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; }
```

Zainab Khyioon

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:

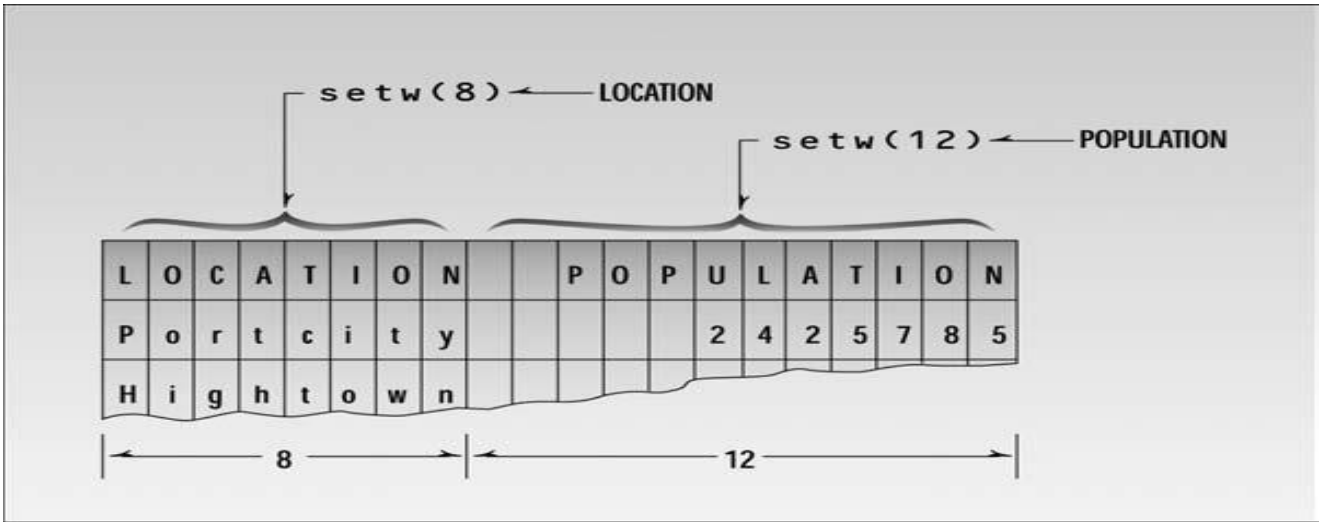
```
LOCATION POP.
Portcity 2425785
Hightown 47
Lowville 9761
```

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:

+

-

*****

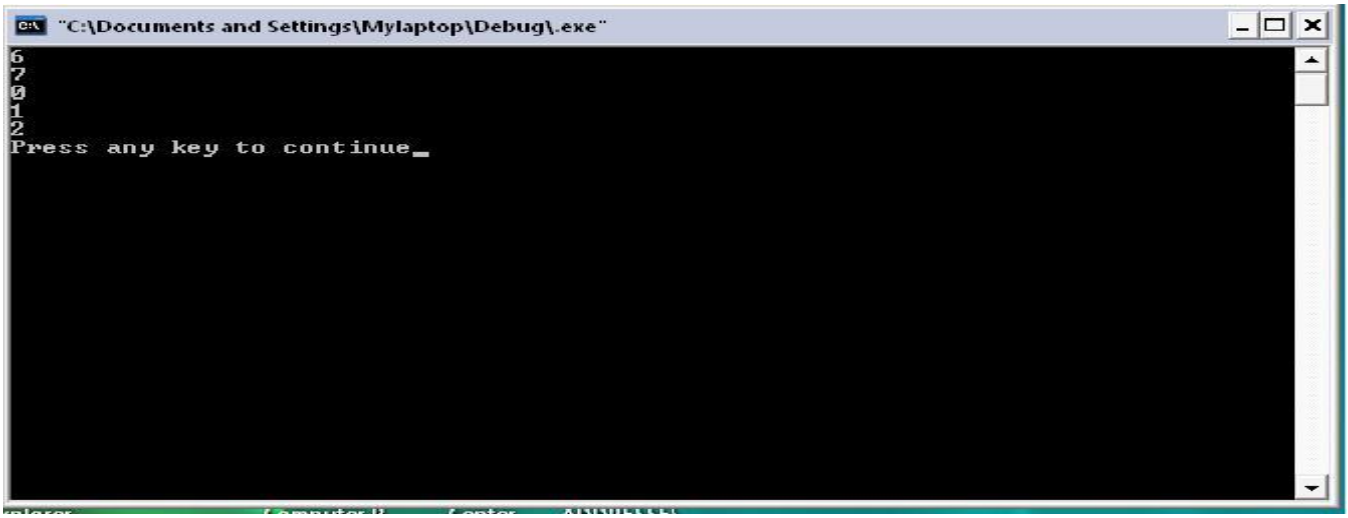
/

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;}
```

