဖိုးတွေအနေနဲ့ သတ်မှတ်ထားပါတယ်။ ASCII character set တွေကို စာအုပ်နောက်ဆုံးစာမျက်နှာတွေမှာ ဖော်ပြထားတာကို လေ့လာကြည့်ပါ။

EXAMPLE 1.8: Several character constants and their corresponding values, as defined by the ASCII character set, are shown below.

Constant	Value
'A'	65
'x'	120
'3'	51
'\$'	36
'.'	32

String Constants

String constant ဆိတ် double quotation marks (" ") ထဲမှာ ရေးထားတဲ့ character တွေကို ဆိုလိပါတယ်။ ဒီ character တွေထဲမှာ blank ပါလည်း လက်ခံပါတယ်။ စာလုံးအရေအတွက် ဘယ်လောက်ပါရမယ်လို့ မကန့်သတ်ပါဘူး။

EXAMPLE 1.9: Several string constants are shown below.

"blue" "YANGON" "25 - 10 - 1947"
"Myanmar" "Line 1\nLine 2\nLine 3"
" " " "

ဒီဥပမာမှာ string constant တစ်ခုဖြစ်တဲ့ "Line 1\nLine 2\nLine 3" ဟာဆိုရင်-
Line 1
Line 2
Line 3

ဆိတ် စာကြောင်းသုံးကြောင်းကို တစ်ကြောင်းတည်းဖြစ်အောင် '\n' ကို အသုံးပြုပြီး ရေးထားတာပါ။ အဲဒီ '\n' ကို Escape sequence လို့ ခေါ်ပါတယ်။ သူ့ကို backslash (\) နဲ့ အမြဲစပြီးတော့ သူ့နောက်မှာ special character တစ်ခု ပါရပါတယ်။ '\n' လို့ ရေးထားတယ်ဆိုရင် ပထမစာကြောင်းရိုက်ပြီးတဲ့အခါမှာ စာကြောင်းအသစ်ကိုကူးပါလို့ ဆိုလိုတာပါပဲ။ '\a' လို့ ရေးထားရင် အချက်ပေး bell သံမြေည်မှာပါ။ C မှာ အသုံးများတဲ့ escape sequence တချို့ကို ဖော်ပြထားပါတယ်။ လေ့လာကြည့်ပါ။

<i>Character</i>	<i>Escape Sequence</i>	<i>ASCII Value</i>
bell (alert)	\a	007
backspace	\b	008
horizontal tab	\t	009
vertical tab	\v	011
new line	\n	010
form feed	\f	012
carriage return	\r	013
quotation mark	\"	034
apostrophe	\'	039
question mark	\?	063
backslash	\	092
null	\0	000

```
main ()
{
    printf ( "Each\\tword\\nis\\ntabbed\\tov\\rone" );
}
```

ဒါ program ကို runလိုက်မယ်ဆိုရင် computer screen မှာ ဒီလိပ်ပေါ်လာမှပါ။

Each word is tabbed over once

```
main ()  
{  
    printf ("Mg Mg told Su Su, \"Let's go !\" \n");  
}
```

ဒါ program ကို ကျွန်ုပ်တော်တိ၊ run လိုက်မယ်ဆိုရင် ဘာပေါ်လာမလဲဆိုတော့---

Mg Mg told Su Su, "Let's go !"

တတိယောက် ဒီဥပမာကတော့ hexadecimal number DB ဆိုတာကို သုံးပြီးတော့ လေးထောင့်ကွက် graphic character တစ်ခုကို ရေးပြနှာပါ။

```
main ()  
{  
    printf ("Here is the character : \xDB");  
}
```

ဒါ program ကို run လိုက်မယ်ဆိုလို ရှိရင် computer မှာ ဒီလိုပေါ်လာမှာပါပဲ။

Here is the character : █

၁.၅ Variables and Arrays

Variable ဆိုတာ program ထဲက information တွေကိုသိမ်းဖို့ အသုံးပြုတဲ့ မသိကိန်းရှင် တစ်ခုပါပဲ။ variable တစ်ခုရဲ့ တန်ဖိုးတွေကို program ထဲမှာ ကြိုက်သလိုပြောင်းလဲလို့ ရပါတယ်။ ဒါပေါ်မယ် သူနဲ့ပတ်သက်တဲ့ data type ကတော့ မပြောင်းလဲပါဘူး။ type declaration မှာ ကြေညာထားတဲ့အတိုင်း အမြှို့နှိမ်မှာပါ။

EXAMPLE 1.10: A C program contains the following lines.

```
int a, b, c;  
char d;  
  
a = 3; b = 5; c = a+b; d = 'a';  
  
a = 4; b = 2; c = a-b; d = 'W';
```

ဒါ program မှာဆိုရင် ပထမလိုင်းနှစ်ကြောင်းဟာ data type ကြည့်သာ ဖြစ်ပါတယ်။ a, b, c တို့ဟာ integer variable တွေပါတဲ့။ d ကတေသာ character type variable ပဲ့။ ဒါ type declaration ဟာ program တစ်ခုလုံးနဲ့ သက်ဆိုင်ပါတယ်။ program ရဲ့ third line မှာ a = 3, b = 5 လို့ assign လုပ်လိုက်တဲ့အတွက် a + b = 8 ဖြစ်သွားပြီလေ။ d ကိုတေသာ character 'a' နဲ့ တွေတယ်လို့ assign လုပ်ထားပါတယ်။ ဒီတန်ဖိုးတွေဟာ အခုထိ ဝယ်မြှုနေတုန်းပါ။ ဒါပေမယ့် program ရဲ့ fourth line က new assignment တွေကြောင့် a, b, c, d တို့ရဲ့ value တွေဟာ ပြောင်းသွားပါပြီ။ ဒါကြောင့်မို့လည်း a, b, c, d တို့ကို variable တွေလို့ ပြောရခြင်းပါပဲ။

Array ဆိုတာကလည်း identifier တစ်မျိုးပါပဲ။ ဒါပေမယ့် သူ့မှာ နာမည်တွေ အများကြီး မထားဘူး။ တစ်ခုပဲထားတယ်။ data item အများကြီးကို သူ့နည်းသူ့ဟန်နဲ့ သိမ်းနိုင်အောင် ဖန်တီးထားပါတယ်။ data item တစ်ခုချင်းကို array element လို့ ခေါ်ထားသလို array element တစ်ခုချင်းကို subscript တွေနဲ့ ချွဲ့ခြားသတ်မှတ်ထားပါတယ်။ Array နဲ့ ပတ်သက်တဲ့ ဥပမာဏေးကို အောက်မှာဖော်ပြထားပါတယ်။ array element ဆိုတာဘာလဲဆိုတာ သိလာမှာပါ။

EXAMPLE 1.11: Suppose that the string “Myanmar” is to be stored in a one dimensional array called **letter**. Since “Myanmar” contains 7 characters, letter will be an 8-element array. Thus, **letter[0]** will represent the letter **M**, **letter[1]** will represent **y**, and so on. The last array element **letter[7]** represents the **null character \0** which indicates the end of the string.

Element Number	Subscript Value	Array Element	String Data Item
1	0	letter[0]	M
2	1	letter[1]	y
3	2	letter[2]	a
4	3	letter[3]	n
5	4	letter[4]	m
6	5	letter[5]	a
7	6	letter[6]	r
8	7	letter[7]	o

Letter ဆိတဲ့ array တစ်ခုကို character type ဖြစ်ကြောင်း ဒီလို declare လုပ်လို.ရပါတယ်။

```
char letter[] = "Myanmar";
```

ဒဲ type declaration ကိုပဲ နောက်ဒီလိုရေးလို.ရပါတယ်။

```
char letter[8] = "Myanmar";
```

ဒီနေရာမှာ array size ကို တိတိကျကျ ဖော်ပြထားပါတယ်။ ဒါပေမယ့် သတိထားရမှာက array size ဟာ သိပ်ငယ်နေရင်လည်း မဖြစ်ပါဘူး။ နောက်ကစာလုံးတွေ ပျောက်သွားနိုင်သလို သိပ်ကြီးပြန်လည်း နောက်မှာ ထပ်တိုးလာမယ့် စာလုံးတွေဟာ သူညေတွေဖြစ်နိုင်သလို အဓိုက်မရှိဘူး။ character တွေ တိုးလာမှာ ထိုးရပါတယ်။

၁.၆ Data Types

C language မှာ data type အမျိုးစုရိနေပါတယ်။ ကိုယ်တွက်ချင်တဲ့ ကဏ္ဍားနဲ့ data type format ချင်း match ဖြစ်မှ program က အဖြေမှန်ထွက်လာမှာပါ။ နှဲ.မို.ရင်တော့ အဖြေထွက်လာပေမယ့် တလွှဲတွေဖြစ်နေမှာပါပဲ။ အောက်ကယေားမှာ basic data type တွေကို ဖော်ပြထားပါတယ်။

<i>Data type</i>	<i>Description and Range</i>	<i>Format</i>	<i>Typical memory requirement</i>
char	single character	%c	1 byte
signed char	-128 to + 127	%d	
unsigned char	0 to 255		
int	integer quantity	%d	2 bytes
signed int	-32768 to + 32767		
unsigned int	0 to 65535		
long int	-2147483648 to + 2147483647	%ld	4 bytes
float	floating-point number 3.4E-38 to 3.4E+38	%f	1 word (4 bytes)
double	double-precision floating-point number 1.7E-308 to 1.7E+308	%lf	2 words
long double	3.4E-4932 to 3.4E+4932	%Lf	10 bytes

Type declaration နဲ့ ပတ်သက်လို့ အောက်မှာ့ပါတယ်။ လေးလာကြည့်ပါ။

EXAMPLE 1.12: A C program contains the following type declarations and statement.

```

float a = 3, b = 4, c;
c = a + b;
printf ("C = %f", c);
c += a + b;
printf ("New C = %f", c);

```

ဒါ program မှာ စေခဲ့တယ်။ statement ဟာ data type declaration ပဲဖြစ်ပါတယ်။ a, b, c တို့ဟာ floating-point variable တွေပါတယ်။ a = 3, b = 4 လို့ initialize လုပ်ထားတာကို ထွေမှာပါ။ အဲလို့မရေးဘဲ assignment statement အနေနဲ့ အောက်မှာရေးလည်းရပါတယ်။ a နဲ့ b

ဘန်ဖိုးတွေသားပြီဆိုရင် C တန်ဖိုးကို computer က တွက်ပြီးသွားပါပြီ။ ပြီးတော့ရင် printf command သုံးပြီး C တန်ဖိုးကို computer screen မှာ ရိုက်ပြခိုင်းပါတယ်။ format က %f နဲ့ ရိုက်ခိုင်းထားလေ့ကြည့် C = 7.000000 လို့ computer မှာ ပေါ်လာပါလိမ့်မယ်။ ဒါကို free format နဲ့ ပြခိုင်းလေ့ ခေါ်ပါတယ်။ C + = a + b ဆိတဲ့ statement ကတော့ incrementing type statement ပါပဲ။ မူလ C = 7 တန်ဖိုးထဲကို a + b ပေါင်းလဒ်ကို ထပ်ပေါင်းပေးပါလို့ ဆိုလိုပါတယ်။ တစ်နည်းအားဖြင့် C = C + a + b ကို တစ်မျိုးပြောင်းရေးထားတာပါ။ အခုဆိုရင် C = 14.000000 ဖြစ်နေပြီနောက်။ ရိုက်ခိုင်းလိုက်ရင် computer screen မှာ C = 14.000000 လို့ ပေါ်နေမှာပါပဲ။

၁.၇ Symbolic Constants

Symbolic character ဆိတာ character အစုတစ်ခု သို့မဟုတ် numeric value တစ်ခု သို့မဟုတ် statement တစ်ခုကို အမည်တစ်ခုနဲ့ အစားပေးထားတဲ့ name ကိုဆိုလိုပါတယ်။ program ကို compile လုပ်တဲ့အခါမှာ symbolic constant တွေအတော့ သူနဲ့သက်ဆိုင်ရာ character sequence တွေ ပြန်ဝင်လာမှာပါ။ Symbolic constant တွေကို program ရဲ့ အစဉ်းမှာ define လုပ်ထားရပါမယ်။ define လုပ်နည်းကို အောက်မှာ နမူနာဖော်ပြထားပါတယ်။ လေ့လာကြည့်ပါ။

EXAMPLE 1.13: A C program contains the following symbolic constant definitions.

# define	TAXRATE	0.25
# define	PI	3.141593
# define	TRUE	1
# define	FALSE	0
# define	FRIEND	"ARKAR"
# define	PRINTX	printf ("%d\n", x)

Symbolic name တွေကို နာမည်ပေးတဲ့အခါမှာ Uppercase letter တွေနဲ့ပဲ ရေးထားပါတယ်။ ဒါမှုလည်း identifier name တွေနဲ့ ကွဲပြားမှာပါ။ နောက်ပြီးတော့ define statement နောက်ဆုံးမှာ semicolon (;) ရေးစရာမလိုဘူးဆိတာကို သတိပြုပါ။

၁.၈ Operators and Expressions

တစ်သီးတြားရှိနေတဲ့ constant တွေ၊ variable တွေ၊ array element တွေကို operator ဘွားပြီး ဆက်စပ်လိုက်တဲ့အခါမှာ expression ဆိုတာ ဖြစ်လာတာပါပဲ။ operator ဘယ်နှစ်မျိုးရှိလဲ။ operator တွေမှာ Arithmetic operators၊ Unary operators၊ Relational and logical operators၊ Assignment operators၊ Conditional operators စသည်ဖြင့် အမျိုးအစားတွေ ဘယ့်ကြီးရှိတာပေါ့။ အဲဒါတွေ သုံးပြီးတော့ expression ဆိုတာ ဖြစ်လာတာပါ။ Operatorကနေ operate အလုပ်ခဲရတဲ့ data item တွေကို operand လို့ ခေါပါတယ်။

Arithmetic Operators

C language မှာ arithmetic operator (၅) မျိုးရှိပါတယ်။ အဲဒါတွေကတော့ ...

Operator	Purpose
+	addition
-	subtraction
*	multiplication
/	division
%	remainder after integer division, which is sometimes referred to as the modulus operator

www.foxitsolutions.com

C language မှာ ထပ်ကိန်း (power) တင်တဲ့ operator မရှိပါဘူး။ power တင်ချင်ရင် library ကိုခေါ်ပြီး pow() ဆိုတဲ့ exponentiation function ကို သုံးရမှာပါ။ အဲဒါအခါမှာ # include ရှာ <math.h> ဆိုတဲ့ header file ထည့်ရေးပေးရပါမယ်။

EXAMPLE 1.14: Suppose that x and y are integer variables, whose values are 10 and 4 respectively. Several arithmetic expressions involving these variables are shown below.

```

main ()
{
    int x = 10, y = 4;

    printf ("x + y = %d\n", x + y);
    printf ("x - y = %d\n", x - y);
    printf ("x * y = %d\n", x * y);
    printf ("x / y = %f\n", x / y);
    printf ("x % y = "); printf ("%d", x % y);
}

```

ဒါ program ကို run လိုက်တဲ့အပါမှာ computer မှာ ဒီလိုပေါ်လာပါလိမ့်မယ်။

```

x + y = 14
x - y = 6
x * y = 40
x / y = 2
x % y = 2

```

EXAMPLE 1.15: Suppose that x and y are floating-point variables whose values are 12.5 and 2.4 respectively. Several arithmetic expressions involving these variables are shown below.

```

main ()
{
    float x = 12.5, y = 2.4;

    printf ("x + y = %f\n", x + y);
    printf ("x - y = %f\n", x - y);
    printf ("x * y = %f\n", x * y);
    printf ("x / y = %f", x / y);
}

```

ဒါ program ကို ကျွန်တော်တို့ run ကြည့်ပါမယ်။ လေးလာကြည့်ပါ။

```

x + y = 14.900000
x - y = 10.100000
x * y = 30.000001
x / y = 5.208333

```

EXAMPLE 1.16: Suppose that c1 and c2 are character-type variables that represent the characters A and Z respectively. Several arithmetic expressions that make use of these variables are shown below.

```

main ()
{
    char c1 = 'A', c2 = 'Z';
    printf ("c1 = %d\n", c1 );
    printf ("c1 + c2 = %d\n", c1 + c2 );
    printf ("c1 + c2 + 5 = %d\n", c1 + c2 + 5 );
    printf ("c1 + c2 + '5' = %d", c1 + c2 + '5' );
}

```

ဒီ program ကို run လိုက်တဲ့အခါ computer မှာ ဒီလိုပေါ်လာမှာပါ။

```

c1 = 65
c1 + c2 = 155
c1 + c2 + 5 = 160
c1 + c2 + '5' = 208

```

www.mkyong.com

ဒီ program မှာဆိုရင် A = 65, Z = 90, '5' = 53 လို့ ASCII character set ယေားကနေ တန်ဖိုးတွေ ယူထားတာပါ။

Expression တစ်ခုရဲ့ value ကို ကိုယ်လိုချင်တဲ့ data type ဖြစ်အောင်လို့, type cast ဆိုတဲ့ ပုံစံနဲ့ force လုပ်ယူလို့ရပါတယ်။ လုပ်ပုံလုပ်နည်းကတော့ ကိုယ်ဖြစ်ချင်တဲ့ data type ကို ကွင်းခတ်ပြီးတော့ expression ရှုံးမှာ ထားပေးရပါမယ်။

EXAMPLE 1.17: Suppose that *i* is an integer variable whose value is 15, and *f* is a floating-point variable whose value is 7.6. The expression $(i + f) \% 4$ is invalid because the first operand $(i + f)$ is a floating-point rather than integer.

ဒီပုံစွဲမှာ $(i + f) \% 4$ ကို print လုပ်ခိုင်:ရင် မှာ:ပါတယ်။ ဘာပြုလိုလဲဆိုတော့ % operator ကို သုံးရင် integer တွေကိုပဲ အကြောင်းရှာခိုင်းလို့ရတာကိုး။ ဒီတော့ $(i + f)$ value ကို integer ဖြစ်အောင် အရေးပေါ် ပြုပြင်ဖို့လိုလာပြီ။ ဘယ်လိုပြုပြင်မလဲဆိုတော့ $(\text{int}) (i + f)$ လို့ ရေးလိုက်ရင် $(i + f)$ value ဟာ လောလောဆယ် integer ဖြစ်သွားပါပြီ။ ဒါကြောင့်မို့လည်း $(\text{int}) (i + f) \% 4$ ကို print လုပ်လိုက်မယ်ဆိုလို့ရှိရင် အကြောင်း (j) ရမှာပါပဲ။

Unary Operators

Unary operator ဆိတ် operand တစ်ခုကို သူ့ကြောင့် new value ဖြစ်သွားနေနိုင်တဲ့ operator တစ်မျိုးပါ။ ဥပမာ Unary minus ဆိုရင် သူ့ကြောင့် operand တွေဟာ လက္ခဏာ ပြောင်းသွားပါမယ်။ ပြီးတော့ increment operator (++) နဲ့ decrement operator (--) ဆိတ်တွေလည်း ရှိပါတယ်။ increment operator ကြောင့် operand တစ်ခုဟာ တစ်တိုးသွားသလို decrement operator ကြောင့် operand ဟာ တစ်လျော့သွားမှာပါ။ ဒီဥစွာတွေ လက်တွေအသုံးချနည်းကို အောက်မှာ ဥပမာတစ်ပုဒ်နဲ့ ရှင်းပြထားပါတယ်။ လေ့လာကြည်ပါ။

EXAMPLE 1.18: A C program includes an integer variable *i* whose initial value is 2. The program shows the utilisation of the increment and decrement operators in two different ways, depending on whether the operator is written before or after the operand.

```
main ()
{
    int i = 2;

    printf ("i = %d\n", i );
    printf ("i = %d\n", ++i );
    printf ("i = %d\n", i );
    printf ("i = %d\n", i++ );
    printf ("i = %d", i );
}
```

ဒါ program ကို run လိုက်တဲ့အခါ computer မှာ ဒီလိုပေါ်လာမှာပါ။

```
i = 2  
i = 3  
i = 3  
i = 3  
i = 4
```

ဒါ program မှာ ရှင်းစရာလေးတွေ ပေါ်နေပါတယ်။ ဒါပေမယ့် printf statement တွေပဲ ကွက်ပြီး ရှင်းပါမယ်။ ပထမ printf ကြောင့် computer မှာ i = 2 လို့ ပေါ်လာတာပါ။ ဒုတိယ statement မှာ ++i လို့ ရေးထားတဲ့အတွက် i တန်ဖိုးကို တစ်ခုအရင်တိုးပြီးမှ အဖြေကို ရိုက်ပါလိမ့်မယ်။ ဒါတော့ i = 3 လို့ computer screen မှာ ပေါ်လာပြီပေါ့။ တတိယ statement ကျတော့ i က 3 လို့ ထပ်ရိုက်ခိုင်းတဲ့အတွက် i = 3 က ထပ်ပေါ်လာတာပါ။ စတုတွေ statement ကတော့ i++ လို့ ရေးထားတာကြောင့် printf လုပ်ပြီးမှ i value ကို တစ်ခုတိုးပါ လိမ့်မယ်။ ဒါကြောင့်လည်း i = 3 ဟာ computer မှာ ဆက်ပေါ်နေတာပါ။ နောက်ဆုံး statement ကျတော့မှ i ထဲက နောက်ဆုံးတန်ဖိုး 4 ကို ရိုက်ပါပြီ။ ဒါဆိုရင် increment operator အကြောင်းကို သဘောပေါက်သွားပြီလို့ ထင်ပါတယ်။

Relational and Logical Operators

C language မှာ relational operator (၄) မျိုးရှုပါတယ်။ အဲဒါတွေကတော့ ...

<u>Operator</u>	<u>Meaning</u>
<	less than
<=	less than or equal to
>	greater than
>=	greater than or equal to

relational operator တွေလို့ အရေးပါတဲ့ equality operator တွေရှိပါသေးတယ်။ အဲဒါ operator နှစ်မျိုးကတော့ ...

<u>Operator</u>	<u>Meaning</u>
<code>==</code>	equal to
<code>!=</code>	not equal to

EXAMPLE 1.19: Suppose that i, j and k are integer variables whose values are 1, 2 and 3 respectively. Several logical expressions involving these variables are shown below.

<u>Expression</u>	<u>Interpretation</u>	<u>Value</u>
<code>i < j</code>	true	1
<code>(i + j) >= k</code>	true	1
<code>(j + k) > (i + 5)</code>	false	0
<code>k != 3</code>	false	0
<code>j == 2</code>	true	1

စောင့်တွက် operator (၆) မျိုးအပြင် C language မှာ logical operator ဟုပါသေးတယ်။ အခြား operator တွေရဲ့ သက်တနဲ့ အဓိပ္ပာယ်က ဒီလိုပါ။

<u>Operator</u>	<u>Meaning</u>
<code>&&</code>	and
<code> </code>	or

EXAMPLE 1.20: Suppose that i is an integer variable whose value is 7, f is a floating-point variable whose value is 5.5, and c is a character variable that represents the character 'w'. Several complex logical expressions that make use of these variables are shown below.

Expression	Interpretation	Value
<code>(i >= 6) && (c == 'w')</code>	true	1
<code>(i >= 6) (c == 119)</code>	true	1
<code>(f < 11) && (i > 100)</code>	false	0
<code>(c != 'p') ((i + f) <= 10)</code>	true	1

Assignment Operators

C language မှာ assignment operator အမျိုးမျိုး ရှိပါတယ်။ အသုံးများဆုံး operator ကတေသာ ညီမျှခြင်း (=) ပါပဲ။ multiple assignment ကို ဒီလိုပုံစံနဲ့ ရေးလိုပါတယ်။

identifier 1 = identifier 2 = ... = expression

ဒီ assignment မှာဆိုရင် operation ကို ညာဘက်ကနေ ဘယ်ဘက်ကို ဦးတည်လုပ်သွားမှုပါ။ operator အတော်များများရဲ့ precedence ကို အောက်မှာ ပေါ်ပေါ်ဖော်ပါတယ်။ လေ့လာ ကြည့်ပါ။

www.foxitsoftware.com

Operator Category

Operators

Associativity

Unary operators	- + + - - ! sizeof	R ----> L
Arithmetic multiply, divide and remainder	* / %	L -----> R
Arithmetic add and subtract	+ -	L -----> R
Relational operators	< <= > >=	L -----> R
Equality operators	= = !=	L -----> R
Logical and	&&	L -----> R
Logical or		L -----> R
Assignment operators	= += -= *= %=	R -----> L

Conditional Operators

လွယ်ကူတဲ့ conditional operation တစ်ခုကို conditional operator (? :) သုံးဖြီးတော့ အလုပ်လုပ်ခိုင်းလို့ ရပါတယ်။ conditional expression တစ်ခုရဲ့ general form က ဒီလိုပါ။

variable = expression 1 ? expression 2 : expression 3

အခုလို conditional expression တစ်ခုကို evaluate လုပ်ရမှာ expression 1 ဟာ မူန်ဘူးလား ဆိုတာကို အရင်ဆုံးဖြတ်ရမှာပါ။ expression 1 က မှန်နောက်ထိရင် (?) အောက်က expression 2 ကို evaluate လုပ်ပြီးတော့ ရတဲ့အဖြစ်ကို variable နဲ့ ညီမျှပေးရပါမယ်။ expression 1 ဟာ မှားနောက် ဆိုရင်တော့ expression 3 ကို evaluate လုပ်လို့ ရတဲ့အဖြစ်၊ variable ကို ညီပေးရမှာပါ။

EXAMPLE 1.21: Here is a C program that makes use of the conditional operator to find the larger number from the two numbers assigned from the keyboard.

```
main ()
{
    int    a, b, larg;
    printf ("Enter two numbers : ");
    scanf ("%d, %d", &a, &b);
    larg = (a > b) ? a : b;
    printf ("The larger number is %d", larg);
}
```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter two numbers : 12, 7

The larger number is 12

- ၁။ ဒီ program မှာဆိုရင် ပထမဆုံး statement က a, b, larg တို့ကို integer ဖြစ်ကြောင်း ကြေညာတာပါ။
- ၂။ ဒုတိယ statement ကတေသာ့ Enter two numbers : ဆိုတဲ့ prompt ကို computer မှာ display လုပ်ခိုင်းတာပါပဲ။
- ၃။ တတိယ statement ကတေသာ့ a နဲ့ b တို့ကူး data တွေကို keyboard ကနေ သွင်းပေးရမှာပါ။
- ၄။ စတုထွေ statement ကတေသာ့ conditional expression ဖြစ်ပါတယ်။ a ဟာ b ထက် ကြီးလားလို့ မေးပြီးတော့၊ ဟုတ်တယ်ဆိုရင် larg ကို a နဲ့တူတယ်လို့ assign လုပ်မယ်။ မဟုတ်ဘူးဆိုရင် larg ကို b နဲ့ assign လုပ်မှာပါ။
- ၅။ နောက်ဆုံး statement ကတေသာ့ larg ထဲကအဖြေကို computer မှာ ပေါ်လာအောင် လုပ်ခိုင်းတာပါပဲ။ ဒါကြောင့်မို့လည်း computer မှာ The larger number is 12 လို့ ပေါ်လာတာပါ။

၁.၉ Library Functions

C language မှာ အသင့်ရေးပြီးသား library function တွေ အများကြီးရှုပါတယ်။ အသုံးများတဲ့ library function အများစုကို ယေားနဲ့ဖော်ပြထားပါတယ်။ ဒီယေားမှာပါတဲ့ Type ဆိုတဲ့ column ဟာ function ကနေ return ပြန်လာမယ့် data type ကို ဆိုလိုတာပါ။ အသုံးပြန်လုံးကို EXAMPLE 1.22 မှာ ဖော်ပြထားပါတယ်။ လေ့လာကြည့်ပါ။

Function	Type	Purpose
abs (i)	int	return the absolute of i
ceil (d)	double	round up to the next integer value (the smallest integer that is greater than or equal to d)
cos (d)	double	return the cosine of d
cosh (d)	double	return the hyperbolic cosine of d
exp (d)	double	raise e to the power d (e = 2.7182818..)
fabs (d)	double	return the absolute value of d
floor (d)	double	round down to the next integer value (the largest integer that does not exceed d)

<u>Function</u>	<u>Type</u>	<u>Purpose</u>
fmod (d1, d2)	double	return the remainder (the noninteger part of the quotient) of d1/d2 with same sign as d1
getchar ()	int	enter a character from the standard input device
log (d)	double	return the natural logarithm of d
pow (d1, d2)	double	return d1 raised to the d2 power
printf (...)	int	send data items to the standard output device
putchar (c)	int	send a character to the standard output device
rand ()	int	return a random positive integer
sin (d)	double	return the sine of d
sqrt (d)	double	return the square root of d
scanf (...)	int	enter data items from the standard input device
tan (d)	double	return the tangent of d
toascii (c)	int	convert value of argument to ASCII
tolower (c)	char	convert letter to lowercase
toupper (c)	char	convert letter to uppercase

www.foxitsoftware.com

EXAMPLE 1.22: This program demonstrates the use of library functions.

```
# include <math.h>

main ()
{
    printf ("abs (-12.954) = %dn", abs (-12.954));
    printf ("ceil (12.001) = %ln", ceil (12.001));
    printf ("ceil (-12.995) = %ln", ceil (-12.995));
    printf ("cos (60)      = %ln", cos (60 * 3.141593 / 180));
}
```

```

printf ("cosh (60)      = %f\n", cosh (60 * 3.141593 / 180));
printf ("exp (2.5)       = %f\n", exp (2.5));
printf ("fabs (-12.995)  = %f\n", fabs (-12.995));
printf ("floor (12.995)  = %f\n", floor (12.995));
printf ("fmod (9, 2.5)    = %f\n", fmod (9, 2.5));
printf ("log (12.5)       = %f\n", log (12.5));
printf ("rand ( )          = %d\n", rand ( ));
printf ("sin (30)          = %f\n", sin (30 * 3.141593/180));
printf ("sqrt (6.25)       = %f\n", sqrt (6.25));
printf ("tan (45)          = %f\n", tan (45 * 3.141593/180));
printf ("atan (1) in radians = %f degrees\n",
      atan (1) * 180 / 3.141593);
printf ("atan2 (1, 1.732051) * 180 / 3.141593 = %f \n",
      atan2 (1, 1.732051) * 180 / 3.141593);
printf ("tolower ('z')     = %c\n", tolower ('z'));
printf ("toupper ('a')     = %c\n", toupper ('z'));
printf ("pow (2.5, 2)       = %f\n", pow (2.5, 2));
}

