## 1.1 Questions

- **1. Dividing a program into functions**
- **a. is the key to object-oriented programming.**
- **b. makes the program easier to conceptualize.**
- **c. may reduce the size of the program.**
- **d. makes the program run faster.**
- **2. A function name must be followed by _____.**
- **3. A function body is delimited by _____.**
- **4. Why is the main() function special?**
- **5. A C++ instruction that tells the computer to do something is called a _____.**
- **7. An expression**
- **a. usually evaluates to a numerical value.**
- **b. indicates the emotional state of the program.**
- **c. always occurs outside a function.**
- **d. may be part of a statement.**

- **9. True or false: A variable of type char can hold the value 301.**
- **10. What kind of program elements are the following?**
- **a. 12**
- **b. 'a'**
- **c. 4.28915**
- **d. JungleJim**
- **e. JungleJim()**
- **11. Write statements that display on the screen**
- **a. the character 'x'**
- **b. the name *Jim***
- **c. the number 509**

## 1.2 Answers to Questions

**1. b, c**
**2. parentheses**
**3. braces { }**
**4. It's the first function executed when the program starts**
**5. statement**
**6.**

```
// this is a comment
```

**/* this is a comment */**

**7. a, d**

**8. a. 4**
**b. 10**
**c. 4**
**d. 4**

**9. false**

**10. a. integer constant.** **b. character constant**
**c. floating-point constant.** **d. variable name or identifier**
**e. function name**

**11. a. cout << 'x';**
**b. cout << "Jim";**
**c. cout << 509;**

**Exercises**

**1. Assuming there are 7.481 gallons in a cubic foot, write a program that asks the user to a number of gallons, and then displays the equivalent in cubic feet.**

**2. Write a program that generates the following table:**

**1990 135**
**1991 7290**
**1992 11300**
**1993 16200**

**Use a single cout statement for all output.**

- **Solutions to Exercises**

**1. // ex2_1.cpp**

```
// converts gallons to cubic feet
#include <iostream.h>
int main()
{
float gallons, cufeet;
cout << "\nEnter quantity in gallons: "; cin >> gallons;
cufeet = gallons / 7.481;
cout << "Equivalent in cublic feet is " << cufeet << endl;
return 0; }
```

**1.3 The setw(n) Manipulator**

**setw changes the field width of output.**

**You can think of each value displayed by cout as occupying a field: an imaginary box with a certain width. The default field is just wide enough to hold the value. That is, the integer 567 will occupy a field three characters wide, and the string "Baghdad" will occupy a field seven characters wide.**

**The WIDTH1 program prints the names of three cities in one column, and their populations in another.**

```
// width1.cpp
// demonstrates need for setw manipulator
#include <iostream.h>
int main()
{Int pop1=2425785, pop2=47, pop3=9761;
cout << "LOCATION " << "POP." << endl
<< "Portcity " << pop1 << endl
<< "Hightown " << pop2 << endl
<< "Lowville " << pop3 << endl;
return 0; }
```

**Here's the output from this program:**

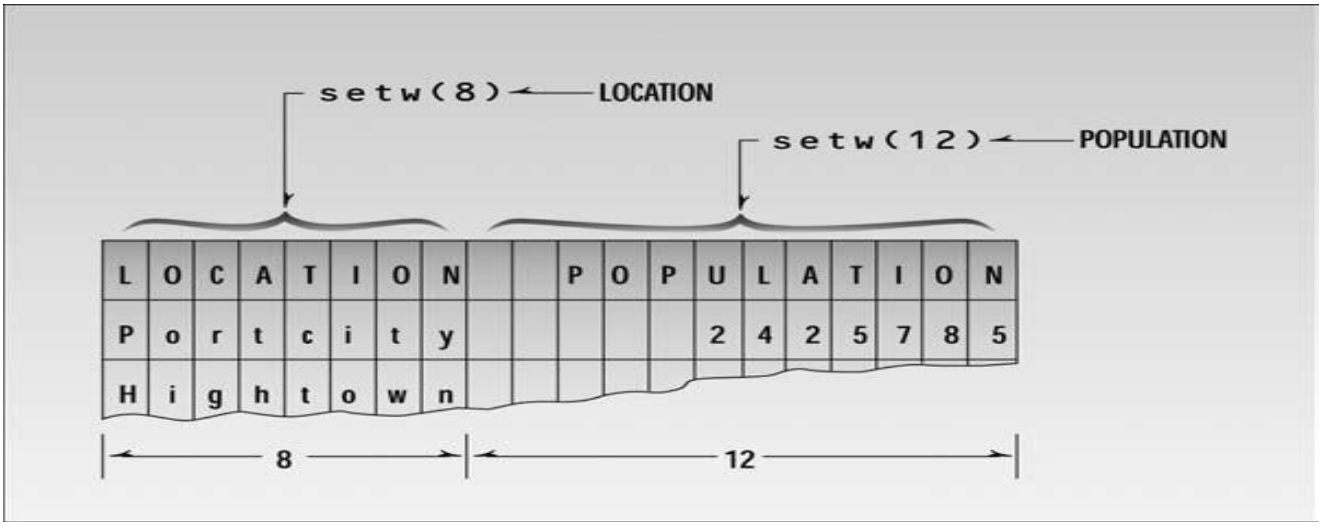
```
LOCATION POP.
Portcity 2425785
Hightown 47
Lowville 9761
```