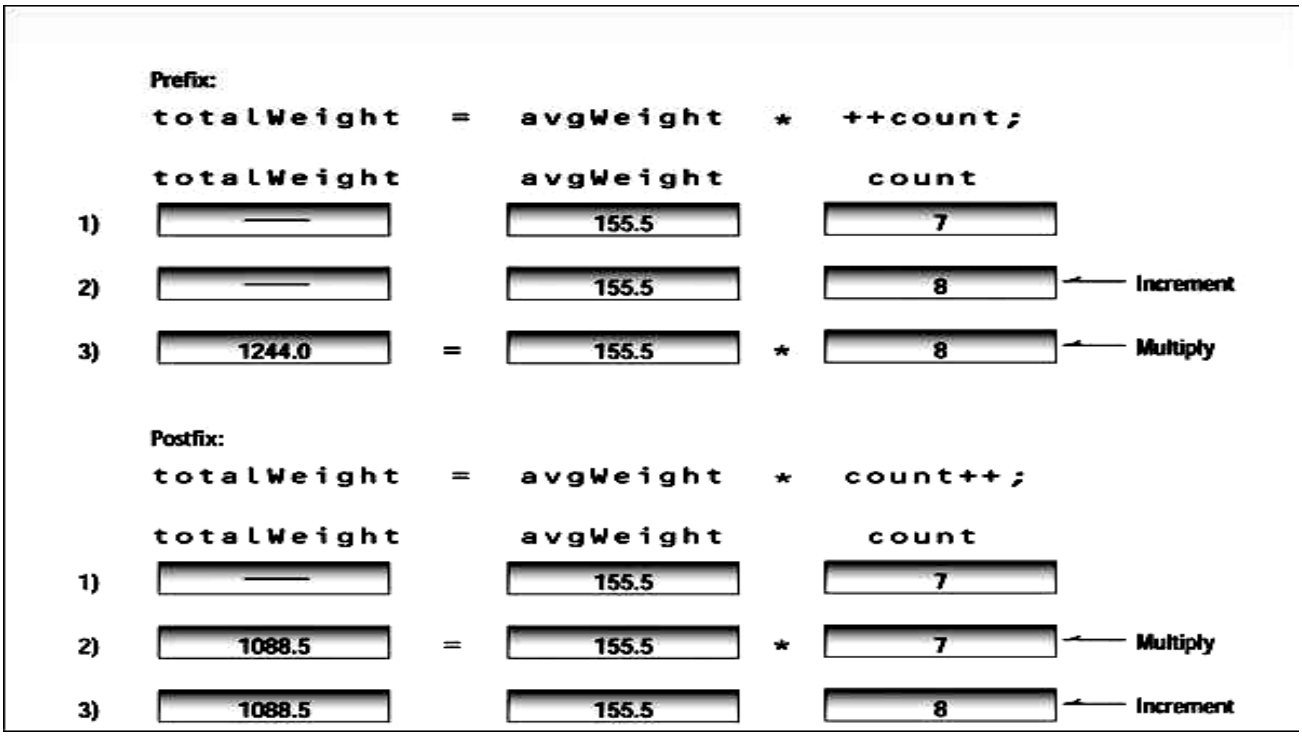


Precedence	Operators
highest (applied first)	()
	* / %
	+ -
	< <= > >=
	== !=
lowest (applied last)	=