```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

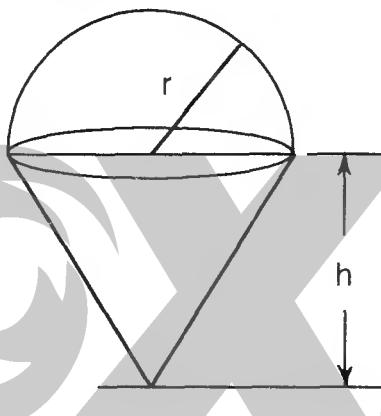
abs (-12.954)	= 12
ceil (12.001)	= 13.000000
ceil (-12.995)	= -12.000000
cos (60)	= 0.500000
cosh (60)	= 1.600287
exp (2.5)	= 12.182494
fabs (-12.995)	= 12.995000
floor (12.995)	= 12.000000
fmod (9, 2.5)	= 1.500000
log (12.5)	= 2.525729
rand ()	= 345
sin (30)	= 0.500000
sqrt (6.25)	= 2.500000

```

tan (45)           = 1.000000
atan (1) in radians = 44.999995 degrees
atan2 (1, 1.732051) * 180 / 3.141593 =29.999994
tolower ('z')      = z
toupper ('a')      = A
pow (2.5, 2)       = 6.25

```

EXAMPLE 1.23: Write a C program that computes the volume of an ice cream cone as shown in the figure.



```

#include <math.h>
#define PI 3.141593

main ( )
{
    float r, h, vol;

    printf ("Enter radius and height\n");
    scanf ("%f, %f", &r, &h);
    vol = (4/3.) * PI * pow (r, 3) / 2 + PI * r * r * h / 3;
    printf ("Volume of ice cream cone = %f\n", vol);
}

```

ဒါ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်လာမှာပါ။

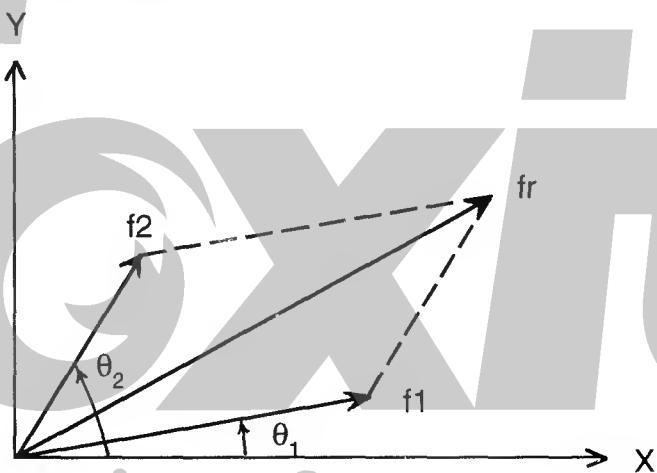
Enter radius and height

1, 1

Volume of ice cream cone = 3.141593

EXAMPLE 1.24: Vectors

Write a C program that reads the values of f and θ (in degrees) of two forces, f_1 and f_2 , sums the horizontal and vertical components of the forces to find the total force acting, and then computes the angle (in degrees) it makes with the X-axis.



```
# include <math.h>
# define PI 3.141593
/* this program computes the magnitude and
direction of the resultant force of the
two vectors.
*/
main ( )
{
    float f1, f2, fx, fy, fr, theta1, theta2, theta;
```

```

printf ("Enter the two vectors (lbs)\n");
scanf ("%f,%f", &f1, &f2);
printf ("Enter the directions (degs)\n");
scanf ("%f,%f", &theta1, &theta2);
theta1 *= PI / 180;
theta2 *= PI / 180;
fx = f1 * cos (theta1) + f2 * cos (theta2);
fy = f1 * sin (theta1) + f2 * sin (theta2);
fr = sqrt (fx * fx + fy * fy);
theta = atan2 (fy, fx) * 180 / PI;
printf ("Resultant force (lbs) = %f\n", fr);
printf ("Angle theta (degs) = %f\n", theta);
}

```

ဒဲ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter the two vectors (lbs)

40,15

Enter the directions (degs)

25,55

Resultant force (lbs) = 53.518509

Angle theta (degs) = 33.055859

EXAMPLE 1.25: This program illustrates the use of assignment operators.

```
# define PRINTX    printf ("%d\n", x)

main ( )
{
    int x = 2, y, z;

    x *= 3 + 2;
    PRINTX;
    /* x = x (3 + 2) = (2) (5) = 10 */

    x *= y = z = 4;
    PRINTX;
    /* z = 4 -> y = 4 -> x = x(y) = (10) (4) = 40 */

    x == (y == z);
    PRINTX;
    /* z = y = 4 -> x == 4 is TRUE -> x = TRUE = 1 */

    }

```

ဒီ program ကို စာဖတ်သူကိုယ်တိုင် run ကြည့်ပြီးတော့ ဘဏ္ဍာဖြစ်လာလဲ ဆိတာကို လေလာကြည့်ရင် operator တွေ၊ precedence သဘောတရားတွေကို ပိနားလည်လာမှာပါ။ လက်တွေ စမ်းကြည့်ပါ။ ဒါဆိုရင် C အခြေခံ သဘောတရားကို မြည်းစမ်းလို့ရသွားမှာပါပဲ။

DATA INPUT နှင့် OUTPUT

C program အတွက် data တွေကို input / output လုပ်ဖို့အတွက် အဓိက function (၆) မျိုး ကို ကျွန်ုတ်တို့ အသုံးပြုပါတယ်။ အဒေါ်တွေကတော့ getchar, putchar, scanf, printf, gets နဲ့ puts တို့ပါဝဲ။ character တစ်ခုချင်းကို အသွင်းအထုတ် လုပ်ချင်တဲ့အခါမှာ getchar နဲ့ putchar တို့ကို သုံးပါတယ်။ scanf နဲ့ printf function တွေကျတော့ character တစ်ခုချင်းပဲဖြစ်ဖြစ်၊ numerical value တွေပဲဖြစ်ဖြစ်၊ ဒါမှုမဟုတ် string တွေပဲဖြစ်ဖြစ် အကုန်လုံးကို အသွင်းအထုတ် လုပ်ပေးနိုင်ပါတယ်။ ဒါပေမယ့် gets နဲ့ puts function တွေကို အသုံးပြုမယ်ဆိုလို့ရှိရင် string ကိုပဲ input / output လုပ်လို့ရမှာပါ။ ဒီ function တွေနဲ့ ပတ်သက်လို့ ပိုပြည့်ပြည့်စုစုပေါင်းလေး ကျွန်ုတ်တို့ ဆက်လေ့လာကြည့်ရအောင်။

၂.၁ getchar Function

getchar function ဟာ character တစ်လုံးချင်းကို ဖတ်ပေးနိုင်ပါတယ်။ ဒီ function ရဲ့ general form က ဒီလိုပါ။

```
c = getchar();
```

ဒီပုံစံမှာ C ဆိုတာ character variable တစ်ခုကို ဆိုလိုပါတယ်။ keyboard ကနေ ကျွန်ုတ်တို့ ရိုက်ထည့်လိုက်တဲ့ single character တစ်ခုကို getchar function ကနေ ဖတ်ပြီးတော့ ညီမျှခြင်းရဲ့ ဘယ်ဘက်က character variable နဲ့ assign လုပ်ပေးမှာပါ။ ဒီတော့ program အစဉ်းက type declaration မှာ ဒီ variable ကို character လို့ ကြေညာပြီးသားဖြစ်ဖို့ လိုပါတယ်။

EXAMPLE 2.1: Here is a simple C program that uses **getchar** function.

```
main ()  
{  
    char c;  
  
    printf ("Enter a character : ");  
    c = getchar ();  
    printf ("you typed : %c", c);  
}
```

ဒါ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

```
Enter a character : h  
You typed : h
```

ဒါ program မှာဆိုရင်...

- I. ပထမ statement ဟာ type declaration ပါ။ c ကို character-type variable လို့ ကြေားပါတယ်။
- II. ဒုတိယ statement ကတေသာ **Enter a character :** ဆိုတဲ့ prompt ကို computer မှာ display လုပ်ခိုင်းတာပါပဲ။
- III. တုတိယ statement ကတေသာ ကျွန်ုတ်တို့ keyboard ကနေ သွင်းလိုက်တဲ့ character တစ်လုံးကို c နဲ့ assign လုပ်ပေးတာပါ။
- IV. အဒေါ်ဖြေတော့မှ နောက်ဆုံး statement ကနေ စောင့်က keyboard ကသွင်းလိုက်တဲ့ character ကို computer display မှာ ပြန် echo လုပ်ပေးမှာ ဖြစ်ပါတယ်။ ဒါကြောင့်မို့လည်း You typed : h ဆိုတဲ့စာကြောင်း အလိုလိပ်ပေါ်လာတာပေါ့။

getchar () နဲ့အလားတူ **function ()** ခု ရှိပါသေးတယ်။ အဲဒေါတွေက **getche**, **getch** နဲ့ **getc** တို့ပါဝဲ။ **getchar** function ကို သုတေသနခါမှာ **keyboard** ကနေ **data** ရှိက်ထည့်ပြီးလေ့ပေါ် program ဟာ ဆက်အလုပ်မလုပ်ပါဘူး။ **data** ကို **computer buffer area** ထဲမှာ သိမ်းထားပါတယ်။ **ENTER** key နှိပ်တော့မှ computer အလုပ်ဆက်လုပ်မှာပါ။ **getche** function ကျတော့ **buffer** မလိုတော့ဘူး။ **data** ရှိက်ထည့်တာနဲ့ **ENTER** key မနှိပ်ဘဲနဲ့ကို အဖြေပေါ်နေပါပြီ။ **getch** ကျတော့ တစ်မျိုး။ ကျွန်ုတ်တို့ရှိက်ထည့်တဲ့ **data** ကို **display** မှာ မပေါ်ခိုင်းပါဘူး။ **getc** function ကျတော့

stdio.h header file ပါမဲ့ run ပါတယ်။ getc function argument ကိုလည်း stdin လို့ ရေးပေးရပါတယ်။ အဲဒီဥစ္စတွေကို ကျန်တော် နမူနာရေးပြထားပါတယ်။ လေလာကြည်ပါ။

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

    c = getche( );
    printf (" You typed : %c\n", c);
    c = getch( );
    printf (" You typed : %c\n", c);
    c = getc (stdin);
    printf (" You typed : %c\n", c);
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်နမှာပါ။

A You typed : A

You typed : B

C

You typed : C

J-J putchar Function

single character တစ်လုံးကို computer display မှာပေါ်လာအောင် putchar function က လုပ်ပေးနိုင်ပါတယ်။ putchar နဲ့ getchar တို့ဟာ အပိုနဲ့အမလို စုတွေရှိနေကြတဲ့ function တွေပါပဲ။ ဒီ function ရဲ့ general form က ဒီလိပ်ပါ။

putchar (c);

ဒီပုံစံမှာ C ဆိတ် character variable တစ်ခုကို ဆိုလိုပါတယ်။ C ရဲ့ content ကို computer display မှာ ပေါ်လာအောင် putchar က လုပ်ပေးမှာပါ။ EXAMPLE 2.1 ကို ဒီလို ပြန်ရေးလို့ရပါတယ်။ လေ့လာကြည့်ပါ။

EXAMPLE 2.2: Here is a simple C program that uses putchar function.

```
# include < stdio.h >
main ()
{
    char c;
    c = getchar ();
    printf ("You typed : ");
    putchar (c);
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင်...

h
You typed : h

computer မှာ ဒီလိုပေါ်လာမှာပါ။

www.foxitsoftware.com

J-2 **scanf Function**

Input data မျိုးစုကို keyboard ကတစ်ဆင့် computer ထဲကို သွင်းချင်တယ်ဆိုလို့ရှိရင် scanf function ဟာ သုံးလို့အလွယ်ဆုံးပါ။ ဘယ် data အမျိုးအစား:တွေပဲဖြစ်ဖြစ် scanf နဲ့ တွဲသုံးလို့ မရတာမရှိပါဘူး။ ဒါပေမယ့် data-type format ကိုတော့ မှန်အောင်ရေးဖို့ အရေးကြိုးပါတယ်။ scanf function ရဲ့ general form က ဒီလိုပါ။

```
scanf (control string , arg1, arg2, arg3, ... );
```

ဒီ ပုံစံမှာ control string ဆိတာက format အတွက် information ပေးဖို့ပါ။ အဲဒီနောက်က arg1, arg2, arg3 ဆိတာတွေကတော့ data item တစ်ခုချင်းအတွက် argument တွေပါပဲ။ data item တစ်ခုချင်းရဲ့ address ကို ဒီ argument တွေနဲ့ပေးပါတယ်။ address မှန်သိအောင်လို့ ကျွန်တော်တို့က data item တွေရဲ့ နာမည်တွေရှိမှာ ampersand (&) သက်တကို ထည့်ပေးဖို့လိုပါတယ်။ control string အတွက်ကျတော့ conversion character တွေရှိမှာ percent sign (%) ကို ထည့်ရမှာပါ။ scanf function နဲ့တွဲသုံးရမယ့် conversion character တွေကို အောက်မှာ ဖော်ပြပေးထားပါတယ်။ လေ့လာကြည့်ပါ။

Conversion Character

Meaning

c	data item is a single character
d	data item is a decimal integer
e	data item is a floating-point value
f	data item is a floating-point value
g	data item is a floating-point value
h	data item is a short integer
i	data item is a decimal, hexadecimal, or octal integer
o	data item is an octal integer
s	data item is a string
u	data item is an unsigned decimal integer
x	data item is a hexadecimal integer
[. .]	data item is a string which may include whitespace characters

EXAMPLE 2.3: Here is a typical application of scanf function.

```
main ()
{
    char item [20];
    int partno;
    float cost;

    printf ("Enter item name, part number and cost\n");
    scanf ("%s %d %f", item, &partno, &cost);
}
```

ဒီ program ကို run လိုက်မယ်ဆိုရင်...

Enter item name, part number and cost
pulley 12345 9.25

ဒီ program မှာ pulley ဆိုတဲ့ character string ကို array အနေနဲ့ data ဖတ်လိုက်တာပါပဲ။ array name က item ပါ။ item က array ဖြစ်လို့မို့ ampersand (&) သင်္ကာတကို သူ့ရှုံးမှာ ထည့်စရာမလိုတော့ပါဘူး။ array ကိုယ်တိုင်က address ဖြစ်နေလို့ပါ။ item အတွက် conversion character ကိုတော့ %s format နဲ့ ရေးပေးရပါမယ်။ ကိန်းပြည့်ကဏ္ဍး 12345 ကိုတော့ partno နဲ့ assign လုပ်ပေးပါတယ်။ partno အတွက် format က %d ပါ။ floating-point variable ဖြစ်တဲ့ cost ကိုတော့ 9.25 နဲ့ assign လုပ်ထားပါတယ်။ သူ့အတွက် format ကတော့ %f ပါပဲ။ အထက်မှာ ပြထားတဲ့ data သွင်းနည်းအပြင် နောက်ထပ် ပုံစွဲတွေနဲ့လည်း data input လုပ်လို့ရပါသေးတယ်။ အဲဒါတွေ ကတော့ ...

pulley
12345
9.25
or
pulley
12345 9.25
or
pulley 12345
9.25

www.foxitsoftware.com

တကယ်လို့ data item တွေကြားမှာ comma ခံပြီး data ထည့်ချင်တယ်ဆိုလို့ရှိရင် conversion character တွေကြားမှာလည်း comma တွေ ရေးပေးရပါမယ်။ ဒီလိုမျိုးပါ။

`scanf (" %s, %d, %f ", item, &partno, &cost);`

EXAMPLE 2.4: This example illustrates the use of the **scanf** function to enter a string consisting of uppercase letter and blank spaces. The string will be of undetermined length but it will be limited to 79 characters plus the null character that is added to the end.

```

main ()
{
    char    text [80];

    scanf ( "%[ ABCDEFGHIJKLMNOPQRSTUVWXYZ ]", text );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုလိုရှိရင် keyboard ကနေ ကျွန်တော်တိ၊ data တွေ ထည့်ပေးရမှာပါ။ data ဟာ character တွေ တန်းစီထည့်ထားတာ ဖြစ်ရပါမယ်။ blank တွေ ပါလို့ ရပါတယ်။ ဒါပေမယ့် စာလုံးအရေအတွက် (80) ထက်တော့ ကျော်လို့မရပါဘူး။ ဥပမာ ဒါ data ဟာ KYAUK MYAUNG ဆိုပါစို့။ data သွင်းပြီးပြီးဆိုရင် KYAUK MYAUNG ဆိုတဲ့စာလုံးတွေရော၊ KYAUK နဲ့ MYAUNG ကြားက space ရော text ဆိုတဲ့ array ထဲကို အကုန်လုံး ရောက်သွားပါပြီ။ ဘာပြုလို့လဲဆိုတော့ conversion character format မှာ uppercase letter A ကနေ Z အတွင်းက character တွေရော၊ Blank space တွေရော အကုန်လက်ခံမယ်လို့ ဉာဏ်ထားတာဘို့။ ဒါပေမယ့် ကျွန်တော်တိက ကျောက်မြောင်းကို ဒီလို Kyauk Myaung သွားရေးမယ်ဆိုရင် text က uppercase letter K တစ်လုံးပဲလက်ခံမှာပါ။ lowercase letter y ကို တွေ့တာနဲ့၊ data ကုန်သွားပြီလို့ စက်က ယူဆပြီပေါ့။ ဒါ data ဖတ်နည်းကို ကျွန်တော်တိနောက်တစ်မျိုး ပြင်ရေးလို့ရပါတယ်။ %[A - Z] လို့ conversion character ကို ပြင်ရေးလိုက်ရင် စောစောကလိုပဲ data သွင်းလို့ရပါတယ်။ တကယ်လို့ %[abc] လို့ ရေးထားမယ်ဆိုလို့ရှိရင် lowercase letter a, b, c (၃) လုံးကလွှဲရင် text က လက်ခံမှာ မဟုတ်ပါဘူး။ ဒီလို %[A - F T - Z] conversion format ဆိုရင် uppercase letter A to F အတွင်းက စာလုံး (၆) မျိုး၊ blank တစ်မျိုး၊ uppercase letter T to Z အတွင်းက စာလုံး (၇) မျိုး စုစုပေါင်း (၁၃) မျိုးကိုပဲ text က လက်ခံမှာပါ။ တဗြားဟာတွေဝင်လာရင် text က data ထဲကနေ ဖြုတ်ချလိုက်မှာပါပဲ။

scanf နဲ့ data ဖတ်နည်း နောက်ပုံစံတစ်မျိုး ရှိပါသေးတယ်။ အဲဒါကတော့ %[^...] ဒီပုံစံပါ။ ဒီဥစ္စာမှာ square brackets ထဲက circumflex (^) သက်တနောက်မှာရှိတဲ့ argument နဲ့မတူသေးမချင်း computer က data တွေကို ဖတ်ပြီးရင်းဖတ်နေမှာပါ။ ပထမတို့က ဖတ်နည်းနဲ့ယဉ်ရင် ပြောင်းပြန်ပေါ့။

EXAMPLE 2.5 : The following C program illustrates the use of **circumflex** statement.

```

main ()
{
    char text [80];

    scanf ("%[^n]", text);
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် keyboard ကနေ ကျွန်တော်တို့ ရှိက်သမျှ character ဆွက် (၇၉ လုံးထက်တော့ကျော်လို့ မရဘူးနော်) text ဆိုတဲ့ array ထဲမှာ assign လုပ်တော့မှာပါ။ ဘယ်လို character မျိုးကိုပဲထည့်ထည့် computer က လက်ခံမှာပါ။ တစ်ခုပဲ သတ်မှတ်ချက်ရှိတယ်။ ENTER ဘစ်ချက်ပုံတိတာနဲ့ data လက်ခံတာ ရပ်သွားပါလိမ့်မယ်။ ကျွန်တော်တို့တွေ string တစ်ခုကို ဒီလိုသွင်းလို့ ရပါတယ်။

My heart leaps when I behold a rainbow in the sky

Data item တွေကို အလုံးအရေအတွက် ကန့်သတ်ချက် (field width format) နဲ့ ဖတ်ချင်ရင် percent sign (%) နဲ့ conversion character [ကြားမှာ field width ထည့်ပေးလို့ ရပါတယ်။

EXAMPLE 2.6: Here is the skeletal structure of a C program that explains the field width format.

```

main ()
{
    int a, b, c;

    scanf ("%3d %3d %3d", &a, &b, &c );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် integer ကေန်း (၃) လုံးတော့ data အဖြစ် ထည့်ပေးရတော့မယ်။ ကဲ- ကျွန်တော်တို့က ဒီလို data သုံးခုကို ထည့်လိုက်ပြီ ဆိုပါစို့။

1 2 3

a = 1, b = 2, c = 3

ဒါဖြင့်ရင် ဒီလို data ထည့်ရင်ကော ဘာဖြစ်မလဲ။

123 456 789

အဲဒါလည်းပဲ အပေါ်ကလိုပဲ။ data တွေက character (၃)လုံးစီ ရှိနေတာဆိုတော့ blank space မရောက်ခင်က integer value တွေကို a,b,c လို့ ယူဆမှာပါ။

a = 123, b = 456, c = 789

ဒါဖြင့်ရင် data တွေကို နောက်တစ်မျိုး ဒီလိုပုံစံနဲ့ ထည့်ကြည့်းမယ်။ ဘာထူးမလဲကြည့်ရအောင်။

123456789

ဒါလည်း မထူးပါဘူး။ a,b,c ကို အရှင်လိုပဲ assign လုပ်မှာပါပဲ။

a = 123, b = 456, c = 789

နောက်ဆုံးအနေနဲ့ ကျွန်တော်တို့ data တွေ ဒီလိုထည့်ကြည့်မယ်။

123 4 56789

ဒီတစ်ခါတော့ ထူးသွားပြီ။ a,b,c တို့ရဲ့ numerical value တွေက ဒီလိုအနေနဲ့ computer ထဲဝင်သွားမှာပါ။

a = 123, b = 4, c = 567

ကျွန်တော်တိ။ ဖြစ်ချင်တာတစ်မျိုး၊ တကယ်ဖြစ်တာက တစ်ခြား။ အဓိပါယ်တစ်ခြားစီ ဖြစ်သွားနိုင်ပါတယ်။ အခုန်ရင် digit နှစ်လုံးဖြစ်တဲ့ ၈ နဲ့ ၉ ဟာ data item မှာ လုံးဝမပါနိုင်တော့ပါဘူး။

EXAMPLE 2.7: Consider a C program that contains the following statements.

```
main ()
{
    int    i;
    float  x;
    char   c;

    scanf ( "%3d %5f %c", &i, &x, &c );
}
```

ဒါ program ကို run တဲ့အခါ item တွေကို ဒီလိုထည့်မယ်ဆိုပါစို့။

10 256.87 T

computer က ဒီလို assign လုပ်တော့မှာပါ။

i = 10, x = 256.8, c = 'T'

EXAMPLE 2.8: Consider the following skeletal outline of a C program.

```
main ()
{
    char   c1, c2, c3;

    scanf ( "%c%c%c", &c1, &c2, &c3 );
}
```

ကျွန်တော်တိ၊ ဖြစ်ချင်တာတစ်မျိုး၊ တကယ်ဖြစ်တာက တစ်ခြား၊ အမိပါယ်တစ်ခြားစီ ဖြစ်သွားနိုင်ပါတယ်။ အခုန်ရင် digit နှစ်လုံးဖြစ်တဲ့ ၈ နဲ့ ၉ ဟာ data item မှာ လုံးဝမပါနိုင်တော့ပါဘူး။

EXAMPLE 2.7: Consider a C program that contains the following statements.

```
main ()
{
    int    i;
    float x;
    char   c;

    scanf ("%3d %5f %c", &i, &x, &c);
}
```

ဒါ program ကို run တဲ့အခါ item တွေကို ဒီလိုထည့်မယ်ဆိုပါစိုး။

10 256.87 T

computer က ဒီလို assign လုပ်တော့မှာပါ။

i = 10, x = 256.8, c = 'T'

EXAMPLE 2.8: Consider the following skeletal outline of a C program.

```
main ()
{
    char   c1, c2, c3;

    scanf ("%c%c%c", &c1, &c2, &c3);
}
```

program အတွက် input data တွေကို ဒီလိုထည့်လိုက်မယ်ဆိုပါစိုး။

1 a 2.0

ဒါဆိုရင် $i = 1$, $x = 2.0$ လို့ computer က assign လုပ်မှာပါ။ character 'a' ကိုတော့ စက်က အသုသလို ကျော်သွားပါလိမ့်မယ်။ ဒါပေမယ့် input data မှာ ဒီလိုထည့်သွေးရင်တော့ scanf function က character 'a' ကို မတွေ့တာနဲ့ ရပ်သွားမှာပါ။

1 2.0

J.4 printf Function

Output data တွေကို standard output device မှာ ရေးချင်တယ်ဆိုရင် printf ဆိုတဲ့ function ကို သုံးရပါတယ်။ printf ဖန်ရှင်ရဲ့ general form ကတော့ ဒီလိုပါ။

printf (control string, arg1, arg2, ...);

control string ဆိုတာက formatting information လိုပေးတဲ့ string အမျိုးအစား တစ်ခုပါ။ arg1, arg2, ... ဆိုတာတွေကတော့ ကျွန်တော်တို့ output လုပ်ချင်တဲ့ data item တွေကို ရည်ညွှန်းတာပါ။ အဒါတွေက constant, single variable, array name စသည်ဖြင့် အကုန်ဖြစ်လို့ ရပါတယ်။ scanf function နဲ့ ကွာတာတစ်ခုကတော့ data item တွေရှေ့မှာ memory address ကို ဖော်ပြတဲ့ (&) operator ကို အသုံးပြုစရာ မလိုပါဘူး။

EXAMPLE 2.10: Here is a simple C program that makes use of the printf function.

```

#include <math.h>

main ()
{
    float i = 2, j = 3;
    printf ( "%f %f %f %f", i,j,i+j, sqrt (i+j));
}

```

ဒါ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်လာမှာပါ။

2.000000 3.000000 5.000000 2.236068

EXAMPLE 2.11: The following C program generates the same floating-point output in two different forms.

```

main ()
{
    double x = 5000, y = 0.005;
    printf ( "%f %f %f %f\n", x, y, x * y, x / y );
    printf ( "%e %e %e %e", x, y, x * y, x / y );
}

```

ဒါ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်လာမှာပါ။

**5000.000000 0.002500 12.500000 2000000.000000
5.000000e+003 2.500000e-003 1.250000e+001 2.000000e+006**

EXAMPLE 2.12: Here is a C program that will read a line of text and then write back as it is entered.

```
main ()  
{  
    char text [80];  
  
    printf ("Enter a sentence\n");  
    scanf ("%[A - Z ]", text );  
    printf ("%s", text );  
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် ...

Enter a sentence

EVERYTHING COMES TO HE WHO HUSTLES WHILE HE WAITS.

EVERYTHING COMES TO HE WHO HUSTLES WHILE HE WAITS.

computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Precision Features

printf function ကိုသုံးပြီးတော့ minimum field width တစ်ခုကို ဘယ်လို specify လုပ်ရတယ် ဆိတာ ပြောပြုပါပြီ။ ဒီတစ်ခါတော့ floating-point value မှာ အသေမတွေ ဘယ်လိုဖြတ်ရမလဲ ဆိတာရယ်။ string ဆွေမှာ character တွေ ဘယ်လိုဖြတ်ယူရမလဲ ဆိတာတွေကို ရှင်းပြပါမယ်။

EXAMPLE 2.13: Here is a program that illustrates the use of the **precision** feature with floating-point numbers.

```

main ()
{
    float      x = 123.456;

    printf ("%7f %7.3f %7.1f\n", x, x, x );
    printf ("%12e %12.5e %12.3e", x, x, x );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

123.456000	123.456	123.5
1.234560e+002	1.23456e++002	1.235e+002

ဒီညွှန်ကိုပဲ minimum field width မဖော်ပြားနဲ့ precision specification နဲ့ ရေးပြလို့
ရပါသေးတယ်။

```

main ()
{
    float      x = 123.456;

    printf ("%f%0.3f %0.1f\n", x, x, x );
    printf ("%e %0.5e %0.3e", x, x, x );
}

```

J.၅ gets and puts Functions

gets နှင့် puts function တွေဟာဆိုရင် computer နဲ့ standard input/output device တွေကြားမှာ string တွေကို transfer လုပ်ဖို့အတွက် အသုံးကျပါတယ်။ string ကို one dimensional character array အနေနဲ့ စက်ကလက်ခံမှာပါ။ string ထဲမှာ whitespace character (blank) တွေပါလို့ရပါတယ်။ gets ဖန်ရှင်ကို သုံးတဲ့အခါမှာ data တွေထည့်ဖြီးတာနဲ့ ENTER key ကိုနိပ်မှ data ကုန်တယ်လို့ စက်ကသိမှာပါ။ gets နဲ့ puts ဖန်ရှင်တွေကို ဘယ်လိုသုံးရလဲဆိုတာကို ရှင်းပြထားပါတယ်။ လေ့လာကြည့်ပါ။

EX 2.12 that reads a line of text into the computer and then writes it back in the original form on the screen.

```
main ()
{
    char    text [80];

    printf ("Enter a sentence\n");
    gets (text);
    puts (text);
}
```

ဤ program ကို runလိုက်မယ့်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter a sentence

Nobody gets to live life backward. Look ahead.

Nobody gets to live life backward. Look ahead.

EXAMPLE 2.15: Here is a C program which illustrates the use of relational and conditional operators.

```
# define PRINT(int)    printf (# int " = %d\n", int)

main ()
{
    int x=1, y=1, z=1;

    x += y += z;
    /* z=1, y=1 -> y = y+z = 1+1 = 2 -> x = x+y = 1+2 = 3 */
    PRINT ( x < y ? y : x );
    /* (x<y) = (3 < 2) = FALSE = 0 -> PRINT (x) = 3 */
}
```

$x < y ? y : x = 3$

EXAMPLE 2.16 : What does the following C program print?

```
# define PRINT(int)    print(#int " = %d\n", int)

main ( )
{
    int      x = 3,      y = 2,      z = 1;

    PRINT ( x < y ? x++ : y++ );
    /* (x<y) = (3<2) = FALSE = 0 -> PRINT(y++) = PRINT(y) = 2
       y = y + 1 = 3 */
    PRINT (x);
    /* x = 3 -> PRINT(x) = 3 */
    PRINT(y);
    /* y = 3 -> PRINT(y) = 3 */
    PRINT ( z += x < y ? x++ : y++ );
    /* (x<y) = (3<3) = FALSE -> z=z+y =1+3 =4 -> y++=3+1=4 */
}
```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

```
x < y ? x++ : y++ = 2
x = 3
y = 3
z += x < y ? x++ : y++ = 4
```

EXAMPLE 2.17: This program uses the library function **hypot** (double, double) to calculate hypotenuse of right-angled triangle.

```

# include <math.h>

main ( )
{
    double side1, side2, hypotenuse;

    printf ("Enter side1, side2\n");
    scanf ("%lf, %lf", &side1, &side2);
    hypotenuse = hypot (side1, side2);
    printf ("Hypotenuse = %lf", hypotenuse);
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင်

```

Enter side1, side2
3,4
Hypotenuse = 5.000000

```

computer မှာ ဒီလို ပေါ်လာမှာပါ။

EXAMPLE 2.18: Averaging Student Exam Scores

Write a C porgram that reads a student's name and three exam scores, and then calculate an average score.

