UNIVERSIDAD NACIONAL
DE ASUNCIÓN

Facultad de
Ingeniería

COMPUTACIÓN

Vectores en C++

Aux. Marcos Antonio Benítez Ocampos

Resolución de ejercicios

Resolución de ejercicios

En este material se propondrá la resolución de dos ejercicios relacionados a los arreglos unidimensionales o vectores en lenguaje de programación C++, así pues, la idea consistirá en que dado dos enunciados al tema mencionado anteriormente, proponer en consecuencia una resolución para los mismos.

Es pertinente mencionar que la resolución propuesta no es única, pudiendo los estudiantes encontrar diversas soluciones para el mismo problema y adecuados a su estilo; la resolución propuesta es a criterio del auxiliar de enseñanza una que podría facilitar la comprensión de la utilización de los arreglos unidimensionales así como también otros comandos del lenguaje de programación C++.

Los ejercicios serán detallados lo más posible en este material.

Ejercicio 1

Elaborar un programa en lenguaje C++ que lea un numero entero N, mayor que 2 y menor o igual que 50, luego cargue un vector vect con N números enteros comprendidos entre 1 y 1000, inclusive, generados al azar (utilice la función semilla) e imprima en el vector generado, luego imprima en pantalla el promedio de los números impares y el de los números pares.

Como todo programa en lenguaje C++ lo primero es la cabecera, donde incluimos las librerías a utilizar y donde es buena costumbre declarar ya las funciones a utilizar en el programa. Entonces las librerías a utilizar serán:

1.1) Iostream, para el flujo de entrada y salida de datos;

1.2) Ctime, librería utilizada para obtener la hora del sistema operativo, utilizaremos la misma para poder generar aleatoriamente los números pedidos.

Observaciones:

- Pueden utilizar indistintamente la librería time.h en vez de ctime, ambas librerías cumplen básicamente con lo mismo, time.h es una librería del lenguaje C propiamente y ctime es de C++.
- Por la versión del compilador que utilizaré para la compilación necesito agregar la librería cstdlib para poder utilizar el comando system("PAUSE"), en otras versiones del compilador esto puede no ser necesario.
- Las librerías cstdlib y stdlib.h se diferencian básicamente en lo mismo que las librerías ctime y time.h, así como las librerías cmath y math.h.

1.3) Como tenemos un número que debe ser entero y perteneciente al intervalo (2, 50], crearemos también una función que valide que dicho número cumpla con esas propiedades. La función entonces recibe como parámetro de entrada un número de tipo **float** y retorna un valor de tipo **int**; nuestra función se llama **valida**.

Entonces, nuestras primeras líneas de programación son las siguientes:

```
#include<iostream>
using namespace std;
#include<cstdlib>
#include<ctime>
int valida(float N);
```

1.4) Seguidamente pasamos a realizar la función **main**. Inicializamos la semilla para poder utilizar la generación de números aleatorios; declaramos la variable **N** (mismo nombre que el dado en el enunciado) y solicitamos al usuario que ingrese un número con las propiedades antes mencionadas y validamos el valor de la variable:

```
int main()
{
   srand(time(0));
   float N;
   cout<<"Ingrese un numero natural del intervalo (2, 50]: ";
   cin>>N;
   N=valida(N);
```

1.5) A continuación declaramos el vector **vect** de **N** componentes y las siguientes variables:

- **pp**: variable utilizada para almacenar el valor del promedio de los números pares.
- **pi**: variable utilizada para almacenar el valor del promedio de los números impares.
- **ci:** variable utilizada para contar la cantidad de números impares.
- **may, men:** variables utilizadas para almacenar el mayor y menor número respectivamente.