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)
```

1.8 Loops

Loops cause a section of program to be repeated a certain number of times. The repetition continues while condition is true.

When the condition becomes false, the loop ends and control passes to the statements following the loop.

There are three kinds of loops in C++ :

1-the for loop,

2-the while loop,

3-the do loop.

1.8.1 The for Loop

The for loop executes a section of code a fixed number of times.It's usually used when you know, before entering the loop, how many times you want to execute the code.

Here's an example, FORDEMO, that displays the squares of the numbers from 0 to 14:

```
#include <iostream.h>
int main()
{
int i; //define a loop variable
for(i=0; i<15; i++) //loop from 0 to 14,
cout << i * i << " "; //displaying the square of j
cout << endl;
return 0;}
```

Here's the output:

0 1 4 9 16 25 36 49 64 81 100 121 144 169 196

Multiple Statements in the Loop Body

```
for(i=1; i<=10; i++)
{
     cout << i  ;
int cube = i*i*i    ;
cubecout << setw(6) << cube << endl   ;
}    //There is no ; here after the brace.
```

```
#include <iostream.h>
#include <iomanip.h> //for setw
int main()
{
int i;   //define loop variable
for(i=1; i<=10; i++) //loop from 1 to 10
{
cout << setw(4) << i;    //display 1st column
int cube = i*i*i; //calculate cube
cout << setw(6) << cube << endl; //display 2nd column
}
return 0;}
```

Here's the output from the program:

```
   1     1
   2     8
   3    27
   4    64
   5   125
   6   216
   7   343
   8   512
   9   729
  10  1000
```

Blocks and Variable Visibility

The loop body, which consists of braces delimiting several statements, is called a block of code. a variable defined inside the block is not visible outside it. Visible means that program statements can access or "see" the variable. In example the variable cube define inside the block.

```
For (i=1; i<=10; i++)
{
cout << setw(4) << i;
int cube = i*i*i;
cout << setw(6) << cube << endl;
}
```

You can't access this variable outside the block; it's only visible within the braces. The statement cube = 10; after the loop body is an Error because the variable cube would be undefined outside the loop.

In the next example it decrements the loop variable. This program, FACTOR, asks the user to type in a number, and then calculates the factorial of this number. Thus the factorial of 5 is 5*4*3*2*1, or 120.)

```
// calculates factorials, demonstrates FOR loop
#include <iostream.h>
int main()
{
int numb;
int fact=1; //long for larger numbers
cout << "Enter a number: ";
cin >> numb; //get number
for(int j=numb; j>0; j--) //multiply 1 by
fact *= j; //numb, numb-1, ..., 2, 1
cout << "Factorial is " << fact << endl;
return 0;}
}
```

The following output shows how large factorials can be, even for small input numbers:

```
Enter a number: 10
Factorial is 3628800
```

Questions

Answers to these questions can be found in Appendix G.

1. A relational operator

a. assigns one operand to another.

b. yields a Boolean result.
c. compares two operands.
d. logically combines two operands.
2. Write an expression that uses a relational operator to return true if the variable george is not equal to sally.
3. Is –1 true or false?
4. Name and describe the usual purpose of three expressions in a for statement.
5. In a for loop with a multistatement loop body, semicolons should appear following
a. the for statement itself.
b. the closing brace in a multistatement loop body.
c. each statement within the loop body.
d. the test expression.
6. True or false: The increment expression in a for loop can decrement the loop variable.
7. Write a for loop that displays the numbers from 100 to 110.
8. A block of code is delimited by _____.
9. A variable defined within a block is visible
a. from the point of definition onward in the program.
b. from the point of definition onward in the function.
c. from the point of definition onward in the block.
d. throughout the function.
10. Write a while loop that displays the numbers from 100 to 110.
11. True or false: Relational operators have a higher precedence than arithmetic operators.

Exercises

***1. Assume that you want to generate a table of multiples of any given number. Write a program that allows the user to enter the number and then generates the table, formatting it into 10 columns and 20 lines. Interaction with the program should look like this (only the first three lines are shown):**

Enter a number: 7

