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: Some Code

```
#include <stdio.h>
#include <stdlib.h>
int main () {
   int numero;
   printf ("Ingrese el valor de numero: ");
   scanf ("%d", &numero);
   if(numero%2==0)
       printf ("\n***El numero es par\n");
   else
       printf ("\n***El numero es impar\n");
   return 0;
}
///
#include <stdio.h>
#define N 10
/* Block
```

```
 * comment */

int main()
{
    int i;

    // Line comment.
    puts("Hello world!");

    for (i = 0; i < N; i++)
    {
        puts("LaTeX is also great for programmers!");
    }

    return 0;
}
```

1.2 Lo mas basico

Codigo 1.2: Some Code

```
public void here() {
    goes().the().code()
}
```

Escribe el siguiente código en un fichero llamado `hello.c`:

Codigo 1.3: Some Code

```
#include <stdio.h>
int main(int argc, char* argv[]) {
  puts("Hola mundo!");
}
```

Ahora compila usando `gcc`:

Codigo 1.4: Some Code

```
$ gcc  -o hello hello.c
```

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