1.6) Seguidamente cargamos las componentes del vector con la función **rand()** teniendo en cuenta que los números generados estén comprendidos en el intervalo [1, 1000]. Luego imprimimos el vector generado.

```
int vect[int(N)], pp=0, pi=0, ci=0, may, men;
for(int i=0;i<N;i++)
   vect[i]=rand()%1000+1;
cout<<"\nEl vector generado es:\n{\t";
for(int i=0;i<N;i++)
   cout<<vect[i]<<"\t\t";
cout<<"}";
```

Observaciones:

- Al declarar el vector de **N** componentes, escribimos "**int**(N)", esto con el fin de evitar una advertencia o bien un error de compilación, pues no hay que olvidar que la variable **N** es de tipo **float** y para declara un arreglo unidimensional se debe colocar la cantidad de componentes del mismo con un número entero o variable de tipo entera en su defecto.
- Al utilizar la estructura de control repetitiva **for**, vemos que tiene una sola sentencia, por lo cual, no es necesario utilizar la llave de apertura y de cierre en el ámbito de dicha estructura de control. Se aclara que esto solo es posible cuando estamos seguros de que la estructura tendrá una sola sentencia, si prefieren colocar las llaves pueden hacerlo.

1.7) Pasamos a la fase de los cálculos y las comparaciones; comenzamos por el algoritmo para determinar el mayor y menor elementos, para esto basta con suponer que ambos son el primer elemento del vector e ir comparando dentro un bucle, en este caso optamos por el ciclo **for**, cada componente del vector, si la componente resulta ser mayor que el mayor que supusimos entonces debemos tomar ese nuevo valor; para el menor, si la componente resulta ser menor que la que supusimos entonces debemos tomar ese nuevo valor, así vamos realizando la comparación para todos los elementos del vector y en un recorrido ya tendremos el mayor y menor elemento del vector. Para determinar los elementos que son impares, dentro de la misma estructura de control verificamos si es divisible por dos (2) cada elemento del vector, si no lo es vamos sumando los elementos impares en la variable **pi** a la par que vamos contando la cantidad de los elementos impares, sino, será par e iremos sumando los elementos pares en la variable **pp**.

```
may=vect[0];
men=vect[0];
for(int i=0;i<N;i++)
{
   if(may<vect[i])
      may=vect[i];
   if(men>vect[i])
      men=vect[i];
   if(vect[i]%2)
   {
      pi+=vect[i];
      ci++;
   }
   else
      pp+=vect[i];
}
```

Observaciones:

- Notar como en las estructuras de control selectivas **if** que tengan una sola sentencia, evitamos utilizar las llaves de apertura y de cierre.
- Resaltamos el hecho de que si un número no es impar, entonces es par y; también que no es necesario contar la cantidad de números pares, pues al tener la cantidad de números impares el resto de los

números en el vector serán pares y se obtiene por diferencia entre la longitud del vector y la cantidad de números impares.

- Recordar que el argumento de una estructura de control selectiva "**if**(argumento)" es tal que si dicho argumento es un valor booleano verdadero (True) se ejecutan las sentencias de la estructura de selección, sino, no. En C++ el valor booleano **False** es equivalente al número entero cero (0), uno distinto de cero es equivalente a **True**. Entonces la expresión "vect[i]%2" es una expresión que devuelve los valores: 0 (False) y 1 (True), es así que si la componente **i** del vector es impar, la expresión devuelve un valor 1 lo que hará que se ejecuten las sentencias de la estructura de control selectiva.

1.8) Seguidamente debemos imprimir el valor almacenado en las variables **may** y **men** además del promedio de los números impares y pares.

```
  cout<<"\n\nEl mayor numero de la lista es: "<<may<<endl;
  cout<<"\n\nEl menor numero de la lista es: "<<men<<endl;
  cout<<"\n\nEl promedio de los numeros pares es: "<<pp/f(N-ci)<<endl;
  cout<<"\n\nEl promedio de los numeros impares es: "<<pi/float(ci)<<endl;
  cout<<"\nFIN DEL PROGRAMA\n";
  system("PAUSE");
  return 0;
}
```

Observación:

- Es importante recordar como manejar los tipos de datos que almacenan nuestras variables: si dividimos dos números enteros o en su defecto dividimos dos variables de tipo entero, el resultado será un número entero si o si, en cambio si dividimos dos números o variables de tipo flotante o un entero y otro flotante o un flotante y otro entero, entonces el resultado será un número de tipo flotante. Por esta razón en nuestro código fuente le cambiamos el tipo de dato al valor que nos da la variable **ci**, no así a la expresión **N-ci**, pues aquí tenemos una diferencia de un número flotante y un entero, lo que da un número flotante.