```
  7   14   21   28   35   42   49   56   63   70
 77   84   91   98  105  112  119  126  133  140
147  154  161  168  175  182  189  196  203  210
```

***2. Write a temperature-conversion program that gives the user the option of converting Fahrenheit to Celsius or Celsius to Fahrenheit. Then carry out the conversion. Use floating-point numbers. Interaction with the program might look like this:**

Type 1 to convert Fahrenheit to Celsius, 2 to convert Celsius to Fahrenheit: 1

Enter temperature in Fahrenheit: 70
In Celsius that's 21.111111

***Answers to Questions**

1. b, c
2. george != sally
3. –1 is true; only 0 is false.
4. The initialize expression initializes the loop variable, the test expression tests the loop variable, and the increment expression changes the loop variable.
5. c, d
6. true

7.

```
for(int j=100; j<=110; j++)
cout << endl << j;
```

8. braces (curly brackets)
9. c
10.

```
int j = 100;
while( j <= 110 )
cout << endl << j++;
```

11. false

***Solutions to Exercises**

1.

```
#include <iostream.h>
#include <iomanip.h> //for setw()
int main()
{int n; //number
cout << "\nEnter a number: ";
cin >> n; //get number
for(int j=1; j<=200; j++) //loop from 1 to 200
{
cout << setw(5) << j*n << " "; //print multiple of n
if( j%10 == 0 ) //every 10 numbers,
cout << endl; //start new line}
```

```
return 0;
}
```

2.

```
#include <iostream.h>
int main()
{   int response;
double temper;
cout << "\nType 1 to convert fahrenheit to celsius,"
<< "\n 2 to convert celsius to fahrenheit: ";
cin >> response;
if( response == 1 )
{
cout << "Enter temperature in fahrenheit: ";
cin >> temper;
cout << "In celsius that's " << 5.0/9.0*(temper-32.0);
   }
```

1.8.2 The while Loop

```
#include <iostream.h>
int main()
{
int n = 99; // make sure n is not initialized to 0
while( n != 0 ) // loop until n is 0
cin >> n; // read a number into n
return 0;
}
```

Here's some sample output.

```
1
27
144
9
0
```

Multiple Statements in a while Loop

it calculates the fourth power of a series of integers. Let's assume that in this program it's important to put the results in a column four digits wide.

1 1
2 16
3 81
4 256
5 625
6 1296
7 2401
8 4096
9 6561

To ensure that the results fit this column width, we must stop the loop before the results become larger than 9999.

```
// prints numbers raised to fourth power
#include <iostream.h>
#include <iomanip.h>                    //for setw
int main()
{ int pow=1;                            //power initially 1
int numb=1;                             //numb goes from 1 to ???
while( pow<10000 )                      //loop while power <= 4 digits
{ cout << setw(2) << numb;              //display number
cout << setw(5) << pow << endl;         //display fourth power
++numb; //get ready for next power
pow = numb*numb*numb*numb;   //calculate fourth power
}
cout << endl;
return 0;
}
```

1.9 The if Statement

The if statement is the simplest of the decision statements.