```

main ( )
{
    char name[20];
    int score1, score2, score3;
    float avg;

    printf ("Enter your name : ");
    scanf ("%[^\n]", name);
    printf ("Score1 ? ");
    scanf ("%d", &score1);

```

```

        printf ("Score2 ? ");
        scanf ("%d", &score2);
        printf ("Score3 ? ");
        scanf ("%d", &score3);
        avg = (score1 + score2 + score3) / 3.0;
        printf ("\n\nName      : % - s\n\n", name);
        printf ("score1 : %-d\n", score1);
        printf ("Score2: %-d\n", score2);
        printf ("Score3: %-d\n\n", score3);
        printf ("Average   : %-5.1f\n", avg);
    }
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှပါ။

Enter your name : AUNG MYINT

Score1 ? 85

Score2 ? 90

Score3 ? 95

Name : AUNG MYINT

Score1 : 85

Score2 : 90

Score3 : 95

Average : 90.0

ဒဲ program မှာ conversion character ထဲက အနှစ်သက်တယာ output ကို ဘယ်ဘက်ဆုံးကပ်ပေးပါလို့ ဆိုလိုပါတယ်။

CONTROL STATEMENTS

၃.၁ if Statement

if statement ဆိတာ conditional statement တစ်ခုပါပဲ။ if ကို သုံးတဲ့အခါမှာ if နောက်က argument ဟာ မှန်လား မှားလားဆိတာကို အရင်စိစစ်ရပါတယ်။ မှန်တယ်ဆိုရင် argument နောက်က statement ကို computer က execute လုပ်ပါလိမ့်မယ်။ ပြီးတော့ရင် next statement ကို ဆင်းသွားမှာပါ။ တကယ်လို့, argument က မှားတယ်ဆိုရင်တော့ သူ့နောက်က အလုပ်ကို မလုပ်တော့ပါဘူး။ next statement ကို တန်းကူးသွားမှာပါပဲ။ if statement ရဲ့ general form က ဒီလိုရှိပါတယ်။

if (argument) statement ;

ဒီပုံစံမှာ if နောက်က argument ဟာ TRUE ဆိုရင် statement ကို computer က execute လုပ်မှာပါ။ FALSE ဆိုရင် မလုပ်ပါဘူး။ if နောက်က statement ထွေဟာ တစ်ခုထက်ပိုလာမယ်ဆိုလို့, ရှိရင် brace ထွေနဲ့ ပိတ်ပေါ့ပို့, မမေ့ပါနဲ့။ if statement အသုံးပြုနည်းကို example program တစ်ခုနဲ့လေ့လာကြည့်ရင် ပို့ရှင်းသွားမှာပါ။ လေ့လာကြည့်ရအောင်။

EXAMPLE 3.1: if Statement

This program illustrates the use of a if statement.

```
main ()  
{  
    int num;  
    printf ("Enter a number\n");  
    scanf ("%d", &num);  
    if (num < 0) printf ("The number is -ive\n");  
    if (num > 0) printf ("The number is +ive\n");  
    if (num == 0) printf ("The number is zero\n");  
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှပါ။

```
Enter a number  
12  
The number is +ive  
Enter a number  
-25  
The number is -ive  
Enter a number  
0  
The number is zero
```

www.toxitsoftware.com

ဒါ program အလုပ်လုပ်သွားပုံကို ကျွန်တော်အကျဉ်းချုပ်းပြီး ရှင်းပြုမယ်။

- III num ဟာ integer variable တစ်ခလို့ အရင်ကြညာပါတယ်။ Enter a number ဆိတဲ့ prompt တစ်ခကို computer မှာပေါ်ခိုးပြီးတော့ cursor ကို နောက်တစ်ကြောင်း ကူးခိုင်းပါတယ်။
- IV keyboard ကနေ 12 ကဏ္နာကို ရိုက်ထည့်ပါတယ်။ ဒါဆိုရင် num = 12 ဖြစ်သွားပြီပေါ့။
- V ပထမဆုံးစတွေ့တဲ့ if statement မှာ num က သုညထက် ငယ်လားလို့မေးပါတယ်။ ဘယ်ငယ်မလဲ။ ဒီတော့ မေးခွန်းက မှားတဲ့အတွက် computer ဟာ next line ကို ဆင်းသွားပါပြီ။
- VI ဒုတိယ if statement ကျတော့ num ဟာ သုညထက် ကြီးလားလို့ မေးပါတယ်။ 12 က သုညထက် ကြီးတာပဲ၊ မှန်တာပေါ့။ ဒါကြောင့်မို့ argument နောက်က အလုပ်ကို လုပ်ပါပြီ။ The number is +ive ဆိတဲ့စာကြောင်းဟာ computer display မှာ ပေါ်လာပြီလေ။ ပြီးတော့ရင် next line ကို computer ဆင်းသွားမှာပါ။

- နောက်ဆုံး: if statement ကတေသာ num ကို သုည့်၏ ညီလားလို့ မေးပြန်ပါတယ်။ မည်ပါဘူး။ FALSE ဖြစ်တဲ့အတွက် computer က next line ကို ဆင်းသွားတဲ့အခါမှာ closing brace နဲ့ တွေ့ပါတယ်။ ဒါကြောင့်မို့ computer program ဒီမှာတင် ပြီးသွားပါပြီ။
- num တန်ဖိုးတွေကို keyboard ကနေ -25၊ 0 လို့ တစ်လျဉ်းစီ ရိုက်ထည့်ပေးရင် computer ဘယ်လို့ အလုပ်လုပ်မလဲဆိုတာကို စာဖတ်သူကိုယ်တိုင် လက်တွေ့လေ့လာကြည့်ချင်ပါတယ်။

၃.၂ go Statement

goto ဆိုတာ program တစ်ခုရဲ့ ပုံမှန်လမ်းကြောင်းကနေ တြေားတစ်နေရာကို လွှတ်လွှတ်လပ်လပ် ရှုနော်သွားချင်တယ်ဆိုရင် အသုံးပြုရမယ့် statement ပါပဲ။ goto ရဲ့ general form က ဒီလိုပါ။

goto LABEL;

LABEL ဆိုတာ ကွန်ပျော်ဘာ သွားချင်တဲ့နေရာဒေသကို အမှတ်သညာပြုထားတဲ့ identifier အမည်ပါ။ Target statement မှာ ဒါ identifier name ကို colon (:) နဲ့ တွဲပြီး ရေးပေးရပါမယ်။

LABEL : TARGET STATEMENT;

EXAMPLE 3.2: Sum of Odd Numbers

Write a C program to compute the sum of odd numbers from START through LAST.

```
main ()
{
    int start, last, odds, sum = 0;

    printf ("Enter start and last\n");
    scanf ("%d,%d", &start, &last);
    if (start % 2 != 0)    odds = start;
    if (start % 2 == 0)    odds = start+1;
    AGAIN :
```

```

if (odds <= last) {
    sum += odds;
    odds += 2;
    goto AGAIN;
}
printf ("Sum of odd numbers %d through %d = %d\n",
       start, last, sum);
}

```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

```

Enter start and last
2, 9
Sum of odd numbers 2 through 9 = 24

Enter start and last
1, 100
Sum of odd numbers 1 through 100 = 2500

```

ဒီ program ဟာ start ဆိုတဲ့ စဉ်းကဏ္န်းတစ်ခုကနောက်နဲ့ last ဆိုတဲ့ နောက်ဆုံးကဏ္န်း တစ်ခုအတွင်းက မဂဏန်းတွေအားလုံးကို ပေါင်းပေးမှာပါ။ start နဲ့ last တို့ဟာ ဖြစ်ချင်ရာ ကဏ္န်းတွေ ဖြစ်နိုင်ပါတယ်။ ဒီ program ဘယ်လို အလုပ်လုပ်သွားတယ်ဆိုတာကို ကျွန်ုတော်တို့ လေ့လာကည့်ရအောင်။

- ၁။ program စစ်ချင်း: start, last, odds, sum တို့ကို integer variable တွေလို့ ကြေညာ လိုက်ပါတယ်။ sum ထဲမှာ မဂဏန်းတွေ အားလုံးပေါင်းလောက် အဖြစ်ထုတ်မှာပါ။ လောလောဆယ်တော့ program က ဘာမှ မတွက်ရသေးဘူးဆိုတော့ sum = 0 ပေါ့။
- ၂။ Enter start and last ဆိုတဲ့ prompt တစ်ခုကို computer မှာ ပေါ်ခိုင်းပါတယ်။ ပြီးတော့ရင် cursor က next line ကို ဆင်းသွားမှာပါ။
- ၃။ keyboard ကနေ 2, 9 လို့ ရိုက်ထည့်လိုက်မယ်ဆိုပါစို့။ ဒါဆိုရင် start = 2, last = 9 ဖြစ်သွားပါပြီ။
- ၄။ အရေးကြီးဆုံးက စဉ်းကဏ္န်း: start ဟာ စုံလား မလားဆိုတာကို စိစစ်ရပါမယ်။ စစ်နည်းကတော့ start ကို 2 နဲ့ စားလို့ မပြတ်ဘူးဆိုရင် start ဟာ မဂဏန်းပေါ့။ ပြတ်တယ်ဆိုရင် စုံဖြစ်ပါတယ်။ start % 2 ဟာ သူည့်နဲ့ မည်ဘူးလားလို့ မေးပါရတယ်။ သူည့်နဲ့ ညီတာပေါ့။ ဒီတော့ start % 2 != 0 ဆိုတဲ့ မေးခွန်းက FALSE ဖြစ်သွားပါပြီ။ if (argument) နောက်က statement ကို execute မလုပ်တော့ပါဘူး။ next line ကို ဆက်ဆင်းသွားမှာပါ။

- ၃ ဒုတိယ if က argument ကျတော့ TRUE ဖြစ်သွားပါပြီ။ ဒါကြောင့် odds = start + 1 = 2 + 1 = 3 လို့ တွက်ယူလိုက်ပါတယ်။ ဆိုလိုတဲ့အခိုဗာယ်က စီးကဏန်းဟာ 2 ဖြစ်ပေမယ့် 3 ကိုပဲ စီးမကဏန်းလို့ ယူလိုက်တာပါ။
- ၄ odds ဟာ last နဲ့ယူဉ်လို့ ငယ်တယ် သို့မဟုတ် ညီတယ်ဆိုရင် if နောက်က statement တွေကို execute လုပ်ပါလိမ့်မယ်။ အခုလည်းပဲ odds = 3 က last = 9 ထက် ငယ်တာမို့ argument က TRUE ပေါ့။
- ၅ if block ထဲကို computer ဝင်လာပါပြီ။ sum += odds ဆိုတာကို sum = sum + odds = 0 + 3 = 3 လို့ computer က တွက်လိုက်ပါတယ်။ sum ထဲကို ပထမဆုံး မကဏန်းတစ်လုံး ထည့်လိုက်တာပါ။
- ၆ 3 ပြီးရင် ဒုတိယ မကဏန်းဟာ 5 မဟုတ်လား။ အဲဒီဥစ္စာ ရဖို့အတွက် odds = odds + 2 = 3 + 2 = 5 ဆိုတဲ့ ညီမျှခြင်းနဲ့ တွက်ယူပါတယ်။ ဒါဆိုရင် ဒုတိယ မကဏန်း ရသွားပြီးလေ။
- ၇ goto နဲ့ AGAIN label ရှိတဲ့နေရာကို ပြန်သွားခိုင်းပါတယ်။ step (6) ကို ပြန်ရောက်သွားပါပြီ။ step (6) ကနေ step (9) ကို ထပ်ခါတလဲ ပတ်နေမှာပါ။ တစ်ချိန်မှာ odds = 11 ဖြစ်သွားပါလိမ့်မယ်။ ဒါဆိုရင် odds = 11 က last = 9 ထက် ကြီးသွားတာကြောင့် argument က TRUE ဖြစ်သွားပါပြီ။ if block ထဲကို ထပ်မဝင်တော့ဘဲနဲ့ အောက်ကိုဆက်ဆင်းသွားမှာပါ။
- ၈ ပြီးတော့ရင် Sum of odd numbers 2 through 9 = 24 လို့ အဖြေရှင်းထုတ်ပါလိမ့်မယ်။ program ဒီမှာတင် ပြီးသွားပါပြီ။
- ၉ start = 1, last = 100 လို့ data သွင်းခဲ့ရင် အဖြေရအောင် computer ဘယ်လိုတွက်သွားလဲဆိုတာကို စာဖတ်သူကိုယ်တိုင် လက်တွေ့ trace လုပ်ကြည့်ပါ။

EXAMPLE 3.3:

Reversing a Number

Write a C program to read an integer number and print out the same number with digits reversed.

```
main ( )
{
    long      num, rev, q, r;
    printf ("Enter a number\n");
    scanf ("%d", &num);
    q = num/10;
    rev = num % 10;
    START : if ( q != 0 ) {
        num = q;
        q = num/10;
        rev = rev * 10 + r;
        goto START;
    }
    printf ("Reversed number is %d", rev);
}
```

```

    r = num % 10;
    rev = rev * 10+r;
    goto START;
}
printf ("The number reversed = %ld\n", rev);
}

```

ဒီ program ကို rum လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter a number

123456789

The number reversed = 987654321

- ဒီ program ရဲ့ ရည်ရွယ်ချက်ကတော့ ကိန်းပြည့်ကဏ္န်းတစ်လုံးကို input data အနေနဲ့
ထည့်ပေးလိုက်မယ်ဆိုရင် ဒီဂဏ်န်းကို ပြောင်းပြန်အစိုးးစဉ်နဲ့ ဖြစ်အောင် computer က လုပ်ပေးမှာပါ။
- ၁။ num, rev, q, r တို့ကို long integer ဂဏ်န်းတွေလို့ declare လုပ်ပါတယ်။
- ၂။ Enter a number ဆိုတဲ့ စာကြောင်းဟာ computer display မှာ ပေါ်လာပါပြီ။
- ၃။ keyboard ဘန် data ကို 123456789 ဆိုတဲ့ ဂဏ်န်းအရှည်ကြီးတစ်ခု ထည့်ပေးလိုက်ပါမယ်။
- ၄။ ဒါဆိုရင် num = 123456789 ဖြစ်သွားမှာပါ။
- ၅။ q = num / 10 = 123456789 / 10 = 12345678 ရမှာပါ။ rev = num % 10 = 9 ပေါ့။
- ၆။ if (q != 0) ဆိုတဲ့ မေးခွန်းက TRUE ပါ။ q က သုညမှ မဟုတ်တာကိုး။ ဒီတော့ if block ထဲကို computer ဝင်လာပါပြီ။
- ၇။ num = q = 12345678
 q = num / 10 = 12345678 / 10 = 1234567
 r = num % 10 = 8
 rev = rev * 10 + r = 9 * 10 + 8 = 98
- ၈။ step (5) ကို ပြန်သွားပါ။
- ၉။ num = q = 1234567
 q = num / 10 = 1234567 / 10 = 123456
 r = num % 10 = 7
 rev = rev * 10 + r = 98 * 10 + 7 = 987
- ၁၀။ step (5) ကို ပြန်သွားပါ။
- ၁၁။ တစ်ချိန်မှာ q = 0 ဖြစ်သွားပါလိမ့်မယ်။ ဒီခါကျရင် if block ထဲက ထွက်လာပြီးတော့ The number reversed = 987654321 လို့ အဖြေရှိက်ပြမှာပါပဲ။ ဒါဆိုရင် program ပြီးသွားပါပြီ။

EXAMPLE 3.4:

Payroll Program

Write a C program that calculates the payroll for four different workers giving overtime if the hours worked are greater than 40 hours.

```
mian ( )
{
    int      class;
    float    wage, hours, pay;

    START : printf ("Enter class (any of 1 to 4)\n");
    scanf ("%d", &class);
    if (class < 1 || class > 4) goto START;
    if (class == 1)    wage = 12;
    if (class == 2)    wage = 15;
    if (class == 3)    wage = 20;
    if (class == 4)    wage = 25;
    printf ("Enter hours worked\n");
    scanf ("%f", &hours);
    if (hours <= 40)  pay = wage*hours;
    if (hours > 40)   pay = wage*40+1.5*wage*(hours-40);
    printf ("Pay = %5.1f\n", pay);
}
```

ဒဲ program ကို run လိုက်မယ်ဆုံးရင်...

Enter class (any of 1 to 4)

5

Enter class (any of 1 to 4)

3

Enter hours worked

50

Pay = 1100.0

computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။ program လမ်းကြောင်းကို စာဖတ်သူကိုယ်တိုင် trace လုပ်ကြည့်ပါ။ မခက်ပါဘူး။

if statement နဲ့ program တစ်ခု ရေးတဲ့အခါမှာ ရွေးစရာလမ်းကြောင်း (၂) ခုရှိလာရင် if-else statement ကို သုံးရမှာပါ။ if နောက်က argument ဟာ မှန်တယ်ဆိုရင် STATEMENT 1 ကို execute လုပ်ပြီးတော့ if-else အပြင်ကို ထွက်သွားမှာပါ။ တကယ်လို့ argument ဟာ FALSE ဖြစ်နေရင် STATEMENT 2 ကို execute လုပ်မှာပါ။ ဒါ statement တွေဟာ တစ်ကြောင်းထက်ပိုလာရင် brace နဲ့ ပိတ်ပေးရပါမယ်။ if-else ရဲ့ general form က ဒီလိုပါ။

```
if ( argument )
    STATEMENT 1;
else
    STATEMENT 2;
```

EXAMPLE 3.5:**Lowercase to Uppercase Conversion.**

Write a C program that reads a lowercase character, converts it to uppercase and then writes out the uppercase equivalent.

```
main ( )
{
    char lower, upper;

    printf ("Enter a lowercase character\n");
    scanf ("%c", &lower);
    if (lower >= 'a' && lower <= 'z')
        upper = 'A' + lower - 'a';
    else
        upper = lower;
    printf ("%c\n", upper);
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter a lowercase character

```
e
E
```

ဒါ program ဟာ lowercase character တစ်ခုကို uppercase character ဖြစ်အောင် ပြောင်းပေးတဲ့ program ပါ။ ဒါ program ဘယ်လို အလုပ်လုပ်သွားတယ်ဆိုတာ လေ့လာကြည့်ရအောင်။

၁။ lower & upper ဆိုတဲ့ variable (j) ခုကို character တွေလို့ declare လုပ်ပါတယ်။

၂။ prompt တစ်ခုကို computer မှာ ပေါ်ခိုင်းပါတယ်။

၃။ input data ကို e လို့ ထည့်လိုက်ပါတယ်။ lower = 'e' ဖြစ်သွားပါပြီ။

၄။ if (lower >= 'a' && lower <= 'z') ဆိုတဲ့ မေးခွန်းမှာ lower က 'a' ထက် ကြီးပါတယ်။ 'z' ထက် ငယ်ပါတယ်။ ဒီတော့ argument က TRUE ပေါ့။ upper = 'A' + lower - 'a' ဖြစ်သွားပါပြီ။ upper ရဲ့ ASCII တန်ဖိုးဟာ upper = 65 + 101 - 97 = 69 လို့ တွက်လို့ ရုပါတယ်။ ASCII တန်ဖိုး 69 ဟာ uppercase letter E ကို ဆိုလိုတာပါ။

၅။ ဒါကြောင့်မို့လည်း printf ("%c\n", upper); လုပ်လိုက်တဲ့အခါမှာ အဖြောက် E ဖွောက်လာတာပေါ့။

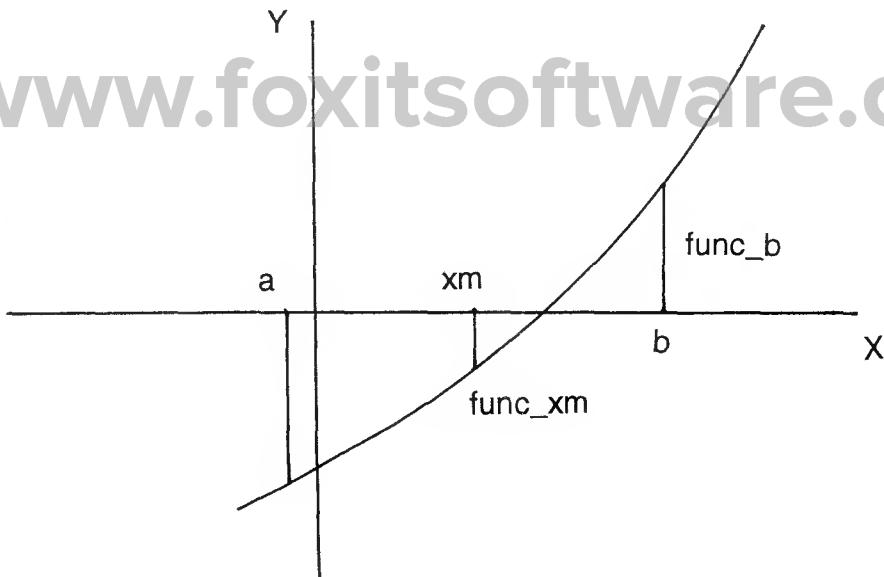
၆။ တကယ်လို့ keyboard ကနေ E အကြီးကို ရိုက်ထည့်ပေးမယ်ဆိုလို့ရှိရင် computer ဘယ်လို့ အလုပ်လုပ်မလဲ။ အဲဒါ စာဖတ်သူကိုယ်တိုင် လက်တွေ့ trace လုပ်ကြည့်ပါ။ အဖြောက်လာမှာပါ။

EXAMPLE 3.6:

Bisection method

Write a C program to find the real roots of the equation

$2x^2 - 5x - 12 = 0$ using the bisection method.



```

#include <math.h>

main ( )
{
    double a, b, xm, func_xm, func_b;

    printf ("Enter a and b\n");
    scanf ("%lf %lf", &a, &b);
    AGAIN:
    if (fabs (a - b) >= 1e-12) {
        xm = (a+b)/2;
        func_xm = 2*xm*xm - 5*xm - 12;
        func_b = 2*b*b - 5*b - 12;
        if (func_xm*func_b < 0)
            a = xm;
        else b = xm;
        goto AGAIN;
    }
    printf ("Root = %5.2f\n", xm);
}

```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter a and b

1, 50

Root = 4.00

Enter a and b

-10, 3

Root = -1.50

- ဒီ program ရဲ့ ရည်ရွယ်ချက်ကတေသူ equation တစ်ခုက မသိကိန်းတစ်ခုရဲ့ တန်ဖိုးကို ရှာချင်တာပါ။ ဒီ program ဘယ်လို အလုပ်လုပ်သွားလဲဆိုတာကို ကျွန်ုတော်တို့ လေ့လာကြည့်ရအောင်။
- အသုံးပြုမယ့် variable တွေကို double လို့ ကြညာလိုက်ပါမယ်။
- a နဲ့ b တို့ရဲ့ တန်ဖိုးတွေကို 1, 50 လို့ ထည့်ပေးလိုက်မယ်ဆိုပါစို့။ a = 1, b = 50 ဖြစ်သွားပြီပေါ့။

၃။ အ - b ဟာ ၁၀ ထက်ကြေးလူးလှုံး မေးပါတယ်။ ကြေးပါတယ်။ ဒေါကက argument
 ဟာ TRUE ပေါ့။ if block ထဲကို computer ဝင်လာဖြီလေ။
 ၄။ $xm = (a+b) / 2$ ဆိတဲ့ equation နဲ့ a နဲ့ b ကြားက midpoint ကို ရှာပါတယ်။
 ၅။ $func_xm = f(xm) = 2(xm^2) - 5(xm) - 12$
 ၆။ $func_b = f(b) = 2(b^2) - 5(b) - 12$
 ၇။ ပုံမှာကြည့်မယ်ဆိုရင် $func_xm$ ရဲ့ value ဟာ -ive ပါ။ x-axis အောက်မှာရောက်နေတာကိုး။
 $func_b$ ကတော့ +ive ပါ။ ဒီတော့ ဒီနှစ်ခုကို မြောက်မယ်ဆိုလို့ရှိရင် အဖြေက -ive ပဲရမှာပါ။ -ive
 ဖြစ်တဲ့အတွက် သုညထက်ငယ်တယ်လေ။ ဒါဆိုရင် if ($func_xm * func_b < 0$) ဆိတဲ့မေးခွန်းက
 မှန်တာပေါ့။ TRUE ဖြစ်သွားတဲ့အတွက် argument နောက်က အလုပ်ကို computer က
 လုပ်မှာပါ။ $a = xm$ လို့ ပြောင်းသွားပါပြီ။ ဆိုလိုတဲ့ အဓိပ္ပာယ်က Root ဟာ $a \neq b$ ကြားမှာလို့
 ပြောတာထက် b နဲ့ xm ကြားမှာလို့ပြောတာက ပိုမျိန်ဘူးလား။ ပထမ a ရဲ့ value ကို cancel
 လုပ်ပြီးတော့ a တန်ဖိုးကို xm ထဲမှာ ထည့်လိုက်တာ ဒီသဘောပါပဲ။ b ရဲ့ value ကတော့ အခု
 step မှာ မပြောင်းလဲပါဘူး။ နိဂုံမှုလအတိုင်းပဲ ဖြစ်ပါတယ်။
 ၈။ step (3) ကနေ step (7) ကို ပြန်ပတ်ခိုင်းပါပြီ။ အဲဒီလိုပတ်နေရင်း တစ်ချိန်မှာ $a - b$ ဟာ 10^{-12}
 ထက် ငယ်သွားပါလိမ့်မယ်။ ဒါဆိုရင် if-else statement က အပြီးထွက်လာပြီးတော့ အဖြေရှိက်မှာပါပဲ။
 $Root = 4.00$ လို့ အဖြေရသွားပါပြီ။ ဒီအဖြေဟာ အမှန်အကန်ပါပဲ။ ဒီ program မှာ if-else
 အသုံးပြုထားတာကို လေ့လာကြည့်ပါ။

၃.၄ if-else if-else Statement

ဒီ statement ဟာလည်း if statement အောက်မှာပဲ အကျိုးဝင်ပါတယ်။ ကျွန်ုတ်တို့ တွေ့
 အော့စောင့်က လေ့လာခဲ့တဲ့ if-else statement ဟာ ရွေးစရာလမ်း (j) ခုပဲရှိတယ်လို့ ပြောခဲ့ပါပြီ။
 argument မှန်ရင် သူ့နောက်က အလုပ်ကို လုပ်မယ်။ မှားရင်တော့ else နောက်က အလုပ်ပဲလုပ်မှာပါ။
 ဒါကြောင့်မို့လည်း သွားစရာလမ်း (j) ခုပဲရှိတာလို့ပြောတာပါ။ တကယ်လို့ သွားစရာလမ်းက သုံးလေးမျိုးဖြစ်လာရင်
 ဘယ်လိုလုပ်ပါမလဲ။ အဲဒါကို ဖြေရှင်းနိုင်တာကတော့ တတိယအမျိုးအစား if-else if-else statement ပါပဲ။
 ၂.၄.၁ general form က ဒီလိုပါ။

```

if ( argument1 )
  STATEMENT1;
else if ( argument2 )
  STATEMENT2;
  
```

```

else if ( argument3 )
    STATEMENT3;
else if
    . . .
else
    STATEMENT(N);

```

ဒီပုံစံမှာ argument1 က မှန်တယ်ဆိုရင် STATEMENT1 ကို execute လုပ်ပြီးတော့ program က အောက်ဆုံးကို ခုန်ဆင်းသွားမှာပါ။ တကယ်လို့ မမှန်ဘူးဆိုရင် argument2 နဲ့ ထပ်စစ်ပါလိမ့်မယ်။ argument2 က TRUE ဆိုရင် STATEMENT2 ကို execute လုပ်ပြီးတော့ if block အောက်ဆုံးကို ခုန်ဆင်းသွားပါ။ ဒီနည်းအတိုင်း TRUE မတော့မချင်း ဆက်စစ်နေမှာပါပဲ။ စစ်သမျှ argument တွေ အားလုံးတစ်ခုမှ မမှန်ဘူးဆိုရင် else နောက်က STATEMENT(N) ကိုပဲ execute လုပ်ပါလိမ့်မယ်။ ဒီ if block နဲ့ ပတ်သက်တဲ့ program လေးတစ်ပုံဒါ ကျွန်တော်တို့ လေ့လာကြည့်ရအောင်။

EXAMPLE 3.7: Leap Test

Write a C program that reads in a number representing a year and then determines whether it is a leap year or not.

```

#define TRUE 1
#define FALSE 0

main ( )
{
    int year, leap;

    printf ("Enter a year (****)\n");
    scanf ("%d", &year);
    if (year % 4 != 0)
        leap = FALSE;
    else if (year % 100 != 0)
        leap = TRUE;
    else if (year % 400 == 0)
        leap = TRUE;
    else
        leap = FALSE;

    if (leap) printf ("%d is a LEAP year\n", year);
    else printf ("%d is NOT a LEAP year\n", year);
}

```

Enter a year (**)**

1995

1995 is NOT a LEAP year

Enter a year (**)**

1996

1996 is a LEAP year

Enter a year (**)**

2000

2000 is a LEAP year

Enter a year (**)**

5000

5000 is NOT a LEAP year

- ဒီ program ဟာဆိုရင် သဲက္ခရာနှစ်တစ်နှစ်က ဝါထပ်လား၊ မထပ်ဘူးလားဆိုတာကို စိစစ်ပေးမှာပါ။ အဲဒီသဲက္ခရာနှစ်ကို 4 နဲ့ အရင်စားကြည့်ပါ။ မပြတ်ဘူးဆိုရင် ဒီနှစ်ဟာ သေချာပေါက် ဝါမထပ်ပါဘူး။ 4 နဲ့ စားလို့ ပြတ်ရင်တော့ ဝါထပ်ချင်ထပ်နိုင်ပါတယ်။ သေတော့ မသေချာသေးဘူး။ ထပ်စစ်ရပါဉိုးမယ်။ 100 နဲ့ ဆက်စားကြည့်ပါ။ စားလို့ မပြတ်ဘူးဆိုရင် အဲဒီနှစ်ဟာ သေချာပေါက် ဝါထပ်ပါတယ်။ ပြတ်တယ်ဆိုရင် အဲဒီ နှစ်ဟာ ဝါထပ်လား၊ မထပ်လား မသေချာပြန်တော့ဘူး။ 400 နဲ့ ထပ်စားကြည့်ပါ။ ပြတ်တယ်ဆိုရင် ဝါထပ်ပါတယ်။ မပြတ်ရင်တော့ ဝါမထပ်တော့ပါဘူး။ အဲဒီနည်းနဲ့ ဒီ program ကို ရေးထားတာပါ။ လေ့လာကြည့်ပါ။
- ၁။ မနှစ်က 1995 ခုနှစ်ပါ။ ဝါမထပ်ပါဘူး။ ဘာပြုလို့လဲဆိုတော့ 1995 ကို 4 နဲ့စားရင် မပြတ်လို့ပါပဲ။ ဒီနှစ်က 1996 ခုနှစ်ပါ။ 1996 ကို 4 နဲ့စားလို့ ပြတ်ပေးမယ့် 100 နဲ့စားတော့ မပြတ်တာကြောင့် ဒီနှစ်ဟာ ဝါထပ်ပါတယ်။
- ၂။ သဲက္ခရာ 2000 ဟာ ဝါထပ်ပါတယ်။ ဘာပြုလို့လဲဆိုတော့ပထမ 4 နဲ့ စားတာပြတ်တယ်။ ဒီတော့ 100 နဲ့ ထပ်စားတယ်။ ပြတ်တယ်။ 400 နဲ့ ထပ်စားတယ်။ ပြတ်သေးတာပဲ။ ဒါကြောင့်မို့ သဲက္ခရာ 2000 ဟာ ဝါထပ်တာပါ။
- ၃။ သဲက္ခရာ 5000 ကျတော့ ဝါမထပ်တော့ဘူး။ ဘာပြုလို့လဲဆိုတော့ ပထမ 4 နဲ့စားတာ ပြတ်တော့ 100 နဲ့ ထပ်စားတယ်။ ပြတ်တယ်။ ဒါပေမယ့် 400 နဲ့စားတော့ မပြတ်တော့ဘူး။ ဒါကြောင့်မို့ သဲက္ခရာ 5000 ဟာ ဝါမထပ်ဘူးလို့ ပြောတာပါ။
- ဒီစစ်နည်းတွေကို computer နားလည်အောင်လို့ if-else if-else statement တွေနဲ့ အသုံးပြုထားပါတယ်။ စာဖတ်သူကိုယ်တိုင် program တစ်ကြောင်းချင်း trace လုပ်ကြည့်ပါ။ အဖြေပေါ်လာမှာပါ။

looping operation တစ်ခုကို ထပ်ခါတလဲ လုပ်ရတော့မယ်ဆိုလို့ရှိရင် while control statement ကို အသုံးပြုရမှာပါ။ while ရဲ့ general form က ဒီလိုရှိပါတယ်။

while (argument) STATEMENT ;

ဒီဥစ္စာရဲ့ အဓိပ္ပာယ်ကတော့ while နောက်က ဘွင်းထဲက argument ဟာ TRUE ဖြစ်နေသမှု ကာလပတ်လုံး STATEMENT ကို computer က ထပ်ခါတလဲ လုပ်ရတော့မှာပါ။ statement တွေဟာ multiple statement တွေဆိုရင် braces နဲ့ ပိတ်ပေးပို့ မမောပါနဲ့။

EXAMPLE 3.8: Consecutive Integer Quantities

Write a C program to display the consecutive digits 0, 1, 2, 3, ..., 9 with all digits in one line separated by one space between them.

```
main ( )
{
    int digit = 0;

    while (digit <= 9) {
        printf ("%d ", digit);
        ++digit;
    }
}
```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

0 1 2 3 4 5 6 7 8 9

ဒီ program ကို ကျွန်ုတ်တို့ လေ့လာကြည့်မယ်ဆိုရင်...

