Constant Objects

Some data must stay the same throughout a program. In C++, you can use a **named constant** to instruct a program to mark those memory locations in which data is fixed throughout program execution.

Named constant: A memory location whose content is not allowed to change during program execution. To allocate memory, we use C++'s declaration statements. The syntax to declare a named constant is:

```
const datatype const_name = value;
```

In C++, const is a reserved word. C++ programmers typically prefer to use uppercase letters to name a named constant.

Ex:

```
const double CONVERSION = 2.54;
const int NO_OF_STUDENTS = 20;
const char BLANK = ' ';
const double PI = 3.1415947;
```

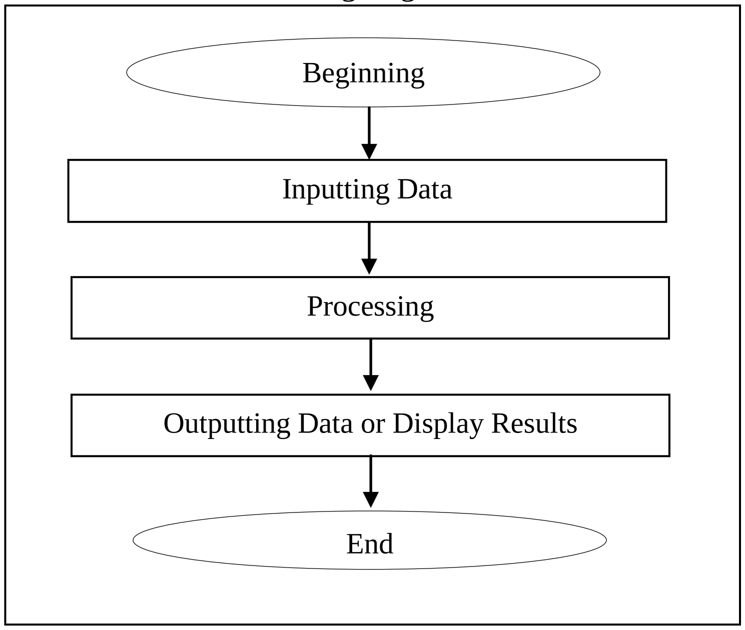
Thus the value of PI for example cannot be modified by the program. Even a statement such as the following will merely result in an error message:

```
PI = PI + 2.0; // invalid
```

Using a named constant to store fixed data, rather than using the data value itself, has one major advantage. If the fixed data changes, you do not need to edit the entire program and change the old value to the new value wherever the old value is used. Instead, you can make the change at just one place, recompile the program, and execute it using the new value throughout.

Program Stages

All most programs are consists of the following stages:



1- Beginning and End:

Each program must have one beginning which is represented by the main function:

Void main () {

While the end is represented by: } for the main function

Note:

Sometimes we use **int main()**, or sometimes **void main()**. Now the question comes into our mind, that what are the differences between these two.

The main() function is like other functions. It also takes arguments, and returns some value. One point we have to keep in mind that the program starts executing from this main() function. So the operating system calls this function. When some value is returned from main(), it is returned to operating system.

The void main() indicates that the main() function will not return any value, but the int main() indicates that the main() can return integer type data. When our program is simple, and it is not going to terminate before reaching the last line of the code, or the code is error free, then we can use the void main(). But if we want to terminate the program using exit() method, then we have to return some integer values (zero or non-zero). In that situation, the void main() will not work. So it is good practice to use int main() over the void main().

The important note that when we use int main() we should use return(0); before the end of the main function.

2- Entering Data

Each program is developed to perform a specific task. So it needs to deal with data related to the task. During the execution of the program, these data must be stored into the memory. C++ compiler uses constant and variable to allocate memory for the entered data.

Putting Data into Variables:
In C++, you can place data into a variable in two ways:
1. Use C++'s **assignment statement**.
2. Use **input** (read) **statements**.

1. Assignment Statement
The assignment statement takes the following form:

variable = expression;

In an assignment statement, the value of the expression should match the data type of the variable. The expression on the right side is evaluated, and its value is assigned to the variable (and thus to a memory location) on the left side. A variable is said to be initialized the first time a value is placed in the variable. In C++, **=** is called the **assignment operator**.

Ex: This program illustrates how data in the variables are manipulated.

```
#include <iostream>
#include <string>
using namespace std;
int main( ) {
int num1, num2;
  double sale;
  char first;

  num1 = 4;
  num2 = 4 * 5 - 11;
  sale = 0.02 * 1000;
  first = 'D';

  cout << "num1 = " << num1 << endl;
  cout << "num2 = " << num2 << endl;
  cout << "sale = " << sale << endl;
  cout << "first = " << first << endl;

return 0;
}
```

Execute:

```
num1 = 4
num2 = 9
sale = 20
first = D
```

Notes:

• This statement is valid in C++:

num = num + 2;

means "evaluate whatever is in num, add 2 to it, and assign the new value to the memory location num." The expression on the right side must be evaluated first; that value is then assigned to the memory location specified by the variable on the left side.

