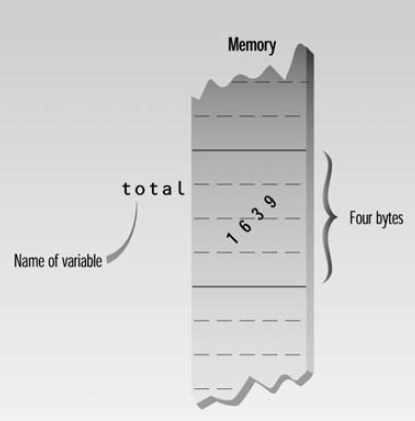
OOP in C++ : အပိုင္း (၃) - Data types and Variables (၁)

Variables ေတြဟာ ပရိုဂရမ္းမင္း ဘာသာတိုင္းအတြက္ အေျခခံ အက်ဆံုး အစိတ္အပိုင္းမ်ား ျဖစ္ၾကပါတယ္။ အခ်က္အလက္ေတြကို ယာယီ သိမ္းဆည္းဖို႔ Variable ေတြကို နာမည္ သတ္မွတ္ေပးထားၿပီး ဖန္တီးႏိုင္သလို တန္ဖိုးေတြလဲ သတ္မွတ္ ထည္႔သြင္းႏိုင္ပါတယ္။ Variables ေတြကို ကြန္ပ်ဴတာ မွတ္ဥာဏ္ (RAM) ရဲ႕ ေနရာအခ်ိဳ႕မွာ ဖန္တီးထားတာ ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ Variable တစ္ခုကို တန္ဖိုးတစ္ခု ထည္႔သြင္း လိုက္တယ္ဆိုတာဟာ တကယ္ေတာ႔ အဲဒီ Variable ကို သတ္မွတ္ေပးထားတဲ႔ RAM ထဲမွာ အဲဒီတန္ဖိုးကို သိမ္းဆည္းထားလိုက္တာပဲ ျဖစ္ပါတယ္။ အဲဒီလို သိမ္းဆည္းထားတဲ႔ ေနရာမွာ အခ်က္အလက္ရဲ႕ အမ်ိဳးအစားေပၚမူတည္ၿပီး ခြဲျခားသိမ္းဆည္းေပးဖို႔ လိုပါတယ္။ ဒါေၾကာင္႔ Variables ေတြကို data types ေတြ ခြဲျခားၿပီး ဖန္တီးေလ႔ ရွိပါတယ္။ ပရိုဂရမ္းမင္း ဘာသာစကားအမ်ားစုမွာ သံုးေလ႔ ရွိတဲ႔ data types ေတြကေတာ႔ integers၊ floating-point numbers ေတြနဲ႔ characters ေတြပဲ ျဖစ္ၾကပါတယ္။

Integer variable ေတြဟာ 130000၊ ဒါမွ မဟုတ္ -23 တို႔လို အျပည္႔ကိန္းေတြကို ေရတြက္ ထည္႔သြင္းရာမွာ အသံုးျပဳပါတယ္။ floating-point numbers ေတြနဲ႔ မတူတဲ႔ အခ်က္ကေတာ႔ integers ေတြမွာ ဒႆမေနာက္က တန္ဖိုးေတြ (အပိုင္းကိန္း တန္ဖိုးေတြ) မပါရွိတာပဲ ျဖစ္ပါတယ္။

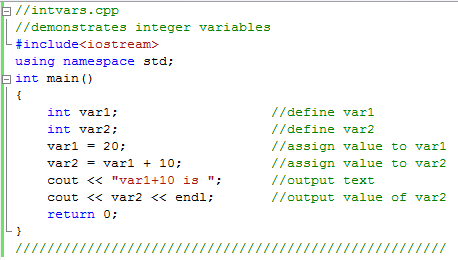
**Defining Integer Variables**

Integer variable ေတြ အမ်ိဳးအစားမ်ားစြာ ရွိပါတယ္။ ဒါေပမယ္႔ အသံုးအမ်ားဆံုးကေတာ႔ int ပဲ ျဖစ္ပါတယ္။ ေနရာယူတဲ႔ မွတ္ဥာဏ္ အရြယ္အစားကေတာ႔ system ေပၚ မူတည္ၿပီး ကြဲျပားႏိုင္ပါတယ္။ ဥပမာ- 32-bit system Windows ေတြမွာ int ဟာ 4 bytes (32 bits) ေနရာယူပါတယ္။ ဒါေၾကာင္႔ int ဟာ -2,147,483,648 ကေန 2,147,483,647 ၾကား တန္ဖိုးရွိတဲ႔ ဂဏန္းေတြကို သိမ္းဆည္းေပးႏိုင္ပါတယ္။



