Lenguaje de Programación C++

C++ no es chino!!!!

你好!

Es más sencillo aprender C++ que chino...

Palabras Reservadas

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

Tipos de datos

Nombre	Descripción	Tamaño*	Rango de valores*
char	Carácter o entero pequeño	1byte	con signo: -128 to 127 sin signo: 0 a 255
short int (short)	Entero corto	2bytes	con signo: -32768 a 32767 sin signo: 0 a 65535
int	Entero	4bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
long int (long)	Entero largo	8bytes	con signo: -2147483648 a 2147483647 sin signo: 0 a 4294967295
bool	Valor booleano. Puede tomar dos valores: verdadero o falso	1byte	true o false
float	Número de punto flotante	4bytes	3.4e +/- 38 (7 digitos)
double	De punto flotante de doble precisión	8bytes	1.7e +/- 308 (15 digitos)
long double	Long de punto flotante de doble precisión	12bytes	3,3621*(e^-4932) hasta 1,18973*(e^4932).

El orden no es obligatorio , pero si necesario

```
int main()
{
    int numero;

    numero = 5 + 5;
    return 0;
}
```

```
int main(){int numero;numero = 5 + 5;return 0;}
```

Comentarios

Hay que usarlos para que cuando otros los lean , no lean en chino

Comentarios de una línea

```
int i=0; //Declaración de variable
```

Comentarios multi-línea

```
float b;/*Esta sentencia (float b;), se ejecutaría
int b; pero esta no porque es comentario*/
```

Tipos de declaraciones

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

int main()
{
    char x = 'a'; // Declaramos y asignamos en la misma línea

    int num; //Declaramos el entero en una línea
    num = 5; //Le asignamos un valor en otra línea

    int num2 = 8; //Asignacion y declaracion al tiempo

    float numero; //Un numero decimal
    numero = 3.5; //Le asignamos un valor al decimal

    float res = numero + num2; //Sumamos dos variables y las asignamos a res
    //3.5 + 8 = 11.5

    res = res + num; //Al valor actual de res le sumamos el valor de num
    //11.5 + 5 = 16.5

    bool valor = false; //Variable booleana
    valor = true; // Pueden ser true o false

    res = res*2; //Duplicamos el valor de res 16.5*2 = 33

    cout << res << endl; //Mostramos el valor de res por pantalla

    return 0;
}
```

```cpp
int x = 3.3;    // Posible aviso: trunca la parte decimal
int num{3.3};   // Error
```

Tipos de declaraciones

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

int main()
{
    const float PI  = 3.1416; //Definimos una constante llamada PI
    cout << "Mostrando el valor de PI: " << PI << endl;
    PI = 2; //Esto generará un error pues PI es de solo lectura (constante)
    return 0;
}
```

Tipos de conversiones

```
int r = 48.1;  // r se almacenará como entero con valor 48
int s = -28.1; // s se almacenará como entero con valor -28
```

```
float x = 19999999;  // x se almacenara como real 20000000
double y = 19999999; // y se almacenara como real 19999999
```

```
bool b = 48.1; // b es cierto
bool b{1};     // OK: inicialización uniforme sólo valores 0 (false) y 1 (true)
bool b{48.1};  // ERROR: inicialización uniforme
```

bool < int

char < int < float < double

```
#include <iostream>
using namespace std;
int main()
{
   int longitud = 18342;
   int altitud = 1123; //Respecto a la base del puerto

   double pendiente = altitud/longitud;

   cout << "La pendiente es " << pendiente << endl;
}
```

Salida en pantalla

01

02

```
#include "iostream"

using namespace std;

int main()
{
    //Se muestra un mensaje por pantalla.
    cout << "Hola Mundo" << " Desde ProgramarYa." << "\n";

    return 0;
}
```

Salida en pantalla

02

```cpp
#include "iostream"
#include "string"

using namespace std;

int main()
{
    //El valor de esta variable se mostrará en pantalla
    string salida1 = "Ejemplo de salida";

    //Este valor también se mostrará en pantalla.
    int numero = 2;

    //Estos valores se concatenarán en una única salida
    string salida2 = "Desde ProgramarYa.";

    //Se concatenan y muestran los valores por pantalla con cout<<
    cout << salida1 << " " << numero << ". " << salida2 << "\n";

   return 0;
}
```

