GUÍA DE APRENDIZAJE PARA CONCURSANTES ICPC Y IOI: TIPOS DE ENTRADA EN C++

1. Introducción

En varias ocasiones nos podemos encontrar problemas o ejercicios que su solución algorítmica es bastante fácil o nos es accesible debido al conocimiento que tenemos en determinada área del conocimiento con la cual se puede solucionar pero nos resulta imposible codificar una solución debido a que presentamos problema con el formato de entrada que se nos presenta los datos iniciales en el problema o ejercicio. A las diferentes formatos de entrada de datos o al menos a los mas comunes y de como debemos abordalos le estaremos dedicando la siguiente guía de aprendizaje.

2. Conocimientos previos

2.1. Lectura de datos

Para la lectura se puede realizar por el método **cin** primitivo de C++ o por **scanf** primitivo de C, en el programa se pueden usar ambos métodos si es necesario pero no es recomendable.

2.2. Esctructura de repetición (bucles o ciclos)

Un **bucle** se utiliza para realizar un proceso repetidas veces. Se denomina también **lazo** o **loop**. El código incluido entre las llaves {} (opcionales si el proceso repetitivo consta de una sola línea), se ejecutará mientras se cumpla unas determinadas condiciones. Hay que prestar especial atención a los bucles infinitos, hecho que ocurre cuando la condición de finalizar el bucle (booleanExpression) no se llega a cumplir nunca. Se trata de un fallo muy típico, habitual sobre todo entre programadores poco experimentados. En el caso de los lenguajes de programación de C++ y Java se cuenta con varias instrucciones de tipo bucle o ciclo como son el **for**,**while** y **do ... while**.

3. Desarrollo

3.1. Entrada simple

Es la entrada mas sencilla que existe donde tu algoritmo deberá solo recibir una vez una determinada entrada y dar la solución a esa entrada.

Ejemplo

La primera y única línea de entrada contiene dos números enteros aa y bb, la cantidad de hijos y de hijas que tiene el ogro Ork.(DMOJ - El Ogro Ork)

Para ser capaces de leer los datos en una entrada con este formato basta con leer los datos en el orden explicado en el formato de entrada.

3.2. Entrada basado en casos de pruebas

Quizás sea el tipo de entrada más común que existe en los problemas o ejercicios de concurso. Se caracteriza porque siempre el primer dato de entrada es un entero que va indicar cuantas ins-

tancias diferentes del problema tu algoritmo debe ser capaz de resolver. Acto y seguido viene las entradas de valores para cada una de las instancias del problema debemos ser capaz de resolver con el mismo algoritmo.

Ejemplo

Una línea que contiene el número de casos T (1 <= T <= 100). Cada una de las siguientes líneas T contendrá el tamaño del lado (1 <= L <= 100) del cuadrado.(DMOJ - Calculando áreas)

La solución a este tipo de entrada es bastante sencilla solo debemos auxiliarnos de una estructura de control ciclica o de bucle dentro de la cual vamos a desarrollar la entrada de datos de una entrada simple para una instancia del problema aplicamos el algoritmo e imprimimos solución. Antes de caer en la estructura repetitiva debemos leer el valor que indicará la cantidad de instancias diferentes del problema.

```
Leer cantidad de casos de pruebas
Mientras la cantidad de casos de prueba sea mayor que 0:
   Leer entrada para una instancia del problema
   Algoritmo solucion
   Imprimir solucion para la instancia del problema
   Decrementar la cantidad de casos de prueba
```

3.3. Entrada hasta determinada condición

Similar al caso anterior nos van a proporcionar varias entradas a para diferentes instancias del problema aqui la diferencia radica en que no nos dicen previamente cuantas instancias del problema sino que cuando la entrada para una determinada instancia del problema cumpla con una determinada condición siginifica que se termino la entrada y nuestro programa debe detenerse.

