FIME

FACULTAD DE INGENIERÍA MECÁNICA Y ELÉCTRICA

TDW

Modulo II

C++

unidos por un cambio transformador

VIVE LA FIME

Instructor:

M.S.I. Mario Alejandro Rocha Moreno

Correo electrónico: thealezroc@gmail.com

COORDINACION DE EDUCACIÓN CONTINUA

unidos por un cambio transformador

VIVE LA FIME

PONDERACIÓN

Prácticas 40%
Tareas 10%
Proyecto 20%
Examen 30%

-Más de una falta equivale a repetir modulo
-Dos retardos es igual a una falta

unidos por un cambio transformador

VIVE LA FIME

Unidad 1
Introducción al Lenguaje C++

Historia del lenguaje C++

El lenguaje C++ se comenzó a desarrollar en el 1980 su autor fue B. Stroustrup, también de la AT&T. Al comienzo era una extensión del lenguaje C que fue denominada C with Classes. Este nuevo lenguaje comenzó a ser utilizado fuera de la AT&T en 1983.

El nombre C++ es también de ese año y hace referencia al carácter de incremento de C (++). Ante la gran difusión y éxito que iba obteniendo en el mundo de desarrolladores, la AT&T comenzó a estandarizarlo internamente en 1987. En 1989 se formó un comité ANSI (seguido algún tiempo después por un comité ISO) para estandarizarlo a nivel americano e internacional

Lenguaje C++ en nuestros tiempos

En la actualidad, el lenguaje C++ es versátil, potente y general. Su éxito entre los programadores profesionales le ha llevado a ocupar el primer puesto como herramienta de desarrollo de aplicaciones.

El lenguaje C++ mantiene las ventajas del C en cuanto a riqueza de operadores y expresiones, flexibilidad, concisión y eficiencia. Además, ha eliminado algunas limitaciones del Lenguaje C original. La evolución del Lenguaje C++ ha continuado con la aparición del lenguaje JAVA, un lenguaje creado simplificando algunas cosas de C++ y añadiendo otras, que se utilizan para el desarrollo de aplicaciones en internet.

Las Principales ventajas que presenta el lenguaje C++ son:

1. Difusión: al ser uno de los lenguajes más empleados en la actualidad posee un gran número de usuarios y existe una gran cantidad de libros, cursos, páginas web etc. Dedicados a él.
2. **Versatilidad:** El lenguaje C++ es un lenguaje de propósito general, por lo que se puede emplear para resolver problemas de cualquier tipo.
3. **Portabilidad:** El lenguaje esta estandarizado y un mismo código fuente se puede compilar en diversas plataformas.
4. **Eficiencia:** Es uno de los lenguajes más rápidos en cuanto a ejecución.
5. **Herramientas:** existe una gran cantidad de compiladores, depuradores, librerías, etc.

Introducción al Depurador de Dev C++

El objetivo de este tema es el conocimiento de la herramienta que proporciona Dev-C++ para la corrección de errores de ejecución: el depurador o debugger. Es primordial que al final de esta sesión el alumno sepa cómo depurar un programa en C++.
Podemos hablar de 3 tipos de Errores: **Sintaxis, de Ejecución y de Lógica**

Como trabajar con el Depurador Dev C++

- **Colocar un punto de parada**
- **Ejecutar un programa en modo de depuración**
- **Ejecución pasó a paso**
- **Inspeccionar variables (whatch)**

Concepto de Programa

Un programa en sentido informático está constituido por un conjunto de instrucciones que se ejecutan ordinariamente de modo secuencial, es decir, cada una a continuación de la anterior.

Componentes del lenguaje C++

- **El Compilador**

El compilador es un elemento más característico del lenguaje C++. Su misión consiste en traducir el lenguaje maquina el programa C++ contenido en uno o más ficheros fuente. El compilador es capaz de detectar ciertos errores durante el proceso de compilación, enviando al usuario el correspondiente mensaje de error.

