
CHAPTER 5

VARIABLES AND OTHER BASIC ELEMENTS IN C++ PROGRAMS

Now, finally, we really begin studying computer programming with the C++ language. Variables are important elements in computer programs, and they are needed in almost every computer program. In this chapter we shall study different types of variables and use the variables in simple arithmetic computations. We shall also explore names and keywords which are very basic elements in source programs.

This chapter, as well as the following chapters, introduce many examples of C++ source programs for you to study. A source program is a textual description of what the computer should do when the compiled version of the source program is executed by a computer. The source programs that you will see in this chapter contain variable declarations followed by executable program statements which do something with the variables. The general structure of the programs is the following:

```
#include <iostream.h>
int main()
{
    Variable declarations.

    Action statements (executable statements)
    that modify the contents of the variables
    that were declared above.
}
```

These are sample pages from Kari Laitinen's book
"A Natural Introduction to Computer Programming with C++".
For more information, please visit
<http://www.naturalprogramming.com/cppbook.html>

5.1 Integer variables (*int, short, long, char*)

A variable in a source program is a basic program element that can be used to store numerical values. The value of a variable usually changes when the program is being executed. When we declare a variable in a program, we actually reserve a few bytes of computer's main memory to be used for a special purpose. The following is an example of a variable declaration (or variable definition):

```
int integer_from_keyboard ;
```

The above source program line means that four bytes (32 bits) of memory are reserved to store an integer that will be read from the keyboard, and those four bytes are referred to with the name `integer_from_keyboard`. Integers are whole numbers that have no decimal point. Integers can be positive or negative.

A variable always has a type, such as `int` which is an abbreviation of the word "integer". The programmer, the person who writes the variable declarations in a program, must give a unique name to each variable. In the declaration above, the name of the variable is `integer_from_keyboard`. Variable declarations, like all C++ statements, must be terminated with a semicolon ;.

In this book we shall assume that variables of type `int` are 32-bit variables which occupy four bytes in the main memory of the computer. The C++ language standard does not actually specify what is the correct size for variables of type `int`. The size of variable type `int` depends on which compiler is in use. There are C++ compilers which use 16-bit `int` variables that occupy only two bytes of memory, but most general-purpose C++ compilers use 32-bit `int` variables these days.

Program `game.cpp`, presented as a program description on the following page opening, is an example program where two variables of type `int` are declared and used. The program is an extremely simple computer game. Unfortunately it is not a fair game because the user of the program will always lose. The program always wins by presenting an integer that is one larger than the number given by the user of the program.

A source program like `game.cpp` is a text file in a computer's hard disk memory before it is compiled. When a source program is compiled and linked, we get an executable version of the program. The compiler is a computer tool that can convert a source program into executable form. The compiler reads and processes a source program file in the same order in which it is written. While compiling program `game.cpp`, the compiler sees first the variable declarations and reserves main memory for the variables. Then it transforms the action statements to machine instructions for the processor.

The action statements (executable statements) in a source program describe the activities a computer performs when the executable version of a source program is run by a computer. The action statements of a program are executed in an order that corresponds with the order in which the statements are written in the source program. In the case of the program `game.cpp`, the computer will perform the following activities:

- First it asks the user to enter an integer from the keyboard.
- Then it reads the integer from the keyboard and stores it in variable `integer_from_keyboard`.
- Then it calculates a value that is one larger than the user-given integer and stores that value in variable `one_larger_integer`.
- In the end, it displays the values of both variables and informs the user that the computer won the game.

There are four action statements in `game.cpp`. Each statement is terminated with a semicolon ;. The variable declarations at the beginning are also statements, but they are not action statements (executable statements). Variable declarations just reserve memory to store information.

Although the memory space that is reserved for an `int` variable is rather large, 4 bytes, there are always limitations how big values can be stored in an `int` variable. A 4-byte `int` variable can store values in the ranges

-2,147,483,648, ... , -1, 0, 1, ... , 2,147,483,647 (decimal numbering system)

-80000000H, ..., -1, 0, 1, ... , 7FFFFFFFH (hexadecimal numbering system)

A 4-byte `int` can thus store 4,294,967,296 (100000000H) different values. To demonstrate the difficulties that arise when the storage capacity of an `int` variable is exceeded, program `game.cpp` is also executed with too great an input value in the program description. As the computer uses 4-byte `int` variables, and it tries to increment the value 2,147,483,647, it results in number -2,147,483,648 and not in 2,147,483,648. To explain this strange behavior of the program, we must remember that the memories of computers can contain only non-negative binary numbers. Negative numbers are represented so that some positive values stored in memory are considered as negative numbers. For example, the value 2,147,483,648 is treated as -2,147,483,648. The values that can be contained in 4-byte `int` variables have the following meanings:

VALUE IN MEMORY	MEANING IN PROGRAM
2,147,483,648 (80000000H)	-2,147,483,648
2,147,483,649 (80000001H)	-2,147,483,647
2,147,483,650 (80000002H)	-2,147,483,646
.	.
.	.
4,294,967,294 (FFFFFFFEH)	-2
4,294,967,295 (FFFFFFFHH)	-1
0	0
1	1
.	.
.	.
2,147,483,646 (7FFFFFFEH)	2,147,483,646
2,147,483,647 (7FFFFFFFH)	2,147,483,647

