Programación Básica

Martin Méndez
Facultad de Ciencias
Universidad Autónoma de San Luis Potosí

Objetivo del Curso

- Estudiar y aplicar los conceptos básicos de programación estructurada en un lenguaje de alto nivel. Al final del curso.

- Al finalizar el programa el alumno deberá ser capaz de diseñar, implementar, y depurar algoritmos sencillos en lenguaje C/C++ o Pyton.

Temario

Unidad	Contenidos
1. Conceptos básicos de programación	1.1.- Estructura básica de un programa 1.2.- Salida a consola mediante cout 1.3.- Compilación y ejecución de un programa 1.4.- Variables y asignación 1.5.- Expresiones aritméticas y jerarquía de operadores 1.6.- Entrada de datos mediante cin 1.7.- Almacenamiento de variables en memoria 1.8.- Apuntadores y operadores de referenciación y dereferenciación 1.9.- Aritmética de apuntadores 1.10.- Ejemplos de programas sencillos
2. Estructuras de decisión	2.1.- Expresiones booleanas y operadores de comparación 2.2.- Operadores booleanos y el tipo bool 2.3.- Instrucción if…else 2.4.- Instrucciones if…else anidadas 2.5.- Instrucción switch 2.6.- Ejemplos de programas

Unidad	Contenidos
3. Estructuras de iteración	3.1.- Motivación para el uso de ciclos 3.2.- Instrucción while 3.3.- Instrucción do...while 3.4.- Instrucción for 3.5.- Instrucciones break y continue 3.6.- Ejemplos de programas
4. Funciones y programación estructurada	4.1.- Ejemplos de funciones de librería: la librería math.h 4.2.- Definición de funciones y paso de parámetros por valor 4.3.- Paso de parámetros por apuntador 4.4.- Paso de parámetros por referencia 4.5.- Funciones recursivas 4.6.- Programación estructurada: motivación y recomendaciones 4.7.- Creación de librerías: archivos de encabezado y de implementación

Unidad	Contenidos
5. Arreglos	5.1.- Motivación 5.2.- Declaración de un arreglo y acceso a sus elementos 5.3.- Recorrido de un arreglo 5.4.- Almacenamiento en memoria: relación entre arreglos y apuntadores 5.5.- Ejemplos de aplicación: sumatorias, histogramas, señales 5.6.- Arreglos bidimensionales y multidimensionales 5.7.- Cadenas de caracteres 5.8.- Manejo de cadenas: librería string.h
6. Introducción al manejo dinámico de memoria	6.1.- Asignación dinámica de memoria para una variable: operadores new y delete 6.2.- Asignación dinámica de memoria para un arreglo 6.3.- Consideraciones para el manejo dinámico de memoria

Evaluación

Exámen	Modalidad de Exámen
1	Examen teórico-práctico de las Unidades 1 y 2 con un peso máximo de 20%
2	Examen teórico-práctico de la Unidad 3 con un peso máximo de 20%
3	Examen teórico-práctico de la Unidad 4 con un peso máximo de 20%
4	Examen teórico-práctico de las Unidades 5 y 6 con un peso máximo de 20%
Ordinario	**Proyecto final con evaluación oral y un peso máximo de 30%**
Título	**Examen teórico-práctico con una duración mínima de 2 horas.**
Regularización	**Examen teórico-práctico con una duración mínima de 2 horas.**

Bibliografía

- C++ Como Programar. Deitel y Deitel. Prentice Hall, 2ª Edición, 1999.
- El Lenguaje de Programación C, Brian Kernighan, Dennis Ritchie, Ed. Prentice Hall, 2ª Edición, 1991.
- Métodos Numéricos para Ingenieros. S.C. Chapra, R.P. Canale. Ed. Mc Graw-Hill, 5ª Edición, 2007

Unidad 1

OBJECTIVES

In this chapter you'll learn:

- To write simple computer programs in C++.
- To write simple input and output statements.
- To use fundamental types.
- Basic computer memory concepts.
- To use arithmetic operators.
- The precedence of arithmetic operators.
- To write simple decision-making statements.