- **El Preprocesador**

Es un componente característico de C++, que no existe en otros lenguajes de programación. El preprocesador actúa sobre el programa fuente, antes de que empiece la compilación propiamente dicha, para realizar ciertas operaciones. Una de estas operaciones es, por ejemplo, la sustitución de constantes simbólicas.

Así, es posible que un programa haga uso repetidas veces del valor 3.141592654, correspondiente al número PI. Es posible definir una constante simbólica llamada PI que se define como 3.141592654 al comienzo del programa y se introduce luego en el código cada vez que hace falta.

- **Librería Estándar**

Con Objeto de mantener el lenguaje lo más sencillo posible, muchas sentencias que existen en otros lenguajes no tienen su correspondiente contrapartida en C++. Por ejemplo, en C++ no hay sentencias para entrada y salida de datos. Es evidente, sin embargo, que esta es una funcionalidad que hay que cubrir de alguna mantener.

Unidad 2
Variables y Constantes

Estructura del lenguaje C++

Un programa de C ++ tiene una estructura muy específica en términos de cómo se escribe el código y algunos elementos clave que utiliza en sus programas de C ++.

Primer Programa en C++, El más simple de los programas de C ++ se muestra aquí

```
1. #include <iostream>
2. int main()
3. {
4.  std::cout << "Hello World!";
5.   return 0;
6. }
```

Para poder llamar un librería, debemos usar *#include,* el cual antecede a la librería ya definida en C++.

Algunas que podemos usar serian:

- **iostream:** Es quizá la más usada e importante, nos permitirá hacer uso del cin y el cout para obtener o imprimir valores por pantalla.
- **math:** Contiene los prototipos de las funciones y otras definiciones para el uso y manipulación de funciones matemáticas.
- **stdlib:** Contiene los prototipos de las funciones, macros, y tipos para utilidades de uso general.
- **string:** Una generalización de las cadenas alfanuméricas para albergar cadenas de objetos. Muy útil para el fácil uso de las cadenas de caracteres, pues elimina muchas de las dificultades que generan los char

Tipos de datos básicos de C++

Pseudocódigo	Lenguaje C++
Entero	Int
Real	Float
Real	Double
Carácter	Char
Lógico	bool

Salidas con cout

Ejemplo:

```
#include <iostream> // USAMOS: cin, cout
using namespace std;
    main() {
      double d1, d2;
      cout << "Introduce dos numeros reales: ";
      cin >> d1 >> d2;
      cout << "La suma es: " << d1 + d2 << endl ;
   }
```

El doble símbolo **<<** se conoce como operador de inserción
Algunas secuencias de escape:
\n nueva línea
\t tabulación horizontal.
**** diagonal invertida
\”

Otra forma de enviar un salto de línea es la utilización de endl.

Salidas y Comentarios

En C++ se puede enviar información a la salida estándar, que como se explicó previamente está conectada por defecto al monitor o pantalla, mediante una instrucción del tipo:

cout << expresion1 << expresion2 << . . . << expresiónN;

Esta instrucción se encarga de convertir el resultado de cada expresión en una cadena de caracteres y de enviar las cadenas de caracteres a la salida estándar. Si se utiliza endl como expresión, entonces se envía un salto de línea a la salida estándar.

Ejemplo:

```
#include <iostream>
using namespace std ;
/* Programa que muestra un numero y su cuadrado */
int main ()
{
   int x = 2;
   cout << "El numero es : " << x << endl ; // muestra x
   cout << "Su cuadrado vale : " << x*x << endl ; // y su cuadrado
   return 0;
}
```

El programa anterior ilustra el uso de esta instrucción, utilizándose expresiones de tipo cadena de caracteres y de tipo entero.

Formateo de salidas con funciones de Flujo

- **Precision**

Fijar el número de cifras después de la coma: precisión

- **Width**

Indica al flujo cuantos espacios debe usar al enviar un elemento a la salida.
El método width solamente afecta a la siguiente salida por pantalla.