Thus, the sequence of C++ statements:

num = 6;

num = num + 2; and the statement:

num = 8;

• Suppose that **x, y,** and **z** are int variables. The following is a legal statement in C++: **x = y = z;**
In this statement, first the value of z is assigned to y, and then the new value of y is assigned to x.

2. Input (Read) Statement

Previously, you learned how to put data into variables using the assignment statement. In this section, you will learn how to put data into variables from the standard input device, using C++'s input (or read) statements.

Putting data into variables from the standard input device is accomplished via the use of **cin** and the operator **>>.** The syntax of **cin** together with >> is:

```
cin >> variable1 >> variable2 >> variable3 ... ;
```

This is called an input (read) statement. In C++, >> is called the **stream extraction operator**.

Ex: This program illustrates how input statements work.

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

main()
{
int feet;
int inches;

cout << "Enter two integers separated by spaces: "<< endl;
cin >> feet >> inches;

cout << "Feet = " << feet << endl;
cout << "Inches = " << inches << endl;
return 0;
}
```

```
Sample Run:
In this sample run, the user input is 23 and 7
Enter two integers separated by spaces: 23 7
Feet = 23
Inches = 7
```

Note: **cin (pronounced ‘‘see-in’’), which stands for** common input**, cout (pronounced ‘‘see-out’’), which stands for** common output.

3- Processing

This stage represents the task of the program.

4- Output Data

Usually the standard output device is the display screen. The C++ **cout** statement is the instance of the ostream class. It is used to produce output on the standard output device which is usually the display screen. The data needed to be displayed on the screen is inserted in the standard output stream (cout) using the insertion operator(<<).

```
#include <iostream>

using namespace std;

int main()
{
	char sample[] = "I Like ";

	cout << sample << " C++ Language";

	return 0;
}
```

Escape sequences

Escape sequences are special characters used in control string to modify the format of an output. These specific characters are translated into another character or a sequence of characters that may be difficult to represent directly. For example, you want to put a line break in the output of a C++ statement then you will use "\n" character which is an escape sequence itself.

An escape sequence consists of two or more characters. For all sequences, the first character will be "\" i.e. backslash. The other characters determine the interpretation of escape sequence. For example, "n" of "\n" tells the cursor to move on the next line.

Escape sequence	**Description**
\'	single quote
\"	double quote
\?	question mark
\\	backslash
\a	audible bell (alert)
\b	backspace
\f	form feed - new page
\n	line feed - new line
\r	carriage return
\t	horizontal tab
\v	vertical tab

New Line (\n)
When a new line is necessary in the output, then this escape sequence is used. For example:
Cout << ”COMPUTER\nSCIENCE”;
First of all, “COMPUTER” is printed and”\n” shifts the cursor to the next line. Then “SCIENCE” is printed on second line. A screenshot of output is shown below:

```
COMPUTER
SCIENCE
```

Tab (\t)
A **TAB** is equal to eight spaces. Whenever TAB button is pressed from the keyboard, then 8 spaces are left blank. This escape sequence performs the functionality of TAB key in the output stream. The code given below will insert a TAB between two words.

cout<<”COMPUTER\tSCIENCE”;

```
COMPUTER        SCIENCE
```

Alert Bell (\a)
This escape sequence is used to play beep during execution. For example:
cout<<”COMPUTER\aSCIENCE”;
First of all, “COMPUTER” is printed then a beep is played and after that “SCIENCE” is printed.

Backspace (\b)
Whenever we want to delete a single character, we press the button “backspace” from our keyboard. The same functionality can be achieved in C++ output with this escape sequence. For example:
cout<<”COMPUTER\bSCIENCE”;

```
COMPUTERSCIENCE
```

First of all, “COMPUTER” is printed and after that “\b” comes which deletes the last character i.e. “R”. After that, “SCIENCE” is printed.

Ex: Sample program

```
#include <iostream>
using namespace std;
int main() {
 cout << "\nThis is\t a string\n\t\t" " with \"many\" escape sequences!\n";
return 0;
}
```

Program output:

```
This is         a string
                with "many" escape sequences!
```

HW: Write a C++ program to generate the screen output shown below.

```
                I
                        "RUSH"
                                \TO\
                        AND
                /FRO/
```