Problemas y soluciones comunes

Introducción

Hemos visto los elementos básicos de la programación estructurada, según la teoría, esta es suficiente para resolver todo tipo de problemas computacionales, pero el poder resolverlos no significa que los problemas se resuelvan fácilmente. De hecho tienen que pasar por varías etapas, la primera es la que vimos en la unidad 1, Algoritmos, si no se imaginan como resolver el problema planteando una estrategia no van a poder empezar a programar.

Aunque esta etapa se volverá tan rutinaria que lo tendrán incorporada sin darse cuenta, pero por ahora antes de escribir un programa lo primero es sentarse a pensar "¿cómo encaro el problema?", "¿qué tengo que conocer que desconozco?", plantear una estrategia y ensayarla (imaginarse como resuelve todas las posibilidades) y luego nos sentamos a escribirla en un lenguaje de programación y a probarla...

...Y ver que falla.

Entonces debemos revisar nuestra estrategia, buscar nuevas cosas para aprender y poder aplicarlas al programa hasta que este salga como queremos.

Existen varios problemas comunes que nos vamos a encontrar y cuya solución a sido largamente ensayada por miles de programadores, vamos a ver algunos de ellos.

Las matemáticas

Si te gustan los juegos y odias las matemáticas no programes, y si no sabes matemáticas y te encanta programar juegos, estos te van a llevar por el mundo de los números hasta que llegue un momento que vas a dar clases de las mismas a más de uno. Matemáticas y videojuegos van de la mano, verás que las matemáticas son hermosas porque harán que aquello que imaginás se haga realidad.

También verán que las soluciones matemáticas comunes no nos satisfacen, sino que, encima de tener que dominar las soluciones comunes, tenemos que darles la vuelta, matemáticamente hablando, para que se adapten a nuestros programas, pero no se preocupen todo está en los libros, solo hay que saber buscarlo, no hay ningún profesor dándoles ejercicios y vigilandolos que no copien, aquí están ustedes buscando una solución para hacer algo parecido a lo que soñaron, así que aprendan a buscar, no le tengan miedo a las matemáticas.

Aprender a contar

Existe una estructura que se llama contador, es algo muy útil y prácticamente básico en programación, de hecho podríamos decir que no existe programa relativamente mediano que no lleve uno.

Por ejemplo nuestro programa del cañon, tiene 5 disparos, ¿Como nos enteramos en que disparo vamos?, muy simple creamos una variable que nos servirá de contador, la iniciamos a 1, y cada vez que disparamos la incrementamos en 1, con solo consultar la variable ya sabemos cuantos disparos se han hecho.

El hecho de que sea algo tan común hace que el C/C++ tenga un operador incremental ++, pero los contadores no necesariamente tienen que contar hacia adelante, supongamos que tenemos un juego con el que empezamos con 5 vidas, entonces creamos una variable vida con valor 5 y cuando perdemos una la decrementamos.

Tampoco es limitante que se incremente o decremente de a uno, puede ser en un valor mayor o menor o incluso variable, por ejemplo en un juego de baloncesto, el tiro normal es de 2 puntos, aunque también está la posibilidad de los tiros de 3 puntos, y como última posibilidad tenemos los tiros libres que valen 1 punto. veamos ejemplos de notación:

```
x=x+2; //un doble en baloncesto, sumamos 2 puntos
x=x+3; //un triple en baloncesto, sumamos 3 puntos
x=x-5; //decrementamos de a 5
x=x+rand()%5; //incrementamos al azar entre 0 y 4 a x
```

Como estas operaciones son muy comunes, C/C++ tiene una forma de escribirla más sintéticas:

```
x+=2; //un doble en baloncesto, sumamos 2 puntos
x+=3; //un triple en baloncesto, sumamos 3 puntos
x-=5; //decrementamos de a 5
x+=rand()%5; //incrementamos al azar entre 0 y 4 a x
```

También es válido para la multiplicación (*) la división (/) y resto (%)

