1 Introducción

Objetivos de aprendizaje
Dominando los temas del presente capitulo Usted podrá.

1. Acceder a las características importantes de C++
2. La estructura general de un programa
3. Utilización de objetos flujo de salida y de entrada para la creación de programas simples

Fundamentos C++

Sin declaración using	Con declaración using

//programa para imprimir texto
#include <iostream>
 	
int main()
{
  std::cout << “Hola\n”;
 	
  return 0;
}
	//programa para imprimir texto
	#include <iostream>
	using std::cout; //
	using std::cin;
	using std::endl;
	
	int main()
	{
	  cout << “Hola” << endl;
	
	  return 0;

Instruccion	Descripcion
#include	Directiva del preprocesadpr
<iostream>	Componente de entrada/salida (objetos cin, cout,cerr)
using	Declaración que elimina necesidad de repetir el prefijo std.
int main()	Funcion principal que retorna un entero
{ }	Definición de un bloque de programa
std::cout	Uso del nombre cout del espacio de nombres std, dispositivo std de salida
::	Operador binario de resolución de alcance
<<	Operador de inserción en flujo
“Hola\n”	Literal Hola + salto de línea (también << std::endl;
;	Finalización de una sentencia
return 0	Punto de finalización correcta de la función

Si la función, como en este caso tiene un encabezado int main() debe tener al menos un return de un valor entero. Una función void nombre() puede finalizar con la instrucción return o sin ella.

Programa que muestra la suma de dos enteros

```
# include <iostream>
int main()
{
    // declaracion de variables
    int numero1;
    int numero2;
    int suma;
     std::cout << “Escriba el primer entero”;
    std::cin >>numero1;
     std::cout << “Escriba el segundo entero”;
    std::cin >>numero2;

    suma = numero1 + numero2;
    std::cout << “La suma de “ numero1 << “ + ” <<numero2 << “ es: “ << suma << std::endl;
    return 0;
}
```

Instrucción	Descripcion
cin	Dispositivo std de entrada
>>	Operador de extracción de flujo
cout	Dispositivo std de salida
<<	Operador de inserción en flujo
+	Operador de suma (unario/binario)
-	Operador de resta (unario/binario)
*	Operador multiplicativo
/	Operador de división
%	Operador de modulo o resto
( )	Operador para agrupar expresiones ej: a * (b+c)
==	Operador de igualdad
>	Mayor
>=	Mayor igual
<	Menor
>=	Menor igual
!=	Operador de desigualdad
=	Operador de asignación
+=	Asignación y suma x+=3; equivale a x = x + 3;
-=	Resta y asignación
*=	Multiplicación y asignación
/=	División y asignación
++	Operador de incremento
--	Operador de decremento
(? :)	Operador condicional (ternario)

Programa que comprara dos enteros, utiliza la declaración using

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
int main()
{
    int numero1;
    int numero2;
    cout << “Escriba dos enteros para comparar”;
    cin >>numero1 >> numero2;

    if (numero1 > numero2)
    cout <<  numero1 << “ > ” <<numero2 << std::endl;

    if (numero1 == numero2)
    cout <<  numero1 << “ == ” <<numero2 << std::endl;