```
#include <iostream.h>
int main()
{
 int x;
cout << "Enter a number: "; cin >> x;
 if( x > 100 )
 cout << "That number is greater than 100\n";
 return 0;}
```

Enter a number: 2000
That number is greater than 100

Multiple Statements in the if body

```
#include <iostream.h>
int main()
{
int x;
cout << "Enter number";
cin >> x;
if( x > 100 )
{
cout << "The number " << x<<endl;
cout << " is greater than 100\n";
}
return 0;
}
```

Here's some output
Enter a number 12345
The number 12345
is greater than 100

Nesting ifs Inside Loops

Example:

PRIME, that nests an if within a for loop. This example tells you whether a number you enter is a prime number. (Prime numbers are integers divisible only by themselves and 1. The first few primes are 2, 3, 5, 7, 11, 13, 17.

```
#include <iostream.h>
#include <process.h> //for exit()
int main()
{
Int  n, j;
 cout << "Enter a number: ";
cin >> n;                          //get number to test
for(j=2; j <= n/2; j++)             //divide by every integer from
if(n%j == 0)                        //2 on up;if remainder is 0,
```

```
 {                                    //it's divisible by j
cout << "It's not prime; divisible by " << j << endl;
 exit(0);  //exit from the program
}
cout << "It's prime\n";
 return 0;
 }
```

Here's outputof the program:

```
 Enter a number: 13
It's prime
Enter a number: 22229
It's prime
Enter a number: 22231
It's not prime; divisible by 11
```

IF example, with an else added to the if:

```
#include <iostream.h>
int main()
{
int x;
cout << "\nEnter a number: ";
cin >> x;
if( x > 100 )
cout << "That number is greater than 100\n";
else
cout << "That number is not greater than 100\n";
return 0;
}
```

Here's output

```
Enter a number: 300
That number is greater than 100
Enter a number: 3
That number is not greater than 100
```

The operation of the if...else statement is shown

Questions

Answers to these questions can be found in Appendix G.

1. A relational operator
a. assigns one operand to another.
b. yields a Boolean result.
c. compares two operands.
d. logically combines two operands.
2. Write an expression that uses a relational operator to return true if the variable george is not equal to sally.
3. Is –1 true or false?
4. Name and describe the usual purpose of three expressions in a for statement.
5. In a for loop with a multistatement loop body, semicolons should appear following
a. the for statement itself.
b. the closing brace in a multistatement loop body.
c. each statement within the loop body.
d. the test expression.
6. True or false: The increment expression in a for loop can decrement the loop variable.
7. Write a for loop that displays the numbers from 100 to 110.
8. A block of code is delimited by _____.
9. A variable defined within a block is visible
a. from the point of definition onward in the program.
b. from the point of definition onward in the function.
c. from the point of definition onward in the block.
d. throughout the function.
10. Write a while loop that displays the numbers from 100 to 110.
11. True or false: Relational operators have a higher precedence than arithmetic operators.
12. How many times is the loop body executed in a do loop?
13. Write a do loop that displays the numbers from 100 to 110.
14. Write an if statement that prints Yes if a variable age is greater than 21.
15. The library function exit() causes an exit from
a. the loop in which it occurs.
b. the block in which it occurs.
c. the function in which it occurs.
d. the program in which it occurs.
16. Write an if...else statement that displays Yes if a variable age is greater than 21, and displays No otherwise.
17. The getche() library function
a. returns a character when any key is pressed.

b. returns a character when Enter is pressed.
c. displays a character on the screen when any key is pressed.
d. does not display a character on the screen.
18. What is the character obtained from cin when the user presses the Enter key?
19. An else always matches the _____ if, unless the if is _____.
20. The else...if construction is obtained from a nested if...else by _____.

Exercises
5. Use for loops to construct a program that displays a pyramid of Xs on the screen. The pyramid should look like this

```
X
XXX
XXXXX
XXXXXXX
XXXXXXXXX
```

except that it should be 20 lines high, instead of the 5 lines shown here. One way to do this is to nest two inner loops, one to print spaces and one to print Xs, inside an outer loop that steps down the screen from line to line. 6. Modify the FACTOR program in this chapter so that it repeatedly asks for a number and calculates its factorial, until the user enters 0, at which point it terminates. You can enclose the relevant statements in FACTOR in a while loop or a do loop to achieve this effect.

7. Write a program that calculates how much money you'll end up with if you invest an amount of money at a fixed interest rate, compounded yearly. Have the user furnish the initial amount, the number of years, and the yearly interest rate in percent. Some interaction with the program might look like this:

```
Enter initial amount: 3000
Enter number of years: 10
Enter interest rate (percent per year): 5.5
```

At the end of 10 years, you will have 5124.43 dollars. At the end of the first year you have 3000 + (3000 * 0.055), which is 3165. At the end of the second year you have 3165 + (3165 * 0.055), which is 3339.08. Do this as many times as there are years. A for loop makes the calculation easy.

8. Write a program that repeatedly asks the user to enter two money amounts expressed in old-style British currency: pounds, shillings, and pence. The program should then add the two amounts and display the answer, again in pounds, shillings, and pence. Use a do loop that asks the user whether the program should be terminated. Typical interaction might be
Enter first amount: £5.10.6
Enter second amount: £3.2.6
Total is £8.13.0
Do you wish to continue (y/n)?
To add the two amounts, you'll need to carry 1 shilling when the pence value is greater than 11, and carry 1 pound when there are more than 19 shillings.
10. Write another version of the program from Exercise 7 so that, instead of finding the final amount of your investment, you tell the program the final amount and it figures out how many years it will take, at a fixed rate of interest compounded yearly, to reach this amount. What sort of loop is appropriate for this problem? (Don't worry about fractional years; use an integer value for the year.)
Answers to Questions
1. b, c
2. george != sally
3. –1 is true; only 0 is false.
4. The initialize expression initializes the loop variable, the test expression tests the loop variable, and the increment expression changes the loop variable.
5. c, d
6. true
7.

```
for(int j=100; j<=110; j++)
cout << endl << j;
```

8. braces (curly brackets)
9. c
10.

```
int j = 100;
while( j <= 110 )
cout << endl << j++;
```

11. false
12. at least once
13.

```
int j = 100;
```

```
do
cout << endl << j++;
while( j <= 110 );
```

14.

```
if(age > 21)
cout << "Yes";
```

15.D

16.

```
if( age > 21 )
cout << "Yes";
else
cout << "No";
```

17. a, c

18. '\r'

19. preceding, surrounded by braces

20. reformatting

Solutions to Exercises

1.

```
#include <iostream.h>
#include <iomanip.h > //for setw()
int main()
{int  n; //number
cout << "\nEnter a number: ";
cin >> n; //get number
for(int j=1; j<=200; j++) //loop from 1 to 200
{
cout << setw(5) << j*n << " "; //print multiple of n
if( j%10 == 0 ) //every 10 numbers,
cout << endl; //start new line
}
return 0;}
```

2.

```
#include <iostream.h>
int main()
{
int response;
```

```
int temper;
cout << "\nType 1 to convert fahrenheit to celsius,"
<< "\n 2 to convert celsius to fahrenheit: ";
cin >> response;
if( response == 1 )
{
cout << "Enter temperature in fahrenheit: ";
cin >> temper;
cout << "In celsius that's " << 5.0/9.0*(temper-32.0);
} else
{
cout << "Enter temperature in celsius: ";
cin >> temper;
cout << "In fahrenheit that's " << 9.0/5.0*temper + 32.0;
}
cout << endl;
return 0;}
```

3.

```
#include <iostream.h>
#include <conio.h> //for getche()
int main()
{
char ch;
Int  total = 0; //this holds the number
cout << "\nEnter a number: ";
while( (ch=getche()) != '\r' ) //quit on Enter
total = total*10 + ch-'0'; //add digit to total*10
cout << "\nNumber is: " << total << endl;
return 0;
}
```

1.9.1 The switch Statement

```
#include <iostream.h>
int main()
{ int speed; cout << "\nEnter 33, 45, or 78: ";
  cin >> speed;
```

```
  switch(speed)
{
   case 33: cout << "USER SELECTED 33   \n";
   break;
   case 45: cout << "USER SELECTED 45 \n ";
   break;
   case 78:
   cout << "USER SELECTED 78 \n";
   break;  }
return 0;
}
```

Here's an example of the output:

```
Enter 33, 45, or 78: 45
```

You can also use type char as in next example.