၁။ digit ကို integer variable လို့ declare လုပ်ပြီးတော့ value ကို သည်လို့ initialize လုပ်ထားပါတယ်။

- while (digit <= 9) အမိပ္ပာယ်ကတော့ ဒါ argument ဟာ TRUE ဆိုရင် while loop ထဲကို computer ဝင်သွားမှာပါ။ စစချင်းမှာ digit = 0 ဆိုတော့ digit <= 9 ဆိုတာ TRUE ဖြစ်နေပါပြီ။
 - ဒီတော့ while loop ထဲကို computer ဝင်လာပါတယ်။ printf statement ကြောင့် computer မှာ သူညေပေါ်လာပါပြီ။ သူညေနောက်မှာ blank တစ်ခုလည်းပါသေးတယ်။ ဘာပြုလို့လဲဆိုတော့ %d နောက်က blank တစ်ခု ထည့်ထားတာကို။ ပြီးတော့ရင် digit ကို တစ်တိုးပါတယ်။ closing brace ကိုတွေ့တဲ့အခါမှာ while loop အစကို ဖြန်သွားပြီပေါ့။
 - အခုတစ်ခါမှာ digit = 1 ဖြစ်နေပါတယ်။ printf statement ကြောင့် သူညာနဲ့ blank အေးမှာ 1 ကို ဆက်ရှိက်မှာပါ။ ပြီးတော့ရင် digit ကို တစ်ခုတိုးပါတယ်။ ဒီနည်းအတိုင်း digit ဟာ 9 ထက် မကျော်မချင်း while loop ကို ပတ်နေမှာပါပဲ။ တစ်ချိန်မှာ digit = 10 ဖြစ်သွားတဲ့အခါကျော်ရင် while loop ကနေ computer အပြီးထွက်လာပြီးတော့ program လည်း ရပ်သွားပါလိမ့်မယ်။
 - ဒါ program ကို ပိုပြီးကျစ်ကျစ်လျစ်လျစ်ဖြစ်အောင်လို့ ဒီလိုပြောင်းရေးလို့ရပါတယ်။ increment operator (++) သုံးသွားပုံကို လေ့လာကြည့်ပါ။

```
main ( )
{
    int      digit = 0;
    while (digit <= 9)
        printf ("%d ", digit++);
}
```

EXAMPLE 3.9: PI series

Write a C program to evaluate the value of PI using the series.

$$\pi / 4 = 1 - 1/3 + 1/5 - 1/7 + 1/9 - 1/11 + 1/13 - \dots$$

```
main ( )
{
    unsigned int n;
    double i = 1, pi = 0;
```

```

while (i <= n) {
    pi += 1/i - 1/(i+2);
    i += 4;
}
printf ("PI = %lf", 4 * pi);
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter n : 65400

PI = 3.141562

ဒါ π series က slow converging series ဖြစ်တာကြောင့် term ကို နည်းနည်းလေး
ပေါင်းခိုင်းတာနဲ့တော့ အဖြေမှန်ရမှာ မဟုတ်ပါဘူး။ စုစုတွဲ term ပေါင်း 65400 အထိ ပေါင်းတာတောင်
အဖြေမှ မမှန်သေးတာ။ term သုံးသန်း (3 millions) ထိ ပေါင်းကြည့်ပါ။

```

main ( )
{
    double n, i = 1, pi = 0;

    printf ("Enter n : ");
    scanf ("%lf", &n);
    while (i <= n) {
        pi += 1/i - 1/(i+2);
        i += 4;
    }
    printf ("PI = %lf", 4*pi);
}

```

Enter n : 3000000

PI = 3.141592

ဒါဆိုအဖြေမှန်နဲ့နဲ့ ရဘွားပါပြီ။ ဒါ program ဘယ်လို အလုပ်လုပ်ဘွားလဲဆိုတာကို စာဖတ်သူကိုယ်တိုင်
trace လိုက်လုပ်ကြည့်ရင် while loop အကြောင်းကို ပိုနားလည်ဘွားမှာပါ။ လေ့လာကြည့်ပါ။

EXAMPLE 3.10: Lowercase to Uppercase Text Conversion
Write a C program that will read a line of lowercase text character-by-character and convert to uppercase using the while statement.

```
# define EOL '\n'

main ( )
{
    char letter[80];
    int i, count;

    i = count = 0;
    letter [count] = getchar ();
    while (letter[count] != EOL)
        ++count;
    letter[count] = getchar ();
}

while (i < count)
{
    putchar (toupper (letter[i]));
    ++i;
}
```

ဒဲ program ကို runလိုက်မယ်ဆိုရင်...

Welcome to C Programming
WELCOME TO C PROGRAMMING

computer မှာ ဒီလိုပေါ်လာမှာပါ။ ဒီဥစ္စာကိုပဲ ကျစ်ကျစ်လျစ်လျစ်ဖြစ်အောင် ဒီလိုပြောင်းရေးလိုပါတယ်။

```

#define EOL '\n'

main( )
{
    char letter[80];
    int i, count;

    i = count = 0;
    letter[count] = getchar();
    while (letter[count] != EOL)
        letter[++count] = getchar();
    while (i < count)
        putchar(toupper(letter[i++]));
}

```

၃။ program ကို လေ့လာကြည့်မယ်ဆိုရင်...

- ၁။ EOL ကို ENTER key နှင့်တူတယ်လို့ define လုပ်ပါတယ်။
- ၂။ letter ကို character အစုံး (80) ပါတဲ့ array လို့ declare လုပ်ပါတယ်။
- ၃။ i = count = 0 လို့ initialize လုပ်ပါတယ်။
- ၄။ keyboard ကနေ သွင်းပေးလိုက်တဲ့ character တွေထဲက ပထမဆုံး character ကို getchar function နဲ့ဖတ်ပြီးတော့ letter [count] = letter [0] နဲ့ တူတယ်လို့ assign လုပ်ပါတယ်။ အခုံ program မှာ letter [0] = 'W' ပါ။
- ၅။ ပထမ while loop ရဲ့ အဓိပ္ပာယ်က letter[count] ဟာ EOL နဲ့ မတူဘူးဆိုရင် while loop ထဲကို computer ဝင်လာမှာပါ။ အခုလည်းပဲ letter [count] = letter [0] = 'W' က ENTER key နှင့်တာနဲ့ မတူတာပဲ။ ဒီတော့ computer က while loop ထဲ ဝင်လာပါပြီ။ keyboard ကနေသွင်းတဲ့ character တွေထဲက ဒုတိယ character ကို letter [++count] = letter [1] နဲ့ တူတယ်လို့ assign လုပ်မှာပါ။ ဒီတော့ letter[1] = 'e' ဖြစ်သွားပါပြီ။ ဒီနည်းအတိုင်း while loop ကို ပတ်နေရင်း တစ်ချိန်မှာ ကျွန်ုင်တော်တို့က ENTER key နှင့်လိုက်တာနဲ့ while loop ကို computer မပတ်တော့ဘူး။ ရပ်သွားပါပြီ။ count ထဲမှာ keyboard ကနေ သွင်းပေးထားတဲ့ character စုစုပေါင်းအရေအတွက်တော့ ရောက်နေပြီဖြစ်ပါတယ်။
- ၆။ နောက်ဆုံး while loop ကတော့ letter array ထဲက character တစ်လုံးစိတို့ uppercase ဖြစ်အောင် ပြောင်းပြီးတော့ display မှာ ရိုက်ပြပါလိမ့်မယ်။ ဒါဆိုရင် program ပြီးသွားပါပြီ။

EXAMPLE 3.11: Write a C program to print the corresponding Celsius to Fahrenheit temperature table.

```
main ( )
{
    float lower, upper, step, cel, fah;

    printf ("Enter lower limit of temperature table\n");
    scanf ("%f", &lower);
    printf ("Enter upper limit\n");
    scanf ("%f", &upper);
    printf ("Enter step size\n");
    scanf ("%f", &step);
    printf ("\n CELSIUS FAHRENHEIT\n\n");
    cel = lower;
    while (cel <= upper) {
        fah = 1.8*cel + 32;
        printf ("10.1f %8.1f\n", cel, fah);
        cel += step;
    }
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter lower limit of temperature table

0

Enter upper limit

100

Enter step size

10

CELSIUS	FAHRENHEIT
0.0	32.0
10.0	50.0
20.0	68.0
30.0	86.0

40.0	104.0
50.0	122.0
60.0	140.0
70.0	158.0
80.0	176.0
90.0	194.0
100.0	212.0

ဒါ program ကို define function တွေ သုံးပြီးတော့ နောက်တစ်မျိုး ပြင်ရေးလို့ရပါတယ်။
လေ့လာကြည့်ပါ။

```
# include <stdio.h>

#define PRINT_HEADING puts(" CELSIUS      FAHRENHEIT\n")
#define PRINT_TABLE(a, b, c) PR(a, b, c)
#define PR(a, b, c) printf ("%10.1" #a      "%10.1" #a "\n", b, c)
#define LOWER    0
#define UPPER   100
#define STEP    10

main ( )
{
    float cel, fah;
    cel = LOWER;
    PRINT_HEADING;
    while ( cel <= UPPER ) {
        fah = 1.8*cel + 32;
        PRINT_TABLE (f, cel, fah);
        cel += STEP;
    }
}
```

include <stdio.h> # define function တွက် aung.h နာမည်၏ C:\TC\INCLUDE directory အောက်မှာ save လုပ်လိုက်ပါ။ ဒါဆိုရင် တတိယပုံစံနဲ့ ပြင်ရေးလို့ရပါတယ်။ ဒါ program ကို run လိုက်မယ်ဆိုလို့ရှိရင် EXAMPLE 3.11 က အဖြော်မှာပါ။ ကျွန်တော်တို့ဘာသာ create လုပ်တဲ့ header file ကို include လုပ်တဲ့အခါမှာ # include "aung.h" လို့ရေးရပါတယ်။

```
# include "aung.h"

main ( )
{
    float cel, fah;

    cel = LOWER;
    PRINT_HEADING;
    while (cel <= UPPER) {
        fah = 1.8*cel + 32;
        PRINT_TABLE (f, cel, fah);
        cel += STEP;
    }
}
```

EXAMPLE 3.12: Write a C program that reads a sequence of positive real numbers and computes their average. A negative number should be entered to signal the end of the input data.

```
main ()
{
    int      k = 0;
    float   num, sum = 0;

    clrscr ( );
    printf ("Enter num : ");
    scanf ("%f", &num);
    while (num > 0)  {
        ++k;
```

```

        sum += num;
        printf ("Enter num : ");
        scanf ("%f", &num);
    }
    if (k == 0)
        printf ("No positive number entered\n");
    else
        printf ("\nAverage = %f\n", sum/k);
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။ clrscr() ဆိတာ clear the screen လို့ ဆိုလိုတာပါ။ program တစ်ခါ run တိုင် screen ပေါက ရေးပြီးသားတွေကို ဖျက်ပစ်မှာပါ။

```

Enter num : -5
No positive number entered

Enter num : 5
Enter num : 6
Enter num : 7
Enter num : 8
Enter num : -5

Average = 6.500000

```

www.kitsoftware.com

EXAMPLE 3.13: Natural Number e

Write a C program that will estimate the natural number e.
The number e is given by the relationship:

$$e = \sum_{i=0}^{\infty} \frac{1}{i!}$$

where $i!$ means i factorial.

$i! = i(i-1)(i-2)(i-3) \dots (3)(2)(1)$
and
 $0! = 1$

```
main ()
{
    int    i = 2;
    float   term = 1, sum = 2;

    while ( i <= 25 ) {
        term /= i;
        sum += term;
        ++i;
    }
    printf ( "Natural Number e = %f", sum );
}
```

ဒဲ program ကို run လိုက်မယ်ဆုံး၏ computer မှာ ခြေလိုပေါ်လာမှာပါ။

Natural Number $e = 2.718282$

www.foxitsoftware.com

- EXAMPLE 3.14:** Write a C program to calculate the annual interest on an amount p for a period of t years with
- 11% annual interest rate for a period less than or equal to 1 year
 - 10% annual interest rate for a period more than 1 year but less than or equal to 3 years
 - 9% annual interest rate for a period more than 3 years but less than or equal to 5 years
 - 5% annual interest rate for a period more than 5 years

```

main ( )
{
    int    i = 0, t;
    float  p, amount, interest, rate;

AGAIN: .
clrscr ( );
printf ("Enter amount and period\n");
scanf ("%f,%d", &p, &t);
if (t < 0) goto AGAIN;
if (t == 0 || t == 1)      rate = 0.11;
else if (t == 2 || t == 3) rate = 0.10;
else if (t == 4 || t == 5) rate = 0.09;
else rate = 0.05;
amount = p;
while (i < t) {
    amount += amount*rate;
    ++i;
}
interest = amount - p;
printf ("\nAmount = %5.2f\n", amount);
printf ("Years     = %2d\n", t);
printf ("Interest   = %5.2f", interest);
}

```

www.foxitsoftware.com

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter amount and period

25000, 5

Amount = 38465.60

Years = 5

Interest = 13465.60

ဒီ program ကို လေ့လာကြည့်မယ်ဆိုရင် ...

- ၁။ data type တွေကို အရင် declare လုပ်ပါတယ်။ ပြီးတော့ အချိန်အတွက် counter 'i' ကိုလည်း သူညလို့, initialize လုပ်ပါတယ်။
- ၂။ screen ကို clear လုပ်ပါတယ်။
- ၃။ Enter amount and period လို့, prompt ပေါ်လာတဲ့အခါမှာ 25000, 5 ကို keyboard ကနေ ကျွန်ုတ်တော်တို့, ရိုက်ထည့်ပေးပါတယ်။ ဒါဆိုရင် $p = 25000$, $t = 5$ ဖြစ်သွားပါပြီ။
- ၄။ တကယ်လို့, t ကို မှားပြီး negative value ထည့်ပေးမိရင် data အသစ်ပြန်တောင်းခိုင်းပါတယ်။ ဒါကြောင့်မို့, step (2) မှာ AGAIN ဆိတ် label ကို ထည့်ရေးထားတာပါ။
- ၅။ if-else if-else statement နဲ့, t ကို စိစစ်ပါတယ်။ else if ($t == 4$ || $t == 5$) ဆိတ် expression ဟာ TRUE ဖြစ်တောင်းခဲ့တယ်။ rate = 0.09 ကို ယူလိုက်ပါတယ်။ ပြီးတော့ရင် if block အပြင်ကို ထွက်လာပါပြီ။
- ၆။ amount = $p = 25000$ လို့, assign လုပ်ပါတယ်။
- ၇။ while ($i < t$) ဆိတ် အဓိပ္ပာယ်က t ဟာ 5 ထက် ငယ်နေသူမျှ ကာလပတ်လို့: looping ကို ပတ်မယ်လို့ ဆိုလိုပါတယ်။ ဒီ looping ထဲမှာ computer က ပတ်နေရင်း ဒီအဖြေဖွေ ထွက်နေပါတယ်။
- | | |
|---------|--------------------|
| $t = 1$ | amount = 27250 |
| $t = 2$ | amount = 29702.5 |
| $t = 3$ | amount = 32375.725 |
| $t = 4$ | amount = 35289.54 |
| $t = 5$ | amount = 38465.60 |
- ၈။ interest = $38465.60 - 25000 = 13465.60$
- ၉။ amount = 38465.60 ; $t = 5$; interest = 13465.60 လို့, အဖြေရှိက်ထုတ်ပြီးတော့ program ပြီးသွားပါပြီ။ ဒီ program ဟာ if-else if-else နဲ့, while statement တို့ကို တွေပြီးသုံးပြထားတာဖြစ်ပါတယ်။

၃.၆ do-while Statement

while statement ကို သုံးပြီးတော့ looping တစ်ခကိုပတ်တဲ့ အခါမှာ ရှေ့ဆက်ပြီး ပတ်သင့် မပတ်သင့်ကို looping စစချင်းမှာပဲ စစ်ဆေးခဲ့တာကို တွေ့မှာပါ။ တစ်ခါတစ်ရဲ ဒီလို့ test မျိုးကို looping ရဲ့ နောက်ဆုံးကျမှု စစ်ဆေးတာမျိုးလည်း ရှိပါတယ်။ အဲဒါဆိုရင် do-while statement ကို အသုံးပြုရမှာပါ။ do-while ရဲ့ general form က ဒီလိုရှိပါတယ်။

```

do
    STATEMENT;
while ( expr );

```

ဒီပုံစံမှာဆိုရင် စစချင်းမှာ looping ထဲကို အနောက်အယုက်၊ အတားအဆီးမရှိ do ကိုကျဉ်းမြှုံး body of loop ထဲကို ကျွန်ုတ်တော်တိ၊ ဝင်လို့ရပါတယ်။ body ထဲက statement တွေ အားလုံးကို execute လုပ်ပြီးသွားရင်တော့ while နဲ့ တွေ့နှုပါ။ ဒီခါကျရင် expr ဆိုတဲ့ argument တစ်ခုက မေးခွန်းထုတ်ပါ လိမ့်မယ်။ expr က TRUE ဆိုရင် d0-while loop ကို ခုတိယအကြိမ် ကျွန်ုတ်တို့ကို ပတ်ခွင့်ပြုမှုပါ။ FALSE ဆိုရင်တော့ ပတ်ခွင့်ပြုမှု မဟုတ်ပါဘူး။ while အောက်က statement တွေဆိုကို တန်းဆင်းဆင်းရမှုပါ။ ဒါဟာ do-while control statement ရဲ့ သဘောတရားပါပဲ။

EXAMPLE 3.15: Factorial of a Number

Write a C program to compute the factorial of a positive integer n. The factorial of n represented by n! is defined as

$$n! = n (n - 1) (n - 2) (n - 3) \dots \dots (3) (2) (1)$$

```

main ( )
{
    long      i = 2, n, nfac = 1;

    clrscr ( );
    printf ( "Enter n : " );
    scanf ( "%ld", &n );
    do
        nfac *= i++;
    while ( i <= n );
    printf ( "Factorial of %ld = %ld", n, nfac );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှုပါ။

Enter n : 5

Factorial of 5 = 120

ဒဲ program ရဲ ရည်ရွယ်ချက်ကတေသာ computer ကို ကဏ္န်းတစ်လုံးပြောလိုက်ရင် အဲဒီကဏ္န်းရဲ factorial ကို ရှာပေးမှာပါ။

- i = 2, nfac = 2 လို့ initialize လုပ်ပါတယ်။ i က counter ပါ။ n! ရဲ အဖြောက်ဆုံးမှာ nfac ထဲရောက်အောင် တွက်မှာပါ။ n ကတေသာ ကျွန်တော်တို့ factorial ရှာမယ့် ကဏ္န်းပါပဲ။
- Enter n : လို့ computer က prompt လုပ်တဲ့အခါကျရင် keyboard ကနေ ကျွန်တော်တို့ 5 ရိုက်ထည့်လိုက်မယ်ဆိုပါစို့။ ဒါဆိုရင် n = 5 ဖြစ်သွားပါပြီ။
- စစ်ချင်း do-while loop ထဲဝင်လာပြီးတော့ nfac = nfac * i = 1 * 2 = 2 လို့ computer က တွက်ပါတယ်။ ပြီးတော့ရင် i ကို တစ်တိုးပါတယ်။ while နဲ့ တွေ့တဲ့အခါမှာ i = 6 ဖြစ်လာတဲ့အခါကျရင် do-while loop ကို computer မပတ်တော့ပါဘူး။
- next line ကို.ဆင်းလာပြီးတော့ factorial of 5 = 120 လို့ အဖြောက်လိုက်ပါပြီ။ ဒါဟာ do-while statement ကို အသုံးပြုနည်းပါပဲ။ program ပြီးသွားပါပြီ။

EXAMPLE 3.16: Fibonacci Numbers

Write a C program to compute and display the first n Fibonacci numbers. The Fibonacci numbers form an interesting sequence in which each number is equal to the sum of the previous two numbers.

```

main ()
{
    int i, n, f1, f2, f;

    clrscr();
    i = f1 = f2 = 1;
    printf ("How many Fibonacci numbers ? ");
    scanf ("%d", &n);
    do {
        if (i < 3) f = 1;
        else f = f1 + f2;
        f1 = f2;
        f2 = f;
        i++;
    }
}
```

```

f2 = f1;
f1 = f;
printf ("i = %3d      Fibo = %5d\n", i++, f);
} while (i <= n);
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်လာမှာပါ။

How many Fibonacci numbers ? 15

i = 1	Fibo = 1
i = 2	Fibo = 1
i = 3	Fibo = 2
i = 4	Fibo = 3
i = 5	Fibo = 5
i = 6	Fibo = 8
i = 7	Fibo = 13
i = 8	Fibo = 21
i = 9	Fibo = 34
i = 10	Fibo = 55
i = 11	Fibo = 89
i = 12	Fibo = 144
i = 13	Fibo = 233
i = 14	Fibo = 377
i = 15	Fibo = 610

- ၁။ program စစချင်းမှာ How many Fibonacci numbers? လို့ prompt ပေါ်လာပြီးတော့ n အတွက် data တောင်းပါတယ်။ ကျွန်ုတ်တို့က 15 လို့ ရိုက်ထည့်ပေးလိုက်မယ်ဆိုရင် n = 15 ဖြစ်သွားပါပြီ။ i = f1 = f2 = 1 လို့ initialize လုပ်ထားပါတယ်။
- ၂။ do-while loop ထဲကို ဝင်လာပါပြီ။ ဒါ loop ထဲမှာ i က 3 ထက် ငယ်နေသူမျှ f ရဲ့ တန်ဖိုးတွေဟာ 1 တွေချည်းပဲလို့ သတ်မှတ်ထားပါတယ်။ i = 3 ဖြစ်လာရင်တော့ f = f1 + f2 = 1 + 1 = 2 လို့ တွက်ယူမှာပါ။ ပြီးတော့ရင် f2 = f1 = 1, f1 = f = 2 လို့ နေရာချင်း လဲပါတယ်။ i နဲ့ f ကို ရိုက်ထဲပါပြီးတော့ i ကို တစ်ခုတို့ပါတယ်။ i = 2 ဖြစ်သွားပါပြီ။
- ၃။ while (i <= n) ဆိုတော့ i = 15 မဖြစ်သေးမချင်း do-while loop ကို computer က ပတ်လိုက်၊ တွက်လိုက် လုပ်နေမှာပါ။ i = 16 ဖြစ်သွားတဲ့အချိန်မှာ program ပြီးသွားပါပြီ။ စာဖတ်သူကိုယ်တိုင် program တစ်ကြောင်းချင်း trace လိုက်လုပ်ကြည့်ရင် ပိုရင်းသွားမှာပါ။

Write a C program to print a list of the prime factors of a number.

```
main ( )
{
    int divisor = 2, adder = 1, n;

    clrscr ( );
    printf ( "Enter n\n" );
    scanf ( "%d", &n );
    printf ( "Prime factors of %d\n", n );
    do {
        if ( n % divisor == 0 ) {
            n /= divisor;
            printf ( "%d", divisor );
        }
        else {
            divisor += adder;
            adder = 2;
        }
    } while ( n/2 > 1 );
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter n

105

Prime factors of 105

3 5 7

ဒါ program ရဲ့ ရည်ရွယ်ချက်ကတော့ ကဏ္န်းတစ်လုံးပေးထားရင် အဲဒီကဏ္န်းရဲ့ ဆွဲကိန်းကဏ္န်းတွေကို ရှာပေးမှာပါ။

- ၁။ program စစ်ချင်: Enter n လို့ prompt ပေါ်လာတဲ့အချိန်မှာ 105 လို့ ကျွန်တော်တို့ ရိုက်ပေးလိုက်ရင် n = 105 ဖြစ်သွားပါပြီ။ divisor = 2, adder = 1 လို့ initialize လုပ်ထားပါတယ်။
- ၂။ do loop ကို စပတ်ပါပြီ။ စစ်ချင်: n % divisorဟာ သုညလားလို့မေးပါတယ်။ 105 ကို 2 နဲ့စားရင် အကြွောင်းသုည မရပါဘူး။ ဒီတော့ else နောက်က အလုပ်ဂိုပဲ computer လှပ်မှာပါ။ divisor ကို တစ်တိုးပေးသလို adder = 2 လို့ assign လုပ်ပါတယ်။

၃။ မြန်မာစာမျက်နှာ၏ အမြဲးအမွှေ့သာ ၁၁၂ ဘာ ၁ ထက် ကြံးရင် ပေးကု ဖြန့်သွားပါတယ်။
 n/2 = 105/2 = 52 ဆိုတော့ do ကို ပြန်သွားပါဖြီ။
 ၄။ if (n % divisor == 0) ဆိုတာ 105 ကို 3 နဲ့စားလို့ ပြတ်တယ်ဆိုရင် n = n/divisor = 105/3 = 35 လို့ ပြင်ရပါမယ်။ အဲဒီ (3) ကိုလည်း print လုပ်ရပါမယ်။
 ၅။ n/2=35/2=17 က ၁ ထက်ကြီးတာမို့ do ကို ပြန်သွားပါဖြီ။ 35 ကို 3 နဲ့ စားလို့ မပြတ်ဘူးဆိုရင် else နောက်က divisor = divisor + adder = 3 + 2 = 5 လို့ပြင်ပြီးတော့ while ဆိုကို ရောက်လာပါတယ်။ n/2 = 17 က ၁ ထက်ကြီးတာမို့ do ကို ပြန်သွားပါတယ်။
 ၆။ 35 ကို 5 နဲ့ စားလို့ပြတ်ရင် n=n / divisor = 35/5 = 7 လို့ ပြင်ရပါမယ်။ အဲဒီ (5) ကိုလည်း print လုပ်ရပါမယ်။ ဒီနည်းအတိုင်း ဆက်လုပ်သွားမယ်ဆိုရင် 105 ရဲ့ ဆွဲကိန်းကဏ္န်းတွေဖြစ်တဲ့ ၃, ၅, ၇ တို့ကို အဖြေထဲတဲ့ရပါဖြီ။

EXAMPLE 3.18: Linear Regression

Write a C program that reads in pairs of data values x and y , calculates, and prints the values of the regression coefficients a and b. Test your program using the following sample test data.