Programa para imprimir texto

```
1  // Fig. 2.1: fig02_01.cpp
2  // Text-printing program.
3  #include <iostream> // allows program to output data to the screen
4
5  // function main begins program execution
6  int main()
7  {
8     std::cout << "Welcome to C++!\n"; // display message
9
10    return 0; // indicate that program ended successfully
11
12 } // end function main
```

```
Welcome to C++!
```

// == comentario de línea; /* xxxxxxxxxxx */ == comentario de varias líneas

★ Todo programa debe comnezar con un comentario que describa su propósito, autor, fecha y hora.

La línea 3 == directiva del preprocesador, mensaje para el preprocesador de C++. Note que empieza con # y esta entre <>. Se indica al preprocesador que debe incluir el contenido del **archivo de encabezado de flujos de entrada y salida <iostream>**

C++ Keywords

Keywords common to the C and C++ programming languages

`auto`	`break`	`case`	`char`	`const`
`continue`	`default`	`do`	`double`	`else`
`enum`	`extern`	`float`	`for`	`goto`
`if`	`int`	`long`	`register`	`return`
`short`	`signed`	`sizeof`	`static`	`struct`
`switch`	`typedef`	`union`	`unsigned`	`void`
`volatile`	`while`			

C++ only keywords

`asm`	`bool`	`catch`	`class`	`const_cast`
`delete`	`dynamic_cast`	`explicit`	`false`	`friend`
`inline`	`mutable`	`namespace`	`new`	`operator`
`private`	`protected`	`public`	`reinterpret_cast`	
`static_cast`	`template`	`this`	`throw`	`true`
`try`	`typeid`	`typename`	`using`	`virtual`
`wchar_t`				

Instrucción

```
8     std::cout << "Welcome to C++!\n"; // display message
```

Comando

Operador

Cadena de Caracteres

Fin de la instrucción

Escape Sequence	Description
`\n`	Newline. Position the screen cursor to the beginning of the next line.
`\t`	Horizontal tab. Move the screen cursor to the next tab stop.
`\r`	Carriage return. Position the screen cursor to the beginning of the current line; do not advance to the next line.
`\a`	Alert. Sound the system bell.
`\\`	Backslash. Used to print a backslash character.
`\"`	Double quote. Used to print a double quote character.

```cpp
// Fig. 2.3: fig02_03.cpp
// Printing a line of text with multiple statements.
#include <iostream> // allows program to output data to the screen

// function main begins program execution
int main()
{
   std::cout << "Welcome ";
   std::cout << "to C++!\n";

   return 0; // indicate that program ended successfully

} // end function main
```

```
Welcome to C++!
```

```cpp
// Fig. 2.4: fig02_04.cpp
// Printing multiple lines of text with a single statement.
#include <iostream> // allows program to output data to the screen

// function main begins program execution
int main()
{
   std::cout << "Welcome\nto\n\nC++!\n";

   return 0; // indicate that program ended successfully

} // end function main
```

```
Welcome
to

C++!
```

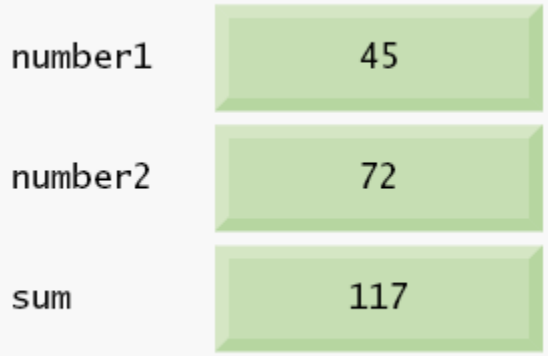
Sumando 2 enteros

```cpp
// Fig. 2.5: fig02_05.cpp
// Addition program that displays the sum of two integers.
#include <iostream> // allows program to perform input and output

// function main begins program execution
int main()
{
   // variable declarations
   int number1; // first integer to add
   int number2; // second integer to add
   int sum; // sum of number1 and number2

   std::cout << "Enter first integer: "; // prompt user for data
   std::cin >> number1; // read first integer from user into number1

   std::cout << "Enter second integer: "; // prompt user for data
   std::cin >> number2; // read second integer from user into number2

   sum = number1 + number2; // add the numbers; store result in sum

   std::cout << "Sum is " << sum << std::endl; // display sum; end line

   return 0; // indicate that program ended successfully

} // end function main
```

```
Enter first integer: 45
Enter second integer: 72
Sum is 117
```

Conceptos de Memoria

Variable names such as number1, number2 and sum actually correspond to **locations in the** computer's memory. Every variable has a name, a type, a size and a value.