```
#include <iostream.h>
#include <conio.h> //for getche()
int main()
{
char dir='a';
int x=10, y=10;
while( dir != '\r' )
   {
    cout << "\nYour location is " << x << ", " << y;
    cout << "\nEnter direction (n, s, e, w): ";
    dir = getche(); //get character
    switch(dir) //switch on it
      {
   case 'n': y--; break; //go north
   case 's': y++; break; //go south
   case 'e': x++; break; //go east
   case 'w': x--; break; //go west
   case '\r': cout << "Exiting\n"; break; //Enter key
   default: cout << "Try again\n"; //unknown char
       } //end switch
    } //end while
```

```
return 0;
}
```

These operators allow you to logically combine Boolean variables (that is, variables of type bool, with true or false values).

1.10 Logical AND Operator

```
#include <iostream.h>
#include <process.h> //for exit()
#include <conio.h> //for getche()
int main()
{
char dir='a';
int x=10, y=10;
while( dir != '\r' )
{cout << "\nYour location is " << x << ", " << y;
cout << "\nEnter direction (n, s, e, w): ";
dir = getche();
        switch(dir)
{
case 'n': y--; break; //update coordinates
case 's': y++; break;
case 'e': x++; break;
case 'w': x--; break;
}
if( x==7 && y==11 )
{cout << "\nYou found the treasure!\n";
exit(0);
 }}
return 0;
}
```

There are three logical operators in C++:

Operator	*Effect*
&&	Logical AND
||	Logical OR
!	Logical NOT

1.10.1 Logical OR Operator