1.9) Finalmente nos resta escribir el cuerpo de la función valida.

```
int valida(float N)
{
  while(N<2||N>50||N!=int(N))
  {
    cout<<"\n\nERROR, ingrese un numero natural del intervalo (2, 50]: ";
    cin>>N;
  }
  return int(N);
}
```

Observación:

- Para el prototipado de una función en C++ se tienen dos opciones, declarar la función en la cabecera y luego escribir el cuerpo de la función después de la función **main** (que es lo que nosotros implementamos) o bien en la cabecera declarar la función y escribir su cuerpo también. Generalmente los programadores tienden a programar con el estilo elegido en nuestro código fuente, pues al revisar un código fuente se espera revisar primeramente la función principal y luego las funciones implementadas. Se aclara que ambos estilos son correctos.

Finalmente nuestro código fuente completo es el siguiente:

```
#include<iostream>
using namespace std;
#include<cstdlib>
#include<ctime>
int valida(float N);
int main()
{
   srand(time(0));
   float N;
   cout<<"Ingrese un numero natural del intervalo (2, 50]: ";
   cin>>N;
   N=valida(N);
   int vect[int(N)], pp=0, pi=0, ci=0, may, men;
   for(int i=0;i<N;i++)
      vect[i]=rand()%1000+1;
   cout<<"\nEl vector generado es:\n{\t";
   for(int i=0;i<N;i++)
      cout<<vect[i]<<"\t\t";
   cout<<"}";
   may=vect[0];
   men=vect[0];
   for(int i=0;i<N;i++)
   {
      if(may<vect[i])
         may=vect[i];
      if(men>vect[i])
         men=vect[i];
      if(vect[i]%2)
      {
         pi+=vect[i];
         ci++;
      }
      else
         pp+=vect[i];
   }
   cout<<"\n\nEl mayor numero de la lista es: "<<may<<endl;
   cout<<"\n\nEl menor numero de la lista es: "<<men<<endl;
   cout<<"\n\nEl promedio de los numeros pares es: "<<pp/f(N-ci)<<endl;
   cout<<"\n\nEl promedio de los numeros impares es: "<<pi/float(ci)<<endl;
   cout<<"\nFIN DEL PROGRAMA\n";
   system("PAUSE");
   return 0;
}
int valida(float N)
{
   while(N<2||N>50||N!=int(N))
   {
      cout<<"\n\nERROR, ingrese un numero natural del intervalo (2, 50]: ";
      cin>>N;
   }
   return int(N);
}
```

Ejercicio 2

Elaborar un programa en lenguaje C++ que lea un numero entero y positivo N, de más de 4 dígitos, si el numero tiene 4 o menos dígitos finalizar el programa; luego imprima en pantalla el numero obtenido con el orden de sus cifras invertido utilizando un vector.

Como todo programa en lenguaje C++ lo primero es la cabecera, donde incluimos las librerías a utilizar y donde es buena costumbre declarar ya las funciones a utilizar en el programa. Entonces las librerías a utilizar serán:

1.1) Iostream, para el flujo de entrada y salida de datos;

Observaciones:

- Por la versión del compilador que utilizaré para la compilación necesito agregar la librería cstdlib para poder utilizar el comando system("PAUSE"), en otras versiones del compilador esto puede no ser necesario.

