Introducción a los algoritmos en c++

OscarGauss :P

PUBLISHED BY PUBLISHER

BOOK-WEBSITE.COM

First printing, March 2013

Contenidos

1. Introducción a la programación en c++

1.1 Introducción

1.1.1 ¿Qué es un Lenguaje de Programación?

Antes de hablar de C++, es necesario explicar que un lenguaje de programación es una herramienta que nos permite comunicarnos e instruir a la computadora para que realice una tarea específica. Cada lenguaje de programación posee una sintaxis y un léxico particular, es decir, forma de escribirse que es diferente en cada uno por la forma que fue creado y por la forma que trabaja su compilador para revisar, acomodar y reservar el mismo programa en memoria.
Existen muchos lenguajes de programación de entre los que se destacan los siguientes:

- C
- C++
- Basic
- Ada
- Java
- Pascal
- Python
- Fortran
- Smalltalk

1.1.2 Historia de C++

C++ es un lenguaje de programación creado por Bjarne Stroustrup en los laboratorios de At&T en 1983. Stroustrup tomó como base el lenguaje de programación más popular en aquella época el cual era C.
El C++ es un derivado del mítico lenguaje C, el cual fue creado en la década de los 70 por la mano del finado Dennis Ritchie para la programación del sistema operativo (un sistema parecido a Unix es GNU/Linux), el cual surgió como un lenguaje orientado a la programación de sistemas (System Programming) y de herramientas (Utilities) recomendado sobre todo para programadores expertos, y que no llevaba implementadas muchas funciones que hacen a un lenguaje más comprensible.
Sin embargo, aunque esto en un inicio se puede convertir en un problema, en la práctica es su mayor virtud, ya que permite al programador un mayor control sobre lo que está haciendo. Años más tarde, un programador llamado Bjarne Stroustrup, creo lo que se conoce como C++.
Necesitaba ciertas facilidades de programación, incluidas en otros lenguajes pero que C no soportaba, al menos directamente, como son las llamadas clases y objetos, principios usados en la programación actual. Para ello rediseñó C, ampliando sus posibilidades pero manteniendo su mayor cualidad, la de

permitir al programador en todo momento tener controlado lo que está haciendo, consiguiendo así una mayor rapidez que no se conseguiría en otros lenguajes.
C++ pretende llevar a C a un nuevo paradigma de clases y objetos con los que se realiza una comprensión más humana basándose en la construcción de objetos, con características propias solo de ellos, agrupados en clases. Es decir, si yo quisiera hacer un programa sobre animales, crearía una clase llamada animales, en la cual cada animal, por ejemplo un pato, sería un objeto, de tal manera que se ve el intento de esta forma de programar por ser un fiel reflejo de cómo los humanos (en teoría) manejamos la realidad.
Se dice que nuestro cerebro trabaja de forma relacional (relacionando hechos), es por ello que cada vez que recuerdas algo, (cuentas un hecho), termina siendo diferente (se agregan u omiten partes).

1.1.3 ¿Qué es C++?

C++ es un lenguaje de programación orientado a objetos que toma la base del lenguaje C y le agrega la capacidad de abstraer tipos como en Smalltalk.
La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitieran la manipulación de objetos. En ese sentido, desde el punto de vista de los lenguajes orientados a objetos, el C++ es un lenguaje híbrido.
Posteriormente se añadieron facilidades de programación genérica, que se sumó a los otros dos paradigmas que ya estaban admitidos (programación estructurada y la programación orientada a objetos). Por esto se suele decir que el C++ es un lenguaje de programación multiparadigma.

1.1.4 Herramientas Necesarias

Las principales herramientas necesarias para escribir un programa en C++ son las siguientes:

1. Un equipo ejecutando un sistema operativo.
2. Un compilador de C++
   - Windows MingW (GCC para Windows) o MSVC (compilador de microsoft con versión gratuita)
   - Linux (u otros UNIX): g++
   - Mac (con el compilador Xcode)
3. Un editor cualquiera de texto, o mejor un entorno de desarrollo (IDE)
   - Windows:
     - Microsoft Visual C++ (conocido por sus siglas MSVC). Incluye compilador y posee una versión gratuita (versión express)
     - Bloc de notas (no recomendado)
     - Editor Notepad++
     - DevCpp (incluye MingW - en desuso, no recomendado, incluye también un compilador)
     - Code::Blocks
   - Linux (o re-compilación en UNIX):
     - Gedit
     - Kate
     - KDevelop
     - Code::Blocks
     - SciTE
     - GVim

- Mac:
  - Xcode (con el compilador trae una IDE para poder programar)

4. Tiempo para practicar
5. Paciencia

Adicional

- Inglés (Recomendado)
- Estar familiarizado con C u otro lenguaje derivado (PHP, Python, etc).

Es recomendable tener conocimientos de C, debido a que C++ es una mejora de C, tener los conocimientos sobre este te permitira avanzar mas rapido y comprender aun mas. Tambien, hay que recordar que C++, admite C, por lo que se puede programar (reutilizar), funciones de C que se puedan usar en C++.