    if (numero1 < numero2)
    cout <<  numero1 << “ < ” <<numero2 << std::endl;
    return 0;
}
```

Palabras Reservadas

C y C++

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

Solo C++

and and_eq asm bitand bitor bool catch class compl const_cast delete dynamic_cast explicit export false friend inline mutable namespace new not not_eq operator or or_eq private protected public reinterpret_cast static_cast template this throw true try typeid typename using virtual wchar_t xor xor_eq

Tipos de datos fundamentales y jerarquia de promoción

Tipos de Datos
long double
double
float
unsigned long int
long int
unsigned int
int
unsigned short int
short int
unsigned char
char
bool

Espacios de nombres

include <iostream>

Sin incluir directiva ni declaracion using

…..

std::cout << “Hola” << std::endl;

……..

#include <iostream>

Con declaracion using

//la declaración using de cada elemento solo incorpora los nombres especificados

using std::cin; usa el objeto cin del espacio std

using std::endl;

cout << “Hola” <<endl;

……

#include <iostream>

Con directiva using

//la directiva using incorpora la totalidad de los nombres del espacio de nombre

using namespace std; usa el espacio de nombre std

cout << “Hola” <<endl;

……

Espacios de nombres

Un programa incluye muchos identificadores definidos con distintos alcances. Una variable de un alcance se “traslapa” si entra en conflicto con una del mismo nombre y distinto alcance.para evitar esto se utilizan los espacios de nombres.

Para acceder a un miembro de un espacio de nombre se lo califica con el nombre del espacio, el operador de resolución de alcance y el nombre del miembro std::cout ….

Otras forma de uso de los espacios de nombre es mediante declaración using o directiva using.

//demostración de espacios de nombre

```
#include <iostream>
using namespace std; //usa el espacio de nombre std
int entero1 = 98 //variable global
//crea espacio de nombres ejemplo
namespace Ejemplo//declara dos constantes y una variable
{
 const doublé PI = 3.14;
 const doublé E  = 2.71;
 int entero1 = 8;
 void imprimirValores(); //prototipo
      namespace interno// espacio de nombres anidado
      {
         enum Anios {FISCAL1 = 1990, FISCAL2, FISCAL3};
       } // fin espacio de nombre interno
} //fin espacio de nombre Ejemplo

namespace //crea espacio de nombre sin nombre
{
    Doublé doubleSinNombre = 88.22;
}
```

```
Int main ()
  cout << doubleSinNombre;
  cout << entero1; //imprime la variable global
  cout << Ejemplo::entero1; // imprime entero1 del espacio de nombre Ejemplo
  Ejemplo::imprimirValores(); //invoca a la función
  Cout << Ejemplo::Interno::FISCAL1; //imprime el valor del espacio de nombre interno
```

Resumen:

1. Los comentarios de una linea comienzan con //, esta línea es omitida por el compilador.
2. Las directivas del preprocesador comienzan con #, estas líneas no terminan con ";" ya que no son parte de C. Permiten incorporar archivos de encabezado. <iostream> contiene información necesaria para utilizar cin y cout-
3. Los programas en C ejecutan la función main.
4. Toda sentencia termina con ;
5. La declaración using std::cout informa al compilador que puede encontrar a cout en el espacio de nombre std y elimina la necesidad de repetir el prefijo std.

Ejercicios

1. Cuál de las siguientes sentencias son correctas para la ecuación algebraica $y=ax^3 + 7$.
    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)
2. Escriba un programa que pida al ususrio dos números e informe la suma, la resta, el producto y el cociente de los mismos
3. Imprima un programa que imprima los números del 1 al 4 en una misma línea, hágalo de las formas siguientes:
    a. Utilizando un solo operador de inserción de flujo
    b. Una única sentencia con 4 operadores de inserción de flujo
    c. Utilizando cuatro sentencias
4. Escriba un programa que reciba tres números por el teclado e imprima la suma, el promedio, el producto, el mayor y el menor de esos números. Escriba un adecuado dialogo en pantalla.
5. Escriba un programa que reciba un numero que represente el radio de un circulo e imprima el diámetro, circunferencia y área.
6. Que imprime el siguiente código
    a. std::cout << "*\n**\n***\n****" <<std::endl;
    b. std::cout << 'A';
    c. std::cout << static_cast< int > 'A'; (que es static_cast? Investigue.)
7. Utilizando solo lo que hemos desarrollado en esta introduccion escriba un programa que calcule los cuadrados y los cubos de los números de 0 a 10 y los muestre por pantalla.
8. Escriba un programa que reciba un numero entero de 5 digitos, que separe el numero en sus digitoe y los muestre por pantalla, uno por línea comenzando por elmas significacivo en la primera línea.

2 Estructuras de control

Objetivos de aprendizaje
Dominando los temas del presente capitulo Usted podrá.

1. Comprender técnicas básicas de descomposición
2. Utilizar análisis de casos, o estructuras de selección
3. Utilizar repeticiones
4. Controlar repeticiones

Sentencia de selección if

