Compilando Conocimiento

Introducción a la Programación Competitiva

Club de Algoritmia ESCOM

Rosas Hernandez Oscar Andrés

Enero 2018

Índice general

Parte I

Introducción a la Programación Competitiva

Capítulo 1

¿Qué es la Programación Competitiva?

1.1. Introducción

1.1.1. ¿Qué hace diferente a un programa de competitiva de uno normal?

Parte II

Un tour de C++ y lo que deberías saber de el

Capítulo 2

¿Porqué C++ ?

2.1. Ventajas de C++

Hay muchas razones por las cuales C++ es uno de los lenguajes más usados en la modernidad en la programación competitiva (si no es que el más).

Puedes escuchar un video muy bonito para mi sobre porque elegir este lenguaje:

`Bjarne Stroustrup: Why I Created C++`

Figura 2.1: Este es el creador del lenguaje y persona a la cual nunca sabras como pronunciar su nombre: Bjarne Stroustrup

Las principales razones por C++ sin un orden en particular son:

- **Te da abstracciones sin costo extra.**

  O como lo diría su creador te da el poder de usar abstracciones de alto nivel pero estando muy cerca del hardware.

  Personalmente creo que esa es mayor ventaja que tiene sobre todos los demás, creo que esa frase resume perfectamente todo el objetivo de C++ .

  Lo que nos dice esto es que podremos representar grafos, matrices, operaciones, clases en general, algoritmos, etc... en nuestros programas con gran facilidad, cosa que no podemos hacer fácilmente por ejemplo en lenguajes como C o la familia de los ensambladores.

  Y podrías decir, pero en Java o en Python podemos representar de una manera igual de fácil (siendo muy exactos no creo que sea el caso, pero esa es otra historia amigos) así que ... ¿porqué usar C++ ?

  Pues porque en todos los demás lenguajes estas pagando un costo (a veces muy grande de varios cientos o miles de veces) por poder representar ideas o conceptos abstractos en un programa, en C++ esta prácticamente garantizado que usar una clase por ejemplo o que usar un arreglo que se auto dimensiona solo no será más costoso que si lo hubieras hecho tu desde ensamblador.

- **Tienes a la STD a tu lado**

  Esta es otra gran ventaja, ya que siguiendo con la mentalidad de abstracciones sin costo extra tenemos gracias a la gran libreria estándar un montón de cosas que no tenemos que hacer desde cero, desde arreglos que cambian de tamaño, arreglos asociativos, pilas, colas, algoritmos de ordenamiento, de búsqueda, etc...

  Así podemos dejar los algoritmos básicos al lenguaje y enfocarnos en las cosas de verdad interesantes.

- **Es compilador, el compilador es tu amigo**

  Otra ventaja más, podemos siempre confiar en el compilador, en que si nuestro programa compila muy probablemente está haciendo lo que debe hacer (cosa que no podemos esperar con python por ejemplo), el compilador es tu amigo, te dira en donde te equivocaste, en donde puede que hallas querido decir otra cosa y muchas veces te dará consejos, además, tras bambalinas está transformando tu código en algo que la computadora puede de verdad entender y además usará toda la información que le diste para muchas veces incluso mejorar tu código en vez de solo "traducirlo" y optimizarlo de maneras que me sorprenden personalmente.

- **Es prácticamente un "super set" de C**

  Es decir, que cualquier código válido de C es válido en C++ , esto es de gran ayuda pues C es uno de los lenguajes más conocidos por lo que puede que la sintaxis sea más sencilla de aprender que otros como Haskell. Otra ventaja es que al estar basados en C conserva muchas de las ventajas de C como su portabilidad, su velocidad de ejecución y la capacidad de tener un gran control de todos los recursos del sistema (memoria y tiempo de vida de un objecto, cough cough Java y su recolector de basura)

- **Tienes un gran control de todos los recursos del sistema**

  Esto es también es muy importante, pues nos dice que en C++ podemos controlar con gran lujo de detalle cuando un pedazo de memoria ya no es usado y deberíamos liberarlo ayudando con esto a aumentar la velocidad de nuestro programa o cuando queremos usar una variable local y cuando queremos usar memoria del heap, tenemos el control de decidir si queremos pasar las cosas por referencia o por valor, si deseamos mover un objecto o si una referencia no podrá ser modificada.

2.2. vs C

El gran problema con C es que es un lenguaje muy pequeño en el sentido en que todo lo tienes que hacer tu, si quieres hacer un problema que involucra cosas medio complejas todas las estructuras las tienes que codear al momento, y en un deporte de tiempo, cada segundo cuenta, así que en resumén, lo que "mata" a C es la falta de algo parecido a la std de C++ .