ဒါေပမယ္႔ ယခင္ Windows system အေဟာင္းေတြနဲ႔ MS-DOS စနစ္ေတြမွာေတာ႔ int ဟာ 2 bytes ပဲ ေနရာယူပါတယ္။ data types ေတြ ေနရာယူတဲ႔ သတ္မွတ္ခ်က္ေတြကိုေတာ႔ LIMITS ဆိုတဲ႔ header file ထဲမွာ ေရးသားထားပါတယ္။ ကၽြန္ေတာ္တို႔အေနနဲ႔ လက္ရွိသံုးေနတဲ႔ compiler ရဲ႕ help ကေနလည္း ရွာေဖြ ဖတ္ရႈႏိုင္ပါတယ္။

ဥပမာ-အေနနဲ႔ int data type သံုးထားတဲ႔ ပရိုဂရမ္ေလးတစ္ခုကို ေရးျပပါ႔မယ္။



အထက္ပါ ပရိုဂရမ္ေလးမွာ int var1; int var2; ဆိုတဲ႔ ကုဒ္ႏွစ္ေၾကာင္းဟာ integer variable var1 နဲ႔ var2 တို႔ကို ေၾကျငာ သတ္မွတ္ေပးလိုက္တာပဲ ျဖစ္ပါတယ္။ ေရွ႕က int ဆိုတဲ႔ keyword ေလးက variable ရဲ႕ type ကို သတ္မွတ္ေပးတာ ျဖစ္ပါတယ္။ အဲဒီ statement ေတြကို declaration ေတြလို႔ ေခၚဆိုၿပီး အျခား program statements ေတြလိုပဲ စာေၾကာင္း အဆံုးမွာ semicolon (;) နဲ႔ ပိတ္ေပးရမွာ ျဖစ္ပါတယ္။

ကၽြန္ေတာ္တို႔ အေနနဲ႔ variables ေတြကို အသံုးမျပဳခင္မွာ ႀကိဳတင္ ေၾကျငာေပးထားရမွာ ျဖစ္ပါတယ္။ ဆန္သြားဝယ္မယ္႔သူဟာ ဆန္အိတ္ကို ႀကိဳယူသြားရသလိုပါပဲ။ ဒါေၾကာင္႔ မ်ားေသာအားျဖင္႔ ပရိုဂရမ္ရဲ႕ အစပိုင္းမွာ variable ေတြကို ေၾကျငာထားေလ႔ ရွိပါတယ္။ ဒါေပမယ္႔ တကယ္က ပရိုဂရမ္ရဲ႕ မည္သည္႔ ေနရာမွာမဆို variable ေတြကို ေၾကျငာေပး ႏိုင္ပါတယ္။

**Declarations and Definitions**

Declaration နဲ႔ Definition ၾကားက ျခားနားခ်က္ကို ရွင္းျပခ်င္ပါတယ္။ Declare လုပ္တာဟာ variable ကို ဘယ္လိုေခၚမယ္၊ ဘာ data type ျဖစ္တယ္ဆိုတာကို program အတြင္းမွာ မိတ္ဆက္ ေၾကျငာေပးတာ ျဖစ္ပါတယ္။ Definition ဆိုတာကေတာ႔ memory ထဲမွာ အဲဒီ variable ကို ေနရာယူလိုက္တာပဲ ျဖစ္ပါတယ္။ အထက္က ပရိုဂရမ္ထဲမွာ ေရးထားတဲ႔ int var1; int var2; ဆိုတဲ႔ ကုဒ္ႏွစ္ေၾကာင္းဟာ declarations ေတြ ျဖစ္သလို definition ေတြလည္း ျဖစ္ပါတယ္။ မ်ားေသာအားျဖင္႔ အခ်ိဳ႕ေနရာေတြမွာ အဲဒီ အခ်က္ႏွစ္ခုက တူေလ႔ ရွိပါတယ္။ ဒါေပမယ္႔ ေနာက္ပိုင္းမွာ definitions မဟုတ္တဲ႔ declarations ေတြ အေၾကာင္းကို အေသးစိတ္ ရွင္းျပသြားမွာ ျဖစ္ပါတယ္။