- **Fill**

Indica el carácter con lo que completa los espacios en blanco cuando se manda un elemento a la salida y se fija el número de espacios con la función width.
Al igual que el método width, el método fill solamente afecta a la siguiente salida por pantalla

- **Setf**

Es una abreviatura de set flags, que significa establecer banderas. Una bandera es algo que indica que se debe efectuar algo de una de dos posibles maneras.

Si damos una determinada bandera como argumento de setf, dicha bandera le dirá a la computadora que escriba las salidas en ese flujo de alguna forma específica. El efecto sobre el flujo depende de la bandera. Banderas:

`ios::fixed`	Formato de punto fijo
`ios::showpoint`	Incluya un punto decimal en números en coma flotante.
`ios::showpos`	Signo + para números positivos
`ios::left`	Siguiente número en el extremo izquierdo.
`ios::right`	Siguiente número en el extremo derecho.

Ejemplo:

```
#include <iostream>
#include <stdlib.h>
//Escribir programa que lea tres números reales y los sume y muestro
el resultado
//con tres decimales.
using namespace std;
int main()
{
                    setlocale(LC_ALL, "Spanish");
float a, b, c, res;
cout << "Introduce 3 números reales para sumar:\n ";
cin >> a >> b >> c;
res = a + b + c;
//Formateo la salida con 2 decimales
cout.precision(3);
cout.setf(ios::fixed);
cout.setf(ios::showpoint);
cout.setf(ios::right);
cout.width(8); cout.fill(' '); cout << a << endl;
cout.width(8); cout.fill(' '); cout << b << endl;
cout.width(8); cout.fill(' '); cout << c << endl;
cout<<"________"<<endl;
cout.width(8); cout.fill(' '); cout << res << endl;
return 0;
}
```

Entrada con cin

Es el flujo de entrada estándar. Supondremos que la entrada estándar es el teclado. Veamos un ejemplo:

```
#include<iostream>
using namespace std;
int main()
{
                    setlocale(LC_ALL, "Spanish");
//delaracion de variables
    string nombre;
    int edad;
    float estatura;
    float peso;
    string correo;
//solicitar datos
cout<<"Ingresa tu nombre: ";
cin>>nombre;
cout<<"Ingresa tu edad: ";
cin>>edad;
cout<<"Cuanto mides: ";
cin>>estatura;
cout<<"Cuanto pesas: ";
cin>>peso;
cout<<"Cuál es tu correo: ";
cin>>correo;
//mostrar datos
cout<<"Hola "<<nombre<<" como te va"<<endl;
cout<<"Tu edad es "<<edad<<" años"<<endl;
cout<<"Tu estatura es "<<estatura<<" m"<<endl;
cout<<"Tu peso es "<<peso<<" kg"<<endl;
cout<<"Tu correo es "<<correo<<endl;
cout<<(8<10)<<endl;
    system("pause");
}
```

Entrada y Salida de caracteres

Todos los datos se envían como caracteres. Cuando un programa envía a la salida el número 10, en realidad lo que envía son los dos caracteres ‘1’ y ‘0’. Qué la computadora interprete el número 10 o los caracteres ‘1’ y ‘0’ depende del programa.

C++ ofrece algunos recursos para la entrada y salida de datos de caracteres

Funciones miembro o métodos

Get

Permite a un programa leer un carácter de entrada y guardarlo en una variable de tip char. Está función toma un argumento, que debe ser una variable de tipo char.

Comparación de cin >> y cin.get:

EJEMPLOS:

```
#include <iostream>
using namespace std;
main()
{

char c1,c2,c3,c4,c5;
//Introduce 4 caracteres en 2 lineas diferentes
cin.get(c1);
cin.get(c2);
cin.get(c3);
cin.get(c4);
 //Imprimo los caracteres leidos
 cout << "Los 4 caracteres leidos son:\n";
cout.put(c1);
cout.put(c2);
cout.put(c3);
cout.put(c4);
}
```

```
#include <iostream>
using namespace std;
main()
{

//Prueba con cin
char c1, c2, c3, c4;
//Introduce 4 caracteres separados por espacios
cin >> c1;
cin >> c2;
cin >> c3;
cin >> c4;
cout << "Los 4 caracteres leídos son:\n";
cout << c1 << c2 << c3 << c4;
```

Put

Es análoga a la función miembro get sólo que se emplea para salida. Mediante put un programa puede enviar a la salida un carácter. La función miembro put recibe un argumento que debe ser una expresión de tipo char (constante o variable).