<u>x</u>	<u>y</u>
2.10	2.90
6.22	3.83
7.17	5.98
10.52	5.71
13.68	7.74

```
main ( )
{
    int      i = 1, n = 5;
    float   x, y, sumx, sumy, sumxx, sumxy, a, b;

    clrscr ( );
    sumx = sumy = sumxx = sumxy = 0;
    printf ("Enter x and y\n");
}
```

```

do {
    scanf ( "%f,%f", &x, &y );
    sumx += x;
    sumy += y;
    sumxx += x * x;
    sumxy += x * y;
    ++i;
} while ( i <= n );
b = ( n * sumxy - sumx*sumy ) / ( n* sumxx - sumx* sumx );
a = ( sumy - b * sumx ) /n;
if ( b < 0 )
    printf ( "\ny = %f %f", a, b );
else
    printf ( "\ny = %f + %f x", a, b );
}

```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်လာမှာပါ။

Enter x and y

2.1, 2.9

6.22, 3.83

7.17, 5.98

10.52, 5.71

13.68, 7.74

y = 2.038392 + 0.402319 X

ဒီ program ဟာဆိုရင် ပေးထားတဲ့ x, y data (5) အဲကို ကိုယ်စားဖြေတဲ့ error အနည်းဆုံး မျဉ်းဖြောင့်တစ်ကြောင်းရဲ့ equation ကို ရှာပေးမှာပါ။ ဒီဥစ္စာကို linear regression လုပ်တယ်လို့ အော်ပါတယ်။ တကယ်လို့များ data point တွေဟာ မျဉ်းဖြောင့်ပေါ်မှာ အားလုံးကျနေမယ်ဆိုလို့ရှိရင် equation က ကွက်တိကို ဖုန်းနေမှာပါ။ ဒီမျဉ်းဖြောင့်ရဲ့ equation ကို $y = a + bx$ လို့ သတ်မှတ်မယ် ဆိုပါတယ့်။ ဒါဆိုရင် a နဲ့ b တို့ကို

$$b = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

$$a = \frac{\sum y - b(\sum x)}{n}$$

ဆိတ် ညီမျှခြင်းတွေနဲ့ တွက်ယူလိုရပါတယ်။ ဒီညီမျှခြင်းတွေမှာ ၂ ဟာ x, y data pair အရေအတွက်ပါ။ အခုံ program မှာ n = 5 လို့ အသေယူထားပါတယ်။ program ကို general ဖြစ်စေခဲင်ရင် ၂ ကို အရှင်ထားပေးပါ။ ဒီ program အလုပ်လုပ်သွားပုံကို ကျွန်တော်ရှင်းမပြတော့ပါဘူး။ စာဖတ်သူကိုယ်တိုင် လေ့လာဖို့ အလုပ်ချုပ်ထားခဲ့ပါမယ်။ လေ့လာကြည့်ပါ။

EXAMPLE 3.19: Taylor Series

Write a C program to estimate the value of sine of x (in radians) using the Taylor series expansion

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \dots$$

```
# include <math.h>
# define PI 3.141593

main ( )
{
    int i = 2;
    double deg, x, term, sum;

    clrscr ( );
    printf ( "Enter deg : " );
    scanf ( "%lf", &deg );
    x = deg * PI / 180;
    term = sum = x;
    do {
        term *= -x*x / ((2*i-2)*(2*i-1));
        sum += term;
        ++i;
    } while (fabs (term) > 1e - 12);
    printf ( "Sine of %4.2lf estimated = %lf\n", deg, sum );
    printf ( "Sine of %4.2lf in exact = %lf", deg, sin ( x ) );
}
```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

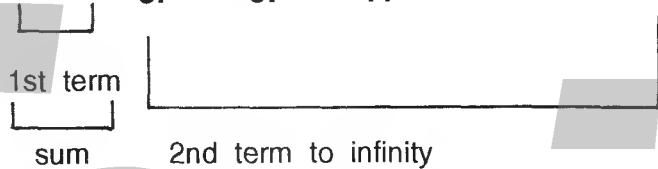
Enter deg : 60

Sine of 60.00 estimated = 0.866025

Sine of 60.00 in exact = 0.866025

ဒီ program ကိုရေးဖို့အတွက် သချာနည်းနဲ့ ဘယ်လိုစဉ်းစားလဲဆိုတော့ ...

$$\sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \dots$$



i = 2 ဖြစ်တဲ့အခါမှာ

$$\frac{\text{2nd term}}{\text{1st term}} = -\frac{x^3}{3!} \cdot \frac{1}{x} = -\frac{x^2}{2.3} = -\frac{x^2}{(2i-2)(2i-1)}$$

i = 3 ဖြစ်မယ်ဆိုရင် ...

$$\frac{\text{3rd term}}{\text{2nd term}} = \frac{x^5}{5!} \cdot \frac{3!}{x^3} = \frac{x^2}{4.5} = \frac{x^2}{(2i-2)(2i-1)}$$

ဒါကြောင့်မို့ ith term = (i-1)th term * $\frac{-x^2}{(2i-2)(2i-1)}$ ဆိုတဲ့ ပုံသေနည်းနဲ့

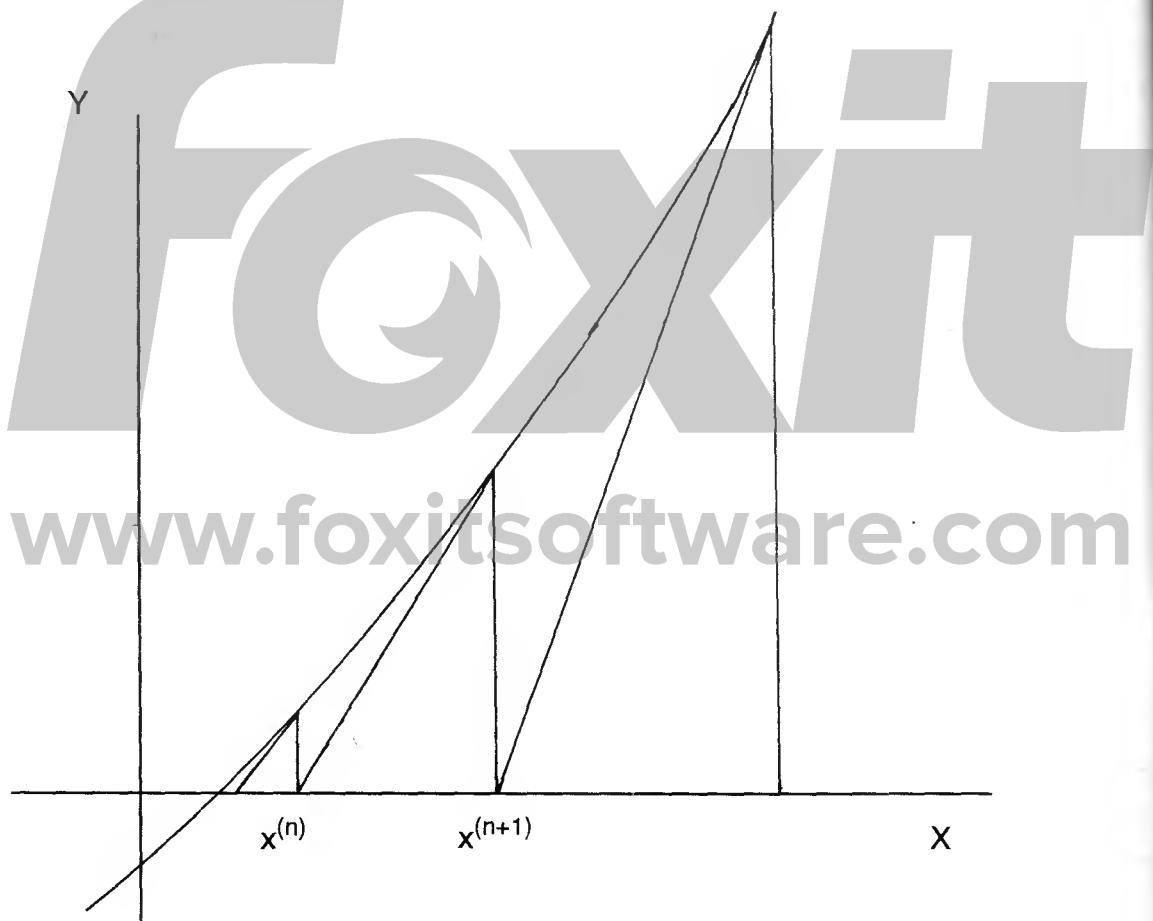
တွက်ယူသွားလို့ရပါတယ်။ term တစ်ခုချင်းကို ဘယ်လောက်ယောက်လဲလို့ စစ်တဲ့အခါမှာ term ကို absolute value ယူပြီးတော့ $1e-12$ ထက် ကြံးနေသမျှ ကာလပတ်လုံး do-while loop ကို ပတ်ပါလို့ program ကို instruct လုပ်ထားပါတယ်။ စာဖတ်သူကိုယ်တိုင် program ကို trace လုပ်ကြည့်ပါ၊ run ကြည့်ပါ။ cosine series ကိုလည်း စမ်းရေးကြည့်စေချင်ပါတယ်။

EXAMPLE 3.20: Newton-Raphson Method

Write a C program to find the roots of an equation which is the function of x using the Newton-Raphson method.

$$x^{(n+1)} = x^{(n)} - \frac{F(x)^{(n)}}{F'(x)^{(n)}}$$

where the superscript (n) denotes values obtained on the nth iteration and (n+1) indicates values to be found on the (n + 1)th iteration. Test your program with the equation used in EXAMPLE 3.6. F' means the derivative of F(x) with respect to x.



```

main ( )
{
    int      i = 1;
    double   x, fx, fpx;

    clrscr ( );
    puts ( "Assume x" );
    scanf ( "%lf", &x );
    do {
        fx = 2*x*x - 5*x - 12;
        fpx = 4*x - 5;
        x -= fx / fpx;
    } while ( ++i < 25 );
    printf ( "Root = %5.2lf", x );
}

```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှပါ။

Assume x
10000
Root = 4.00

Assume x
-5000
Root = -1.50

www.foxitsoftware.com

ဒီ program ဟာဆိုရင် equation တစ်ခုက မသိကိန်းတစ်ခုရဲ့ တန်ဖိုးကို Newton method နဲ့ ရှာတဲ့နည်းဖြစ်ပါတယ်။ Bisection method ထက် ပိုကောင်းပါတယ်။ ဘာပြုလိုလဲဆိုတော့ အဖြစ်၊ အထောက်^{ဦး} ခန်းမျိုးထားတာတောင် အကြောင်ရော အများ^{ဦး} မတွက်ဘဲနဲ့ အဖြမ်နှစ်ရတာကိုး။ တစ်ခုပဲရှိတယ်။ ဒီ method ကို သုံးမယ်ဆိုလိုရှိရင် equation ကို differentiate လုပ်ဖို့လိုတဲ့အတွက် အခက်အခဲရှိတာပေါ့။ ကောင်းမြှုပ်နည်းပေါ်တော်လေးကို တစ်ဆင့်ရှုံးလေ့လာကြည့်ရအောင်။

- ၁။ Iteration counter 'i' ကို ၁ လို့ initialize လုပ်ပါတယ်။
- ၂။ Assume x လို့ prompt တစ်ခု ပေါ်လာတဲ့အခါမှာ ကျွန်တော်တို့က keyboard ကနေ 10000 လို့ ရိုက်ထည့်ပေးလိုက်မယ် ဆိုပါစို့။ ဒါဆိုရင် $x = 10000$ ဖြစ်သွားပါပြီ။
- ၃။ do loop ထဲကို ဝင်လာပြီးတော့ $f(x)$ နဲ့ $f'(x)$ တို့ရဲ့ တန်ဖိုးတွေကို တွက်ယူပါတယ်။ ပြီးတော့ရင် Newton ပုံသေနည်းကိုသုံးပြီး $x = x - f(x) / f'(x)$ ကို ဆက်တွက်ပါတယ်။ ညီမျှခြင်းရဲ့ ညာဘက် x ဟာ assumed value ပါ။ while နဲ့ တွေ့တဲ့အခါမှာ i ကို တစ်တိုးပြီးတော့ 25 ထက် ငယ်လားလို့ မေးပါတယ်။ ငယ်တယ်ဆိုရင် step (3) ကို ပြန်လုပ်မှာပါ။
- ၄။ ဒီနည်းအတိုင်း do-while loop ကို 25 ကြိမ်ပတ်ပြီးတဲ့အခါမှာ အဖြော်ပါပြီ။ အဖြော် x ထဲမှာ ရှိနောတယ်လေ။ ဒီခါကျရင် do-while loop ထဲက အပြီးထွက်လာပြီးတော့ computer မှာ $\text{Root} = 4.00$ လို့ အဖြော်ပြုပါလိမ့်မယ်။ program လည်း ဒီမှာတင် ပြီးသွားပါပြီ။

၃.၇ for Statement

for statement ဆိုတာ C language မှာ looping လုပ်ဖို့အတွက် အသုံးများဆုံး control statement တစ်ခုပါပဲ။ ဒီ statement ရဲ့ general form က ဒီလိုပါ။

for (expr1; expr2; expr3) STATEMENT;

ဒီပုံစံမှာ expr1, expr2, expr3 တို့ရဲ့ဆိုလိုရင်း အမိုးယူကတော့ expr1 ဟာ looping ရဲ့ အစဉ်း counter ဖြစ်ပါတယ်။ တရှုံးက initial index လို့လည်း ပြောပါတယ်။ expr2 ကတော့ looping ကို ရှုံးဆက်ပတ်သင့်၊ မပတ်သင့်ဆိုတာကို စိစစ်တဲ့ expression တစ်ခုပါ။ ဒီ expression ဟာ မှန်နေသူမျှ looping ဟာ ဆက်ပတ်နေမှာပါပဲ။ expr3 ကတော့ counter ကို increment သို့မဟုတ် decrement လုပ်တာဖြစ်ပါတယ်။ increment operator နဲ့ decrement operator တွေကို for statement ထဲမှာ ထည့်သုံးလို့ရပါတယ်။ for (; expr2; expr3) လို့လည်း ရေးလို့ရတာရှုပါတယ်။ for (; ;) ဆိုရင်တော့ never ending loop ဖြစ်သွားပါလိမ့်မယ်။ infinite loop လို့လည်း ဓာတ်ပါတယ်။ ပြဿနာမတက်အောင် ကြည့်သုံးပါ။

EXAMPLE 3.12:

Temperature Conversion

Write a C program to convert the temperatures in Celsius to Fahrenheit.

```
main ( )
{
    int      cel;
    float    fah;

    clrscr ( );
    puts ("CELSIUS      FAHRENHEIT\n");
    for (cel = -40; cel <= 40; cel += 10) {
        fah = 1.8 * cel + 32;
        printf ("%5d      %10.2f\n", cel, fah);
    }
}
```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

CELSIUS	FAHERNHEIT
-40	-40.00
-30	-22.00
-20	-4.00
-10	14.00
0	32.00
10	50.00
20	68.00
30	86.00
40	104.00

ဒဲ program ဟာ စင်တိဂရိတ်ကနေ အရင်ပိုက်ကို conversion လုပ်တဲ့ program ပါ။ အနှစ် ၄၀ ဒီဂရိစင်တိဂရိတ်ကနေ အပေါင်း ၄၀ ဒီဂရိ စင်တိဂရိတ်ကို ၁၀ ဒီဂရိစင်တိဂရိတ်ထိ တိုးပြီးတော့ ဒီဂရိ အရင်ပိုက်ကို ပြောင်းပေးသွားပါတယ်။ for loop ဘယ်လို အလုပ်လုပ်သွားလဲဆိုတာကို ကြည့်ရအောင်။

J" FAHRENHEIT සිතු heading ගී රෙපිටයි॥ තම්බෙනාද්:මාසිං:පිටයි॥
 for loop අ.ගෝ.පිටි॥ initial condition ම cel = -40 පි॥ විනො fah = 1.8 * cel
 + 32 = -40 ලදී computer ම තුන්දිග්බිටයි॥ cel අ. fah ගී computer මා print
 ලදුවූ:තාක් closing brace } අ. ගෝ.පිටයි॥ cel ගී 10 එහි:පිටයි॥ cel = -40 + 10
 = -30 ප්‍රේත්වා:පිටි॥ විච්චා cel = -30 භා last counter ප්‍රේත්තා cel = 40 නග්‍යයෙහා:
 තුළා:ලදී. මෙහි:පිටයි॥ වි expression භා TRUE ප්‍රේත්තාම්. for loop ගී නග්‍යපත්ති:පිටයි॥
 2" තම්බෙනාද්:මාසිං:පිටයි॥ විච්චා cel = 50 ප්‍රේත්ලාබිඩ්මයි॥ විච්චා cel = 50 භා 40 නග්‍ය නය්‍යයෙනුමලා:
 තුළා:මලා॥ වි expression භා FALSE ප්‍රේත්තාමෙනාද් for loop පෙන්ම computer
 මංද්‍යෙනුපිවු:॥ වොග් ගිහෙවුමද්:ලාපී:තොශු program පී:වු:පිටි॥

EXAMPLE 3.22: Integration

Write a C program to find the value of integral

$$\int_0^{\pi} \log_e (5 - 4 \cos x) dx \quad \text{using the rectangular rule}$$

```
# include<math.h>

main ( )
{
    int    n;
    double a, b, x, h, h2, integral = 0;

    clrscr ( );
    puts ( "Enter a, b, n" );
    scanf ( "%lf,%lf,%d", &a, &b, &n );
    h = ( b - a ) / n;
    h2 = h/2;
    for ( x = h2; x <= b - h2; x += h )
        integral += h * log ( 5 - 4 * cos ( x ) );
    printf ( "Integral = %7.6lf", integral );
}
```

සි program ගී run ලදිග්මයිඩ්රිං ඡායාප්‍රේට්‍රුමාපි॥

```
Enter a, b, n  
0, 3.141593, 200  
Integral = 4.355173
```

ဒီ integral ရဲ့ exact value က 4.3551723 ဖြစ်ပါတယ်။ integration ရဲ့ error က 0.0000013 ပဲဖြစ်တာမို့။ ဒီ program ဟာ ကောင်းကောင်းအသုံးချလို့ ရတာပေါ့။ အသုံးပြုကြည့်ပါ။

၃.၈ Nested Loops

looping တစ်ခုကို တစ်ခြား looping တစ်ခု သို့မဟုတ် တစ်ခုထက်ပိုတဲ့ looping တွေက င့်ထားခြင်းကို nested loops လို့ခေါ်ပါတယ်။ inner looping နဲ့ outer looping တို့ရဲ့ control structure အမျိုးအစားတွေဟာ တူစရာမလိုပါဘူး။ inner loop မှာ for နဲ့သုံးမယ်။ outer loop မှာ while သုံးမယ်။ ဒါလည်းရတာပါပဲ။ ဒါမှုမဟုတ် outer loop မှာ do-while သုံးပြီးတော့ inner loop မှာ for ကိုသုံးမယ်ဆိုလည်း ဖြစ်တာပါပဲ။ တခုပဲသတိထားရမှာက loop တစ်ခုနဲ့တစ်ခု crossing ဖြစ်လို့ ပရပါဘူး။ အပြင် loop က အတွင်း loop ကို completely င့်ထားတာဖြစ်နေရပါမယ်။

EXAMPLE 3.23: Converting Several Lines of Text to Uppercase

Write a C program using the nested for loop continuing the conversion until the first character in a line is an asterisk.

```
# define      EOL      '\n'  
  
main( )  
{  
    char      letter [80];  
    int       tag, k;  
  
    clrscr( );  
    for ( ; (letter [0] = getchar( )) != '*' ; ) {  
        for ( k = 1; ( letter [k] = getchar( )) != EOL; ++k );
```

```

tag = k;
for (k = 0; k < tag; ++k)
    putchar ( toupper ( letter [k] ) );
printf ( "\n\n" );
}
printf ( "Good Bye" );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် ကျွန်တော်တိ၊ ရိုက်ထည့်သမျှ စာကြားတွေကို (စာတစ်ကြားဟာ 79 character ထက်မကျော်ရဘူးနော်) uppercase letter တွေအဖြစ် ပြောင်းပေးမှာပါ။ program ကို ရပ်စေချင်ရင် data အဖြစ် asterisk (*) တစ်လုံးရိုက်ထည့်ပေးရပါမယ်။ ဒါဆိုရင် computer ၏ Good Bye လို့ နှုတ်ဆက်ပြီးတော့မှ ပြန်လည်ပေးမှာပါ။

Now is the time for all good men

NOW IS THE TIME FOR ALL GOOD MEN

to come to the aid of their country

TO COME TO THE AID OF THEIR COUNTRY

+

Good Bye

www.foxitsoftware.com

EXAMPLE 3.24: Solving Indeterminate Equations

Write a C program to determine the three unknowns x, y, and z which are defined by two equations as:

$$x + y = z = 25$$

$$2.5 x + 5 y + 0.25 z = 25$$

where x,y and z are integers.

```

main ( )
{
    int x,y,z, sum;
    for ( x = 1; x <= 10; ++x )
        for ( y = 1; y <= 5; ++y ) {
            z = 25 - x - y;
            sum = 2.5 * x + 5 * y + 0.25 * z;
            if ( sum == 25) goto RESULT;
        }
    RESULT: printf ( "X = %d\n", x );
    printf ( "Y = %d\n", y );
    printf ( "Z = %d\n", z );
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

```

X = 2
Y = 3
Z = 20

```

EXAMPLE 3.25: Drawing Two Doagonal Lines

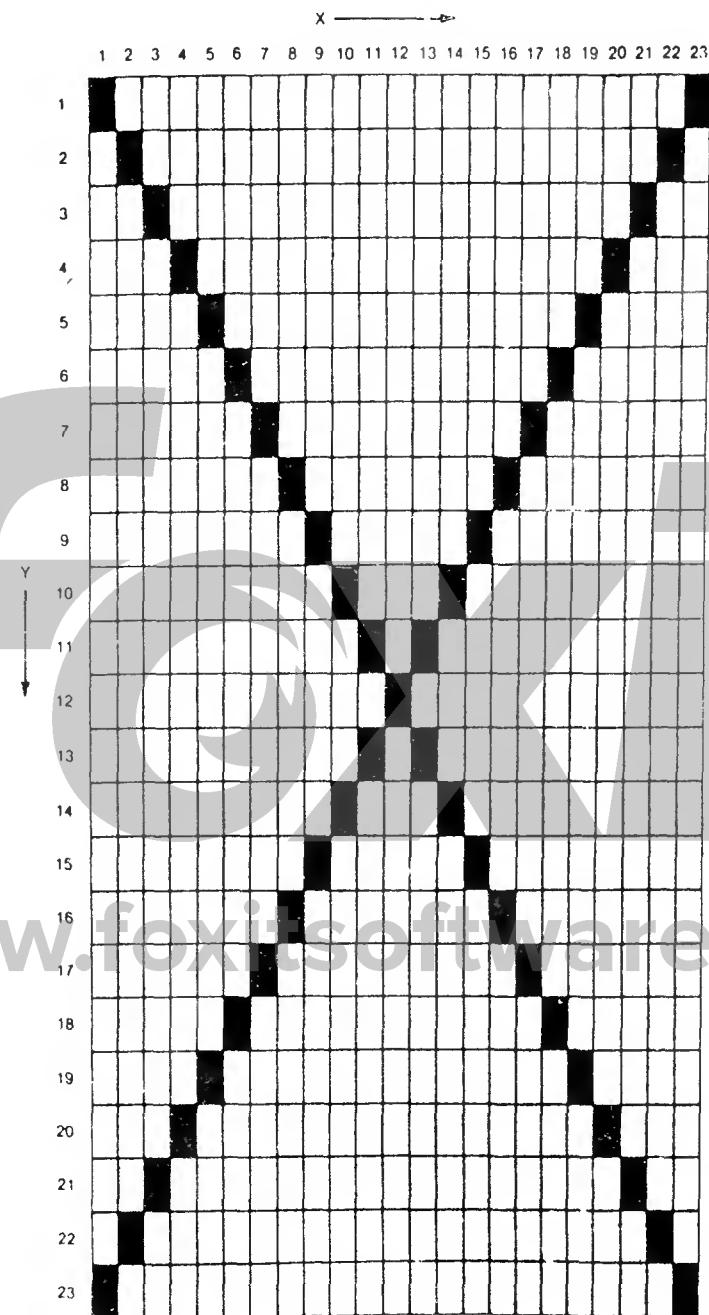
Write a C program to print two diagonal lines on the screen.

```

main ( )
{
    int i,j;
    clrscr ( );
    for ( j=1; j<10; j++ ) {
        for ( i=1; i<10; i++ )
            if ( i==j )
                printf ( "\xDB" ); /* print black squares */
            else if ( i == 10 - j )
                printf ( "\xDB" );
            else
                printf ( "\xB0" ); /* print gray squares */
        printf ( "\n" );
    }
}

```

ဒီ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်လာမှပါ။



EXAMPLE 3.26: Finding an AP Sequence

Write a C program to find an AP series in the form
 $x, x+1, x+2, x+3, \dots, x+n-1$
whose sum of all consecutive integers, S determined
by $nx + n(n-1)/2$ is known. n is the total number of
consecutive integers.

```
main ( )
{
    int i, j, k, s, n, x;

    clrscr ();
    printf ("nEnter the sum : ");
    scanf ("%d", &s);
    printf ("\nThe AP series is \n");
    for (n = 2; n*(n-1) < 2*s; n++) continue;
    n--;
    for (i = n; ; i--) {
        k = i * (i - 1) / 2;
        if ((int) ((s - k) % i) != 0) continue;
        x = (s - k) / i;
        for (j = 0; j < i; j++)
            printf ("%d ", x + j);
        printf ("\n");
        break;
    }
}
```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter the sum : 27

The AP series is

2 3 4 5 6 7

၃၄ switch Statement

statement တွေအများကြီးပါဝင်တဲ့ variable group အပ်စုတွေကနဲ့ ကိုယ်လိုချင်တဲ့ statement group ဆိုရင် ရွှေ့ပြီးရောက်သွားစေချင်ရင် switch ဆိုတဲ့ statement ကို သုံးရပါမယ်။ ရွှေ့ပုံရွှေ့နည်းကတော့ switch နောက်က ကွင်းထဲမှာ ရေးထားတဲ့ expression ရဲ့ current value ပေါ်မှာ မူတည်ပြီး selection လုပ်သွားမှာပါ။ switch ရဲ့ general form က ဒီလိုရှိပါတယ်။

```
switch (expr) {  
    case expr1:  
        statement1;  
        break;  
    case expr2:  
        statement 2;  
        break;  
    .....  
    case exprn:  
        statement n;  
        break;  
    default:  
        .....;  
}
```

www.foxitsoftware.com

ဒီပုံစံမှာဆိုရင် switch နောက်က expr ရဲ့ current value ဟာ expr1 ရဲ့ value နဲ့ တူမယ် ဆိုလိုရှင် ပထမအပ်စုထဲက statement 1 ကို execute လုပ်ပြီးတော့ break တွေ့တာနဲ့ closing brace } အောက်ကိုဆင်းသွားမှာပါ။ တကယ်လို့ expr2 နဲ့ တူမယ်ဆိုရင် ဒုတိယအပ်စုထဲက statement2 ကို execute လုပ်ပြီးတော့ closing brace အောက်ကိုဆင်းသွားမှာပါပဲ။ expr 1 ကနဲ့ exprn အထိ တစ်ခုမှ မတူဘူးဆိုရင်တော့ default အောက်က အပ်စုကို execute လုပ်မှာပါ။ ပြီးတော့ရင် switch block က ထွက်သွားမှာဖြစ်ပါတယ်။ တကယ်လို့ expr က exprn နဲ့တူတယ်ဆိုရင် statement(n) ကို execute လုပ်မှာပါ။ ဒါပေမယ့် case block မှာ break ရေးမထားဘူးဆိုရင် default နောက်က အလုပ်တွေကို computer ဆက်လုပ်းမှာပါပဲ။ ဒါဆိုရင် အဖြေဟာ ကျွန်ုတ်တို့ထင်သလို ဟုတ်မှာမဟုတ်တော့ ပါဘူး။ သတိထားပါ။

EXAMPLE 3.27: Program Payroll

Write a C program to calculate the payroll for four different classes of workers.

```
main ( )
{
    int      class;
    float    hours, wage, pay;

    printf ("Enter Class and Hours worked :\n");
    scanf ("%d,%f", &class, &hours);

    switch ( class ) {
        case 1: wage = 4.0; break;
        case 2: wage = 5.36; break;
        case 3: wage = 7.36; break;
        case 4: wage = 8.75; break;
        default: printf ( "ERROR" );
                  goto EXIT;
    }
    if ( hours > 40 ) pay = 40 * wage + 1.5 * wage * ( hours - 40 );
    else pay = hours * wage;
    printf ( "Pay = %6.2f", pay );
    EXIT : printf ( "\n" );
}
```

ဒါ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter Class and Hours worked:

1, 50

Pay = 220.00

Enter Class and Hours worked :

12, 50

ERROR

EXAMPLE 3.20. Sorting Letters

Write a C program to sort letters into vowels and consonants.

```
main( )
{
    char ch;

    printf ("Enter the letter ? ");
    ch = getche( );

    switch ( ch ) {
        case 'a':
        case 'e':
        case 'i':
        case 'o':
        case 'u':
            printf ("\n%c is a vowel\n", ch );
            break;
        default:
            printf ("\n%c is a consonant\n", ch );
    }
}
```

ဒါ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။
www.toxitsoftware.com

Enter the letter ? u
u is a vowel

Enter the letter ? c
c is a consonant

၃.၁၀ break Statement

break ဆိတ် looping သို့မဟုတ် switch အပ်စကနေ exit လုပ်ဖို့အတွက် အသုံးပြုတာပါ။ while, do-while, for, switch statement တွေနဲ့ break ကို တွေသုံးလို့ရပါတယ်။ ဒါပေမယ့် if-statement ကနေ exit လုပ်တာကိုတော့ break နဲ့ သုံးလို့မရပါဘူး။ goto ပဲ သုံးလို့ရမှာပါ။ break ရဲ့ general form က ဒီလိုပါ။

break;

EXAMPLE 3.29: Write a C program to get a command from the user and take the requested action.

```
# define TRUE 1
# define FALSE 0

main()
{
    int char
    flag = TRUE;
    key;

    clrscr();
    puts ("Press ? for help.");
    while (flag) {
        printf ("Command : ");
        key = getch();
        switch (key) {
            case 'q':
            case 'Q':
                puts ("\nDon't try again. Good luck.");
                flag = FALSE
                break;
            case 'h':
            case 'H':
            case '?':
```

```
    puts ("\nHere is the menu.");
    puts ("h or H or ?      : HELP");
    puts ("m or M      : MESSAGE");
    puts ("q or Q      : QUIT\n");
    break;

case 'm':
case 'M':
    puts ("\nHere is the message for you.");
    puts ("Programming is disappointing.\n");
    break;

default :
    puts ("\nUnknown command ... \n");
    break;
}
```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှပါ။

Press ? for help.

Command : ?

Here is the menu.

h or H or ? : HELP

m or M : MESSAGE

q or Q : QUIT

<http://www.toronto.ca>

Command :

Here is the message for you.

Programming is disappointing.

Command : x

Unknown command ...

Command : q

Don't try again. Good luck.