Ejemplo

La entrada contiene uno o más casos de prueba. Cada caso consiste de un entero N par con 6<=N<1000000. La entrada temina con un 0.(DMOJ - Conjeturas de Goldbach)

La solución a este tipo de entrada es parecida a la anterior con la diferencia que voy a estar leyendo datos de entrada para una instancia del problema mientras dicha entrada cumpla no determinada condicion.

```
Leer entrada para una instancia del problema
Mientras la entrada leida cumpla con determinada condicion:
   Algoritmo solucion
   Imprimir solucion para la instancia del problema
   Leer entrada para una instancia del problema
```

3.4. Entrada hasta fin de fichero

Similar al caso anterior nos van a proporcionar varias entradas para diferentes instancias del problema aqui la diferencia radica en que no existe una determinada condición en la entrada de datos de una instancia del problema que cuando cumpla siginifica que se termino la entrada y nuestro programa debe detenerse. En nuestro caso el programa debe terminar cuando no exista mas datos de entradas.

Ejemplo

La entrada tiene varios casos de prueba:
La primera línea de cada caso de prueba contiene dos números enteros N ($1 \leq N \leq 1000000$) *y M* ($1 \leq M \leq 10000$)*, la cantidad de elementos de la secuencia y el número de consultas respectivamente. M líneas siguen con consultas de uno de los dos tipos explicados.*

- *En caso de que sea una consulta de tipo 1, la línea será de la forma 0 x y k, donde x,y,k son enteros,*($1 \leq x \leq y \leq N$,$1 \leq k \leq 10^9$)*.*
- *En caso de que sea una consulta de tipo 2, la línea será de la forma y1 x y, donde x,y son enteros,* ($1 \leq x \leq y \leq N$)*.*

(DMOJ - Multiplicación en Rango)

Existen multiples ejercicios donde no se especifican el fin de la entrada de datos. Este tipo de ejercicios son los que se conocen que su entrada de datos es hasta fin de fichero. Para solucionar este problema vamos auxiliarnos en que las funciones de captura de datos cuando son invocadas devuelven sin pudieron o no leer información. Conociendo lo anterior podemos elaborar la siguiente solución.

```
Mientras se pueda leer el primer dato de la entrada de una instancia del
   problema:
   Leer el resto de los datos de entradas de la instancia del problema
   Algoritmo solucion
   Imprimir solucion para la instancia del problema
```

4. Implementación

4.1. Entrada basado en casos de pruebas

```
//Declaro la variable que define la cantidad de casos
int cases;
cin>>cases;
while(cases--){
   //Leer entrada del caso inesimo
   //Algoritmo
   //Impresion de los resultados del caso inesimo
}
```

4.2. Entrada hasta determinada condición

```
int X,Y;
cin>>X>>Y;
//Se procesa entradas mientras la entrada se diferente de 0 0
while(X!=0 || Y!=0){
   //Algoritmo
   //Impresion de los resultados del caso inesimo
   cin>>X>>Y;
}
```

4.3. Entrada hasta fin de fichero

```
//Primer dato de entrada que para este caso en un entero
//pero puede ser de cualquier otro tipo
int n;
while(cin>>n){
   //Leer resto de la entrada si existe
   //Algoritmo
   //Impresion de los resultados
}
```

5. Complejidad

Una vez visto cada una de las posibles formatos de entradas estamos claro que la complejidad en la entrada simple será O(1) mientras en los demás casos la complejidad sería O(N) donde N sería la cantidad de entradas a procesar pero no debemos preocuparnos por esto por que los tiempos establecidos por los problemas siempre tienen en cuenta la entrada de datos.

6. Aplicaciones

El saber como leer cada tipo de entrada que nos puede facilitar y aclarar el trabajo de codificación de la solución del problema. Es muy común que a veces la mayor complejidad de un ejercicio en su codificación es el formato de entrada por tanto es útil conocer las variantes mas comunes de entrada y como tratarlas como parte de la solución.