GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: PLANTILLA SOLUCIÓN DE C++

1. Introducción

A pesar que no todos los problemas se resuelven con el mismo o algoritmos similares la forma de codificar una solución algorítmica puede ser común lo cual nos lleva a construir o elaborar nuestra propia plantilla de codificación la cual siempre usaremos independientemente del algoritmo a codificar.

2. Conocimientos previos

2.1. Biblioteca o librería

En C++, se conoce como librerías (o bibliotecas) a cierto tipo de archivos que podemos importar o incluir en nuestro programa. Estos archivos contienen las especificaciones de diferentes funcionalidades ya construidas y utilizables que podremos agregar a nuestro programa, como por ejemplo leer del teclado o mostrar algo por pantalla entre muchas otras más.

La declaración de librerías, tanto en C como en C++, se debe hacer al principio de todo nuestro código, antes de la declaración de cualquier función o línea de código, debemos indicarle al compilador que librerías usar, para el saber que términos estaran correctos en la escritura de nuestro código y cuáles no. La sintaxis es la siguiente: #include <nombre de la librería>

2.2. Macro

Los macros son muy utilizados en C y C++. Estos básicamente son un alias que podemos incluir en nuestro código el cual, al momento de compilar, sera reemplazado por lo que hayamos definido. Para la declarar una macro se utiliza la directiva #define seguido de la definición de la macro. Esto puede ser útil si tienes que escribir varias veres el mismo código y quieres ahorrar tiempo de tipeo, o bien, si quieres que sea mas legible.

3. Desarrollo

Para conformar una plantilla que sirva de base para la codificación de cualquier algoritmo solución a un problema utilizando C++ vamos a dividir o seccionar dicha plantilla en seis secciones la cual vamos a describir cada una a continuación:

1. **Declaración de bibliotecas:** Lo primero que debemos definir en nuestro código son las bibliotecas o librerías que va utilizar nuestro código. Para incluirlas basta con utilizar la directiva #include seguido con el nombre de la biblioteca o libreria entre los operadores de menor que y mayor que. Cada inclusión se debe hacer por línea. El incluir biblioteca o librería que no es utilizada luego por el código no afecta en nada, ni en tiempo y memoria. En la versiones modernas de C++ con la siguiente inclusión:

```
#include <bits/stdc++.h>
```

es más que suficiente ya que esta biblioteca contiene a gran parte de las bibliotecas que usaremos para desarrollar soluciones a problemas o ejercicios de concursos.

2. **Declaración de macros:** La definición de macros mejora la legibilidad de nuestras soluciones y aumenta la reutilización de las mismas. No son de uso obiligatorio es solo una sugerencias que damos. Entre las macros que podemos citar su utilización están la del salto de linea (ENDL) que sustituye a la función **endl** por el alto costo computacional de esta última. Para definir una macro usamos la

3. **Inclusión de la std:** Esta definida por la siguiente instrucción:

```
using namespace std;
```

   El motivo es el de dar acceso al espacio de nombres (namespace) std, donde se encuentra encerrada toda la librería estándar. El motivo de encerrar la librería estándar en un espacio de nombres no es otro que el de hacer más sencilla la creación de proyectos muy grandes, de manera que el proyecto no deje de compilar debido a que se han escogido los mismos nombres para dos funciones, clases, constantes o variables. Es decir, las funciones que normalmente llamarías como **std::cout** solo tendrías que usar **cout**.

4. **Declaración de las variables globales:** Posterior a la inclusión de la std puede venir la declaración de las variables globales que necesitemos en nuestra solución. Esto tampoco es obligatorio pues podemos tener soluciones que no requieren variables globales.

5. **Declaración e implementación métodos auxiliares:** Como mismo nos puede suceder con las variables globales nos puede suceder que nuestra solución necesitemos la implementación y utilización de métodos auxiliares los cuales invocaremos en el método *main* por tanto la definición e implementación de estos métodos auxiliares deben hacerse previamente a la del método *main*.

6. **Declaración e implementación del método main:** Es elemento que junto con la inclusión de la bibliotecas y el std no pueden faltar en la solución. Es el método por el cual comienza la ejecución de nuestro algoritmo. Las principales instruccciones y las llamadas a funciones o métodos auxiliares deben hacerse desde él. Siempre todo por encima de la instrucción **return 0;**

4. Implementación

```
#include <bits/stdc++.h>
//Definir resto bibliotecas que necesites en la solucion
#define ENDL '\n'
#define OPTIMIZAR_IO ios_base::sync_with_stdio(0); cin.tie(0); cout.tie(0);
#define PRESICION(x) cout.setf(ios::fixed,ios::floatfield); cout.precision(x);
//Definir resto de macros personalizadas a gusto
```

```cpp
using namespace std;

//Declarar variables globales

//Declarar metodos auxiliares


int main() {

   //Optimizacion del cin y cout
   OPTIMIZAR_IO
   //Precicion de impresion del cout si es necesario
   PRESICION(2)

   // A partir de aqui desarrollamos e implemantamos nuestro algoritmo el cual
       debe incluir:
   // a.Declarar las variables locales del metodo main
   // b.Leer datos
   // c.Otras instrucciones propias del algoritmo
   // d.Imprimir resultados

   return 0;
}
```

5. Complejidad

Aunque pudiera parecer que escribir una solución basada en una plantilla pudiera complejizar una solución nada es más falso. Primero la complejidad del algoritmo no va aumentar por *montarte* sobre algo ya prestablecido al contrario va a ayudar ahorrar tiempo tipeo o codificación.

6. Aplicaciones

El uso de programar utilizando una plantilla base es muy utilizado por los programadores de concurso. Cada uno se elabora su propia plantilla base acorde a sus gustos y necesidades. Aumenta la reutilización de determinadas sentencias de código en diversas soluciones. Hace que la solución este más legible para realizar una depuración en caso que este mal la solución o exista un error.

7. Ejercicios propuestos

A continuación un grupo de ejercicios los cuales ya puedes hacer una vez visto esta guía y las anteriores a ella:

- DMOJ - Hello World

- DMOJ - El Ogro Ork
- DMOJ - No divisibles
- DMOJ -Cortes al tablero de ajedrez
- DMOJ - Coordenadas del Cuadrado No se guíe por los casos de pruebas, están mal.