Variable Names

Variavle ေတြကို နာမည္ေပးတဲ႔ ေနရာမွာ လိုက္နာရတဲ႔ စည္းကမ္း အခ်ိဳ႕ ရွိပါတယ္။ နာမည္ကို စာလံုး အႀကီး ဒါမွမဟုတ္ အေသးနဲ႔ ေရးႏိုင္ပါတယ္။ 1 ကေန 9 အထိ ဂဏန္းေတြ ကိုလဲ ထည္႔သြင္းႏိုင္ေပမယ္႔ အစ စာလံုးကိုေတာ႔ အကၡရာနဲ႔ပဲ စရမွာ ျဖစ္ပါတယ္။ နာမည္ ၾကားမွာ space ျခားလို႔ မရေပမယ္႔ under-score (\_) ကို ေတာ႔ အသံုးျပဳလို႔ ရပါတယ္။ နာမည္ကို စိတ္ႀကိဳက္ စာလံုးေရ အနည္းအမ်ား ေပးႏိုင္ပါတယ္။ စာလံုး အႀကီးနဲ႔ အေသးကို compiler က ခြဲျခား သတ္မွတ္ထားတာ ျဖစ္လို႔ var ဟာ VAR နဲ႔ မတူႏိုင္တာကို သတိျပဳရမွာ ျဖစ္ပါတယ္။

ေနာက္ ျခြင္းခ်က္ တစ္ခုက C++ မွာ အသံုးျပဳဖို႔ သတ္မွတ္ထားတဲ႔ keywords ေတြကိုေတာ႔ နာမည္ေပးတဲ႔ အခ်ိန္မွာ ခ်န္လွပ္ထားခဲ႔ရမွာ ျဖစ္ပါတယ္။ (ဥပမာ- if, else, while, do, case, new, class, int, float, double, char,...)။

C++ ပရိုဂရမ္မာ အမ်ားစုဟာ variable ေတြကို နာမည္ေပးဖို႔ စာလံုးအေသးေတြပဲ သံုးေလ႔ ရွိပါတယ္။ အခ်ိဳ႕ကေတာ႔ စာလံုးအႀကီးနဲ႔ အေသးကို ေရာေရးေလ႔ ရွိပါတယ္။ (ဥပမာ - IntVar, dataCount)။ const ေတြကို ေရးသားတဲ႔ ေနရာမွာေတာ႔ စာလံုးအႀကီးေတြပဲ သံုးေလ႔ ရွိၾကပါတယ္။ ဘယ္လိုပံုစံမ်ိဳးပဲေရးေရး ပရိုဂရမ္တစ္ပုဒ္မွာေတာ႔ သတ္သတ္မွတ္မွတ္ စတိုင္ တစ္မ်ိဳးနဲ႔ပဲ ေရးသားသင္႔ပါတယ္။ C++ မွာ class ေတြနဲ႔ function ေတြကို နာမည္ေပးတဲ႔ ေနရာမွာလဲ အထက္ပါ ဥပေဒသမ်ားကိုပဲ အသံုးျပဳပါတယ္။

Variable ေတြကို နာမည္ေပးတဲ႔ ေနရာမွာ အဲဒီ variable ရဲ႕ ရည္ရြယ္ခ်က္ကို လူတိုင္း နားလည္လြယ္မယ္႔ နာမည္မ်ိဳးကို ေပးဖို႔ လိုပါတယ္။ ဥပမာ ေျပာရရင္ bT ဒါမွမဟုတ္ t ဆိုတာထက္စာရင္ boilerTemperature လို႔ နာမည္ေပးတာမ်ိဳးက ပိုမိုရွင္းလင္းလြယ္ကူေစပါတယ္။