Entrada en pantalla

```cpp
#include "iostream"
#include "string"

using namespace std;

int main()
{
    cout << "Hola! Este es un ejemplo en C++" << "\n" << "Por favor ingrese su nombre:" << "\n";
    //La instrucción \n es un salto de línea Mostrando los textos separados

   string nombre;//En esta variable estará almacenado el nombre ingresado.
   cin >> nombre; //Se lee el nombre

   cout << "Bienvenido al sistema " << nombre << ". Gracias por usar nuestra aplicación" << "\n";


   return 0;
}
```

Entrada en pantalla

```cpp
#include "iostream"
#include "string"

using namespace std;

int main()
{
    //Mensaje de bienvenida
    cout << "Hola! Aqui podras realizar sumas" << "\n";

    //Se declaran los números que se sumarán (pueden ser decimales)
    float numero1, numero2;

     //Se pide el primer numero
    cout << "Por favor ingrese el primer valor: " << "\n";

    //Se asigna el primer valor a numero1
    cin >> numero1;

    //Se pide el segundo numero
    cout << "Por favor ingrese el segundo valor: " << "\n";

    //Se asigna el segundo valor a numero2
    cin >> numero2;

    //Se muestra el resultado.
    cout << numero1 << " + " << numero2 << " = " << numero1 + numero2;

   return 0;
}
```

Tipos de operadores

Operadores Matemáticos

Operador	Descripción
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo

Precedencias

El valor determinado por una expresión se calcula siguiendo un orden de evaluación en función de unas reglas de precedencia

Precedencia Operaciones

Precedencia	Operador	Descripción	Asociatividad
1	`::`	Resolución de ámbito	Izquierda a derecha
2	`(expresion)`	Expresión entre paréntesis	Izquierda a derecha
5	`* / %`	Multiplicación, división y división entera	Izquierda a derecha
6	`+ -`	Suma y resta	Izquierda a derecha
7	`<< >>`	Extracción e inserción	Izquierda a derecha
16	`=`	Asignación	Derecha a izquierda

https://en.cppreference.com/w/cpp/language/operator_precedence

Precedencia Operaciones

```
// Operaciones con constantes literales enteras
(3 + 2)*5          // 1º) (3 + 2) 2º) 5*5: Resultado 25
(3 + 2)*12/5       // 1º) (3 + 2), 2º) 5*12, 3º) 60/5: Resultado 12
(3 + 2)*(12/5)     // 1º) (3 + 2), 2º) (12/5) 3º) 5*2: Resultado 10
```

```
a*(((b + 2) + 3) + z)
```

https://en.cppreference.com/w/cpp/language/operator_precedence

Operadores De asignación

```
y = y + 1.5;
```

```
y = (2 + 3.5)/3.1 + a;  // OK
vel = x;                // OK
masa = 5.6;             // OK
x + y = u + w;          // ERROR
2 = z;                  // ERROR
```

```
x = a = 2;  // Primero se asigna a = 2, luego x = a
```

```
a += b;    // equivale :  a = a + b
a -= b;    // equivale :  a = a - b
a *= b;    // equivale :  a = a * b
a /= b;    // equivale :  a = a / b
a %= b;    // equivale :  a = a % b
```

```
a + = b;    // ¡ERROR!
```

Importante

El operador de asignación no es equivalente a la igualdad matemática: una sentencia como la anterior, y = y + 1.5;, no tiene sentido en matemáticas.

Operadores Condicionales

```
#include "iostream"
#include "string"
#include "stdlib.h"

using namespace std;

int main()
{
    string password = "";
    cout << "Ingrese la contrasenia: ";
    cin >> password;
    if(password == "myClave")
    {
        cout << "Contrasenia correcta. Bienvenido";
    }
    else
    {
        cout << "Contrasenia incorrecta.";
    }

    system("PAUSE");

    return 0;
}
```

Operadores Condicionales

```
# include "iostream"

using namespace std;

int main()
{
    cout << "Ingrese la Opción a ejecutar: ";
    int opcion = 0;
    cin >> opcion;

    switch(opcion)
    {
        case 1: cout << "Usted ha seleccionado la opción 1";
        break;
        case 2: cout << "Usted ha seleccionado la opción 2";
        break;
        case 3: cout << "Usted ha seleccionado la opción 3";
        break;
        default: cout << "Usted ha ingresado una opción incorrecta";
    }
    // system("PAUSE"); //Solo ponla si no te da error

    return 0;
}
```

Operadores Ciclos

```
#include "iostream"
#include "stdlib.h"

using namespace std;
int main()
{

    for(int i=10; i > 0; i--)
    {//Notemos que escribir i-- es similar a escribir i = i - 1
        cout << i << endl;
    }
    system("PAUSE");
    return 0;
}
```