Aunque para problemas sencillos C también puede ser una opción, medio rara porque podrías entonces también hacerlo en C++ pero esta bien :)

2.3. vs Java

Con toda honestidad hay un porcentaje de la comunidad de programación competitiva que usan Java, así que si que es una opción biable, sobretodo por su gran libreria estandar y también porque en C++ no hay algo parecido a `BigInteger` y `BigDecimal` y suelen ser muchos los problemas que lo requieran, así que si bien C++ podría ser tu lenguaje por defecto es importante que también conozcas lo básico de Java (O Kotlin si eres de los niños cool)

2.4. vs Python

Python es un gran lenguaje pero tiene todas las de perder en programción competitiva pues a ser interpretado y sin tipos sus programas acaban siendo muy lentos incluso usando el algoritmo correcto, eso si, hay varias aplicaciones útiles de python, como que todos los enteros tienen infinita presición por defecto (aka `BigInteger` como en Java).

Así que tampoco es una mala idea aprenderlo por si se necesita un día, pero definitivamente no es la mejor idea para ser tu lenguaje por defecto en programción competitiva.

Capítulo 3

Tipos de datos númericos

3.1. Integers: Enteros

3.1.1. int

3.1.2. long

3.1.3. size_t

3.1.4. unsigned

3.2. Floating Points: Puntos Flotantes

3.3. Complex: Números Complejos

Capítulo 4

Bases del Lenguaje

4.1. Condicionales

4.2. Ciclos

4.3. Funciones

4.3.1. Recursion: Recursividad

4.4. References and value types: Referencias o Valores

4.4.1. Value types: Variables de valor

4.4.2. Pointers: Apuntadores

4.4.3. References: Referencias

4.4.4. Move: Semánticas de Movimiento

Capítulo 5

Containers: Contenedores de la STD

5.1. std::vector

5.1.1. std::array

5.2. std::string

5.3. std::map

5.4. std::set

Capítulo 6

Cosas cool la sintaxis

6.1. auto

6.2. for (auto x : container)

Parte III

Ideas de las Ciencias de la Computación

Capítulo 7

La Complejidad

7.1. Cotas y Notaciones

7.1.1. Big O Notation: Notación de O grande

Capítulo 8

Optimización

8.1. Límites de Tiempo de Ejecución

8.2. Límites de Memoria

Capítulo 9

Problemas NP

Parte IV

Estructuras de Datos

Capítulo 10

Arrays

Capítulo 11

Stacks LIFO: Pilas

Capítulo 12

Queue FIFO: Colas

Capítulo 13

Linked Lists: Listas Enlazadas

Capítulo 14

Binary Trees: Arboles Binarios

14.1. BTS: Arboles de Búsqueda

14.2. AVL - RedBlackTree: Arboles Autobalanceables

14.3. Trie

Capítulo 15

Heaps

Capítulo 16

Hash Tables

Parte V

Algoritmos Generales

Capítulo 17

Search: Búsquedas

17.1. Linear Search: Búsqueda Lineal

17.2. Binary Search: Búsqueda Binaria

17.3. Ternary Search: Búsqueda Ternaria

17.4. Upper Bound

17.5. Lower Bound

Capítulo 18

Sorting: Ordenamiento por Comparaciones

18.1. Bubble Sort

18.2. Selection Sort

18.3. Merge Sort

18.4. Quick Sort

Capítulo 19

Sorting: Ordenamiento NO por Comparaciones

19.1. Bucket Sort

Parte VI

Programación es solo matemáticas aplicadas

Capítulo 20

Binary: Explotando el Binario

20.1. Bits

20.1.1. Manejo de Bits

20.1.2. Operaciones con Bits

20.2. Conversiones entre Sistemas

20.3. Binary Exponentiation: Exponenciacion Binaria

20.4. Binary Multiplication: Multiplicación Binaria

Capítulo 21

Roots: Encontrar Raíces de ecuaciones

21.1. Newton - Ranphson

Capítulo 22

Teoría de Números

22.1. Divisibilidad

22.1.1. Euclides

22.2. Modulos

22.3. Fibonacci

22.4. Primos y Factores

22.4.1. Eratosthenes Sieve: Criba de Eratóstenes

22.4.2. Prime Factorization: Factorización

22.4.3. Divisores

22.4.4. Euler Totient: La Phi de Euler

Capítulo 23

Probabilidad

23.1. Inclusión Exclusión

Capítulo 24

Geometría

Parte VII

Técnicas de Solución

Capítulo 25

Ad-Hoc

Capítulo 26

Recursividad y BackTracking

Capítulo 27

Divide and Conquer: Divide y Vencerás

Capítulo 28

Greedy

Capítulo 29

Programación Dinámica

Parte VIII

Grafos y Flujos

Capítulo 30

Grafos y Gráficas

30.1. Representaciones

30.2. BFS: Breadth-first Search

30.3. DFS: Depth-first Search

30.4. Dijsktra: Camino más cercano