Assignment statements

အထက္ပါ ပရိုဂရမ္ထဲက var1 = 20; var2 = var1 + 10; ဆိုတဲ႔ ကုဒ္ ႏွစ္ေၾကာင္းကို ေလ႔လာၾကည္႔ၾကရေအာင္။ ညီမွ်ျခင္း သေကၤတ (=) ဟာ တကယ္ေတာ႔ သူ႔ရဲ႕ ညာဘက္က တန္ဖိုးေတြကို ဘယ္ဘက္က variable ေတြထဲကို assign ျပဳလုပ္ေပးလိုက္တာပဲ ျဖစ္ပါတယ္။ ပထမ ကုဒ္စာေၾကာင္းမွာပါတဲ႔ var1 ဟာ ယခင္က တန္ဖိုး မရွိေသးေပမယ္႔ ဒီကုဒ္ေၾကာင္႔ တန္ဖိုး 20 ျဖစ္လာပါတယ္။

Integer Constants

အဲဒီ ပထမ စာေၾကာင္းမွာ ပါတဲ႔ 20 သည္ integer constant ပဲ ျဖစ္ပါတယ္။ constants ေတြဟာ ပရိုဂရမ္ တစ္ပုဒ္မွာ တန္ဖိုး မေျပာင္းလဲဘဲ တည္ရွိေနၾကပါတယ္။ integer constant ေတြကေတာ႔ ဒႆမ တန္ဖိုး မပါရွိဘဲ integer ရဲ႕ သတ္မွတ္ တန္ဖိုးေတြ အတြင္းမွာ ရွိတဲ႔ အျပည္႔ကိန္း တန္ဖိုးေတြပဲ ျဖစ္ပါတယ္။ ကုဒ္ ဒုတိယ စာေၾကာင္းမွာပါတဲ႔ (+) သေကၤတကေတာ႔ var1 ကို 10 ေပါင္းထည္႔ေပးလိုက္တာပဲ ျဖစ္ပါတယ္။ အဲဒီ 10 ကလဲ ေနာက္ထပ္ integer constant တစ္ခုပဲ ျဖစ္ပါတယ္။ အဲဒီလို ေပါင္းလို႔ ရလာတဲ႔ တန္ဖိုးကို (=) operator က var2 ထဲကို assign ျပဳလုပ္ေပးလိုက္တာပဲ ျဖစ္ပါတယ္။

Output Variations

နမူနာ ပရိုဂရမ္ထဲက cout << "var1+10 is "; ဆိုတဲ႔ စာေၾကာင္းက ကၽြန္ေတာ္တို႔ ယခင္ သင္ခန္းစာေတြမွာ ေလ႔လာခဲ႔သလိုမ်ိဳး string constant တစ္ခု (var1+10 is ) ကို display မွာ ျပသေပးမွာ ျဖစ္ပါတယ္။ ေနာက္ စာသား တစ္ေၾကာင္း ျဖစ္တဲ႔ cout << var2 << endl; ကေတာ႔ variable var2 ထဲမွာ သိမ္းဆည္းထားတဲ႔ တန္ဖိုးကို ျပသေပးမွာ ျဖစ္ပါတယ္။ ဒါေၾကာင္႔ ဒီပရိုဂရမ္ကို run ၾကည္႔လိုက္မယ္ဆိုရင္ var1+10 is 30 ဆိုတာကို ျမင္ရမွာ ျဖစ္ပါတယ္။ မွတ္သားရမွာ တစ္ခုက cout နဲ႔ << operator ေတြဟာ integer နဲ႔ string ေတြကို ခြဲျခား ကိုင္တြယ္ႏိုင္စြမ္း ရွိတယ္ ဆိုတာပါပဲ။ ဒါေၾကာင္႔ C language မွာ printf() အသံုးျပဳသလို (display လုပ္ရမယ္႔ variable သာမကပဲ) data type ေတြ ခြဲျခားဖို႔ %d %c %s ေတြ ထည္႔ေပးစရာမလိုဘဲ အဆင္ေျပေျပ လုပ္ေဆာင္ႏိုင္စြမ္း ရွိပါတယ္။