Aunque No es obligacion aprender C, es recomendable tener nociones sobre la programación orientada a objetos en el caso de no tener conocimientos previos de programación estructurada. Asimismo, muchos programadores recomiendan no saber C para saber C++, por ser el primero de ellos un lenguaje imperativo o procedimental y el segundo un lenguaje de programación orientado a objetos.

1.1.5 Consejos iniciales antes de programar

Con la práctica, se puede observar que se puede confundir a otros programadores con el código que se haga. Antes de siquiera hacer una línea de código, si se trabaja con otros programadores, ha de tenerse en cuenta que todos deben escribir de una forma similar el código, para que de forma global puedan corregir el código en el caso de que hubieran errores o rastrearlos en el caso de haberlos.

También es muy recomendable hacer uso de comentarios (comenta todo lo que puedas, hay veces que lo que parece obvio para ti, no lo es para los demás) y tratar de hacer un código limpio y comprensible, especificando detalles y haciendo tabulaciones, aunque te tome un poco mas de tiempo, es posible que mas adelante lo agradezcas tu mismo.

1.1.6 Ejemplos

Codigo 1.1: Ejemplo de C++

```cpp
#include <iostream>

using namespace std;

int main()
{
    int numero;
    cin>>numero;
    if(numero%2==0){
        cout<<"El numero es par\n";
    }else{
        cout<<"EL numero es impar\n";
    }
    return 0;
}
```

1.2 Lo mas basico

1.2.1 Proceso de desarrollo de un programa

Si se desea escribir un programa en C++ se debe ejecutar como mínimo los siguientes pasos:

1. Escribir con un editor de texto plano un programa sintácticamente válido o usar un entorno de desarrollo (IDE) apropiado para tal fin
2. Compilar el programa y asegurarse de que no han habido errores de compilación
3. Ejecutar el programa y comprobar que no hay errores de ejecución

Este último paso es el más costoso, por que en programas grandes, averiguar si hay o no un fallo prácticamente puede ser una tarea totémica.
Un archivo de C++ tiene la extención *cpp* a continuación se escribe el siguiente codigo en c++ del archivo 'hola.cpp'

Codigo 1.2: Hola Mundo

```
1  // Aquí generalmente se suele indicar qué se quiere con el programa a hacer
2  // Programa que muestra 'Hola mundo' por pantalla y finaliza
3
4  // Aquí se sitúan todas las librerias que se vayan a usar con include,
5  // que se verá posteriormente
6  #include <iostream> // Esta libreria permite mostrar y leer datos por consola
7
8  int main()
9  {
10      // Este tipo de líneas de código que comienzan por '//' son comentarios
11      // El compilador los omite, y sirven para ayudar a otros programadores o
12      // a uno mismo en caso de volver a revisar el código
13      // Es una práctica sana poner comentarios donde se necesiten,
14
15      std::cout << "Hola Mundo" << std::endl;
16
17      // Mostrar por std::cout el mensaje Hola Mundo y comienza una nueva línea
18
19      return 0;
20
21      // se devuelve un 0.
22      //que en este caso quiere decir que la salida se ha efectuado con éxito.
23 }
```

Mediante simple inspección, el código parece enorme, pero el compilador lo único que leerá para la creación del programa es lo siguiente:

Codigo 1.3: Hola Mundo Compilado

```
1 #include <iostream>
2 int main(void){ std::cout << "Hola Mundo" << std::endl; return 0; }
```

Como se puede observar, este código y el original no difieren en mucho salvo en los saltos de línea y que los comentarios, de los que se detallan posteriormente, están omitidos y tan sólo ha quedado "el esqueleto" del código legible para el compilador. Para el compilador, todo lo demás, sobra.

Aquí otro ejemplo

Codigo 1.4: Hello World

```
#include <iostream>

int main()
{
    std::cout << "Hola Mundo" << std::endl;
    std::cout << "Hello World" << std::endl;
    std::cout << "Hallo Welt" << std::endl;
    return 0;
}
```

Para hacer el código mas corto debemos incluir *using namespace std* con lo que le estamos diciendo al compilador que usaremos el espacio de nombres std por lo que no tendremos que incluirlo cuando usemos elementos de este espacio de nombres, como pueden ser los objetos cout y cin, que representan el flujo de salida estándar (típicamente la pantalla o una ventana de texto) y el flujo de entrada estándar (típicamente el teclado).
Se veria de la siguiente manera:

Codigo 1.5: Using Namespace Std

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hola Mundo" << endl;
    cout << "Hello World" << endl;
    cout << "Hallo Welt" << endl;
    return 0;
}
```

Los pasos siguientes son para una compilación en GNU o sistema operativo Unix, para generar el ejecutable del programa se compila con g++ de la siguiente forma:

Codigo 1.6: Compilar

```
| g++ hola.cpp -o hola
```

Para poder ver los resultados del programa en acción, se ejecuta el programa de la siguiente forma:

Codigo 1.7: Ejecutar

```
|./hola
```

Y a continuación se debe mostrar algo como lo siguiente:

Codigo 1.8: Resultado