Figure 5-1 shows how a 4-byte integer variable looks like when it is in the main memory of a computer. Those four bytes are reserved from somewhere in the main memory. We do not need to know the exact memory address. We only need to know that there are four contiguous memory locations.,

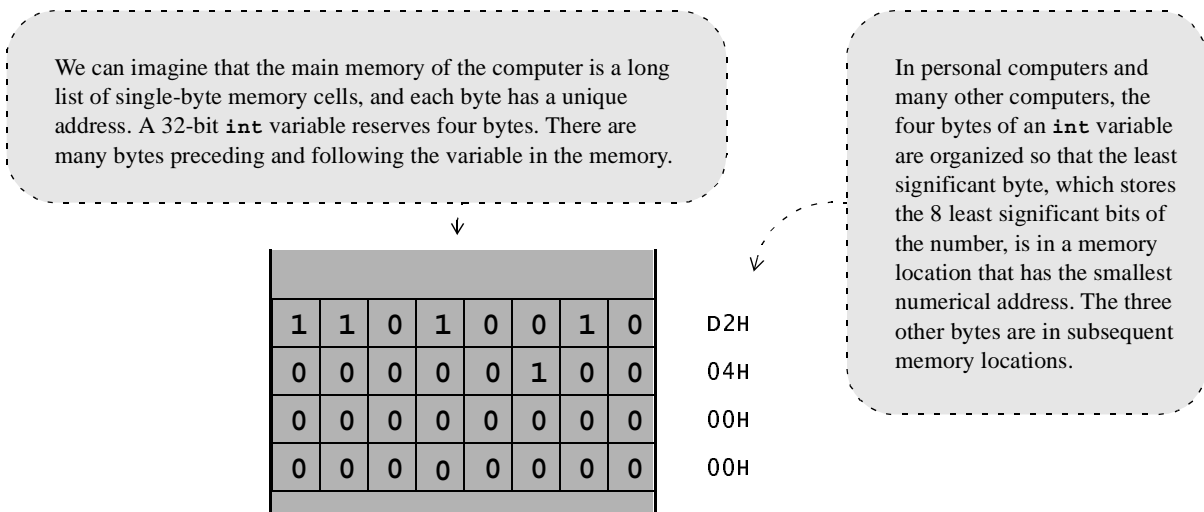


Figure 5-1. Value 1234, 4D2H, stored in a 4-byte `int` variable in a computer's main memory.

This line is not actually part of the program. This is a comment line that gives documentary information to the reader of the program. A double slash // marks the beginning of a comment line. The compiler ignores the double slash and the text that follows it on the same line.

Here two integer variables are declared. The names of the variables are `integer_from_keyboard` and `one_larger_integer`. The variables are referred to with these names later in the program.

```
> // game.cpp (c) 1997-2001 Kari Laitinen
#include <iostream.h>

int main()
{
    int integer_from_keyboard ;
    int one_larger_integer ;

    cout << "\n This program is a computer game. Please, type in "
         << "\n an integer in the range 1 ... 2147483646 : " ;

    cin >> integer_from_keyboard ;

    one_larger_integer = integer_from_keyboard + 1 ;

    cout << "\n You typed in " << integer_from_keyboard << "."
         << "\n My number is " << one_larger_integer << "."
         << "\n Sorry, you lost. I won. The game is over. \n" ;
}
```

This line of source code reads an integer from the keyboard and stores the read integer into variable `integer_from_keyboard`. The execution of the program stays on this line until the user of the program has entered an integer.

Texts inside double quote characters " " are strings of characters that will be displayed on the screen. `\n` among the text means that the text will begin from a new line. `\n` is said to be the newline character.

game.cpp - 1.+ A program that implements a simple computer game.

`cout` and `cin` are objects that represent the screen and the keyboard in C++ programs. These objects are pronounced "see-out" and "see-in". The output operator `<<` is used to output data to `cout`, the screen. The input operator `>>` is used to read data from `cin`, the keyboard.

After this assignment statement has been executed, the value of variable `one_larger_integer` is one greater than the value stored in `integer_from_keyboard`.

```

> cout << "\n This program is a computer game. Please, type in "
    << "\n an integer in the range 1 ... 2147483646 : " ;

-> cin  >>  integer_from_keyboard ;

one_larger_integer = integer_from_keyboard + 1 ; <-----

cout << "\n You typed in " << integer_from_keyboard << "."
    << "\n My number is " << one_larger_integer << "." <-----
    << "\n Sorry, you lost. I won. The game is over. \n" ;

```

It is possible to output many types of data in a single output statement. Here the values of the integer variables are displayed between strings of characters given inside double quotes. The output operator `<<` is placed between different types of data. A semicolon `;` terminates the entire output statement.

game.cpp - 1 - 1. The action statements of the program.

```
D:\book2cpp>game
```

```
This program is a computer game. Please, type in
an integer in the range 1 ... 2147483646 : 1234
```

```
You typed in 1234.
My number is 1235.
Sorry, you lost. I won. The game is over.
```

```
D:\book2cpp>game
```

```
This program is a computer game. Please, type in
an integer in the range 1 ... 2147483646 : 2147483647
```

```
You typed in 2147483647.
My number is -2147483648.
Sorry, you lost. I won. The game is over.
```

game.cpp - X. In the second execution too large an input value is given to the program.