Operadores Ciclos

```
#include "iostream"

using namespace std;
int main()
{
    int numero;
    cout <<  "Ingrese un numero ";
    cin >> numero;
    while(numero <= 100)
    {
        cout <<  "Ingrese un numero ";
        cin >> numero;
    }
    system("PAUSE");
    return 0;
}
```

Operadores Ciclos

```
#include "iostream"

using namespace std;
int main()
{
    int opcion = -1;

    //Mostramos el menú hasta que la opción sea cero
    while(opcion != 0)
    {
        cout << "Ingrese 1 para mostrar mensaje\n";
        cout << "Ingrese 2 para hacer algo más\n";
        cout << "Ingrese 0 para salir\n";

        cout << "Ingrese una opción a ejecutar\n";
        cin >> opcion;
    }

    system("PAUSE");
    return 0;
}
```

Operadores Ciclos

```
#include "iostream"

using namespace std;
int main()
{
    int opcion = -1;

    //Mostramos el menú hasta que la opción sea cero
    while(opcion != 0)
    {
        cout << "Ingrese 1 para mostrar mensaje\n";
        cout << "Ingrese 2 para hacer algo más\n";
        cout << "Ingrese 0 para salir\n";

        cout << "Ingrese una opción a ejecutar\n";
        cin >> opcion;
    }

    system("PAUSE");
    return 0;
}
```

Operadores Ciclos

```
#include "iostream"
using namespace std;

int main()
{
    int matriz[10][10];

    for(int i = 0; i < 10; i++)//Ciclo externo
    {
        //Notemos que en ciclo interno no usamos la variable 'i' sino 'j'
        //Si usaramos i, el comportamiento sería inesperado
        for(int j = 0; j < 10; j++)//Ciclo interno
        {
            //Llenamos la matríz con los números del 1 al 100
            matriz[i][j] = (i+1)*(j+1);
            //(i+1)*(j+1) va desde 1x1 hasta 10x10
        }
    }

    for(int i = 0; i < 10; i++)//Ciclo externo
    {
        for(int j = 0; j < 10; j++)//Ciclo interno
        {
            //Mostramos por pantalla el valor de la fila i columna j
            cout << matriz[i][j] << "\n";
        }
    }

    system("pause");
    return 0;
}
```

Estructuras de datos

Vectores

```
int vector2[3];
vector2[0] = 1;
vector2[1] = 3;
vector2[2] = 10;
```

```
float vector4[5] = {10.5, 5.1, 8.9, 10, 95.2}; //Array con 5 elementos
float numero5 = vector4[4]; //Para acceder al elemento 5, se usa el índice 4
float primerNumero = vector4[0]; //Para el primer elemento se usa el índice 0
```

```
#include "iostream"

using namespace std;

int main()
{
    int edades[] = {1,2,9,8,16,32,9,50,36,20,1,87};
    int limite = (sizeof(edades)/sizeof(edades[0]));
    for (int i = 0; i < limite; i++)
    {
            cout<<edades[i]<<endl;
    }
}
```

Estructuras de datos

```
#include "iostream"
#include "string"
using namespace std;

int main()
{
    string titulos[5];
    string autores[5];
    cout << "Por favor ingrese la siguiente información de los Libros: \n";
    for(int i = 0; i < 5; i++)
    {
        cout << "\n******* Libro " << i + 1 << "********:\n";
        cout << "Titulo: ";
        //cin >> titulos[i]; //No funciona con espacios
        getline(cin, titulos[i]);
        cout << "Autor: ";
        //cin >> autores[i]; //No funciona con espacios
        getline(cin, autores[i]);
    }
}
```

Estructuras de datos /Matrices

```
#include "iostream"
#include "stdio.h"
#include "string"

using namespace std;

int main()
{
    string libros[5][2];
    cout << "Por favor ingrese la siguiente información de los Libros: \n";
    string titulo ,autor;
    for(int i = 0; i < 5; i++)
    {
        cout << "\n******* Libro " << i + 1 << "********:\n";
        cout << "Titulo: ";
        getline(cin,titulo);
        cout << "Autor: ";
        getline(cin,autor);
        libros[i][0] = titulo;
        libros[i][1] = autor;
    }

    system("pause");

    return 0;
}
```