```
if (expresión) {lista de sentencias};[else {lista de sentencias}]
Sin clausula else
if (nota >= 4){
   cout << “Aprobado”;
}

Con clausula else
if (nota >= 4){
   cout << “Aprobado”;
}
else{
  cout << “No aprobado”;
}

Anidados
If (nota > 9)
    cout << “Distinguido”;
else if (nota > 8)
    cout << “Sobresaliente”;
else if(nota > 6)
    cout << “Muy bueno”;
else if (nota >= 4)
    cout << “Bueno”;
else
    cout << “No aprobo”;
```

Estructura de repeticion
while

```
int i = 0;
while (I < 10 ){
    i++;
}
```

for

```
for (int I = 0; I < 10;i++)
        ;
```

Desarrollar una funcion que calcule el promedio de varios valores

```
#include <iostream>
using std::cout;
using std::cin;
using std::endl;
int calcularPromedio() //con cantidad conocida a priori ejemplo 10 valores
{
    Int cantidad, valor, suma;
    Suma = 0;
    cantidad = 1;
    while  (cantidad <= 10){
        cout >> “Ingrese un valor “;
        cin >> valor; cout << endl;
        suma +=valor;
        cantidad ++;
    }
    return suma/cantidad;
}
Alternative con ciclo for
    for(cantidad = 1, suma = 0; cantidad <=10; cantidad ++){
        ……..
int calcularPromedio() //con fin por valor centinela
{
    Int cantidad, valor, suma;
    Suma = 0;
    cantidad = 0;
    cout >> “Ingrese un valor “;
    cin >> valor;
    while  (valor > 0){
        suma +=valor;
        cantidad ++;
        cout >> “Ingrese un valor “;
        cin >> valor;
    }
    return (cantidad > 0?suma/cantidad: 0;
}
Alternativa con ciclo for
    for(cantidad = 0, suma = 0; valor > 0; cantidad ++){
```

Anexo I Introducción a las clases y objetos

3

Objetivos de aprendizaje
Dominando los temas del presente capitulo Usted podrá.

1. Comprender que son clases, objetos, funciones miembro y miembros de datos
2. Como definir clases y funciones miembro

Definición de una clase con una función miembro sin parametros

```
#include<iostream>
using std::cin;
using std::cout;

//definicion de la clase
class LibroCalificaciones
{
public:
        void MostrarMensaje()
        {
                cout << “Bienvenidos” << endl;
        }

};  // fin de definición de la clase


int main()
{
    LibroCalificaciones miLibroCalificaciones; //Crea un objeto de la clase LibroCalificaciones

    miLibroCalificaciones.MostrarMensaje(); //invoca a la función miembro del objeto

    return 0;
}
```

Definición de una clase con función miembro con parámetros

```
#include<iostream>
using std::cin;
using std::cout;

#include<string> //pone a disposicion la clase string
using std::string;
using std::getline;

//definicion de la clase
class LibroCalificaciones
{
public:
        void MostrarMensaje(string nombreCurso)
        {
                cout << “Bienvenidos” << nombreCurso << endl;
        }
};  // fin de definición de la clase

int main()
{
    string nombreDelCurso;
    LibroCalificaciones miLibroCalificaciones; //Crea un objeto de la clase LibroCalificaciones
    cout << “Escriba el nombre del curso” << endl;
    getline (cin, nombreDelCurso)


    miLibroCalificaciones.MostrarMensaje(nombredelCurso);
    return 0;
}
```

Agregar otras funciones

```
#include<iostream>
using std::cin;
using std::cout;

#include<string> //pone a disposicion la clase string
using std::string;
using std::getline;
//definicion de la clase
class LibroCalificaciones
{
public:
        void establecerNombreCurso(string nombre)
        {
           nombreCurso = nombre;
        }
```