ကၽြန္ေတာ္တို႔ အေနဲ႔ cout ႏွစ္ေၾကာင္းက ထြက္လာတဲ႔ အေျဖက တစ္ေၾကာင္းတည္း ျဖစ္ေနတာကို သတိထားမိမွာပါ။ အလိုအေလ်ာက္ တစ္ေၾကာင္း ဆင္းသြားတာမ်ိဳး မရွိပါဘူး။ လိုအပ္ရင္ ကၽြန္ေတာ္တို႔ ကိုယ္တိုင္ ထည္႔သြင္းေပးဖို႔ လိုအပ္မွာ ျဖစ္ပါတယ္။ ေနာက္ပိုင္းမွာ '\n' လိုမ်ိဳး escape sequence ေတြနဲ႔ manipulator ေတြ အသံုးျပဳပံုကို အေသးစိတ္ ရွင္းလင္းသြားမွာ ျဖစ္ပါတယ္။

The endl Manupulator

cout ေနာက္ဆံုး စာေၾကာင္းထဲက ေနာက္ဆံုး စကားလံုးဟာ endl ျဖစ္ပါတယ္။ တကယ္ေတာ႔ တစ္ေၾကာင္း ဆင္းခ်င္တဲ႔ ေနရာမွာသံုးရတဲ႔ manipulator တစ္မ်ိဳးပဲ ျဖစ္ပါတယ္။ '\n' နဲ႔ သေဘာသဘာဝခ်င္း တူညီေပမယ္႔ အသံုးျပဳရတာ အဓိပၸါယ္ ပိုမိုရွင္းလင္းေစပါတယ္။ manipulator ေတြကေတာ႔ output stream ကို ျပဳျပင္ပံုေဖာ္ေပးႏိုင္တဲ႔ ညႊန္ၾကားခ်က္ေတြပဲ ျဖစ္ပါတယ္။ endl အေနနဲ႔ '\n' နဲ႔ မတူတဲ႔ အခ်က္ကေတာ႔ output buffer ကို flush ျပဳလုပ္ေပးႏိုင္တာပဲ ျဖစ္ပါတယ္။

**Other Integer Types**

အထက္မွာ တင္ျပခဲ႔သလိုပဲ int အျပင္ အျခား integer data types ေတြ ရွိေနပါေသးတယ္။ အသံုးအမ်ားဆံုးကေတာ႔ long နဲ႔ short ပဲ ျဖစ္ပါတယ္။ (တကယ္ေတာ႔ char ဆိုတာလဲ integer data type လို႔ ေျပာလို႔ ရႏိုင္ပါတယ္။ ေနာက္ပိုင္းမွာ အလ်င္းသင္႔သလို ရွင္းျပေပးသြားပါ႔မယ္။) ကၽြန္ေတာ္တို႔ အထက္မွာ တင္ျပခဲ႔သလိုမ်ိဳး int ဟာ system ေပၚမူတည္ၿပီး အရြယ္အစား ကြဲျပားေပမယ္႔ long နဲ႔ short တို႔ကေတာ႔ system ေပၚ မူတည္မေနဘဲ အရြယ္အစား သတ္မွတ္ခ်က္ ပံုေသ ျဖစ္ပါတယ္။

long အေနနဲ႔ 4 bytes အျမဲေနရာယူေလ႔ရွိၿပီး 32-bit Windows system မွာ ရွိတဲ႔ int နဲ႔ အရြယ္အစားခ်င္း တူပါတယ္။ ဒါေၾကာင္႔ long ဟာ -2,147,483,648 ကေန 2,147,483,647 ၾကား တန္ဖိုးရွိတဲ႔ ဂဏန္းေတြကို သိမ္းဆည္းေပးႏိုင္ပါတယ္။ long ကို ေနာက္တစ္နည္း long int လို႔လဲ ေရးႏိုင္ပါတယ္။ အဓိပၸါယ္ကေတာ႔ အတူတူပဲ ျဖစ္ပါတယ္။ 32-bit system မွာ long ကို သံုးရတာဟာ int နဲ႔ တူညီေနလို႔ သိပ္မထူးျခားေပမယ္႔ 16-bit system ေတြမွာေတာ႔ 4-bytes ေနရာယူတဲ႔ integer type တစ္ခုကို ဖန္တီးဖို႔ အဆင္ေျပေစပါတယ္။ short ကေတာ႔ 2-bytes ေနရာယူၿပီး 16-bit system က int နဲ႔ အရြယ္အစား တူညီပါတယ္။ short ဟာ -32,768 ကေန 32,767 ၾကားမွာရွိတဲ႔ ဂဏန္းေတြကို သိမ္းဆည္းႏိုင္ပါတယ္။ short ကိုေတာ႔ ေခတ္သစ္ Windows system ေတြမွာ သာမန္အားျဖင္႔ အသံုးျပဳေလ႔ မရွိဘဲ memory ကို ေခၽြတာဖို႔ သိပ္လိုမွပဲ သံုးၾကပါေတာ႔တယ္။ int ဟာ short ထက္ ႏွစ္ဆ အရြယ္အစား ပိုႀကီးၿပီး အခ်က္အလက္ေတြကိုလဲ ပိုမိုျမန္ဆန္စြာ ေရးဖတ္ႏိုင္လို႔ အသံုးမ်ားပါတယ္။ ကၽြန္ေတာ္တို႔ အေနနဲ႔ long data type အမ်ိဳးအစား constant တစ္ခုကို ဖန္တီးခ်င္တယ္ ဆိုရင္ ဂဏန္းတန္ဖိုးရဲ႕ ေနာက္မွာ L ကို ကပ္ထည္႔ေပးလိုက္ရမွာ ျဖစ္ပါတယ္။ ဥပမာ -