Ignore

Este método permite descartar caracteres existentes en el buffer de entrada.

```
#include <iostream>

using namespace std;

int main(){
   char cadena1, cadena2;

   cout<<"introduzca dos palabra: ";
   cadena1=cin.get();
   cin.ignore(100,' ');//utilización de cin.ignore donde va ignorar
el carácter de espacio
   cadena2=cin.get();

   cout<<"las iniciales de las palabras son: "<<cadena1<<"
"<<cadena2;

   return 0;
}
```

Definición de Variables y operador de Asignación

En C++ el operador de asignación de una expresión a una variable es el carácter igual “=”. Para definir una variable hay que utilizar una expresión del tipo:

tipo nombre_var;

Donde tipo es un tipo valido y nombre_var es el identificador de la variable.
Existen ciertas reglas que definen los identificadores de variable válidos. Por ejemplo, un identificador de una variable no puede coincidir con el nombre de una palabra reservada del lenguaje, como return o case. Puede consultar en un manual estas reglas. En general, puede utilizar nombres de variables que incluyan caracteres del alfabeto, salvo la ˜n, y números y algún carácter especial como.

Ejemplo:

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

int main ()
{
   int x;
   double d = 2.5;
   cout << x << ' ' << d << endl ;
   x = 8;
   cout << x << ' ' << d << endl ;
   return 0;
}
```

Unidad 3

Expresiones, Sentencias y Operadores

Sentencia

Las sentencias son unidades completas, ejecutables en sí mismas. Ya se verá que muchos tipos de sentencias incorporan expresiones aritméticas, lógicas o generales como componentes de dichas sentencias.

Por ejemplo:

```
espacio = espacio_inicial + velocidad * tiempo;
```

Tipos de Operadores

- **Operadores de Asignación**

Permite asignar a una variable el valor de una expresión, por ejemplo:

```
x= 1;
a=26.610;
nombre = “Pedro”;
```

- **Operadores Aritméticos**

Nombre del Operador	Simbología
Suma	+
Resta	-
Multiplicación	*
División	/
Resto de la división entera	%

- **Operadores de incremento y decremento (++) y (--)**

Son los operadores que actúan sobre un solo operador para producir un nuevo valor. Por eso el nombre de unarios, porque para poder funcionar solo necesitan de un operador

Operador	Propósito
-	Menos Unario: Es el signo menos que va delante de una variable, constante o expresión.
++	Operador Incremento: Hace que la variable, constante o expresión se aumente en uno.
--	Operador Decremento: Hace que su variable, constante o expresión disminuye en uno.

- **Operadores Relacionales**

Nombre del Operador	Simbología
Menor que	<
Menor o igual que	<=
Mayor que	>
Mayor o igual que	>=
Igual que(Comparación)	==
Diferente	!=

- **Operadores Lógicos AND, OR y NOT**

Nombre del Operador	Simbología	
AND	&&	Si todas las condiciones están unidas por AND y todas se cumplen, se realizarán las acciones .
OR	||	Si las condiciones están unidas por OR y una de las condiciones se cumplen, se realizarán las acciones.
NOT	!=	No es igual al resultado que se ve reflejado del lado derecho.

- **Operadores de Direcciones**

Una dirección de memoria contiene un byte de información, una variable dependiendo de su tipo puede ocupar uno o más bytes.
Un apuntador solo almacena la dirección del primer byte de la variable sin importar que esta sea de más de 1 byte.
* indirección ()
& dirección o referencia ()

Ejemplo:

```
int a;
int * area;
area = &total;
```