Aritmética

C++ operation	C++ arithmetic operator	Algebraic expression	C++ expression
Addition	+	$f + 7$	`f + 7`
Subtraction	-	$p - c$	`p - c`
Multiplication	*	bm or $b \cdot m$	`b * m`
Division	/	x / y or $\frac{x}{y}$ or $x \div y$	`x / y`
Modulus	%	$r \bmod s$	`r % s`

Reglas de precedencia

Operator(s)	Operation(s)	Order of evaluation (precedence)
( )	Parentheses	Evaluated first. If the parentheses are nested, the expression in the inner-most pair is evaluated first. If there are several pairs of parentheses "on the same level" (i.e., not nested), they are evaluated left to right.
*, /, %	Multiplication, Division, Modulus	Evaluated second. If there are several, they are evaluated left to right.
+ -	Addition Subtraction	Evaluated last. If there are several, they are evaluated left to right.

(expresion) = Paréntesis; sirve para agrupar expresiones, ej. Multiplicar a*(b+c) es diferentes de a*b+c.

Ejemplos

Algebra: $m = \dfrac{a + b + c + d + e}{5}$

C++:
```
m = ( a + b + c + d + e ) / 5;
```

Algebra: $y = mx + b$

C++:
```
y = m * x + b;
```

Algebra: $z = pr\%q + w/x - y$

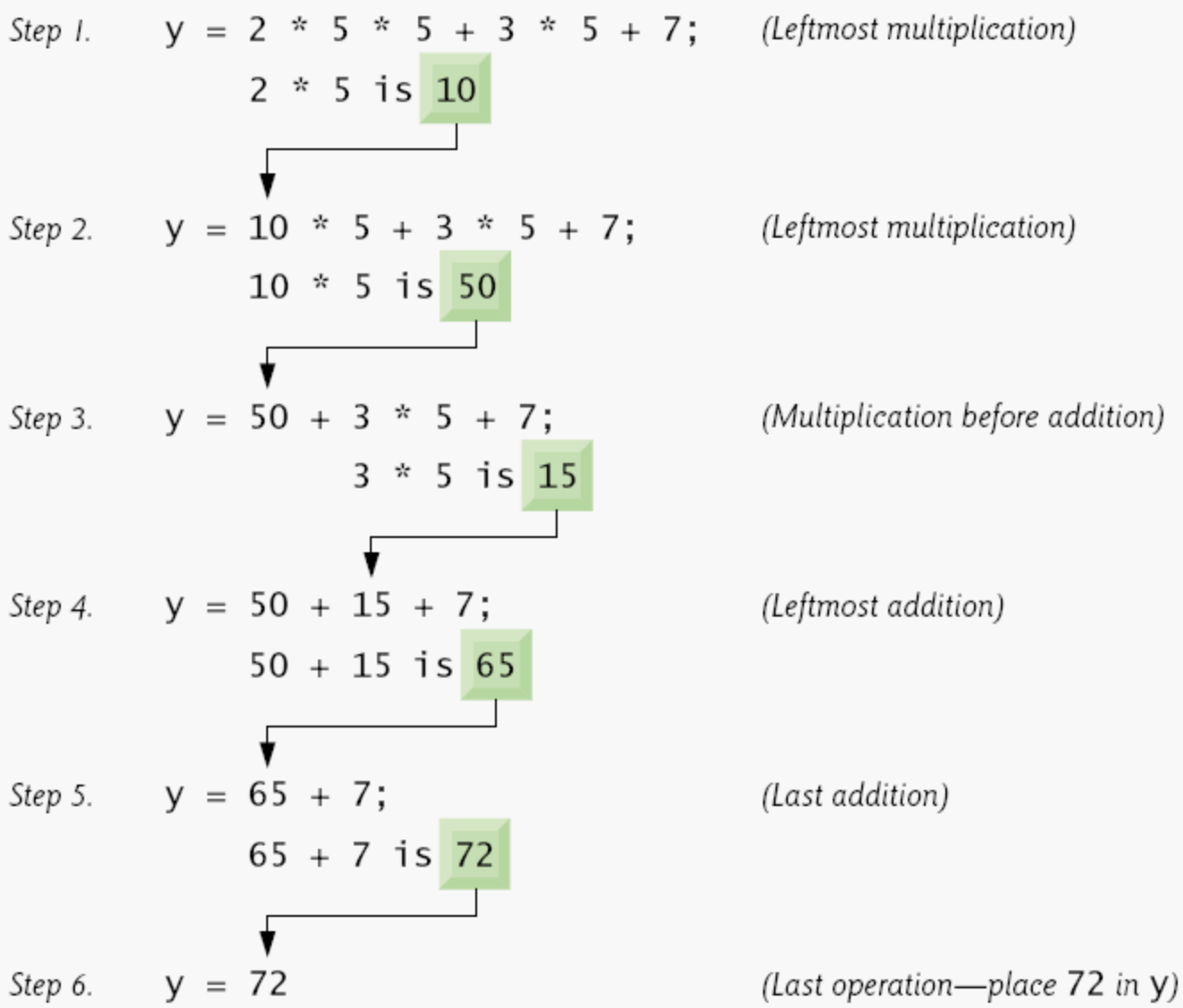
C++:
```
z = p * r % q + w / x - y;
```

6 1 2 4 3 5

of a second-degree polynomial ($y = ax^2 + bx + c$):

```
y = a * x * x + b * x + c;
```

6 1 2 4 3 5


Step 1.
y = 2 * 5 * 5 + 3 * 5 + 7;
2 * 5 is 10
(Leftmost multiplication)
Step 2.
y = 10 * 5 + 3 * 5 + 7;
10 * 5 is 50
(Leftmost multiplication)
Step 3.
y = 50 + 3 * 5 + 7;
3 * 5 is 15
(Multiplication before addition)
Step 4.
y = 50 + 15 + 7;
50 + 15 is 65
(Leftmost addition)
Step 5.
y = 65 + 7;
65 + 7 is 72
(Last addition)
Step 6.
y = 72
(Last operation—place 72 in y)

Toma de Decisiones: operadores de igualdad y relacionales

- Instrucción **if (Condición)**

Si se cumple la condición (es decir, si es verdadera), se ejecuta la instrucción que se encuentra en el cuerpo de la instrucción **if**.

Si la condición no se cumple (es falsa), el cuerpo no se ejecuta.

Standard algebraic equality or relational operator	C++ equality or relational operator	Sample C++ condition	Meaning of C++ condition
Relational operators			