**Here's a variation of this program, WIDTH2, that uses the setw manipulator.**

```
// width2.cpp
// demonstrates setw manipulator
#include <iostream.h>
#include <iomanip.h> // for setw
int main()
{
int pop1=2425785, pop2=47, pop3=9761;
cout << setw(8) << "LOCATION" << setw(12)
<< "POPULATION" << endl
<< setw(8) << "Portcity" << setw(12) << pop1 << endl
<< setw(8) << "Hightown" << setw(12) << pop2 << endl
```

**<< setw(8) << "Lowville" << setw(12) << pop3 << endl;**
**return 0;**

**The setw manipulator causes the number (or string) that follows it in the stream to be printed within a field n characters wide, where n is the argument to setw(n).**




Here's the output of WIDTH2:

```
LOCATION  POPULATION
Portcity     2425785
Hightown          47
Lowville        9761
```

**In previous the output was**
**LOCATION POP.**
**Portcity 2425785**
**Hightown 47**
**Lowville 9761**

**Variable Type Summary**

| | *Numerical Range* | | *Digits of* | *Bytes of* |
|---|---|---|---|---|
| *Keyword* | *Low* | *High* | *Precision* | *Memory* |
| `bool` | false | true | n/a | 1 |
| `char` | –128 | 127 | n/a | 1 |
| `short` | –32,768 | 32,767 | n/a | 2 |

## 1.4 Arithmetic Operators

**C++ uses the four normal arithmetic operators:**

**+**

**-**

*****

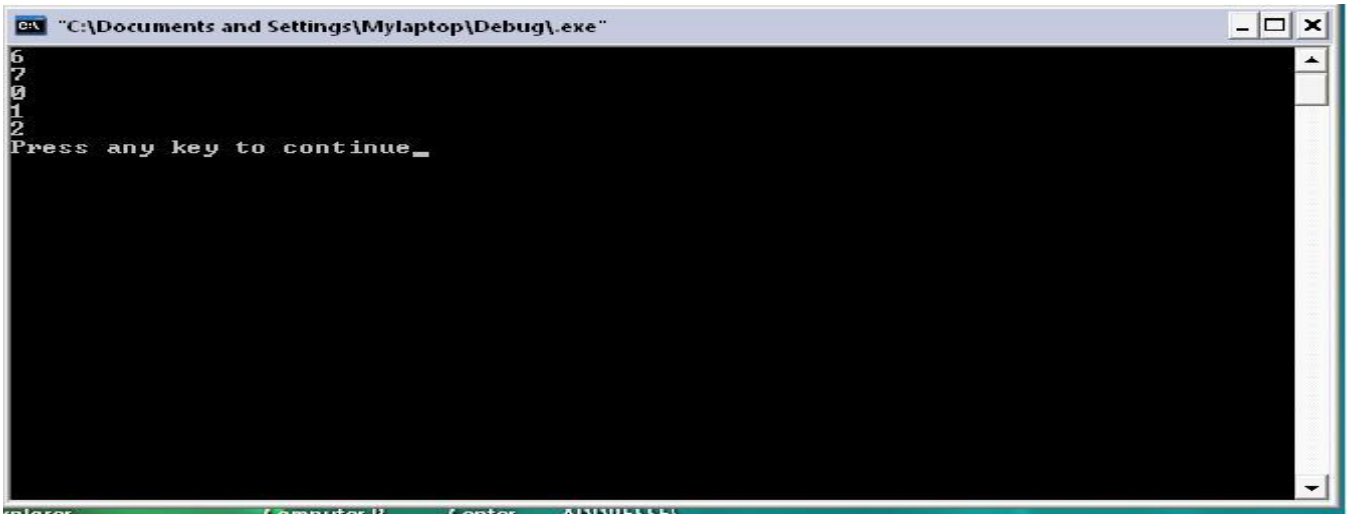
**/**

**for addition, subtraction, multiplication, and division. These operators work on all the data types, both integer and floating-point.**