- ၁။ program စစ်ဆေးစွာ flag = TRUE = 1 လို့ initialize လုပ်ထားပါတယ်။
- ၂။ screen clear လုပ်ပြီးတော့ Press ? for help ဆိုတဲ့ prompt တစ်ခု computer မှာ ပေါ်လာပါပြီ။
- ၃။ while (flag) ဟာဆိုရင် TRUE ဖြစ်တာကြောင့် while loop ထဲကို computer ဝင်လာပါတယ်။ Command : ဆိုတဲ့ prompt ပေါ်လာတဲ့အခါမှာ ကျန်တော်တို့ keyboard ကနေ ? ကို ရှိက်ထည့်မယ်ဆိုပါစို့။ key = '?' ဖြစ်သွားပါပြီ။
- ၄။ Switch (key) ဆိုတဲ့ အမိဘယ်က key = '?' ဖြစ်တဲ့ case နေရာကို သွားပါလို့ ဆိုလိုပါတယ်။ ပြီးတော့ရင် case '?' " ကြောင့် **Here is the menu.**

h or H or ? : HELP
m or M : MESSAGE
q or Q : QUIT

ဆိုတဲ့ စာတွေဟာ computer မှာ ပေါ်လာပါပြီ။ နောက်ပြီးတော့ break ကို တွေ့တာနဲ့ switch block ထဲကနေ အပြင်ထွက်သွားပါတယ်။ flag = TRUE ကတော့ မပြောင်းပါဘူး။

- ၅။ ဒီနည်းအတိုင်း while loop ကို ပတ်ပြီးရင်း ပတ်နေမှာပါ။ တစ်ချိန်မှာ ၁ ကို ရှိက်ထည့်လိုက်မယ် ဆိုပါတော့။ key = 'q' ဖြစ်သွားပြီလေ။ ဒီတော့ case 'q' : ရှိတဲ့နေရာကို computer သွားမှာပါ။ ပြီးတော့ရင်

Don't try again : Good luck.

ဆိုတဲ့စာကြောင်းဟာ computer မှာ ပေါ်လာပါလိမ့်မယ်။ flag = TRUE အစား: flag = FALSE လို့ ပြန်ပြောင်းလိုက်ပါတယ်။ break ကို တွေ့တာနဲ့ switch block က ထွက်ပြီးတော့ while (flag) ကို ပြန်သွားတဲ့အခါမှာ flag က FALSE ဖြစ်နေတဲ့အတွက် while loop ထဲကို computer မဝင်တော့ပါဘူး။ အောက်ဆုံးကို ကျဉ်ဆင်းသွားပါပြီ။ program ဒီမှာတင် ပြီးသွားပြီ ဖြစ်ပါတယ်။

EXAMPLE 3.30: Repeat EX 3.24 using the break statement.

```
main ()
{
    int      x,, y, z;
    float   sum;

    for (x = 1; x <= 10; ++x) {
        for (y = 1; y <= 5; ++y) {
            z = 25 - x - y;
```

```

        sum = 2.5 * x + 5 * y + .25 * z;
        if ( sum == 25.0 ) break;
    }
    if ( sum == 25.0 ) break;
}
printf ( "X = %d\n", x );
printf ( "Y = %d\n", y );
printf ( "Z = %d\n", z );
}

```

ဒဲ program ကို လေ့လာကြည့်မယ်ဆိုရင် break statement ဟာ တစ်ခါရေးရင် looping ထဲက တစ်ခါပဲ ထွက်လိုပါတယ်။ နှစ်ခါထွက်ချင်တယ်ဆိုလိုရှိရင် inner loop မှာ break တစ်ခါ၊ outer loop မှာလည်း break တစ်ခါ၊ နှစ်ခါရေးပေးပို့ လိပါတယ်။ goto ဆိုရင် တစ်ခါတည်းနဲ့ ကိစ္စပြီးတာပေါ့။ ဒဲ program ကို run မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

X = 2
Y = 3
Z = 20

EXAMPLE 3.31: Write a C program to calculate the day on which any date

```

main ( )
{
    int     day, month, year, x, y, z, day_of_weak;
    char    ch;

    clrscr ( );
    puts ( "Enter a date in the form : day/month/year" );
    scanf ( "%d%c%d%c%d", &day, &ch, &month, &ch, &year );

```

```

x = year % 100;
y = year / 100;
z = month - 2;
if ( z <= 0 ) {
    z += 12;
    --x;
}
day_of_week = (( 13 * z - 1 ) / 5 + day + x + x / 4 + y / 4 - 2 * y ) % 7;
printf ( "\n%d/%d/%d is a ", day, month, year );
switch ( day_of_week ) {
    case 0 : puts ( "Sunday" ); break;
    case 1 : puts ( "Monday" ); break;
    case 2 : puts ( "Tuesday" ); break;
    case 3 : puts ( "Wednesday" ); break;
    case 4 : puts ( "Thursday" ); break;
    case 5 : puts ( "Friday" ); break;
    case 6 : puts ( "Saturday" );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် ...

Enter a data in the form : day/month/year
22/10/1996

22/10/1946 is a Tuesday

computer မှာ အလိပ်ပေါ်လာမှာပါ။ ဒါ program ဘယ်လိုအလုပ်လုပ် သွားတယ်ဆိုတာကို ကျွန်ုတ်တို့ trace လုပ်လို့ရပါတယ်။ date of birth ၂၂/၁၀/၁၉၉၆ ဆိုပါစီ။ ဒီနေ့ဟာ ဘာနေ့ကျလလို့ တွက်ကြည့်လို့ရပါတယ်။ day = 22, month = 10, year = 1996 ဖြစ်သွားပါပြီ။ ဒီတော့ x = year % 100 = 96, y = year / 100 = 19 နဲ့ z = month - 2 = 8 တို့ဖြစ်ပါတယ်။ if (z <= 0) ဆိုတဲ့ expression ၂ FALSE ဖြစ်တာမို့ အောက်ဆက်ဆင်ပြီး၊ day_of_weak ကို computer ၂ တွက်တော့မှာပါ။ day_of_weak = $(\frac{13*8 - 1}{5} + 22 + 96 + \frac{96}{4} + \frac{19}{4} - 2 * 19)$

$\% 7 = (\frac{103}{5} + 22 + 96 + 24 + 4 - 38) \% 7 = (20 + 118 + 28 - 38) \% 7 =$
 $128 \% 7 = 2$ ရပါတယ်။ ပြီးတော့ရင် switch (day_of-week) ကိုတွေ့တဲ့အခါမှာ case 2: ဆိုကို computer သွားပြီးတော့ Tuesday ကို ရိုက်ပြပါလိမ့်မယ်။ သူ.ရှေ့မှာ 22/10/1996 is a ဆိုတာကို switch ထဲ မဝင်ခင်ကတည်းက ရိုက်ခဲ့ပြီးပြီလေ။ ဒါကြောင့်မို့ computer မှာ 22/10/1996 is a Tuesday လို့ ပေါ်လာတာပါ။ break ကို တွေ့တာနဲ့ switch block က အရင်ထွက်မယ်။ နောက်ထပ် closing brace တွေ့တာနဲ့ program ရပ်သွားပါပြီ။

၃.၁၁ continue Statement

looping တစ်ခုကို ပတ်နေတုန်းမှာ loop ထဲက statement တွေကို အကုန် execute မလုပ်ဘဲနဲ့ လမ်းတစ်ဝင်းကနောက်တစ်ခုကိုမဲ့ ပြန်ပတ်ချင်ရင် continue ဆိုတဲ့ statement ကို သုံးရပါတယ်။ နောက်တစ်ခုကိုမဲ့ ပြန်ပတ်ချင်ရင် continue ဆိုတဲ့ statement ကို သုံးရပါတယ်။ continue ကို while, do-while, for တို့နဲ့ တွဲပြီး သုံးနိုင်ပါတယ်။ continue ရဲ့ general form က ဒီလိုပါ။

continue;

EXAMPLE 3.32: Averaging a List of Nonnegative Numbers

Write a C program to find the average of the nonnegative integers from a list of numbers.

```
main ( )
{
    int      n, count, navg = 0;
    float    x, avg, sum = 0;

    printf ( "How many numbers ? " );
    scanf ( "%d", &n );
```

```

for ( count = 1; count <= n; ++count ) {
    printf ("X = ");
    scanf ("%f", &x );
    if ( x < 0 ) continue;
    sum += x;
    ++navg;
}
avg = sum / naeg;
printf ("\n The average is %0.2f", avg);
}

```

ဒါ program ဟာဆိုရင် ကျွန်တော်တိ၊ ထည့်လိုက်တဲ့ data တွေထက် positive ကဏ္နာ:တွေကိုပဲ ရွှေ့ပေါင်းပြီးတော့ average ရှာပေးတဲ့ program ပါ။ x က သုညထက် ငယ်တယ်ဆိုရင် sum += x; နဲ့ ++navg; ဆိုတဲ့ statement (j) ခက်း skip လုပ်ပြီးတော့ for loop ကို နောက်တစ်ကြိမ် ပတ်ပါ လိမ့်မယ်။ x ဟာ သုညထက် မငယ်ဘူးဆိုရင် အဲဒီကဏ္နာ:ကို sum ထဲမှာ ထည့်ပေါင်းမှာပါ။ နောက်ပြီးတော့ counter ကို တစ်ခုတိုးမှာပါ။ ဒီနည်းအတိုင်း program ကို တွက်သွားမယ်ဆိုရင် computer မှာ ဒီလို ပေါ်လာမှာပါ။

How many numbers ? 6

X = 1

X = -1

X = 2

X = -2

X = 3

X = -3

The average is 2.00

```

main ( )
{
    int      i, j, k, x = 0;

    clrscr ( );
    for (i = 0; i < 5; ++i)
        for (j = 0; j < i; ++j) {
            switch (i + j - 1) {
                case -1:
                case 0:
                    x += 1;
                    break;
                case 1:
                case 2:
                case 3:
                    x += 3;
                    continue;
                default :
                    x += 2;
            }
            printf ("x = %d\n", x);
        }
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

```

x = 1
x = 15
x = 20
x = 22
x = 24

```

ဒီအဖြေ ဘယ်လိုရသွားတယ်ဆိုတာကို စာဖတ်သူကိုယ်တိုင် trace လုပ်ပြီးတော့ အဖြေရှာကြည့်စေခဲ့ပါတယ်။ ဒါဆိုရင် break, continue တို့ အကြောင်းတွေကို ပိုနားလည်လာပါလိမ့်မယ်။

၃.၁၂ comma Statement

looping တစ်ခုထဲမှာပဲ expression နှစ်ခုကို သုံးပြီးတော့ တစ်ခကရှုံးကိတိုးပြီး looping ပတ်သွားချိန်မှာ ကျွန်ုတ်တစ်ခုက အောက်ပြန်ပြီးပတ်သွားစေချင်ရင် expression တွေကြားမှာ comma (,) operator ကို သုံးရပါမယ်။ သူ့ကို ဘယ်လိုသုံးသွားလဲဆိုတာကို အောက်ကပ္ပါဒာမှာ လေ့လာကြည့်ပါ။

EXAMPLE 3.34: Searching for Palindromes

A palindrome is a word, phrase or sentence that reads the same way either forward or backward. Write a C program that will enter a line of text containing a word, phrase or a sentence and determine whether or not the text is a palindrome. The comparisons will include punctuations and blank spaces.

```
# define EOL      '\n'
# define TRUE     1
# define FALSE    0
main( )
{
    char    letter [80];
    int     tqag, k, kback, flag, loop = TRUE;

    while ( loop ) {
        flag = TRUE;
        printf ("Enter a word, phrase or sentence :\n");

        for ( k = 0; ( letter [k] = getchar () ) != EOL; ++k );
        if (( toupper ( letter [0] == 'E' ) && ( toupper ( letter [1] ) == 'N' )
              && ( toupper ( letter [2] ) == 'D' )) break;
        tag = k - 1;
        for ( ( k = 0, kback = tag ); k < tag / 2; ( ++k, --kback ) ) {
            if ( letter [k] != letter [ikback] ) {
                flag = FALSE;
                break;
            }
        }
    }
}
```

```
        for (k = 0; k <= tag; ++k)
            putchar ( letter [k] );
        if ( flag ) printf ("IS a palindrome\n");
        else printf ("is NOT a palindrome\n");
    }
}
```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter a word, phrase or sentence:

PULLED

PULLED IS a palindrome

Enter a word, phrase or sentence :

FALSE

FALSE is NOT a palindrome

Enter a word, phrase or sentence :

END

FOXIT
www.foxitsoftware.com

FUNCTIONS

function ဆိတာ မိမိဘသာ ရပ်တည်လို့ရပြီး အဓိပ္ပာယ်ရှိရှိ တည်ဆောက်ထားတဲ့ self-contained program တစ်ခုကို ခေါ်တာပါ။ C program မှန်ရင် အနည်းဆုံး function တစ်ခုတော့ ပါရမယ်လို့ ဖြောခြားပါပြီးနော်။ အဒီ function ကို main လို့ ခေါ်တာလည်း သိပြီးသားပါ။ ဒီတစ်ခုတော့ main function အပြင် တစ်ခြား function တွေကိုလည်း လေ့လာကြမယ်လေ။ function တွေကို ဘာအတွက် အသုံးပြုရတာလဲ၊ အကျိုးရှိလို့လားလို့၊ မေးနိုင်ပါတယ်။ အကျိုးကျေးဇူး အများကြီး ရှိတာပေါ့။ function တွေကို အသုံးပြုခြင်းအားဖြင့် main program မှာ ထပ်ခါတလဲ လုပ်ရမယ့် အလုပ်တွေ ရှိနေတယ်ဆိုလို့ရှိရင် main ထဲမှာ အဲဒါတွေနဲ့ ရွှေပြအောင် ရေးစရာမလိုတော့ဘူး။ function အသစ်တစ်ခု သပ်သပ်ရေးပြီးတော့ အဲဒီကိုပဲ သွားလိုက်ပြန်လိုက် လုပ်ခိုင်းနေလို့ရတယ်လေ။ ဒီတော့ program လည်း ရှင်းသွားသလို ပိုလည်း efficient ဖြစ်သွားပါတယ်။ function ကို အသုံးပြုရတဲ့ နောက်ရည်ရွယ်ချက်တစ်ခုကတော့ data တွေကို program တစ်နေရာကနေ တစ်နေရာကို transfer လုပ်ဖို့အတွက် function ကို အသုံးပြုပါတယ်။ ကောင်းပြီ၊ function ဆိတာကို အဓိပ္ပာယ်အရှင်းဆုံး ပေါ်သွားအောင်လို့ program အလွယ်တစ်ခု ကျွန်းတော်ရေးပြပါမယ်။ လေ့လာကြည့်ပါ။

EXAMPLE 4.1: Here is a simple C program that contains two functions : main () and func (). When executed, the program displays a message "I love C" on the screen.

```
main ( )
{
    printf ("I ");
    func ();
    printf ("C" )
}
```

```

func ()
{
    printf ("love ");
}

```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

I love C

ဒီ program ကို လေ့လာကြည့်မယ်ဆိုရင်...

၁။ main function ထဲက printf ("I "); ကို computer က အရင်ဆုံး execute လုပ်ပါဖြိုး။ I နောက်မှာ space တစ်ခု တမင်ချွန်ထားတာကို သတိပြုကြည့်ပါ။ ဒါကြောင့်မို့လည်း computer display မှာ I (space) ဆိုတာ အရင်ပေါ်လာတာပါ။

၂။ func(); ဆိုတဲ့ အဓိပ္ပာယ်ကတော့ 'func' function ရှိတဲ့နေရာကို သွားပါလို့ ဆိုလိပါတယ်။ func နောက်က ကွင်း (j) ခုဟာ စလာဖြစ်တဲ့အတွက် main ကနေ ဘာ argument (data) မှ func ဆိုကို ယူမသွားပါဘူး။ computer က main ကနေ func ဆိုကို ဒီတိုင်းလက်ချဉ်းကူးသွားမှာပါ။ func ထဲကိုရောက်လာပါဖြိုး။ computer လုပ်ရမယ့်အလုပ်ကတော့ printf ("love "); ပါဝါ။ ဒီတော့ love ကို I (space) နောက်မှာ ကပ်ရှိပါဖြိုးတော့ space တစ်ခု ချွန်ထားခဲ့ပါလိမ့်မယ်။ အခုခံဗိုရင် computer မှာ I love ဆိုတာ ပေါ်နေပြီလေ။ closing brace နဲ့တွေ့တဲ့အခါမှာ func ကနေ main ဆိုကို ပြန်သွားလို့ရပါဖြိုး။

၃။ main ထဲက func (); statement အောက်ကနေရာကို computer ရောက်လာပါတယ်။ အဲဒီမှာ computer လုပ်ရမယ့်အလုပ်ကတော့ printf ("C"); ဖြစ်ပါတယ်။ C ကို I (space) love (space) နောက်မှာ ကပ်ရှိပါလိမ့်မယ်။ ဒါကြောင့်မို့ computer display မှာ နောက်ဆုံးပေါ်လာတဲ့ အဖြေကတော့ I love C ဖြစ်ပါတယ်။ closing brace နဲ့တွေ့တဲ့အခါမှာ program ပြီးသွားပါဖြိုး။

၄။ ဒီ function နဲ့ သဘောတရားတူတဲ့ ဥပမာတစ်ခုကို ကျွန်ုတ်ပောပြုပါမယ်။ လူတစ်ယောက်ဟာ အိမ်မှာ စာရေးနေရာင်းတန်းလန်းကနေထပြီး အလူတစ်ခုကို သွားပါတယ်။ အလူအတွက် ဘာလက်ဖွဲ့မှ ပါမသွားဘူး။ ဒါပေမယ့် အလူမှာတော့ တွေ့ကရာအကုန်တုတ်မယ်။ ဝရင်အိမ်ကို ကိုယ်ခါလက်ခါ ပြန်လာပါတယ်။ အိမ်ရောက်လာကဲ့အခါမှာ ရေးလက်စစာကို ဆက်ရေးနေမယ်ဆိုပါစို့။ ဒီဥပမာမှာ အိမ်က main function နဲ့ တူပါတယ်။ func ကတော့ အလူပေါ့။ အလူနဲ့အိမ်ဟာ ဆက်စပ်မှု ရှိပေမယ် အတွင်းရေးကိစ္စချင်း မပတ်သက်ပါဘူး။ တစ်ခြားစီပါ။ main က အလုပ်နဲ့ func က အလုပ် (j) ခုကို ဆက်စပ်ပေးတာပဲ ရှိပါတယ်။

EXAMPLE 4.2:

This is another version of EX 4.1. In this program, main () first calls func1 (), which then calls func2 (). Next, func2 () displays "I" and then returns to func1 (), that prints "love" and then returns to main () which prints "C".

```
main ()
{
    func1 ();
    printf ("C");
}

func1 ()
{
    func2 ();
    printf ("love ");
}

func2 ()
{
    printf ("I ");
}
```



ဒါ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ပေါ်လေမယ့် message ကဟော
EXAMPLE 4.1 က အတိုင်း I love C ပါပဲ။ program execute လုပ်သွားတဲ့ step တွေကို မြားတွေ
နဲ့ပဲ ကျွန်တော် ရေးပြသွားပါမယ်။ လေမရှည်အောင်လိုပါ။ main () → func1 () → func2 () → printf
("I ") → printf (" love ") → printf ("C") အတိုင်း computer က function တွေထက်
ဝင်လိုက်ထွက်လိုက် လုပ်သွားရင် I love C ဆိုတဲ့ message ကို ရိုက်ပေးပါလိမ့်မယ်။

EXAMPLE 4.3:

Printing a Message Surrounded by a Border

Write a C program to print a message surrounded by a border. Use functions to draw the elements of the border.

```

#define WIDTH 41
#define ROWS 4

main ()
{
    clrscr ();
    line ();
    sides ();
    printf ("t * t  U AUNG MYINT \t*t *\n");
    sides ();
    line ();
}

line ()
{
    int i;

    for (i = 1; i <= WIDTH; ++i) {
        if (i == 1) putchar ('t *');
        putchar ('*');
    }
    putchar ('\n');
}

```

```

sides ()
{
    int i;

    for (i = 0; i <= ROWS; ++i)
        printf ("t * \t\t\t\t * \n");
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer ပေါ်မှာ ဒီလိပ်လာမှာပါ။

U AUNG MYINT

EXAMPLE 4.4: Finding Square Root of a Number

Write a C program to display the square root of a number that is entered from the keyboard.

```
main ( )
{
    float      root;
    float      get_sqroot( );
    root = get_sqroot ( );
    printf ( "\nSquare root = %f", root );
}

float      get_sqroot ( )
{
    int       i;
    float     xn, xr;

    printf ( "Enter a positive real number ? " );
    scanf ( "%f", &xn );
    printf ( "Assume its root ? " );
    scanf ( "%f", &xr );

    for ( i = 1; i <= 50; ++i )
        xr = ( xr + xn / xr ) / 2;
    return ( xr );
}
```

ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါပဲ။

Enter a positive real number ? 124

Assume its root ? 12.5

Square root = 11.135529

Enter a positive real number ? 124

Assume its root ? 100

Square root = 11.135529

ဒီ program ရဲ့ ရည်ရွယ်ချက်ကတော့ `get_sqroot()` ဆိတဲ့ function တစ်ခုကို အသုံးပြုပြီးတော့ given number တစ်ခုရဲ့ square root ကို ရှာပေးမှာပါ။ ဒီ function အလုပ်လပ်ပုံက EXAMPLE 4.1 က function နဲ့ မတူပါဘူး။ စောစောက ကျွန်ုတ်ရှင်းပြခဲတဲ့ ဥပမာနဲ့ ပုံစံတစ်မျိုး ကဲ့သွားပါတယ်။ လူတစ်ယောက်ဟာ အိမ်မှာစာရေးနေရင်း တန်းလန်းကထပြီးတော့ အလူအိမ်တစ်ခုကို သွားပါတယ်။ ဘာလက်ဖွဲ့ပစ္စည်းမှလည်း ယူမသွားဘူးလို့ ပြောခဲ့ပါတယ်။ ပြန်လာရင်တော့ တစ်ခုခု ယူလာခဲ့မယ်လို့ အိမ်မှာ ပြောခဲ့တယ်လေ။ main function မှာ `root = get_sqroot();` လို့ ရေးထားတာ။ ဒီသော့ပေါ်ပါပဲ။ `get_sqroot()` က ပြန်လာရင် `data` တစ်ခုခုတော့ ယူလာတော့မှာပါ။ အဲဒီယူလာတဲ့ `data` ကို `root` နဲ့ ညီမယ်လို့လည်း ပြောထားခဲ့ပါတယ်။ computer က `get_sqroot` function ရှိတဲ့နေရာကို သိသွားပါပြီ။ ဒါပေမယ့်ဘာ `data` မှတော့ ယူမသွားဘူးနော်။ စောစောကလူဟာ အလူအိမ်မှာ အဝအပြုတုတ်ပြီးတော့ အပြန်မှာအိမ် အတွက် မုန်တစ်ခု ပြန်ယူလာခဲ့သလိုပဲ `get_sqroot` function ထဲမှာ `calculation` တွေလုပ်ပြီးတော့ function က ပြန်အတွက်မှာ အဖြေကို `return` လုပ်ပေးပါတယ်။ `return (xr)` ဆိတဲ့ အဓိပ္ပာယ်က `get_sqroot() = xr` လို့ `assign` လုပ်ပြီး main function ကို ပြန်ယူမယ်လို့ ဆိုလိုပါတယ်။ အဲဒီလို `assign` လုပ်လို့ရအောင်ကို `xr, get_sqroot()` တို့ကို `float` တွေလို့ `declare` လုပ်ထားတာ တွေ့မှာပါ။ main function ကို ပြန်ရောက်လာတဲ့အခါးကျွန်ုတ် `root = get_sqroot()` လို့ ထပ်ညီပေးပါတယ်။ ဒါမှလည်း `get_sqroot` function က `data` ဟာ main ကိုရောက်လာမှာပါ။ ဒီအဖြေကို လိုချင်ရင် `root` ကို `print` လုပ်ရုပါပဲ။ ဒါဟာ ဒုတိယအမျိုးအစား function တစ်ခုလို့ မှတ်လိုက်ပါ။

EXAMPLE 4.4:**Lowercase to Uppercase Character Conversion**

Write a C program that reads a lowercase character, converts it to uppercase and then writes out the uppercase equivalent.

```
/* this is main program */

main ()
{
    char lower, upper;
    char lower_to_upper ( char lower );

    printf ( "Enter a lowercase character ? " );
    scanf ( "%c", &lower );
    upper = lower_to_upper ( lower );
    printf ( "\n The uppercase equivalent is %c", upper );
}

char lower_to_upper ( char c1 )
{
    char c2;

    c2 = ( c1 >= 'a' && c1 <= 'z' ) ? ( 'A' + c1 - 'a' ) : c1;
    return ( c2 );
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter a lowercase character ? e

The uppercase equivalent is E

ဒါ program ကို လေလာကြည့်မယ်ဆိုလိုရှိရင် function ကို call ခေါကတည်းက data ယူသွားတာကို တွေ့ရပါတယ်။ အဲဒီ ဥစ္စာက lowercase letter 'e' ပါ။ lower_to_upper ဆိုတဲ့ function က return ပြန်လာတဲ့အခါကျတော့ uppercase letter 'E' ကို ပြန်သယ်လာပါတယ်။ ဒါက တတိယအမျိုးအစား function ပါပဲ။ ဒါ function မျိုးကို အသုံးပြုတဲ့ example program တွေကို လေလာဖို့အတွက် ရေးပြထားပါတယ်။ လေလာကြည့်ပါ။

EXAMPLE 4.6:

Finding a Largest Number

Write a C program to find the largest of four integer numbers using a function call.

```

main()
{
    int      num1, num2, num3, num4, larg;

    clrscr();
    puts ("Enter four integer numbers");
    scanf ("%d,%d,%d,%d, &num1, &num2, &num3, &num4);
    larg = larger (larger (num1, num2), larger (num3, num4));
    printf ("Largest number = %d", larg);

}

larger (int a, int b)
{
    int c;
    c = a > b ? a : b;
    return (c);
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်လာမှာပါ။

```

Enter four integer numbers
15, 21, -50, 16
Largest number = 45

```

EXAMPLE 4.7:**Numerical Integration**

Write a C program to calculate the value of the integral.

$$\int_0^{\pi} \log_e (5 - 4 \cos x) dx$$

using the Simpson's method.

```
# include <math.h>

main()
{
    int j, n;
    float a, b, delx, sum, sumodd, sumeven;
    float fx( float a );

    clrscr();
    puts ("Enter a, b, n");
    scanf ("%f,%f,%d", &a, &b, &n);
    delx = ( b - a ) / n;
    sumodd = fx( a + delx );
    sumeven = 0;
    for ( j = 2; j <= n - 1; ++j )
        if ( j % 2 != 0 )
            sumodd += fx( a + delx * j );
        else
            sumeven += fx( a + delx * j );
    sum = (delx / 3) * (fx(a) + 4 * sumodd + 2 * sumeven + fx(b));
    printf ("Lower limit = %9.6f\n", a );
    printf ("Upper limit = %9.6f\n", b );
    printf ("No of strips = %3d\n\n", n );
    printf ("Integral value = %10.7f", sum );
}

float fx( float x )
{
    return (log ( 5 - 4 * cos (x) ) );
}
```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

```
Enter a, b, n
0, 3.141593, 50

Lower limit = 0.000000
Upper limit = 3.141593
No of strips = 50

Integral value = 4.3551722
```

exact integral value က 4.3551723 ဖြစ်တာမို့ error က 0.0000001 ပဲရှိပါတယ်။ Simpson formula က rectangular rule ထက် ပိုကောင်းတာကို နှင့် ယျွှေ့ကြည့်ရင် သိနိုင်ပါတယ်။ Simpson formula နဲ့ ပတ်သက်တဲ့ သခံဗျာရှင်းလင်းချက်တွေကို Numerical method စာအုပ်တွေမှာ လေ့လာကြည့်ရင် နားလည်သွားမှာပါ။

4.2 Accessing a Function

function တစ်ခုကို access လုပ်ချင်တယ်ဆိုလို့ရှိရင် အဲဒီ function ရဲ့နာမည်ကို အရင်ရေးပြီးတော့ ဘူးနောက်ကကွင်းထဲမှာ argument တွေကို တန်ဖိုးပြီး (ကြားမှာ comma ခံပြီးတော့) ရေးပေးရပါတယ်။ ကွင်းထဲမှာ argument တစ်လုံးမှ မရှိဘဲလည်း ထားလို့ရပါတယ်။ main function ထဲကနေ call ခေါ်တဲ့ argument တွေကို actual arguments လို့ ခေါ်ပါတယ်။ main function အပြင်က function တွေထက် argument တွေကိုတော့ formal arguments လို့ ခေါ်တာပါပဲ။

EXAMPLE 4.8: Raising Power to a Number

Write a C program to create the function power () to compute the value of integer x raised to the yth power.

```

#include <math.h>

main ()
{
    long int x, y;
    long int power ( long int x, long int y);

    clrscr ();
    puts ("Enter x and y");
    scanf ("%ld,%ld", &x, &y);
    printf ("%ld to the power of %ld = %ld\n",
            x, y, power ( x, y));
}

longint power ( long int a, long int b)
{
    long int temp = 1;

    for ( ; b > 0; b--) temp *= a;
    return temp;
}

```

www.foxitssoftware.com

Enter x and y

9,7

9 to the power of 7 = 4782969

တစ်ခါတစ်ရုံ return statement ကို function definition မှာ ထည့်မရေးဘဲ ချွန်ထားလည်းရပါတယ်။ အဲဒီခါကျရင် ဘာ information မှ မပါဘဲ calling portion of the program ကို return ပြန်သွားမှာပါ။ အခု example လေးကို လေ့လာကြည့်ပါ။

EXAMPLE 4.9:

Selecting a Larger Number

Write a C program that accepts two integer quantities and determines the larger number.

```
main ()
{
    int a, b;

    printf ("Enter two integer numbers\n");
    scanf ("%d, %d", &a, &b);
    larg (a, b);
}

larg (x, y)
int x, y;
{
    int z;

    z = (x >= y) ? x : y;
    printf ("\nLarger number is %d", z);
}
```

ဒါ ပရီဂမ ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါပဲ။

Enter two integer numbers

7, 11

Larger number is 11

EXAMPLE 4.10: Search for a Maximum

Write a C program to find the particular value of x that causes the function $y = x \cos(x)$ to be maximized within the interval bounded by $x = 0$ and $x = \pi$.