```
        string obtenerNombreCurso()
        {
                return nombreCurso;
        }

        void mostrarMensaje()
        {
                cout << "Bienvenidos" << obtenerombreCurso() << endl;
        }

private: //especificador de acceso para miembros solo accedidos por funciones de la clase
        string nombreCurso

}; // fin de definición de la clase

int main()
{
   string nombreDelCurso;
   LibroCalificaciones miLibroCalificaciones; //Crea un objeto de la clase LibroCalificaciones
   cout << "El nombre inicial del curso es" << miLibroCalificaciones.obtenerNombreCurso();

   cout << "Escriba el nombre del curso" << endl;
   getline (cin, nombreDelCurso);
   miLibroCalificaciones.establecerNombreCurso(nombreDelCurso);

   miLibroCalificaciones.MostrarMensaje();
   return 0;
}
```

Las variables que se declaran en el cuerpo de la función se las denominan **variables locales**, las variables se pierden al terminar la función (salvo si son declaradas static). Las funciones de la clase se denominan **funciones miembro**. Los atributos son variables que se declaran en la definición de una clase pero fuera de las funciones miembros de la clase. Estas variables se llaman **miembros de datos**. En el ejemplo anterior la clase libroCalificaciones contiene un miembro de datos llamado nombreCurso.

Cuando se crea un objeto de la clase libroCalificaciones su miembro de datos nombreCurso se inicializa con la cadena vacia. Las clases pueden tener un constructor, es una función especial que se define con el nombre de la clase, no devuelven valores (tampoco void), si la clase no posee constructor el compilador proporciona un constructor predeterminado.

Incializar objetos mediante constructores

```
#include<iostream>
using std::cin;
using std::cout;
#include<string> //pone a disposicion la clase string
using std::string;
using std::getline;
//definicion de la clase
```

```
class LibroCalificaciones
{
public:
        LibroCalificaciones (string nombre) // el constructor
        {
            establecerNombreCurso(nombre);
        }

        void establecerNombreCurso(string nombre)
        {
           nombreCurso = nombre;
        }

        string obtenerNombreCurso()
        {
                return nombreCurso;
        }

         void mostrarMensaje()
        {
                cout << "Bienvenidos" << obtenerombreCurso ()<< endl;
        }

private: //especificador de acceso para miembros solo accedidos por funciones de la clase
        string nombreCurso

};  // fin de definición de la clase

int main()
{
    LibroCalificaciones miLibroCalificaciones("Curso Algoritmos");

    cout << "Se creo el curso " << miLibroCalificaciones.obtenerNombreCurso;
    return 0;
}
```

Diagrama de clase UML de la clase LibroCalificaciones

LibroCalificaciones
- nombreCurso: string
<constructor> + LibroCalificaciones (nombre:string) + establecerNombreCurso (nombre : string) + obtenerNombreCurso(): string + mostrarMensaje()

Colocar una clase en un archivo separado (cabecera) para su reutilización

```
//LibroCalificaciones.h para definer la clase separada de la function main()
#include<iostream>
```

```
using std::cin;
using std::cout;
#include<string> //pone a disposicion la clase string
using std::string;

//definicion de la clase
class LibroCalificaciones
{
public:
        LibroCalificaciones (string nombre) // el constructor
        {
            establecerNombreCurso(nombre);
        }

        void establecerNombreCurso(string nombre)
        {
           nombreCurso = nombre;
        }

        string obtenerNombreCurso()
        {
                return nombreCurso;
        }

         void mostrarMensaje()
        {
                cout << "Bienvenidos" << obtenerombreCurso ()<< endl;
        }

private: //especificador de acceso para miembros solo accedidos por funciones de la clase
        string nombreCurso

};  // fin de definición de la clase
```

Un programa que utiliza la clase definida por el usuario