```
Hola Mundo
```

1.2.2 Sintaxis

Sintaxis es la forma correcta en que se deben escribir las instrucciones para el computador en un lenguaje de programación específico. C++ hereda la sintaxis de C estándar, es decir, la mayoría de programas escritos para el C estándar pueden ser compilados en C++.
El punto y coma El punto y coma es uno de los simbólos más usados en C, C++; y se usa con el fin de indicar el final de una línea de instrucción. El punto y coma es de uso obligatorio.
Ejemplo

Codigo 1.9: Sintaxis

```
1  clrscr(); //Limpiar pantalla, funciona con la librería conio de Borland C++
2  x = a + b;
3
4  string IP = "127.0.0.1"; // Variable IP tipo string
5  cout << IP << endl; // Devuelve 127.0.0.1
6
7  char Saludo[5] = "Hola"; // Variable Saludo tipo char
8  cout << Saludo[0] << endl; // Igual a H
9  cout << Saludo[1] << endl; // Igual a o
10 cout << Saludo[2] << endl; // Igual a l
11 cout << Saludo[3] << endl; // Igual a a
```

El punto y coma se usa también para separar contadores, condicionales e incrementadores dentro de un sentencia for
Ejemplo

Codigo 1.10: Sintaxis

```
1  for (i=0; i < 10; i++) cout << i;
```

Espacios y tabuladores Usar caracteres extras de espaciado o tabuladores (caracteres tab) es un mecanismo que nos permite ordenar de manera más clara el código del programa que estemos escribiendo, sin embargo, el uso de estos es opcional ya que el compilador ignora la presencia de los mismos. Por ejemplo, el segundo de los ejemplos anteriores se podría escribir como:

Codigo 1.11: Sintaxis

```
1  for (int i=0; i < 10; i++) { cout << i * x; x++; }
```

y el compilador no pondría ningún reparo.

1.2.3 Comentarios

Existen dos modos basicos para comentar en c++:
//
Comentan solo una linea de código

/*Comentario*/
Comentan estrofas de código
A continuación un ejemplo:

Codigo 1.12: Comentarios

```
#include <iostream>

using namespace std;

int main()
{
    /*
    cout    es para imprimir
    <<      se utiliza para separar elementos para cout
    endl    es un salto de linea
    */

    //En español
    cout << "Hola Mundo" << endl;
    //En ingles
    cout << "Hello World" << endl;
    //En aleman
    cout << "Hallo Welt" << endl;
    return 0;
}
```

1.2.4 Impresión

Para la impresión se utilliza *cout* de la siguiente manera:

Codigo 1.13: Impresión

```
// Programa que muestra diversos textos por consola

// Las librerías del sistema usadas son las siguientes
#include <iostream>

using namespace std;
// Es la función principal encargada de mostrar por consola diferentes textos
int main(void)
{
    // Ejemplo con una única línea, se muestra el uso de cout y endl
    cout << "Bienvenido. Soy un programa. Estoy en una linea de
        codigo." << endl;

    // Ejemplo con una única línea de código que se puede fraccionar
    // mediante el uso de '«'
    cout << "Ahora "
         << "estoy fraccionado en el codigo, pero en la consola me
            muestro como una unica frase."
```

```
17            << endl;
18
19      // Uso de un código largo, que cuesta leer para un programador,
20      // y que se ejecutará sin problemas.
21      // *** No se recomienda hacer líneas de esta manera,
22      // esta forma de programar no es apropiada ***
23      cout << "Un gran texto puede ocupar muchas lineas."
24           << endl
25           << "Pero eso no frena al programador a que todo se pueda
                poner en una unica linea de codigo y que"
26           << endl
27           << "el programa, al ejecutarse, lo situe como el
                programador quiso"
28           << endl;
29
30      return 0; // Y se termina con éxito.
31  }
```

Consola:

Codigo 1.14: Resultado Impresión

```
Bienvenido. Soy un programa. Estoy en una linea de codigo.
Ahora estoy fraccionado en el codigo, pero en la consola me muestro como una unica
    frase.
Un gran texto puede ocupar muchas lineas.
Pero eso no frena al programador a que todo se pueda poner en una unica linea de
    codigo y que
el programa, al ejecutarse, lo situe como el programador quiso
```

1.2.5 Datos primitivos

En un lenguaje de programación es indispensable poder almacenar información, para esto en C++ están disponibles los siguientes tipos que permiten almacenar información numérica de tipo entero o real:

Nombre	Descripción	Tamaño	
bool	Valor booleano, puede tomar dos valores(verdadero o falso)	1byte	
char	Carácter o entero pequeño	1byte	
short int	Entero corto	2bytes	Con
int	Entero	4bytes	Con signo: -2
long long	Entero largo	8bytes	Con signo: -9223372036854775
float	Número de punto flotante	4bytes	
double	De punto flotante de doble precisión	8bytes	

1.2.6 Variables

1.2.7 Lectura

2. Estructura de datos 1

3. Programación modular

Bibliography

Books

[Smi12] John Smith. *Book title*. 1st edition. Volume 3. 2. City: Publisher, Jan. 2012, pages 123–200.

Articles

[Smi13] James Smith. “Article title”. In: 14.6 (Mar. 2013), pages 1–8.

Index

Lista de Codigos