```
# include <math.h>

main ()
{
    double a, b, xmax, ymax;
    double search_max ( double a, double b );
    double eqn ( double a );

    clrscr ( );
    puts ( "Enter a, b" );
    scanf ( "%lf,%lf", &a, &b );
    xmax = search_max ( a, b );
    ymax = eqn ( xmax );
    printf ( "\nxmax = %lf\n", xmax );
    printf ( "ymax = %lf", ymax );
}

double search_max ( double a, double b )
{
    int i = 1;
    double xl, xr, yl, yr, eps = 1e-7;
    double eqn ( double xl );

    while ( i++ < 30 ) {
        xl = a + ( b - a - eps ) / 2;
        xr = xl + eps;
        yl = eqn ( xl );
        yr = eqn ( xr );
    }
}
```

```

    if ( yl > yr )  b = xr;
    else a = xl;
    if ( i % 5 == 0 )
        printf ( "i = %3d      a = %lf      b = %lf\n",
                  i, a, b );
    else continue;
}
return a;
}

double eqn ( double x )
{
    return x * cos ( x );
}

```

ဒဲ program ကို runလိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

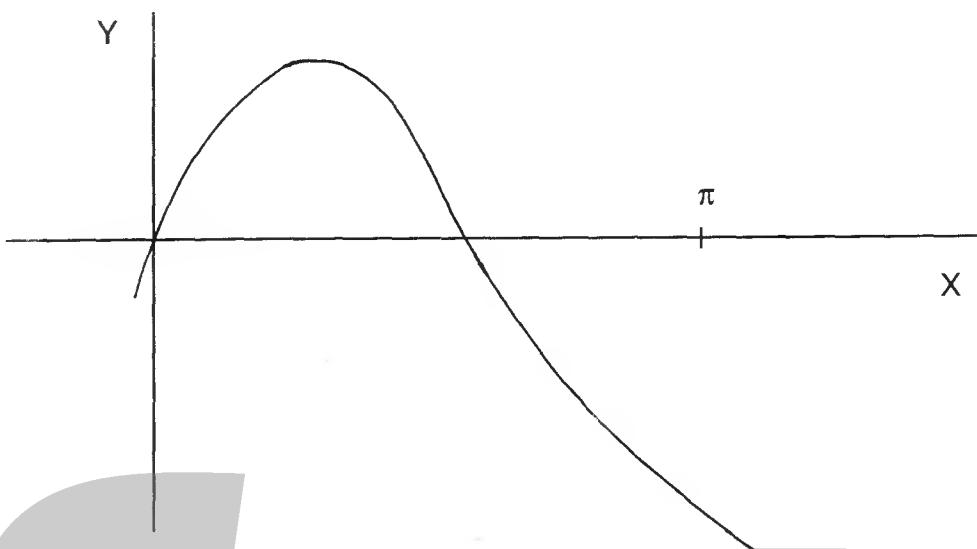
Enter a, b

0, 3.141593

i = 5	a = 0.785398	b = 0.981748
i = 10	a = 0.859029	b = 0.865165
i = 15	a = 0.860180	b = 0.860372
i = 20	a = 0.860330	b = 0.860336
i = 25	a = 0.860334	b = 0.860334
i = 30	a = 0.860334	b = 0.860334

xmax = 0.860334

ymax = 0.561096



ဒါ program ရဲ့ ရည်ရွယ်ချက်ကတော့ $y = x \cos(x)$ ဆိတဲ့ function တစ်ခုရဲ့ $x = 0$ နဲ့ $x = \pi$ interval အတွင်းမှာ အကြံးဆုံးဖြစ်နေမယ့် y_{\max} တန်ဖိုးကို ရှာပေးမှာပါ။ ပြီးတော့ရင် အဲဒီ y_{\max} ရှိတဲ့ x_{\max} နေရာကိုလည်း ရှာပေးမှာ ဖြစ်ပါတယ်။ ဒါ program မှာ function (၃) ခု ပါပါတယ်။

၁။ main function ထဲမှာ a နဲ့ b တို့ရဲ့ တန်ဖိုးတွေကို တောင်းတဲ့အခါမှာ $a = 0$, $b = 3.141593$ လို့ ကျွန်တော်တို့ ထည့်ပေးလိုက်ပါတယ်။ search_max (a, b) လို့ function call ခေါ်လိုက် ပါတယ်။ ဒါ function type ကို အပေါ်မှာ declare လုပ်ထားပုံကို လေ့လာကြည့်ပါ။ search_max function ကနေ main ကို ပြန်ရောက်လာတဲ့အခါမှာ y_{\max} ရှိတဲ့ x position ကို အဖြေ သယ်လာပြီလေး။ အဲဒီ x တန်ဖိုးကို x_{\max} လို့ assign လုပ်ပါတယ်။ ပြီးတော့ရင် eqn (x_{\max}) ကို function call ခေါ်ပါတယ်။ ဒါ function type ကိုလည်း အပေါ်မှာ declare လုပ်ထားတာကို တွေ့တယ်မဟုတ်လား။ function eqn () ကနေ main ကို ပြန်ရောက်လာတဲ့အခါမှာ maximum y တန်ဖိုးကို သိသွားပါပြီ။ အဲဒါကို y_{\max} နဲ့ assign လုပ်ပေးပါတယ်။ x_{\max} , y_{\max} တို့ရဲ့ တန်ဖိုးတွေကို print လုပ်ပြီးရင် program ပြီးသွားပြီ ဖြစ်ပါတယ်။

၂။ search_max function ရဲ့ အလုပ်ကတော့ a နဲ့ b ကို သိရင် x_{\max} ကိုရှာပေးမှာပါ။ ပထမ x_l နဲ့ x_r တို့ကို ရှာပါတယ်။ $x_l = a + \frac{b - a}{2} - \frac{\text{eps}}{2} = a + \frac{b - a - \text{eps}}{2}$ နဲ့
 $x_r = x_l + \text{eps}$ ဖြစ်ပါတယ်။ x_l နဲ့ x_r ကို သိရင် y_l နဲ့ y_r တို့ကို function eqn () ကို call ခေါ်ပြီးတော့ ရှာလိုက်ပါပြီ။ y_l နဲ့ y_r ကို သိသွားတဲ့အခါမှာ y_l က y_r ထက် ကြီးလားလို့ မေးပါတယ်။ ဟုတ်တယ်ဆိုရင် $b = \pi$ အစား $b = x_r$ လို့ ပြောင်းပေးမှာပါ။ အဲလိုမဟုတ်ဘူးဆိုရင် $a = 0$ အစား $a = x_r$ လို့ ပြောင်းပါမယ်။ for loop ကို အကြိမ် 30 ပတ်နိုင်းပါတယ်။ ဒါပေမယ့် a နဲ့ b တို့ရဲ့ ပြောင်းနေတဲ့ အဖြေတွေကိုတော့ 5 ကြိမ်ခြားပြီးတော့ အဖြေထွက်ထားပါတယ်။

အခါနလိုဖြစ်အောင် continue အသုံးပြထားတာကို လေ့လာကြည့်ပါ။ 25 ကြိမ်မြောက်မှာ a နဲ့ b တူသွေးတာကို တွေ့ရပါတယ်။ ဒါကြောင့်မို့ $x_{\max} = a = b$ လို့ ပြောလို့ရပါတယ်။ main function ဆိတ် ဒီအဖြေ သယ်သွားလို့ရပြီပေါ့။

21
function eqn () ဟာဆိတ်ရင် search_max function နဲ့ main function တို့မှာ ပါဝင်တဲ့ statement တွေမှာ လိုသလို ခေါ်သူ့နေတာကို တွေ့ရမှာပါ။ ဥပမာ $yl = \text{eqn}(xl)$, $yr = \text{eqn}(xr)$ နဲ့ $ymax = \text{eqn}(xmax)$ တို့ ဖြစ်ကြပါတယ်။

4.J Defining Own Constants

program တစ်ခုမှာပါဝင်တဲ့ constant တွေကို စစချင်းမှာပဲ define လုပ်ထားမယ်ဆိုရင် ဒါ program ဟာ ကြည့်လို့လွယ်သလို ပြန်ပြင်လို့လည်းလွယ်မှာပါ။ ဒီနည်းကိုသုံးပြီးတော့ စက်ရိုင်းတစ်ခုရဲ့ ရေးပြထားပါတယ်။ လေ့လာကြည့်ပါ။

EXAMPLE 4.11: Defining Constants

Write a C program that uses the preprocessor directives defining PI and square (x) to compute the area of a circle.

```
# define square(x) x*x
# define PI    3.141593
main ()
{
    float      radius, area;
    printf( "Enter the radius ? " );
    scanf( "%f", &radius );
    area = PI * square( radius );
    printf( "The area of circle is %f", area );
}
```

ဒီ program မှာ square (x) ဆိတဲ့ ဖန်ရှင်ကို x^2 နဲ့ညီတယ်လို့. define အရင်လုပ်ထားပါတယ်။
 ပြီးတော့ PI တန်ဖိုးကိုလည်း 3.141593 နဲ့ ညီတယ်လို့. define လုပ်ပါတယ်။ ဒါကြောင့်မို့. area =
 PI * square (radius) ဆိတဲ့ assignment statement ကို တွေ့တဲ့အခါမှာ area = 3.141593 *
 radius * radius ဆိုပြီးတော့ computer က ပြောင်းတွက်သွားမှာပါ။ ဒီ program ကို run လိုက်မယ် ဆိုရင်
 computer မှာ ဒီလိုပေါ်လာမှာပါ။

**Enter the radius ? 5.5
The area of circle is 95.033188**

ဒီ program ကိုပဲ macros နဲ့ mathematical operation အကုန်လုံးကို ပြောင်းရေးလို့ ရပါတယ်။
 လေ့လာကြည့်ပါ၍။

```
# define     square ( x )      .x * x
# define     PI                 3.141593
# define     circle_area ( r )   PI * square ( r )

main ()
{
    float    radius, area;

    printf ( "Enter the radius ? " );
    scanf ( "%f", & radius );
    area = circle_area ( radius );
    printf ( "The area of circle is %f", area );
}
```

ဒီ program ကို စောစောက data ကိုပဲထည့်ပြီး run လိုက်မယ်ဆိုရင် စောစောကလို့ အမြဲ
 ထွက်လာမှာပါ။

၄.၃ Creating Own Header Files

C language မှာ ကျွန်တော်တိ၊ ကိုယ်ပိုင် header file တွေတည်ဆောက်လို ရပါတယ်။ အဲဒီ file တွေကို program မှာ preprocessor macro အနေနဲ့ထည့်ရေးရမှာပါ။ ကျွန်တော်တိ၊ အလွယ် သိချင်တဲ့ information တွေကို header file တွေဖွင့်ပြီး library လုပ်ထားရင်တောင် ရပါတယ်။

EXAMPLE 4.12: Here is a C program that demonstrates the development of our own header file that is used to compute the square and cube of a number entered from the keyboard.

```
# include "head.h"
main ()
{
    float num, total;

    printf ("Enter a number ? ");
    scanf ("%f", &num);
    total = sqr ( num );
    printf ("The square of %f is %f\n", num, total );
    total = cube ( num );
    printf ("The cube of %f is %f", num, total );
```

ဒါ program မှာ head.h ဆိုတဲ့ header file ထဲကို ကျွန်တော်တိ၊ program တွေမှာသုံးမယ့် # define statement တွေကို ရေးထည့်ပြီးတော့ save လုပ်ထားတာပါ။

```
# define sqr ( x )      x * x
# define cube ( x )     x * x * x
```

သတိပြုရမယ့် အချက်တစ်ချက်က header file ရေးတဲ့အခါမှာ ရေးနေကျ ဒီ (< >) သင်္ကာတအစာ: ဒီသင်္ကာတ (" ") ကိုပဲသုံးပါလို့ သတိပေးချင်ပါတယ်။ စွာစွာက program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလို့ ပေါ်လာမှာပါ။

```
Enter a number ? 5
The square of 5.000000 is 25.000000
The cube of 5.000000 is 125.000000
```

၄.၄ Passing Argument by Value

value တစ်ခုကို function ထဲ pass လုပ်ရာမှာ ပထမအဆင့်က actual argument ရဲ့ copy ပိုပါတယ်။ function ထဲကျတော့ ဒီ argument ကို ဤောက်သလိုတွက်ရင်:ချက်ရင်: ထွက်ချင်တဲ့အဖြေတွေ အသစ်ထပ်ထွက်လာမှာပါပဲ။ argument ရဲ့မူလတန်ဖိုးတွေဟာ function ထဲမှာ ပြောင်းသွားနိုင်ပါတယ်။ ဒါပေမယ့် calling routine က argument ရဲ့တန်ဖိုးကတော့ လုံးဝမပြောင်းပါဘူး။ ဒီလိုနည်းနဲ့ function ထဲကို argument pass လုပ်တာကို passing by value လို့ခေါ်ပါတယ်။

EXAMPLE 4.13: Passing Arguments

Here is a simple C program containing a function that alters the value of its argument.

```
main ()
{
    int      x = 10;

    printf ("\nX = %d from main before calling the function", x );
    modify (x);
    printf ("\n\nX = %d from main after calling the function", x );
}
```

```

modify ( x )
Int x;
{
    x * = 3;
    printf ("\n\nX = %d from the function after being modified", x );
}

```

X = 10 from main before calling the function

X = 30 from the function after being modified

X = 10 from main after calling the function

ဒါ program မှာ X = 10 လို့, main မှာ ပြန်လည်သာထားပါတယ်။ actual argument ရဲ့တန်ဖိုးကို 10 အနေနဲ့ function modify ထဲကို pass ဝင်သွားပါတယ်။ ဒါ function ထဲမှာ X = 30 ဖြစ်သွားပါတယ်။ ဒါပေါ်မယ့် main ကို ပြန်ရောက်တဲ့အခါကျတော့ X = 10 ပြန်ဖြစ်သွားပါတယ်။ ဒီလိုနည်းနဲ့, argument pass လုပ်တာကို passing by value လို့ခေါ်ပါတယ်။

EXAMPLE 4.14: Runge - Kutta Method

Write a C program to approximate the initial value problem.

$$\frac{dx}{dt} = 2 + (x - t - 1)^2$$

$$x(1) = 2$$

whose exact solution is $x(t) = 1 + t + \tan(t - 1)$ using the Runge - Kutta method of order 4. Its formulas are as follows:

$$x(t+h) = x(t) + (F1 + 2 * F2 + 2 * F3 + F4) / 6$$

where

$$F1 = h * F(t, x)$$

$$F2 = h * F(t + h/2, x + F1/2)$$

$$F3 = h * F(t + h/2, x + F2/2)$$

$$F4 = h * F(t + h, x + F3)$$

```

#include <stdio.h>
#include <math.h>
main ()
{
    int      i, nstep = 72;
    float    t = 1, x = 2, h = 0.0078125, h2, start, dprt, prt, exact;
    float    dif = 0, F1, F2, F3, F4;
    double   rk4 ( double t, double x );

    printf (" T          X          ERROR\n");
    printf ("%10.8lf  %10.8lf  %10.8lf\n", t, x, dif );
    h2 = h / 2;
    start = t;
    drpt = 4 * h;
    prt = start + dprt;
    for ( i = 1; i <= nstep; ++i ) {
        F1 = h * rk4 ( t, x );
        F2 = h * rk4 ( t + h2, x + F1 / 2 );
        F3 = h * rk4 ( t + h2, x + F2 / 2 );
        F4 = h * rk4 ( t + h, x + F3 );
        x += ( F1 + 2 * F2 + 2 * F3 + F4 ) / 6;
        exact = 1 + t + tan ( t - 1 );
        dif = x - exact;
        t = start + h * i;
        if ( ( t - prt ) >= 0 )
            printf ("%10.8lf  %10.8lf  %10.8lf\n", t, x, dif );
        prt += dprt;
    }
}

double rk4 ( double t, double x )
{
    double   z;

    z = 2 + ( x - t - 1 ) * ( x - t - 1 );
    return ( z );
}

```

ဒဲ program တာ first order ODE (initial value problem) တစ်ခုကု Hungre-Kutta (4th order) method နဲ့ solveလုပ်တဲ့ programပါ။ T အမျိုးမျိုးအတွက် X တန်ဖိုးတွေကို တွက်ပေးထားပါတယ်။ approximateနဲ့ exact solution တို့ ကွာခြားမှုကို ERRORနဲ့ နိုင်းယဉ်ပေးထားပါတယ်။

T	X	ERROR
1.00000000	2.00000000	0.00000000
1.03125000	2.06251018	0.01563088
1.06250000	2.12508151	0.01565192
1.09375000	2.18777563	0.01568844
1.12500000	2.25065514	0.01574073
1.15625000	2.31378411	0.01580920
1.18750000	2.37722861	0.01589441
1.21875000	2.44105728	0.01599705
.....
.....
.....

၄.၅ Recursion

recursion ဆိုတာ function တစ်ခုကနေ same function ထဲကို ထပ်ခါတစ်လဲ ပြန် call ခေါ်ပြီးထော့ သတ်မှတ်ထားတဲ့ condition တစ်ခုမရောက်သေးမချင်း looping ပတ်နေတာကိုဆိုလိုပါတယ်။

EXAMPLE 4.15: Calculating Factorials

Write a C program to calculate the factorial of a positive integer using recursion.

```
main ()
{
    int n;
    long double facto ( int n );

    printf ( "Enter a number ? " );
    scanf ( "%d", &n );
    printf ( "Factorial of %d is %Lf", n, facto ( n ) );
}
```

```

Long double      facto ( int n )
{
    if ( n <= 1 ) return ( 1 );
    else return ( n * facto ( n - 1 ) );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter a number ? 6
Factorial of 6 is 720.000000

recursive program တစ်ခုကို run တဲ့အခါမှာ recursion function call တွေဟာချက်ခြင်း execute လုပ်ပြီး အဖြေထုတ်မပေးပါဘူး။ အဖြေတွေကို (တိုင်တစ်တိုင်မှာ အပေါက်ပါတဲ့ သံပြားအစိုင်းတွေကို ထည့်စွပ်ထား ပေးသလို) ပြောင်းပြန်ထပ်စီသွားပါတယ်။ သတ်မှတ်ထားတဲ့ condition တစ်ခုကို ရောက်လာလို့ recursion ပုံသွားတော့မှ အဖြေတွေကို (တိုင်ကနေ သံပြားစိုင်းတွေ တစ်ခုချင်း ထုတ်ပေးသလို) ပြောင်းပြန် အစီအစဉ်နဲ့ computer က ထုတ်ပေးသွားမှာပါ။ ဒါဟာ recursion ရဲ့ ထူးခြားတဲ့ ဂိမ်းပေါ်မယ်။ ဒီပုံစံကို ဘယ်လို ရှင်းသွားတယ်ဆိုတာ ကျွန်ုတ် အစီအစဉ်ချုပြုးတော့ ရေးပြပါမယ်။ လေ့လာကြည့်ပါ။

$n! = n \times (n-1)!$
 $(n-1)! = (n-1) \times (n-2)!$
 $(n-2)! = (n-2) \times (n-3)!$
 $(n-3)! = (n-3) \times (n-4)!$

 $\dots\dots\dots$
 $2! = 2 \times 1!$

ပထမ function call မှာ $n!$ ကို $n \times (n-1)!$ လို့သတ်မှတ်ပြီး execute မလုပ်သေးဘဲ တိုင်မှာ စွပ်လိုက်ပါတယ်။ ဒီဟာတိုင်မြဲအောက်ဆုံးမှာ ရှုနေမှာပဲ။ ပြီးတော့ ဒုတိယအကြိမ် function call ကိုခေါ်တဲ့အခါ argument က $(n-1)$ ဖြစ်သွားပါပြီ။ ဒီတော့ $(n-1)!$ ကိုလည်း $(n-1) \times (n-2)!$ လို့ သတ်မှတ်ပြီးတော့ execute မလုပ်သေးဘဲ တိုင်မှာပဲထပ်စွပ်ပြန်ပါတယ်။ ဒီလိုနည်းနဲ့ recursion ကို ဆက်လုပ်သွားတာ argument က အနုတ်ဖြစ်တာနဲ့ process ဟာ ရပ်သွားပြီးတော့ actual value တွေကို reverse order နဲ့ computer က တွက်ထုတ်ပေးပါတော့တယ်။

$1! = 1$
 $2! = 2 \times 1! = 2 \times 1 = 2$
 $3! = 3 \times 2! = 3 \times 2 = 6$
 $4! = 4 \times 3! = 4 \times 6 = 24$
.....
 $n! = n \times (n - 1)! = ...$

EXAMPLE 4.16: Printing Backward

Write a C program that reads a line of text on a char-by-char basis and then writes them in reverse order.

```

#include <stdio.h>
#define EOL '\n'

main ()
{
    void reverse ( void );
    puts ( "Enter a line of text below" );
    reverse ( );
}

void reverse ( void )
{
    char ch;

    if ( ( ch = getchar ( ) ) != EOL ) reverse ( );
    putchar ( ch );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် ကျွန်တော်တိ၊ ရေးထည့်တဲ့စာကြောင်းကို ပြောင်းပြန်အနေနဲ့ computer မှာ ပေါ်လာမှာပါ။

Enter a line of text below

Summer is hot but winter is cold

dloc si retniw tub toh si remmuS

EXAMPLE 4.17: Future Populations

Suppose that a country of 45 million people is experiencing an annual population growth rate of 2% and a steady immigration of 20,000 people each year. Write a C program to estimate the total population after 10 years.

```
int          n = 10, r = 2;
long int      p = 45000000, x = 20000;

main ()
{
    long int pop ( int n );

    clrscr ();
    printf (" \n The population will be %ld millions",
            pop (n) / 1000000 );
}

long int      pop ( int n )
{
    if ( n == 0 )
        return (p);
    else
        return (( 1 + r / 100.) * pop (n - 1 ) + x );
}
```

ဒါ program ၊ run လိုက်စယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

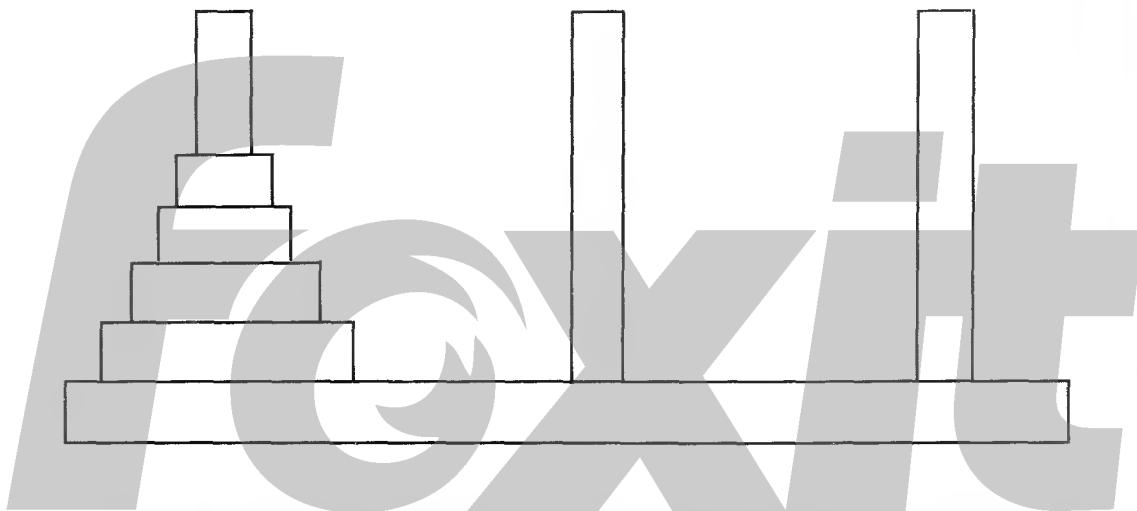
The population will be 55 millions

EXAMPLE 4.13:

The Towers of Hanoi

The towers of Hanoi is a wellknown children's game played with three poles and a number of different sized disks. Each disk has a hole in the center, allowing it to be stacked around any of the poles. Initially the disks are stacked on the leftmost pole in the order of decreasing size.

The object of the game is to transfer the disks from the leftmost pole to the rightmost pole, without ever placing a larger disk on top of a smaller disk. Only one disk may be moved at a time.



```
#include <stdio.h>
main ()
{
    void    transfer ( int, char, char, char );
    int     n;

    printf ( "Welcome to the TOWERS OF HANOI \n\n" );
    printf ( "How many disks ? " );
    scanf ( "%d", &n );
    printf ( "\n" );
    transfer ( n, 'L', 'R', 'C' );
}
```

```

void transfer( int n, char from, char to, char temp )
{
    if( n > 0 ) {
        transfer( n - 1, from, temp, to );
        printf( "Move disk %d from %c to %c\n", n, from, to );
        transfer( n - 1, temp, to, from );
    }
    return;
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် disk တွေမျှနည်းကို computer မှ ဒီလိုပေါ်လာမှာပါ။

Welcome to the TOWERS OF HANOI

How many disks ? 3

Move disk 1 from L to R

Move disk 2 from L to C

Move disk 1 from R to C

Move disk 3 from L to R

Move disk 1 from C to L

Move disk 2 from C to R

Move disk 1 from L to R

PROGRAM STRUCTURE

program တစ်ခုမှာပါဝင်တဲ့ variable တွေကို အမျိုးအစား ခွဲပြီး ဖော်ပြတဲ့နည်း (၂) နည်းရှုပါတယ်။ data type နဲ့ ခွဲခြားဖော်ပြတဲ့နည်းရယ်၊ storage class နဲ့ ခွဲခြားဖော်ပြတဲ့နည်းရယ်လို့ ရှုပါတယ်။ data type ဆိတာကတော့ variable တွေဟာ integer number လား၊ floating-point number လား၊ ဒါမှမဟုတ် character လား၊ ဆိတာကို ခွဲခြားဖော်ပြတာပါ။ storage class ဆိတာကတော့ ဒါ variable တွေ program ထဲမှာ ဘယ်လောက်တည်ပြုလဲ၊ သူတို့ တွေရဲ့ ဉာဏ်ရောက်မှုက ဘယ်လောက်ကျယ်လဲဆိတာကို ကြည့်ပြီး ခွဲခြားဖော်ပြတာပါပဲ။ C language မှာ storage class (၄) မျိုးရှုပါတယ်။ အဲဒါတွေကတော့ automatic၊ external၊ static နဲ့ register တို့ ပါပဲ။ သူတို့ တွေအတွက် keyword တွေက auto၊ extern၊ static နဲ့ register တို့ဖြစ်ပါတယ်။ ဒီအခန်းမှာ register storage class ကလွှဲရင် အားလုံးကို ဆွဲ့နွဲ့တင်ပြမှာပါ။

www.foxitsoftware.com ၅.၁ Automatic Variables

automatic variable ဆိတာ function တစ်ခုတည်းနဲ့ပဲ သက်ဆိုင်တဲ့ local variable တစ်ခုကို ခေါ်တာပါ။ function တစ်ခုထဲက variable တစ်ခုဟာ နောက် function တစ်ခုထဲက variable နဲ့ နာမည်ချင်းတူနေဖတော် ဘာမှမဆိုင်ပါဘူး။ သူတို့ တွေရဲ့ scope ဟာ ကိုယ်ရောက်နေတဲ့ function နဲ့ပဲ သက်ဆိုင်ပါတယ်။ function တစ်ခုမှာပါဝင်တဲ့ variable တွေရဲ့ storage class ကို ဘာမှာကြညာမထားရင် စက်ကအလိုလို automatic variable တွေလို့ သတ်မှတ်မှာပါ။ ဒါကြောင့်မို့ ရှေ့အခန်းတွေမှာ ဖော်ပြခဲ့တဲ့ example တွေက variable တွေအားလုံးဟာ automatic variable တွေပေါ့။ auto လို့ ထည့်ကြညာရင်လည်း အမှားမဖြစ်ပါဘူး။

EXAMPLE 5.1: Calculating Factorials

Write a C program to calculate factorials.

```
main ()  
{  
    auto int    n, i;  
    long int   facto ( int n );  
  
    for ( i = 1 ; i <= 3 ; ++i ) {  
        printf ( "\nEnter n : " );  
        scanf ( "%d", &n );  
        printf ( "\n%d != %ld", n, facto ( n ) );  
    }  
}  
  
long int facto ( int n )  
{  
    auto    int i;  
    auto    long int fac = 1;  
  
    if ( n > 1 )  
        for ( i = 2; i <= n ; ++i ) fac *= i;  
    return ( fac );  
}
```

www.foxitsoftware.com

ဒီ program ကို လေ့လာကြည့်မယ်ဆိုရင် automatic variable တွေကို initialize လုပ်ပေးလို ရတယ်ဆိုတာ တွေရမှာပါ။ ဒါပေမယ့် automatic variable တွေဟာ defined function ကနဲ အဖြင့် ထွက်သွားပြီဆိုရင် နဂါး value ကို ဆက်တည်မြို့အောင် မထိန်းနိုင်တော့ပါဘူး။ အရှင်းဆုံး ဥပမာပြုမယ်ဆိုရင် အခါ program မှာပဲ main function က (i) ရဲ့ value ဟာ အစက ဘယ်လိုပုဂ္ဂနေပါစေ facto ဆိုတဲ့ function ထဲကို ရောက်သွားတဲ့အခါမှာ ဒုတိယတွေတဲ့ (i) နဲ့ ပထမ (i) ဟာ ဘာမှ မပတ်သက်တော့ပါဘူး။ နာမည်ချင်းတူပေမယ့် ကိုယ့်လုပ်ပိုင်ခွင့် ကိုယ်ရှိနေကြတယ်လေ။ ဒါဟာ automatic storage ရဲ့သဘောပါပဲ။ ကျွန်ုတ်တို့ ဒီ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter n : 7

7 ! = 5040

Enter n : 10

10 ! = 3628800

Enter n : 12

12 ! = 479001600

EXAMPLE 5.2: Average Length of Several Lines of Text.

Write a C program that will read several lines of text and determine the average number of characters (including punctuation and blank spaces) in each line.

```
main ()
{
    int    n;          /* no of chars in a given line */
    int    count = 0;   /* total no of lines */
    int    sum = 0;     /* total no of characters */
    float avg;         /* avg no of chars per line */
    int    linecount ( void );

    printf ("Enter the text below \n");
    while ( ( n = linecount () ) > 0 ) {
        sum += n;
        ++count;
    }
    avg = ( float ) sum / count;
    printf ("Avg no of characters per line : %5.2f", avg );
}
```

```

int linecount ( void )
{
    char    line[80];
    int     count = 0;

    while ( ( line[count] = getchar ( ) ) != '\n' )
        ++count;
    return ( count );
}

```

ဒါ program ကို ဆလဲလာကြည့်မယ်ဆိုရင် main function က count ဆိုတဲ့ variable ဟာ ကျွန်တော်တို့ input ထည့်လိုက်တဲ့ paragraph က အင်္ဂလိပ်စာကြောင်း အရေအတွက်ကို ဆိုလိုပါတယ်။ ဒါပေမယ့် linecount function က count နဲ့ main က count ဟာ နာမည်ချင်းတူပေမယ့် အလုပ် သဘောချင်း ဘာမှမဆိုင်ပါဘူး။ သူ့အလုပ်က ကျွန်တော်တို့ input ထည့်လိုက်တဲ့ အင်္ဂလိပ်စာ တစ်ကြောင်းချင်းက character အရေအတွက် စုစုပေါင်းကို သိမ်းမှာပါ။ ဒါဟာ automatic storage ရဲ့သော့သူပါပဲ။ နောက်ပြားစရာ တစ်ခု ကျွန်သေးတာက avg ပါ။ avg ဟာ float လို့ အစဉ်းမှာ ကြည်းသားပါတယ်။ avg ကို ရှာဖို့ အတွက် sum ကို count နဲ့ စားပါတယ်။ ဒါပေမယ့် sum ရေား count ပါ integer တွေချည်းဖြစ် နေတာမို့ တစ်ခုခုကို float ဖြစ်အောင်လို့ type cast လုပ်ပေးရမယ်ဆိုတာကို မမေ့ပါနဲ့။

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှာပါ။

Enter the text below

Now is the time for all good men

to come to the aid of their country.