>	>	x > y	x is greater than y
<	<	x < y	x is less than y
≥	>=	x >= y	x is greater than or equal to y
≤	<=	x <= y	x is less than or equal to y
Equality operators			
=	==	x == y	x is equal to y
≠	!=	x != y	x is not equal to y

```cpp
// Fig. 2.13: fig02_13.cpp
// Comparing integers using if statements, relational operators
// and equality operators.
#include <iostream> // allows program to perform input and output

using std::cout; // program uses cout
using std::cin; // program uses cin
using std::endl; // program uses endl

// function main begins program execution
int main()
{
   int number1; // first integer to compare
   int number2; // second integer to compare

   cout << "Enter two integers to compare: "; // prompt user for data
   cin >> number1 >> number2; // read two integers from user

   if ( number1 == number2 )
      cout << number1 << " == " << number2 << endl;

   if ( number1 != number2 )
      cout << number1 << " != " << number2 << endl;

   if ( number1 < number2 )
      cout << number1 << " < " << number2 << endl;

   if ( number1 > number2 )
      cout << number1 << " > " << number2 << endl;

   if ( number1 <= number2 )
      cout << number1 << " <= " << number2 << endl;

   if ( number1 >= number2 )
      cout << number1 << " >= " << number2 << endl;

   return 0; // indicate that program ended successfully

} // end function main
```

```
Enter two integers to compare: 3 7
3 != 7
3 < 7
3 <= 7
```

```
Enter two integers to compare: 22 12
22 != 12
22 > 12
22 >= 12
```

```
Enter two integers to compare: 7 7
7 == 7
7 <= 7
7 >= 7
```

Operators				Associativity	Type
()				left to right	parentheses
*	/	%		left to right	multiplicative
+	-			left to right	additive
<<	>>			left to right	stream insertion/extraction
<	<=	>	>=	left to right	relational
==	!=			left to right	equality
=				right to left	assignment

expression such as x = y = 0 is evaluated as if it had been written x = (y = 0)

Recuerde que:

First Program in C++: Printing a Line of Text

• Single-line comments begin with //. You insert comments to document your programs and improve their readability.