Un contador normalmente se utiliza junto con un bucle, generalmente para indicarle cuando el bucle a terminado:

ejemplo:

imprimir todos los números entre 1 y 100

```
int n; //contador

n=1; //empezamos en 1

while(n<=100) //mientras n sea menor o igual a 100
{
   cout<<n<<" "; //imprimimos n
   ++n; //incrementamos el contador
}
```

El for está para escribir más sintético:

```
for(int n=1;n<=100;++n) cout<<n<<" ";
```

Qué pasa si queremos los número pares empezando del 2

```
int n; //contador

n=2; //empezamos en 2

while(n<=100) //mientras n sea menor o igual a 100
{
   cout<<n<<" "; //imprimimos n
   n+=2; //incrementamos el contador
}
```

y con for sería:

```
for(int n=2;n<=100;n+=2) cout<<n<<" ";
```

Les dejo como ejercicio imprimir de 100 a 1 y de 100 a 1 decrementando de a 2, si no les sale consulten en el foro.

Si les sale veamos cosas un poco más complicadas:

imprimir el factorial del número n (pista: este es un ejemplo para *=) (pista 2: van a necesitar 2 contadores para resolverlo) donde n lo piden al ingresar al programa

imprimir los valores de la serie de Fibonacci hasta el término n, donde n lo piden al ingresar al programa.

Menor y Mayor

Otra cosa que es muy común es buscar cosas, y entre ellas el menor y el mayor de una serie.

Supongamos que buscamos el menor, entonces solo tenemos que recordar el menor hasta el momento, cuando llega un nuevo número lo comparamos con el que recordábamos si es menor a este el nuevo número sera el menor que recordemos, sino seguirá siendo el anterior.

veámoslo en programa:

```
int menor; // nuestra variable para recordar el menor.
int n; //número ingresado

cin>>n; //ingresamos el primer número
menor=n; //el primer número seguro que es el menor de todos los ingresados hasta ahora.

while(n==999999) //cuando se ingrese 999999 termina el programa (es para que no quede pidiendo números para siempre)
{
   cin>>n;
   if(n<menor) menor=n;
}
cout<<"\nEl menor de todos los ingresados es: "<<menor;
```

Para buscar el mayor es similar solo que debemos recordar el mayor y comparar con este cada número que ingresa. Se los dejo de ejercicio a ver como les sale. Ya saben por problemas consultar en el foro.

Si ya lo resolvieron veamos algo un poco más difícil:

Igual que el ejemplo anterior pero buscando el mayor y el menor de todos los números ingresados.

Buscar los dos números menores, por ejemplo si la lista es 5, 7, 3, 9, -1, 16 debería mostrar 3 y -1

Y ahora uno combinado

Sacar el promedio de una lista de números sin tener en cuenta ni el mayor ni el menor, la entrada de números finaliza ingresando 999999 que obviamente no debe ser incluido para sacar el promedio.

Y aplicado a los juegos

La idea es implementar las tiradas de dados descritas en esta página:

http://www.fororol.com/foros/index.php?topic=137.0

Los dados de rol tienen la particularidad de ser de varias caras, es decir existen de 4, 5, 6 (el que usamos para la generala), 8, etc. los valores de cada cara se numeran de 1 al numero de caras, por ejemplo, uno de cuatro caras tendrá 1, 2, 3 y 4, uno de 6 caras (los dados comunes) 1, 2, 3, 4, 5, 6, uno de 20 caras tendrá los valores del 1 al 20.

Pero además puede tener restricciones en las tiradas dependiendo de las situaciones, como penalidades o ventajas en el juego.

Simulemos entonces:

lanza 2 dados de 20 caras, relanzando cualquier resultado necesario para que los resultados sean mayores de diez

Ejemplo de resultado: ~~3~~ 8 ~~10~~ ~~3~~ 17, ~~2~~ ~~10~~ 9 ~~10~~ 14, total 31 (nota: el tachado no se puede hacer en consola)

lanza 2d10 x veces y muestra el resultado total de los cuales elegir el mejor de todos, x debe pedirlo al principio

Ejemplo:

para 3 tiradas:

2 6 total: 8
4 5 total: 9
1 5 total: 6

la mejor tirada 4 5 total 9

Cualquier duda preguntar en el foro.