Funciones

```
int funcionEntera()//Función sin parámetros
{
    int suma = 5+5;
    return suma; //Acá termina la ejecución de la función
    return 5+5;//Este return nunca se ejecutará
    //Intenta intercambiar la línea 3 con la 5
    int x = 10; //Esta línea nunca se ejecutará
}
```

Funciones

```
bool funcionBool(int n, string mensaje)//Función con dos parámetros
{
    if(n == 0)//Usamos el parámetro en la función
    {
        cout << mensaje;//Mostramos el mensaje
        return 1; //Si n es cero retorna 1
        return true;//Equivalente
    }
    return 0;//Este return sólo se ejecuta cuando n NO es cero
    return false;//Equivalente
}
```

Procedimientos

```
void procedimiento(int n, string nombre)
{
    if(n == 0)
    {
        cout << "hola" << nombre;
        return;
    }
    cout << "adios" << nombre;
}
```

Invocar funciones

```
int main()
{
    funcionEntera(); //Llamando a una función sin argumentos

    bool respuesta = funcionBool(1, "hola"); //Asignando el valor retornado a una

    procedimiento(0, "Juan");//Invocando el procedimiento

    //Usando una función como parámetro
    procedimiento(funcionBool(1, "hola"), "Juan");

    return 0;
}
```

Librerias

```
#include <string.h>
#include <iostream>

using namespace std;

int main ()
{
    cout << ("Hola! Por favor ingrese su nombre ....\n");
    string cadena = "Hola "; //Se le da un valor inicial al string

    string nombre; //Esta cadena contendrá el nombre
    cin >> nombre; //Se lee el nombre

    cadena = cadena + nombre; //Se juntan el saludo con el nombre usando "+"
    cout << (cadena); //Se muestra el resultado final.
    return 0;
}
```

中华人民共和国国庆节

Librerias

Librerías Estandar de C++ (Standar Template Library o *STL*)

A continuación pondré algunas de las librerías de uso más común de C++ y que forman parte de las librerías estandar de este lenguaje.

- **fstream:** Flujos hacia/desde ficheros. Permite la manipulación de archivos desde el programar, tanto leer como escribir en ellos.
- **iosfwd:** Contiene declaraciones adelantadas de todas las plantillas de flujos y sus typedefs estándar. Por ejemplo ostream.
- **iostream:** Parte del a *STL* que contiene los algoritmos estándar, es quizá la más usada e importante (aunque no indispensable).
- **La biblioteca list:** Parte de la *STL* relativa a contenedores tipo list; listas doblemente enlazadas
- **math:** Contiene los prototipos de las funciones y otras definiciones para el uso y manipulación de funciones matemáticas.

中华人民共和国国庆节

Librerias

- memory: Utilidades relativas a la gestión de memoria, incluyendo asignadores y punteros inteligentes (auto_ptr).
- "auto_ptr" es una clase que conforma la librería memory y permite un fácil manejo de punteros y su destrucción automaticamente.
- Biblioteca new: Manejo de memoria dinámica
- numeric: Parte de la librería numérica de la STL relativa a operaciones numéricas.
- ostream: Algoritmos estándar para los flujos de salida.
- queue: Parte de la STL relativa a contenedores tipo queue (colas de objetos).
- Librería stdio: Contiene los prototipos de las funciones, macros, y tipos para manipular datos de entrada y salida.

Librerias

中华人民共和国国庆节

- Librería stdlib: Contiene los prototipos de las funciones, macros, y tipos para utilidades de uso general.
- string: Parte de la STL relativa a contenedores tipo 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 d elas dificultades que generan los char
- typeinfo: Mecanismo de identificación de tipos en tiempo de ejecución
- vector: Parte de la STL relativa a los contenedores tipo vector; una generalización de las matrices unidimensionales C/C++
- forward_list: Esta librería es útil para implementar con gran facilidad listas enlazadas simples.
- list: Permite implementar listas doblemente enlzadas (listas enlazadas dobles) facilmente.

中华人民共和国国庆节

Librerias

- iterator: Proporciona un conjunto de clases para iterar elementos.
- regex: Proporciona fácil acceso al uso de expresiones regulares para la comparación de patrones.
- thread: Útil para trabajar programación multihilos y crear múltiples hilos en nuestra aplicación.
- time: Útil para obtener marcas de tiempo durante ejecución. Se usa con frecuencia para conocer el tiempo exacto durante un programa.
- Nota: Para más información sobre las librerías estandar de C++ y STL puedes ingresar aquí

再见!

Nos vemos en la próxima clase