**The Remainder Operator**

**There is a fifth arithmetic operator that works only with integer variables int. It's called the remainder operator, and is represented by the percent symbol (%). This operator (also called the modulus operator) finds the remainder when one number is divided by another. The REMAIND program demonstrates the effect.**

```
#include <iostream.h>
int main()
{cout << 6 % 8 << endl // 6
<< 7 % 8 << endl // 7
<< 8 % 8 << endl // 0
<< 9 % 8 << endl // 1
<< 10 % 8 << endl; // 2
return 0;}
```

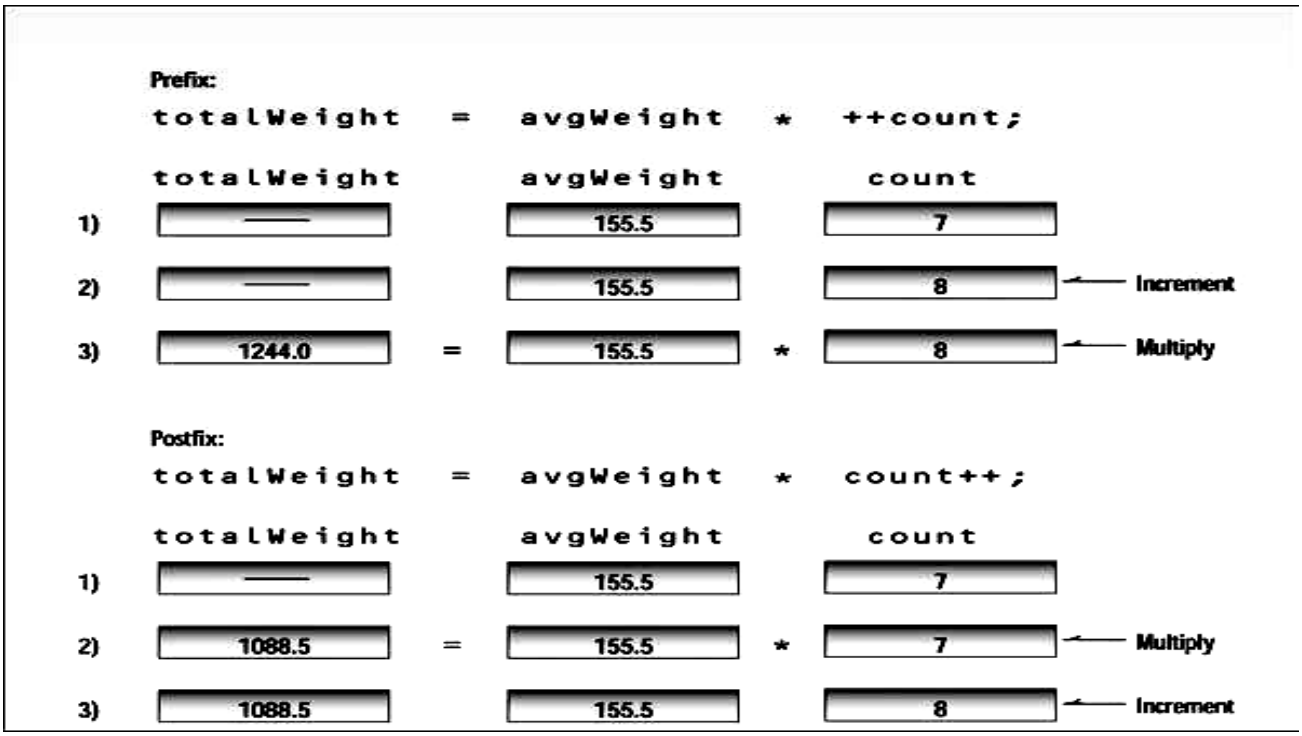


| Precedence | Operators |
|---|---|
| highest (applied first) | () |
| | * / % |
| | + - |
| | < <= > >= |
| | == != |
| lowest (applied last) | = |

**Increment Operators**
**count = count + 1; // adds 1 to "count"**
**++count; // adds 1 to "count"**
**The ++ operator increments (adds 1 to) its argument.**

## 1.5 Prefix and Postfix

**The increment operator can be used in two ways:**
**as a prefix, the operator precedes the variable; and as a postfix, the operator follows.**
**For example**
**totalWeight = avgWeight * ++count;**
**Is the multiplication performed before or after count is incremented? In this case count is incremented first. How do we know that? Because prefix notation is used ++count. If we had used postfix notation, count++, the multiplication would have been performed first, then count would have been incremented.**
**This is shown in Figure bellow:**