• Comments do not cause the computer to performany action when the program is run—they are ignored by the C++ compiler and do not cause anymachine-language object code to be generated.

• A preprocessor directive begins with # and is a message to the C++ preprocessor. Preprocessor directives are processed by the preprocessor before the program is compiled and don't end with a semicolon as C++ statements do.

• The line #include <iostream> tells the C++ preprocessor to include the contents of the input/output stream header file in the program. This file contains information necessary to compile programs that use std::cin and std::cout and operators << and >>.

• White space (i.e., blank lines, space characters and tab characters) makes programs easier to read. White-space characters outside of literals are ignored by the compiler.

• C++ programs begin executing at main, even if main does not appear first in the program.
• The keyword int to the left of main indicates that main "returns" an integer value.

• A left brace, {, must begin the body of every function. A corresponding right brace, }, must end each function's body.

• A string in double quotes is sometimes referred to as a character string, message or string literal. White-space characters in strings are not ignored by the compiler.

• Every statement must end with a semicolon (also known as the statement terminator).

• Output and input in C++ are accomplished with streams of characters.

• The output stream object std::cout—normally connected to the screen—is used to output data. Multiple data items can be output by concatenating stream insertion (<<) operators.

• The input stream object std::cin—normally connected to the keyboard—is used to input data. Multiple data items can be input by concatenating stream extraction (>>) operators.

• The std::cout and std::cin stream objects facilitate interaction between the user and the computer. Because this interaction resembles a dialog, it is often called conversational computing or interactive computing.

• The notation std::cout specifies that we are using a name, in this case cout, that belongs to "namespace" std.

• When a backslash (i.e., an escape character) is encountered in a string of characters, the next character is combined with the backslash to form an escape sequence.

• The escape sequence \n means newline. It causes the cursor (i.e., the current screen-position indicator) to move to the beginning of the next line on the screen.

• A message that directs the user to take a specific action is known as a prompt.

• C++ keyword return is one of several means to exit a function.

Another C++ Program: Adding Integers

• All variables in a C++ program must be declared before they can be used.

• A variable name in C++ is any valid identifier that is not a keyword. An identifier is a series of characters consisting of letters, digits and underscores (_). Identifiers cannot start with a digit. C++ identifiers can be any length; however, some systems and/or C++ implementations may impose some restrictions on the length of identifiers.

- C++ is case sensitive.

- Most calculations are performed in assignment statements.

- A variable is a location in the computer's memory where a value can be stored for use by a program.

- Variables of type int hold integer values, i.e., whole numbers such as 7, –11, 0, 31914.

Memory Concepts

• Every variable stored in the computer's memory has a name, a value, a type and a size.

• Whenever a new value is placed in a memory location, the process is destructive; i.e., the new value replaces the previous value in that location. The previous value is lost.

• When a value is read from memory, the process is nondestructive; i.e., a copy of the value is read, leaving the original value undisturbed in the memory location.

• The std::endl stream manipulator outputs a newline, then "flushes the output buffer."

Arithmetic

• C++ evaluates arithmetic expressions in a precise sequence determined by the rules of operator precedence and associativity.

• Parentheses may be used to group expressions.

• Integer division (i.e., both the numerator and the denominator are integers) yields an integer quotient. Any fractional part in integer division is truncated—no rounding occurs.

• The modulus operator, %, yields the remainder after integer division. The modulus operator can be used only with integer operands.

- Conditions in if statements are commonly formed by using equality operators and relational operators. The result of using these operators is always the value true or false.

- The declaration using std::cout; is a using declaration that informs the compiler where to find cout (namespace std) and eliminates the need to repeat the std:: prefix. Once we include this using declaration, we can, for example, write cout instead of std::cout in the remainder of a program.

Decision Making: Equality and Relational Operators

• The if statement allows a program to take alternative action based on whether a condition is met. The format for an if statement is
if (*condition*) *statement*;

If the condition is true, the statement in the body of the if is executed.

If the condition is not met, i.e., the condition is false, the body statement is skipped.