Avg no of characters per line : 34.00

ဒါ program ကိုပဲ ပိုပြီးကျစ်လျစ်အောင် linecount function ကို ဒီလိပ်ပြောင်းရေးလို့ ရပါကယ်။ လေ့လာကြည့်ပါ။

```
int linecount (void)
{
    int count = 0;

    while ( getchar () != '\n' ) ++count;
    return ( count );
}
```

EXAMPLE 5.3: Area of a Circle by Integration

Write a C program that calculates the approximate area of a circle by finding the sum of areas of rectangles.

```
# include < math.h >
# define PI      3.141593

main ()
{
    int n;
    double area, r;
    double circle ( double r, int n );

    printf ( "Enter R, N : " );
    scanf ( "%lf, %d" , &r, &n );
    area = circle ( r, n );
    printf ( "Approximate area = %lf \n", area );
    printf ( "Exact area       = %lf", PI * r * r );
}

double circle ( double r, int n )
{
    double h, h2, x, area = 0;
```

```

h = r/n;
h2 = h/2;
for ( x = h2; x <= r - h2; x += h)
    area += h * sqrt( r * r - x * x );
return ( 4 * area );
}

```

ဒဲ program ဟာ စက်ခိုင်းတစ်ခုရဲ့ ရရှိယာကို $\text{area} = \pi r^2$ ဆိုတဲ့ ပုံသေနည်းကိုမသုံးဘဲ Numerical Integration Method ကိုသုံးပြီးတွက်တဲ့နည်းပါ။ ဘယ်လိုတွက်သွားလဲ ဆိုတာကိုတော့ အသေးစိတ်ရှင်းမပြတော့ ပါဘူး။ အသေးစိတ်သိသုတေသနဆိုရင် EASY WAY TO PROGRAMMING IN FORTRAN 77 စာအုပ်က စာမျက်နှာ (၉၅) EXAMPLE 4.8 ကို လေ့လာပါ။ ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလို ပေါ်လာမှာပါ။

Enter R, N : 5,1500
 Approximate area = 78.539965
 Exact area = 78.539825

၁၂ External Variables

www.foxitsoftware.com

external variables ဆိုတာ automatic variable တွေနဲ့ လုံးဝမတူလဲဆိုတော့ သူတို့တွေဟာ automatic variable တွေလို function တစ်ခုထဲမှာပဲ confine လုပ်ခံမနေရပါဘူး။ external variable တွေရဲ့ scope ဟာ သူတို့ကို define စလုပ်တဲ့ နေရာကစပြီးတော့ရင် program အသုံး ထိတောင်ကျယ်ဝန်းပါတယ်။ ဒါကြောင့်လို့လို့ ဒီ variable တွေဟာ function (j) ခု (r) ခု စာကန် program ကြီးတစ်ခုလုံးကိုတောင် span လုပ်နိုင်တာပါ။ external variable တွေရဲ့ အထူးခြားဆုံး အရည်အချင်းကတော့ function တွေကြားမှာ ကူးကလာဖြန့်ပြီး information transfer လုပ်ပေးနိုင်တဲ့ အချက်ပါပဲ။ အထူးသဖြင့် argument မျိုးတွေမသုံးဘဲနဲ့ function ထဲကို information transfer လုပ်နိုင်တဲ့ ဂုဏ်သွေးထူးတွေ သူတို့မှာ ရှိနေပါတယ်။ ရှေ့အခန်းတွေမှာ အသုံးများခဲ့တဲ့ return statement ဟာဆိုရင် data item တစ်ခုပဲ return လုပ်ပေးနိုင်ပေမယ့် external variable တွေကျတော့ data item အများကြီးကို တစ်ချိန်တည်း data transfer လုပ်ပေးနိုင်ပါတယ်။ ဒါဟာ external variable တွေရဲ့ သဘောတရားပါပဲ။

EXAMPLE 5.4:**Search for a Maximum.**

Write a C program to find the particular value of x that causes the function $y = x \cos(x)$ to be maximized within the interval bounded by $x = 0$ and $x = \pi$.

```
# include < math.h >
# define EPS 1e - 7
double a, b, xl, xr, yl, yr;

main ()
{
    double xmax, ymax;
    void squeeze ( void );
    double equation ( double xmax );

    printf ( "Enter a, b : \n" );
    scanf ( "%lf, %lf", &a, &b );
    do
        squeeze ( );
    while ( ( yl != yr ) && ( ( b - a ) > EPS ) );
    xmax = ( xl + xr ) / 2;
    ymax = equation ( xmax );
    printf ( "\nXMAX = %8.6lf      YMAX = %8.6lf", xmax, ymax );
}

void squeeze ( void )
{
    double equation ( double xl );

    xl = a + ( b - a - EPS ) / 2;
    xr = xl + EPS;
    yl = equation ( xl );
    yr = equation ( xr );
```

```

    if ( yl > yr ) {
        b = xr;
        return;
    }

    if ( yl < yr ) a = xl;
    return;
}

double equation ( double x )
{
    return ( x * cos ( x ) );
}

```

ဒီ program တာဆိရင် $y = x \cos(x)$ ဆိတဲ့ equation ရဲ့ $x = 0$ နဲ့ $x = \pi$ ဆိတဲ့ interval (j) ခုကြားက အကြိုးဆုံး y တန်ဖိုးကို ရှာပေးတဲ့ program ပါ။ ဒီ program ကို run လိုက်မယ်ဆိရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

Enter a, b :
0, 3.141593
XMAX = 0.860334 YMAX = 0.561096

အခိုပါယ်ကတော့ $x \cos(x)$ function ရဲ့ y-direction က maximum value ရှိတဲ့နေရာကို $x = 0.860334$ နေရာမှာလို့ ဖော်ပြုနေတာပါ။ အခုံ program ကို ဒီထက်ပိုတို့အောင် ရေးလို့ရပါသေးတယ်။ declaration လပ်ရမယ့် ဥစ္စာမှန်သမျှကို external variable တွေအနေနဲ့ main function အပြင်မှာ ရေးပစ်လိုက်ရင် ပိုရင်းသွားမှာပါပဲ။ လေ့လာကြည့်ပါ။

```

#include <math.h>
#define EPS 1e - 7
double a, b, xl, xr, yl, yr, xmax, ymax;
double equation ( double xl);
void squeeze ( void );

```

```

main ()
{
    printf ( "Enter a, b : \n" );
    scanf ( "%lf, %lf", &a, &b );
    do
        squeeze ( );
    while ( ( yl != yr ) && ( ( b - a ) > EPS ) );
    xmax = ( xl + xr ) / 2;
    ymax = equation ( xmax );
    printf ( "\nXMAX = %8.6lf      YMAX = %8.6lf", xmax, ymax );
}

void squeeze ( void )
{
    xl = ( a + b - EPS ) / 2;
    xr = xl + EPS;
    yl = equation ( xl );
    yr = equation ( xr );
    if ( yl > yr )  b = xr;
    else    a = xl;
    return;
}

double equation ( double x )
{
    return ( x * cos ( x ) );
}

```

ဒါ program ကို ကျွန်တော်တို့တွေ စောစောက data တွေနဲ့ပဲ run ကြည့်မယ်ဆိုရင် ပထမအဖြေး
 ထွက်လာမှာပါပဲ။ ဘာမြိုလဲဆိုတော့ a, b, xl, xr, yl အစရိတ် variable တွေကို external value
 တွေဖြစ်အောင် လုပ်ထားတာမို့။ သူတို့ရဲ့ value တွေကို main မှာဖြစ်ဖြစ်၊ တစ်ခြား function မှာပဲဖြစ်ဖြစ်
 ဘစ်နေရာရာကနေ ပြောင်းသွားတာနဲ့ program ကြိုးတစ်ခုလုံးမှာ အလိုလိုပြောင်းသွားတာကိုး။ ရင်းအောင်ပြောရရင်
 a နဲ့ b တို့ရဲ့ မူလတန်ဖိုးက 0 နဲ့ π ပါ။ main function ကနေ squeeze function လဲလည်း

ရောက်သွားရော a သို့မဟုတ် b တစ်ခုခုရဲ့ value ဟာပြောင်းလွှဲသွားပါတယ်။ ဒီလိုပြောင်းသွားတဲ့ a + b တို့ရဲ့တန်ဖိုးအသစ် တွေကို main ဆီ transfer လုပ်လာစေချင်ရင် return လို့ ရေးလိုက်ရဲ့နဲ့ data တွေ ဒီဘက်ရောက်လာမှုပါ။ return (a) + return (b) လို့ရေးစရာ မလိုပါဘူး။ ဘာပြုလို့လဲဆိုတော့ a + b တို့ဟာ external variables လို့ စက်က အလုံလုံသိနေတာကို။

EXAMPLE 5.5: Average Length of Several Lines of Text

Write a C program to modify the program presented in EX 5.2.

```

int n, sum = 0, lines = 0;
float avg;
int linecount ( void );
main ()
{
    printf ( "Enter the text below\n" );
    while ( ( n = linecount () ) > 0 ) {
        sum += n;
        ++ lines;
    }
    avg = ( float ) sum / lines;
    printf ( "Avg no of characters per line : %5.2f", avg );
}
int linecount ( void )
{
    int count = 0;
    while ( ( getchar ( ) != '\n' ) ++ count;
    return ( count );
}

```

ဒါ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှုပါ။

Enter text below :

Life is fragile.

Please handle with care.

Avg no of character per line : 20.00

EXAMPLE 5.6: Evaluating Gamma Function.

Write a C program to evaluate the value of a Gamma function which is defined by the integral

$$\Gamma(n) = \int_0^\alpha x^{n-1} e^{-x} dx$$

Gamma function ဟာ engineering problem တွေမှာအသုံးပင်ပါတယ်။ ဘာပြုလိုလဲဆိုတော့ Bessel's equation တွေကို Gamma function နဲ့ဖြေရှင်းလို့ရတာကိုး။ ပြီးတော့ Gamma function မှာ recursive nature ရှိတာကြောင့်မို့လို့ factorial ရှာတဲ့အခါမှာ အသုံးပင်ပါတယ်။ အထူးသဖြင့် real ကဏ္နားတွေ၊ အနုတ်လက္ခဏာပါတဲ့ ကိန်းတွေရဲ့ factorial ကို ရှာချင်ရင် Gamma function နဲ့ ရှာလို့ရပါတယ်။ gamma function ရဲ့ recursive relationship က ဒီလိုပါ။

$$\Gamma(n+1) = n \Gamma(n)$$

$$\Gamma(2) = 1 \Gamma(1) = 1 = 1!$$

$$\Gamma(3) = 2 \Gamma(2) = 2(1!) = 2!$$

$$\Gamma(4) = 3 \Gamma(3) = 3(2!) = 3!$$

$$\dots\dots\dots$$

$$\Gamma(n+1) = n \Gamma(n) = n!$$

လို့, ရေးလို့ရပါတယ်။ ရိုးရိုး factorial ရှာတဲ့အခါမှာဆိုရင် negative တို့၏ real argument ဘို့တွေအတွက် တန်ဖိုးရှာလို့မရပါဘူး။ ဥပမာ (- 0.5) ! ကိုရှာပါဆိုရင် ဘယ်ရှာလို့ရမလဲ။ ဒါပေမယ့်

Gamma function ကိုသုံးရင်တော့ အဖြေရှာလို့ရပါတယ်။ Integration နည်းကိုမသုံးဘဲ Stirling's approximation နည်းကိုပဲသုံးရင် Gamma function ရဲ့ တန်ဖိုးကို ရှာလို့စိုလွယ်ပါတယ်။ တကယ် exact အဖြေနဲ့တော့ နည်းနည်းကွာနိုင်ပါတယ်။ လေ့လာကြည့်ပါ။ Stirling's equation ကတော့ --

$$\Gamma(x) = \sqrt{\frac{2\pi}{x}} \cdot x^x \cdot e^{-x} \text{ ဖြစ်ပါတယ်။}$$

အဲဒီမှာ $y = \frac{1}{12x} - \frac{1}{360x^3} - x$ ဖြစ်ပါတယ်။

ဒါ approximation ကိုသုံးပြီးတော့ $\Gamma(0.5)$ ရဲ့ value ကိုရှာပါဆိုရင် ဒီလိုရှာရမှာပါ။

$$\begin{aligned}\Gamma(0.5) &= \sqrt{\frac{2\pi}{x}} (0.5)^{0.5} - \frac{16}{e^{45}} \\ &= \sqrt{\pi} (\sqrt{2}) (e^{-\frac{16}{45}}) = 0.99\sqrt{\pi} \\ &\approx \sqrt{\pi}\end{aligned}$$

တကယ်တော့ $(-0.5)!$ ဟာဆိုရင် $\Gamma(0.5)$ နဲ့ အတူတူပါပဲ။ ဒါကြောင့်မိ $(-0.5)! = \pi$ ဖြစ်ပါတယ်။

```
#include <math.h>
double x, y, z, pi = 3.141593;
void gamma ( double x, double y );

main ()
{
    printf ("      X      GAMMA (X)\n");
    for ( x = .5; x <= 10; x += .5 ) {
        y = 1 / ( 12 * x ) - 1 / ( 360 * pow ( x, 3 ) ) - x;
        printf ("%.2f %.20f\n", x, y );
    }
}
```

```

        gamma ( x, y );
        printf ( "%10.2lf
    }
}

void gamma ( double x, double y )
{
    z = sqrt ( 2 * pi / x ) * pow ( x, x ) * exp ( y );
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိပ်ပေါ်လာမှပါ။

X	GAMMA (X)
0.50	1.757
1.00	0.999
1.50	0.886
2.00	1.000
2.50	1.329
3.00	2.000
3.50	3.323
4.00	6.000
4.50	11.632
5.00	24.000
5.50	52.343
6.00	120.000
6.50	287.885
7.00	720.000
7.50	1871.254
8.00	5040.000
8.50	14034.408
9.00	40320.002
9.50	119292.467
10.00	362880.017

www.foxitsoftware.com

computer က တွက်ပေးလိုက်တဲ့ အဖြေတွေကို လေ့လာကြည့်မယ်ဆိုရင် Stirling ရဲ approximation နည်းဟာ မဆိုဘူးဆိုတာ တွေ.ရမှာပါ။ $1! = 1.000$, $2! = 2.000$, $3! = 3.000$ စသည်ဖြင့် ကွက်တိမှန်လာပြီးတော့ $8!$ ကျမှု error တွေ.လာရပါတယ်။ error က 0.002 ပါ။ $9!$ ကျတော့ error က 0.017 ပါ။ Stirling's equation ကို series ပို၍ည်ရည်ယူလိုက်ရင် အဖြေပိုမှန်လာမှာပါ။

၅.၃ Static Variables

ကျွန်တော်တို့ static variables တွေ အကြောင်းကို မပြောခင် single-file program နဲ့ multifile program(j) ခုရဲ့ခြားနားမှုကို အရင်ရင်ပြချင်ပါတယ်။ single-file program ဆိုတာ main function ရော တစ်ခြား function တွေရော အကျွန်လုံးကို ဖိုင်တစ်ခုတည်းမှာ ရေးထားတာကို ဆိုလိုပါတယ်။ ဒါပေမယ့် multifile program မှာကျတော့ ဒီ function တွေ အများကြီးကို ဖိုင်တွေခွဲပြီး သိမ်းမှာပါ။ လိုတဲ့အခါကျမှ ပြန်ပေါင်းပြီးတော့ သုံးမယ့်သဘောပေါ့။ static variable ထွေဟာ automic variable တွေလိုပဲ သူတို့ကို define လုပ်တဲ့ function နဲ့ပဲဆိုင်ပါတယ်။ ဒီနှစ်ခုဟာ ဆင်သလိုရှိပေမယ့် တကယ်တော့ မတူပါဘူး။ automic variable တွေရဲ့ value ဟာ သူတို့ရောက်နေတဲ့ function ကနေ အပြင်ထွက်သွားပြီ ဆိုရင် ဒီတန်ဖိုးဟာ အလိုလိုပျက်သွားပါတယ်။ ဒါပေမယ့် static variable ကျတော့ function အပြင်ကို ထွက်လိုက်ဝင်လိုက် လုပ်ရင်နဲ့ပဲ function မှာ နောက်ဆုံး ရှိနေတဲ့ value ကိုသက်ပြီး အသုံးပြုနေမှာပါ။ ဒါဟာ static variable ရဲ့သဘောတရားပါပဲ။ သူ.၏ usage ကို ရှင်းရှင်းလင်းလင်း ဖြင့်သွားအောင်လို့ example လေးတစ်ခုနဲ့ ကျွန်တော်ရှင်းပြပါမယ်။

EXAMPLE 5.7: Static Variables

This program illustrates the use of static integer variable called num. Its value is retained by the computer, because its previous value is used whenever the function that declares it is called again.

```
# include <stdio.h>
#define PRINTX printf ("IN MAIN : NUM = %d\n", num)

main ()
{
    int num = 2;
```

```

PRINTX;
func () ;
PRINTX;
func () ;
PRINTX;
}

func ()
{
    static int i, num = 0;

    for ( i = 1; i <= 3; ++i )
        printf ("IN FUNC : NUM = %d\n", ++num);
    printf ("\n");
}

```

ဒါ program ကို လေ့လာကြည့်မယ်ဆိုရင် main function မှာ num ရဲ့ local value ဟာ ပေါ့။ မဟုတ်ဘူးလား။ ဒီတော့ main အတွက် စစချင်းရိုက်မယ့်အဖြေဟာ ၂ ပဲဖြစ်ရမှာပါပဲ။ ပြီးတော့မှ call ခေါ်လိုက်တဲ့ func() ဆိုကို computer ထွက်သွားပါတယ်။ func() ထဲမှာ num ကို static variable လို့ကြည်းထားတာကြောင့် func() ထဲမှာဖြစ်လာမယ့် num ရဲ့ value တွေဟာ (၁) ရယ်၊ (၂) ရယ်၊ (၃) ရယ်တို့ဖြစ်ပါတယ်။ ဒီတော့ func() မှာ num ရဲ့နောက်ဆုံး static တန်ဖိုးဟာ (၃) ဖြစ်မှာပေါ့။ မဟုတ်လား။ ဒါပြီးတော့မှ computer ဟာ main ဖန်ရှင်ကို ပြန်သွားပါတယ်။

ဒါအချင်မှာ num ရဲ့ value ဟာ (main ထဲမှာတော့) နှစ်ပဲဖြစ်နေတုန်းပါ။ ဒီတော့ computer က main အတွက် num = 2 လို့ ပြန်ရိုက်ပါပြီ။ ပြီးတော့ func() ကို ပြန်သွားပါတယ်။ func() ထဲမှာ num ရဲ့ နောက်ဆုံးတန်ဖိုးဟာ (၃) လို့ပြောထားတာ မှတ်မိတယ်နော်။ ဒီတော့ func() ထဲမှာ looping ခုတိယအကြိမ် ထပ်ပတ်လိုက်တာနဲ့ num ရဲ့ တန်ဖိုးတွေဟာ ၄၊ ၅၊ ၆ ဖြစ်ကုန်မှာပါပဲ။ ဒါပေမယ့် main ပြန်ရောက်တဲ့အခါကျတော့ num ရဲ့ value ဟာ (၂) ပဲဖြစ်မြို့ဖြစ်လျက်ပါ။ ဒါ program ကို ကျွန်တော်တို့ run လိုက်မယ်ဆိုရင် computer မှာ ဒီလိုပေါ်လာမှာပါ။

IN MAIN	:	NUM = 2
IN FUNC	:	NUM = 1
IN FUNC	:	NUM = 2
IN FUNC	:	NUM = 3

```
IN MAIN      : NUM = 2  
IN FUNC      : NUM = 4  
IN FUNC      : NUM = 5  
IN FUNC      : NUM = 6  
IN MAIN      : NUM = 2
```

EXAMPLE 5.8: Generating Fibonacci Numbers

Write a C program to sum a sequence of the Fibonacci numbers, in which each number is equal to the sum of the previous two numbers.

$F_i = F_{i-1} + F_{i-2}$ where F_i refers to the Fibonacci number. The first two Fibonacci numbers are defined to equal 1.

```
#include <stdio.h>  
#define PRINTX printf( "\nHow many Fibonacci numbers ? " )  
main ()  
{  
    int          i, n;  
    long int     term, sum = 0;  
    long int     fibo ( int n );  
  
    PRINTX;  
    scanf ( "%d", &n );  
  
    for ( i = 1; i <= n; ++i )  
        term = fibo ( i );  
        sum += term;  
        printf ( "\nI = %3d   F = %3ld" " SUM = %ld", i, term, sum );  
}  
}
```

```

long int fibo ( int count )
{
    static long int f1 = 1, f2 = 1;
    long int f;

    f = ( count < 3 ) ? 1 : f1+f2;
    f2 = f1;
    f1 = f;
    return ( f );
}

```

ဒဲ program ကို run လိုက်မယ်ဆိုရင်--

How many Fibonacci numbers ? 10

I = 1	F = 1	SUM = 1
I = 2	F = 1	SUM = 2
I = 3	F = 2	SUM = 4
I = 4	F = 3	SUM = 7
I = 5	F = 5	SUM = 12
I = 6	F = 8	SUM = 20
I = 7	F = 13	SUM = 33
I = 8	F = 21	SUM = 54
I = 9	F = 34	SUM = 88
I = 10	F = 55	SUM = 143

computer မှာ အဲဒီလိုပေါ်လာမှာပါ။ ဘယ်လိုကြောင့်ပေါ်လာလဲဆိုတောက် program မှာ trace လုပ်ကြည့်စေချင်ပါတယ်။

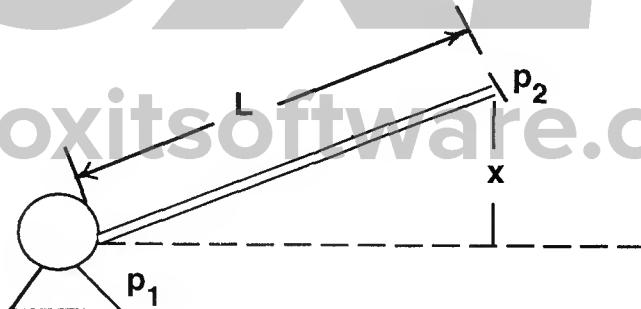
၅.၄ Multifile Programs

program တစ်ခုဟာ သိပ်ကြီးလာမယ်ဆိုရင် ဖိုင်တွေခွဲရေးပြီးတော့ compile လုပ်၊ link လုပ်ပြီးတော့ executable ဖိုင်ရအောင်လုပ်တဲ့နည်းဟာ အကောင်းဆုံးပါ။ အဲဒီလို အနေအထားမျိုးရှိတဲ့ ဖိုင်တွေ ကို multifile program လို့ ခေါ်ပါတယ်။ multifile တွေကို executable program ဖြစ်အောင် လုပ်တဲ့နည်း (၂) နည်း ရှိပါတယ်။ ပထမနည်းက command-line message တွေကိုသုံးပြီးတော့ ဖိုင်တစ်ခု ချင်းကို compile လုပ်၊ link လုပ်တဲ့နည်းပါ။ ဒုတိယနည်းကတော့ Make utility ကို သုံးတဲ့နည်းပါ။ Make ဟာ compiling process၊ linking process တွေကို အလိုအလျောက် လုပ်ပေးတဲ့ program တစ်ခုပါပဲ။ ဒီနည်းမှာ multifile program ကို ဖိုင်တွေခွဲပြီးတော့ နာမည်တစ်မျိုးစီအဲသိမ်းပါတယ်။ ပြီးတော့ executable ဖြစ်လာမယ့်ဖိုင်ကို Project menu ကအောင် (Open project မှာ) filename.prj ဆိုတဲ့ ဖိုင်နာမည်ပေးပြီးတော့ Project file အသစ်တစ်ခု ဖွင့်ရပါတယ်။ ပြီးတော့မှ ဒီဖိုင်ထဲကို စောစောက C program ဖိုင်တဲ့တွေကို add လုပ်ပေးရပါတယ်။ နောက်ပြီး control (Ctrl) + F9 key တွေကို တစ်ဖြို့တည်းနိုပ်လိုက်ရင် project ထဲကဖိုင်တွေဟာ အလိုလို compile လုပ်၊ link လုပ်သွားပြီးတော့ exe ဖိုင်ထွက်လာမှာပါ။ ဒီဖိုင်ရဲ့ နာမည်ကတော့ project ဖိုင်နာမည်နဲ့ အတူတူဖြစ်နေမှာပါ။ ဒီနည်းကိုသုံးပြီးတော့ လွယ်တဲ့ပုံစံတစ်ပုံကို program ရေးကြည့်ရအောင်။

EXAMPLE 5.9:

Fluid Flow

Write a C program to determine the necessary pump discharge pressure for the installation shown in the figure.



- q = oil flow rate, bbl / day
- L = pipe length, miles
- X = elevation, ft
- d = inside pipe diameter, in.
- μ = oil viscosity, cps

P_1 = upstream pressure, psia
 P_2 = downstream pressure, psia

```
#include <stdio.h>
void readata ( void );
float d, l, q, spgr, vis, p2, x ;
void calc ( void );
float rho, v, re, f, deltapf, rhodx, p1;
void output ( void );

main ()
{
    readata ();
    calc ();
    output ();
}
```

ဒဲ program ကို ကျန်တော်တိ. ဒီမှာပဲရပ်ထားရင် မပြည့်စုပါဘူး။ ဘာပြုလိုလဲဆိုတော့ data တွေကို
ဖော်ပေးမယ့် readata () ဆိုတဲ့ function ရယ်၊ flow calculation တွေကို တွက်ပေးမယ့် calc () ဆိုတဲ့
function ရယ်နဲ့၊ အဖြေထုတ်ပေးမယ့် output () function တွေ မပါလိုတဲ့ပဲ။ ဒါပေမယ့် ကိစ္စမရှိပါဘူး။
ကျန်တော်တိ. ဘယ်လို အလုပ်ဖြစ်အောင် လုပ်သွားတယ်ဆိုတာကို လေ့လာကြည့်ပါ၌။ အခုရေးပြီးတဲ့ ပရိုဂရမ်ကို
ဖိုင် name တစ်ခုပေးပြီးတော့ သိမ်းပါမယ်။ ဖိုင် name က mainprog. c ဆိုပါတော့။ အဲဒါပြီးတော့မှ
readata () function ကို ကျန်တော်တိ ဆင်ရေးပါမယ်။

```
void readata ( void )
{
    extern float d, l, q, spgr, vis, p2, x;

    printf ( "Enter the inside pipe diameter ( in ) : " );
    scanf ( "%f", &d );
    printf ( "Enter the pipe length ( miles ) : " );
    scanf ( "%f", &l );
```

```

printf ( "Enter the oil flowrate ( bbl / day ) : " );
scanf ( "%f", &q );
printf ( "Enter the sp gr of oil : " );
scanf ( "%f", &spgr );
printf ( "Enter the oil viscosity ( cps ) : " );
scanf ( "%f", &vis );
printf ( "Enter the downstream pr ( psia ) : " );
scanf ( "%f", &p2 );
printf ( "Enter the elevation ( ft. ) : " );
scanf ( "%f", &x );
}

```

ဒဲ program ကိုတော့ readata.c ဆိုတဲ့ ဖိုင် name နဲ့ ကျွန်ုပ်တော်တိ၊ saveလုပ်ပါမယ်။ ပြီးတော့မှ
တတိယ program ကို ဆက်ရေးပါမယ်။

```

#include < math.h >
{
    extern float d, q, spgr, vis, p2, x;
    extern float rho, v, re, f, deltapf, rhodx, p1;

    d = 12;
    vis = 0.000672;
    rho = spgr * 62.4;
    l = 5280.0;
    v = q * 5.6146 / ( 60.0 * 60.0 * 24 * 0.785 * d * d );
    re = d * v * rho / vis;
    if ( re < 1000 ) f = 16 / re;
    if ( re > 4000 ) f = 0.04 / pow ( re, 0.194 );
    deltapf = 2 * f * l * v * v * rho / ( 32.2 * d );
    rhodx = rho * x;
    p1 = ( rhodx + deltapf + p2 * 144 ) / 144;
}

```

ဒဲ program ကျေတော့ calc.c ဆိုတဲ့ဖိုင် name နဲ့ ကျွန်ုပ်တော်တိ၊ save လုပ်ပါမယ်။ ပြီးတော့မှ
နောက်ဆုံးဖိုင်ဖြစ်တဲ့ output.c ကို ဆက်ရေးမှာပါ။

```

void output ( void )
{
    extern float p1;

    printf ( "In Pump Discharge Pressure = %0.2f psia ", p1 );
}

```

ဒီဖိုင်တွေကို ရေးပြီးတဲ့အခါမှာ project ဖိုင်တစ်ခုကို (Alt + P key တွေကို နိုင်ပြီးတော့) ဖွင့်ပါမယ်။ project ဖိုင် name ကို gflow.prj လို့ ကျွန်တော်တို့ ပေးလိုက်ပါမယ်။ ပြီးတော့ စောောကရေးပြီးသိမ်းထားတဲ့ ဖိုင်တွေအားလုံးကို gflow.prj ထဲမှာ addလုပ်ထည့်ပါမယ်။ addလုပ်တဲ့နည်းကတော့ del key ကို နိုင်ပြီးတော့ ဖိုင်တစ်ခုချင်းကို ရေးထည့်ရမှာပါ။ ဥပမာ C:\TC\OUTPUT mainprog.c လို့ရေးလိုက်မယ်ဆိုရင် root directory C:\ အောက်က TC subdirectory အောက်က OUTPUT subdirectory ထဲမှာရှိတဲ့ mainprog.c ဆိုတဲ့ ဖိုင်ကို gflow.prj ဖိုင်ထဲမှာ add လုပ်လိုက်တာပါပဲ။ ဒီနည်းနဲ့ပဲ ကျွန်တဲ့ ဖိုင်တွေကို ဆက် add လုပ်သွားပါမယ်။ အဲဒါပြီးတော့မှ Ctrl + F9 key တွေကို နှုပ်လိုက်ရင် gflow.exe ဆိုတဲ့ run ဖိုင်ကို ကျွန်တော်တို့ရမှာပါ။ ဒါဆိုရင် pipeline gas transport problem ကို ဖြေရှင်းလို့ ရပါပြီ။ ဒီလိုဖြေရှင်းတဲ့နည်းကို multifile compilation လို့ခေါပါတယ်။ ကျွန်တော်တို့ လက်တွေ ကြည့်ရအောင်လား။

Enter the inside pipe diameter (in.) : 8
Enter the pipe length (miles) : 17
Enter the oil flowrate (bbl / day) : 40000
Enter the specific gravity of oil : 0.83
Enter the oil viscosity (cps) : 9.5
Enter the downstream pressure (psia) : 14.7
Enter the elevation (ft.) : 328

Pump Discharge Pressure = 986.25 psia