```
#include<iostream>
using std::cin;
using std::cout;
#include "LibroCalificaciones.h" // clase de usuario

int main()
{
   LibroCalificaciones miLibroCalificaciones("Curso Algoritmos");
   cout << "Se creo el curso " << miLibroCalificaciones.obtenerNombreCurso;
   return 0;
}
```

Separar la interfaz de la implementación

Interfaz de la clase: la interfaz establece que operaciones se pueden hacer sin determinar como hacerlas. Son las funciones miembro public de la clase.
Para separar la interfaz de la implementación (para ocultar los detalles de la implementación) se divide la implementación de la clase en dos archivos, el archivo encabezado .h y el archivode código fuente .cpp
Definicion de la interfase

```
//LibroCalificaciones.h para definer la interfase de la clase
#include<iostream>
#include<string> //pone a disposicion la clase string
using std::string;

//definicion de la clase
class LibroCalificaciones
{
public:
        LibroCalificaciones (string nombre); // el constructor
        void establecerNombreCurso(string nombre);
        string obtenerNombreCurso():
        void mostrarMensaje();

private: //especificador de acceso para miembros solo accedidos por funciones de la clase
        string nombreCurso

};  // fin de definición interfase de la clase
```

Detalle de la implementacion

```
//LibroCalificaciones.h para definer la clase separada de la function main()
#include<iostream>
using std::cin;
using std::cout;
#include "LibroCalificaciones.h"
//definicion de la clase
class LibroCalificaciones
{
public:
        LibroCalificaciones::LibroCalificaciones (string nombre) // el constructor
        {
            establecerNombreCurso(nombre);
        }

        void LibroCalificaciones::establecerNombreCurso(string nombre)
        {
           nombreCurso = nombre;
        }

        string LibroCalificaciones::obtenerNombreCurso()
        {
                return nombreCurso;
```

```
        }

        void LibroCalificacionesmostrarMensaje()
        {
                cout << "Bienvenidos" << obtenerombreCurso ()<< endl;
        }

private: //especificador de acceso para miembros solo accedidos por funciones de la clase
        string nombreCurso

};  // fin de definición de la clase
```

En el caso de necesitar que una cadena no supere un tamaño determinado, por ejemplo si se desea que la cadea tenga un máximo de 25 caracteres se puede resolver utilizando algunos métodos de la clase string como sigue

```
void estanlecerNombreCurso(string nombre)
{
        If(nombre.length() <= 25)
           nombreCurso = nombre;
        else
           nombreCurso = nombre.substr(0, 25);
}
```

Resumen

1. Para realizar una tarea un programa requiere funciones. Estas ocultan las tareas complejas que realizan
2. Una función en una clase es una función miembro de la clase.
3. Para que un programa pueda realizar las tareas que se describe en la clase se debe crear un objeto de esa clase.
4. Cada llamada a una función miembro es un mensaje.
5. Un objeto tiene atributos que se especifican como datos miembros de la clase.
6. Las definiciones de las clases definen en los datos miembro los atributos y en las funciones miembro el comportamiento.
7. Las clases comienzan con la palabra reservada class, pon convención su nombre y el del constructor se escriben en PascalCase, y los miembros en camelCase.
8. Los miembros public pueden ser invocados por otras funciones.
9. Cada clase creada es un nuevo tipo lo que hace extensible al lenguaje.
10. Para llamar a una función miembro de un objeto se utiliza el operador ".".
11. Cuando un programa instancia un objeto de una clase sus miembros de dato private se encapsulan y solo las funciones miembro de la clase lo pueden utilizar.
12. Cuando las funciones se empaquetan adecuadamente los programadores pueden reutilizarlas.
13. Las interfaces definen la forma de interactuar con la clase.

Ejercicios

1. Defina la diferencia entre el prototipo y la definición de una función
2. Que es un constructor predeterminsdo?

3. Cual es el propósito de un miembro dato?
4. Como puede un programa utilizar la clase string sin la declaración using?
5. Cree una clase llamada fecha que incluya como miembro de datos un dia, un mes y un año, todos de tipo entero. Proporcione una función establecer y una función obtener para cada miembro y una función mostrar fecha que muestre año, mes y dia separado por /.