**Here's an example**

```
// increm.cpp
// demonstrates the increment operator
#include <iostream>
int main()
{
int count = 10;
cout << "count=" << count << endl; //displays 10
cout << "count=" << ++count << endl; //displays 11 (prefix)
cout << "count=" << count << endl; //displays 11
cout << "count=" << count++ << endl; //displays 11 (postfix)
cout << "count=" << count << endl; //displays 12
return 0; }
```

**Here's the program's output:**

```
count=10
count=11
count=11
count=11
count=12
```

**The Decrement (--) Operator**

**The decrement operator, --, behaves like the increment operator, except that it subtracts 1 from its operand.**

## 1.6 Library Functions

**Many activities in C++ are carried out by library functions. These functions perform file access, mathematical computations, and dat conversion. The next example, SQRT, uses the library function sqrt() to calculate the square root of a number entered by the user.**

```
// sqrt.cpp
// demonstrates sqrt() library function
#include <iostream.h> //for cout, etc.
#include <cmath.h> //for sqrt()
int main()
{
double number, answer; //sqrt() requires type double
cout << "Enter a number: ";
cin >> number; //get the number
answer = sqrt(number); //find square root
cout << "Square root is "
<< answer << endl; //display it
return 0;
}
```

**output from the program:**

```
Enter a number: 1000
Square root is 31.622777
```

**Exercises**

**3. Write a program that generates the following output:**

```
10
20
19
```

**Use an integer constant for the 10, an arithmetic assignment operator to generate the 20, and a decrement operator to generate the 19.**

**4. Write a program that displays your favorite poem. Use an appropriate escape sequence for the line breaks. If you don't have a favorite poem, you can borrow this one by Ogden Bash:**

**Candy is dandy,**
**But liquor is quicker.**

**5. A library function, islower(), takes a single character (a letter) as an argument and returns a nonzero integer if the letter is lowercase, or zero if it is uppercase. This function requires the header file CTYPE.H. Write a program that allows the user to enter a letter, and then displays either zero or nonzero, depending on whether a lowercase or uppercase letter was entered. (See the SQRT program for clues.)**

**6. On a certain day the British pound was equivalent to $1.487 U.S., the French franc was $0.172, the German deutschemark was $0.584, and the Japanese yen was $0.00955. Write a program that allows the user to enter an amount in dollars, and then displays this value converted to these four other monetary units.**

**7. You can convert temperature from degrees Celsius to degrees Fahrenheit by multiplying by 9/5 and adding 32. Write a program that allows the user to enter a floating-point number representing degrees Celsius, and then displays the corresponding degrees Fahrenheit.**

**Solution**
**3.**

```
// ex2_3.cpp
// exercises arithmetic assignment and decrement
#include <iostream.h>
int main()
{
int var = 10;
cout << var << endl; // var is 10
var *= 2; // var becomes 20
cout << var-- << endl; // displays var, then decrements it
cout << var << endl; // var is 19
return 0;
}
```

## 1.7 LOOPS AND DECISIONS

### Relational Operators

A relational operator compares two values. The values such as char, int, and float.

The comparison involves such relationships as equal to, less than, and greater than.

The result of the comparison is true or false.

**Our first program, RELAT, demonstrates relational operators in a comparison of integer variables and constants.**

```
// demonstrates relational operators
#include <iostream.h>
int main()
{
int numb;
cout << "Enter a number: "; cin >> numb;
cout << "numb<10 is " << (numb < 10) << endl;
cout << "numb>10 is " << (numb > 10) << endl;
cout << "numb==10 is " << (numb == 10) << endl;
return 0;}
```

**Here's the output when the user enters 20:**

```
Enter a number: 20
numb<10 is 0
numb>10 is 1
numb==10 is 0
```

**Here's the complete list of C++ relational operators:**

| *Operator* | *Meaning* |
|---|---|
| > | Greater than (greater than) |
| < | Less than |
| == | Equal to |
| != | Not equal to |
| >= | Greater than or equal to |
| <= | Less than or equal to |

**The first two lines are assignment statements that set the values of the variables harry and jane.**

```
jane = 44;          //assignment statement
harry = 12;         //assignment statement
(jane == harry)     //false
(harry <= 12)       //true
(jane > harry)      //true
(jane >= 44)        //true
(harry != 12)       // false
(7 < harry)         //true
(0)                 //false (by definition)
(44)                //true (since it's not 0)
```