```
#include <iostream.h>
#include <process.h> //for exit()
#include <conio.h> //for getche()
int main()
{char dir='a';  int x=10, y=10;
while( dir != '\r' )
{
cout << "\n\nYour location is " << x << ", " << y;
if( x<5 || x>15 )
cout << "\nBeware: dragons lurk here";
cout << "\nEnter direction (n, s, e, w): ";
dir = getche();
switch(dir)
{
case 'n': y--  ; break;
case 's': y++; break;
case 'e': x++; break;
case 'w': x-- ; break;
} //end switch
} //end while
return 0;
} //end main()
```

1.10.2 Logical NOT Operator

The logical NOT operator ! is a unary operator—that is, it takes only one operand. The effect of the ! is that the logical value of its operand is reversed: If something is true, ! makes it false;
if it is false, ! makes it true.
Precedence Summary

Operator type	*Operators*	*Precedence*
Unary	!, ++, --, +, -	Highest
Arithmetic	Multiplicative *, /, %	
	Additive +, -	
Relational	Inequality <, >, <=, >=	
	Equality ==, !=	
Logical	And &&	
	Or ||	
Conditional	?:	
Assignment	=, +=, -=, *=, /=, %=	Lowest

Questions

20. The else...if construction is obtained from a nested if...else by _____.

21. Write a switch statement that prints Yes if a variable ch is 'y', prints No if ch is 'n', and prints Unknown response otherwise.

22. Write a statement that uses a conditional operator to set ticket to 1 if speed is greater than 55, and to 0 otherwise.

23. The && and || operators

a. compare two numeric values.

b. combine two numeric values.

c. compare two Boolean values.

d. combine two Boolean values.

24. Write an expression involving a logical operator that is true if limit is 55 and speed is greater than 55.

25. Arrange in order of precedence (highest first) the following kinds of operators: logical, unary, arithmetic, assignment, relational, conditional.

26. The break statement causes an exit

a. only from the innermost loop.

b. only from the innermost switch.

c. from all loops and switches.

d. from the innermost loop or switch.

27. Executing the continue operator from within a loop causes control to go to _____.

Exercises

***4. Create the equivalent of a four-function calculator. The program should ask the user to a number, an operator, and another number. (Use floating point.) It should then carry out the specified arithmetical operation: adding, subtracting, multiplying, or dividing the two**

numbers. Use a switch statement to select the operation. Finally, display the result. When it finishes the calculation, the program should ask whether the user wants to do another calculation. The response can be 'y' or 'n'. Some sample interaction with the program might look like this:

Enter first number, operator, second number: 10 / 3
Answer = 3.333333
Do another (y/n)? y
Enter first number, operator, second number: 12 + 100
Answer = 112
Do another (y/n)? n

6. Modify the FACTOR program in this chapter so that it repeatedly asks for a number and calculates its factorial, until the user enters 0, at which point it terminates. You can enclose the relevant statements in FACTOR in a while loop or a do loop to achieve this effect.

7. Write a program that calculates how much money you'll end up with if you invest an amount of money at a fixed interest rate, compounded yearly. Have the user furnish the initial amount, the number of years, and the yearly interest rate in percent. Some interaction with the program might look like this:

Enter initial amount: 3000
Enter number of years: 10
Enter interest rate (percent per year): 5.5
At the end of 10 years, you will have 5124.43 dollars. At the end of the first year you have 3000 + (3000 * 0.055), which is 3165. At the end of the second year you have 3165 + (3165 * 0.055), which is 3339.08. Do this as many times as there are years. A for loop makes the calculation easy.

8. Write a program that repeatedly asks the user to enter two money amounts expressed in old-style British currency: pounds, shillings, and pence. The program should then add the two amounts and display the answer, again in pounds, shillings, and pence. Use a do loop that asks the user whether the program should be terminated. Typical interaction might be

Enter first amount: £5.10.6
Enter second amount: £3.2.6
Total is £8.13.0
Do you wish to continue (y/n)?

To add the two amounts, you'll need to carry 1 shilling when the pence value is greater than 11, and carry 1 pound when there are more than 19 shillings.

11. Create a three-function calculator for old-style English currency, where money amounts are specified in pounds, shillings, and pence. The calculator should allow the user to add or subtract two money amounts, or to multiply a money amount by a floating-point number. (It doesn't make sense to multiply two money amounts; there is no such thing as square money. We'll ignore division. Use the general style of the ordinary four-function calculator in Exercise 4)

12. Create a four-function calculator for fractions. Here are the formulas for the four arithmetic operations applied to fractions:
Addition: a/b + c/d = (a*d + b*c) / (b*d)
Subtraction: a/b - c/d = (a*d - b*c) / (b*d)
Multiplication: a/b * c/d = (a*c) / (b*d)
Division: a/b / c/d = (a*d) / (b*c)
The user should type the first fraction, an operator, and a second fraction. The program should then display the result and ask whether the user wants to continue.

Answers to Questions
20. reformatting
21.

```
switch(ch)
{
case 'y':
cout << "Yes";
break;
case 'n':
cout << "No";
break;
default:
cout << "Unknown response";
}
```

22. ticket = (speed > 55) ? 1 : 0;
23. d
24. limit == 55 && speed > 55
25. unary, arithmetic, relational, logical, conditional, assignment
26. d

27. the top of the loop
28. b

4.

```
#include <iostream.h>
int main()
{float n1, n2, ans;
char oper, ch;
do
{cout << "\nEnter first number, operator, second number: ";
cin >> n1 >> oper >> n2;
switch(oper)
{case '+': ans = n1 + n2; break;
case '-': ans = n1 - n2; break;
case '*': ans = n1 * n2; break;
case '/': ans = n1 / n2; break;
default: ans = 0;}
cout << "Answer = " << ans;
cout << "\nDo another (Enter 'y' or 'n')? ";
cin >> ch;
} while( ch != 'n' );
return 0;}
```