longvar = 7678L; //assigns long constant 7678 to longvar

အသံုးနည္းတဲ႔ integer data types ေတြ ရွိပါေသးတယ္။ compiler ေတြေပၚမူတည္ၿပီး ကြဲျပားႏိုင္ပါတယ္။ အဲဒီ integer ေတြက သူတို႔ ေနရာယူတဲ႔ bits အတိအက်ကို သတ္မွတ္ထားပါတယ္။ (1-byte = 8-bits)။ အဲဒီ type ေတြရဲ႕ အမည္ေရွ႕မွာ underscores ႏွစ္ခု ကို ေရွ႕က ခံထားပါတယ္။ ဥပမာ- \_\_int8, \_\_int16, \_\_int32, \_\_int64 တို႔ပဲ ျဖစ္ပါတယ္။ \_\_int8 က 32-bit system က char နဲ႔ တူညီၿပီး၊ \_\_int16 က short နဲ႔ \_\_int32 က int နဲ႔ long တို႔နဲ႔ တူညီပါတယ္။ \_\_int64 ကေတာ႔ အႀကီးဆံုး integer value ကို ဂဏန္း ၁၉ လံုးအထိ သိမ္းဆည္းႏိုင္ပါတယ္။ ဒီ type ေတြကို သံုးရာမွာ အရြယ္အစား ပံုေသျဖစ္တဲ႔ အားသာခ်က္ ရွိေပမယ္႔ လက္ေတြ႔မွာေတာ႔ အသံုးနည္းတာကို ေတြ႔ရပါတယ္။

ေနာက္အပတ္ေတြမွာေတာ႔ အျခား data types ေတြနဲ႔ input ျပဳလုပ္ပံုေတြကို ဆက္လက္ ေဆြးေႏြးသြားပါမယ္ခင္ဗ်ာ။ ဤ ေဆာင္းပါးပါ အခ်က္အလက္မ်ားကို စမ္းသပ္ရာမွာ အခက္အခဲ ရွိေနဦးမယ္ဆိုရင္ စာေရးသူရဲ႕ တင္ျပပံု ခ်ိဳ႕ယြင္းခ်က္ေၾကာင္႔သာ ျဖစ္ပါတယ္။ သိလိုသည္မ်ားကိုလည္း aungwh2013@gmail.com ကို ဆက္သြယ္ၿပီး ေမးျမန္းႏိုင္သလို ျပင္ဦးလြင္သားမ်ား အတြက္လည္း ေအာက္ေဖာ္ျပပါ လိပ္စာရွိ ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ၌လည္း ေလ႔လာ စံုစမ္းႏိုင္ပါတယ္။

**အကိုးအကား**

Object-Oriented Programming in C++(4th edition), Robert Lafore, Copyright©2002 by Sams Publishing: ISBN 0-672-32308-7

Dr. ေအာင္ဝင္းထြဋ္ (bluephoenix)

http://www.tech4mm.com

ေတာ္ဝင္ ကြန္ပ်ဴတာ စင္တာ

၁၇၉ စ၊ သုမဂၤလာ၊ ေစ်းေလး အေနာက္ဘက္၊ ျပင္ဦးလြင္ၿမိဳ႕