Ejercicios

Escriba un programa que: calcule el producto de 3 enteros

- Declare las variables x, y, z and result de typo int.
- Pida (Prompt) al usuario 3 enteros.
- Lea 3 enteros del teclado y almacenelos en las variables x, y, z.
- Compute el producto de los 3 enteros contenidos en las variables x, y, z, y asigne el resultado a al variable result.
- Imprima en pantalla “Ll producto es”, seguido del valor de la variable result.
- Regrese un valor de main indicando que el programa a terminado.

```cpp
// Calculate the product of three integers
#include <iostream> // allows program to perform input and output

using std::cout; // program uses cout
using std::cin; // program uses cin
using std::endl; // program uses endl

// function main begins program execution
int main()
{
   int x; // first integer to multiply
   int y; // second integer to multiply
   int z; // third integer to multiply
   int result; // the product of the three integers

   cout << "Enter three integers: "; // prompt user for data
   cin >> x >> y >> z; // read three integers from user
   result = x * y * z; // multiply the three integers; store result
   cout << "The product is " << result << endl; // print result; end line

   return 0; // indicate program executed successfully
} // end function main
```

¿A que ecuación algebraica corresponden las siguientes instrucciones?.

a) y = a * x * x * x + 7;
b) y = a * x * x * (x + 7);
c) y = (a * x) * x * (x + 7);
d) y = (a * x) * x * x + 7;
e) y = a * (x * x * x) + 7;
f) y = a * x * (x * x + 7);

Implemente el código para comprobar

¿Cuál es el valor de x?

a) x = 7 + 3 * 6 / 2 - 1;
b) x = 2 % 2 + 2 * 2 - 2 / 2;
c) x = (3 * 9 * (3 + (9 * 3 / (3))));

Implemente el código para comprobar

Desarrolle un programa que pida al usuario que escriba 2 números, que obtenga los números del usuario e imprima la suma, producto, diferencia y cociente de los números.

Desarrolle un programa imprima los números del 1 al 4 en la misma línea, con cada par de números adyacentes separado por un espacio. Haga esto de varias formas:

a) Utilizando una instrucción con un operador de inserción de flujo.
b) Utilizando una instrucción con 4 operadores de inserción de flujo.
c) Utilizando 4 instrucciones

Implemente un programa que pida al usuario que escriba 2 números enteros, que obtenga los números de usuario e imprima el número más grande, seguido de las palabras 'es más grande'. Si los número son iguales, imprima el mensaje 'Estos números son iguales'

Desarrolle un programa que pida al usuario que escriba 3 números, que obtenga los números del usuario e imprima la suma, producto, el promedio menor y mayor. Cada resultado debe estar en una línea diferente e indicar a que se refiere.

Implemente un programa que lea la base y la altura como número entero y que imprima su diámetro, perímetro y área.

Implemente un programa que lea el radio de un círculo como número entero y que imprima su diámetro, circunferencia y área. Use el valor de pi=3.14.1659. Realice todos los cálculos en instrucciones de salida.

Escriba un programa que imprima las siguientes figuras

Escriba un programa que lea 5 enteros y determine e imprima los enteros mayor y menor del grupo.

Escriba un programa que lea un entero y determine e imprima si es par o impar.

Escriba un programa que lea 2 enteros y determine e imprima si el primero es un múltiplo del segundo

Here is a peek ahead. In this chapter you learned about integers and the type int. C++ can also represent uppercase letters, lowercase letters and a considerable variety of special symbols. C++ uses small integers internally to represent each different character. The set of characters a computer uses and the corresponding integer representations for those characters are called that computer's character set. You can print a character by enclosing that character in single quotes, as with

cout << 'A'; // print an uppercase A

You can print the integer equivalent of a character using static_cast as follows:

cout << static_cast< int >('A'); // print 'A' as an integer

This is called a **cast operation.** When the preceding statement executes, it prints the value 65 (on systems that use the **ASCII character set).** Write a Program that prints the integer equivalent of a character typed at the keyboard. Store the input in a variable of type char. Test your program several times using uppercase letters, lowercase letters, digits and special characters (like $).

Write a program that inputs a five-digit integer, separates the integer into its individual digits and prints the digits separated from one another by three spaces each. [*Hint: Use the integer division* and modulus operators.] For example, if the user types in 42339, the program should print:

```
4   2   3   3   9
```

write a program that calculates the squares and cubes of the integers from 0 to 10 and uses tabs to print the following neatly formatted table of values:

```
integer square  cube
0       0       0
1       1       1
2       4       8
3       9       27
4       16      64
5       25      125
6       36      216
7       49      343
8       64      512
9       81      729
10      100     1000
```