1.2) Como tenemos un número que debe ser entero, positivo y de más de 4 dígitos, crearemos también una función que valide que dicho número cumpla con esas propiedades. La función entonces recibe como parámetro de entrada un número de tipo **float** y retorna un valor de tipo **int**; nuestra función se llama **valida**.

Entonces, nuestras primeras líneas de programación son las siguientes:

```
#include<iostream>
using namespace std;
#include<cstdlib>
int valida(float N);
```

1.3) Seguidamente pasamos a realizar la función **main**; declaramos la variable **N** (mismo nombre que el dado en el enunciado) y solicitamos al usuario que ingrese un número con las propiedades antes mencionadas y validamos el valor de la variable:

```
int main()
{
  float N;
  cout<<"Ingrese un numero natural de mas de 4 digitos: ";
  cin>>N;
  N=valida(N);
```

1.4) Atendiendo a la instrucción de que si el usuario ingresa incorrectamente el valor de **N** el programa debe finalizar, codificamos eso, teniendo en cuenta que nuestra función **valida** ya verifica todo, si el número fue mal ingresado, devolverá el valor 0.

```
  if(N==0)
      return 0;
```

1.5) A continuación implementamos el algoritmos que nos va a permitir determinar la cantidad de dígitos del número ingresado y para eso declaramos las variables: **aux:** para almacenar el valor del número ingresado; **cd**: para almacenar la cantidad de dígitos de **N**, luego declaramos el vector con la cantidad de componentes correspondiente.

```
  int cd=0, aux=N;
  while(aux)
  {
    aux/=10;
    cd++;
```

```
    }
    int vect[cd];
    aux=N;
```

1.6) A continuación, mediante una estructura de control repetitiva **for** cargamos los dígitos de **N** en el vector, es pertinente aclarar que los dígitos se cargan en secuencia inversa.

```
    for(int i=0;i<cd;i++)
    {
        vect[i]=aux%10;
        aux/=10;
    }
```

1.7) Seguidamente imprimimos las componentes del vector desde su primera componente hasta la última, eso dará el número **N** pero con el orden de sus cifras invertidas.

```
    cout<<"\nEl numero "<<N<<" con el orden de sus cifras invertido es: ";
    for(int i=0;i<cd;i++)
        cout<<vect[i];
    cout<<"\nFIN DEL PROGRAMA\n";
    system("PAUSE");
    return 0;
}
```

1.8) Finalmente nos resta escribir el cuerpo de la función valida.

```
int valida(float N)
{
    while(N<0||N!=int(N))
    {
        cout<<"\n\nERROR, ingrese un numero natural de mas de 4 digitos: ";
        cin>>N;
    }
    int cont=0, aux=N;
    while(aux)
    {
        aux/=10;
        cont++;
    }
    if(cont<5)
        return 0;
    else
        return int(N);
}
```

Finalmente nuestro código fuente completo es el siguiente:

```
#include<iostream>
using namespace std;
#include<cstdlib>
int valida(float N);
int main()
{
   float N;
   cout<<"Ingrese un numero natural de mas de 4 digitos: ";
   cin>>N;
   N=valida(N);
   if(N==0)
      return 0;
   int cd=0, aux=N;
   while(aux)
   {
      aux/=10;
      cd++;
   }
   int vect[cd];
   aux=N;
   for(int i=0;i<cd;i++)
   {
       vect[i]=aux%10;
       aux/=10;
   }
   cout<<"\nEl numero "<<N<<" con el orden de sus cifras invertido es: ";
   for(int i=0;i<cd;i++)
      cout<<vect[i];
   cout<<"\nFIN DEL PROGRAMA\n";
   system("PAUSE");
   return 0;
}
int valida(float N)
{
   while(N<0||N!=int(N))
   {
      cout<<"\n\nERROR, ingrese un numero natural de mas de 4 digitos: ";
      cin>>N;
   }
   int cont=0, aux=N;
   while(aux)
   {
      aux/=10;
      cont++;
   }
   if(cont<5)
      return 0;
   else
      return int(N);
}
```