Implementación de pseudocódigo en C++

Objetivos

- Implementar en C++ algoritmos escritos en pseudocódigo.

A continuación, se indica cómo se implementa el pseudocódigo visto en clase utilizando el lenguaje C++. Se estructura en los siguientes apartados:

1) Declaración de variables.
2) Escribir valores por pantalla.
3) Leer valores por teclado.
4) Instrucción condicional "si".
5) Bucle "mientras".
6) Bucle "para".
7) Definición de funciones.
8) Llamada a funciones.

1) Declaración de variables

Pseudocódigo	Implementación en C++

1.1) Variables simples
`x:natural`
`x:entero`
`y:real`
`y:real`
`a,b:carácter`
	`int x;`
	`int x;`
	`float y;`
	`double y;`
	`char a,b;`

1.2) Vectores En los algoritmos en pseudocódigo generalmente los vectores empiezan en la posición 1. Cuando se utilice la posición 0 de los vectores en el pseudocódigo, se especificará en el algoritmo.	En C++ los vectores empiezan por la posición 0. Para mantener el pseudocódigo de los algoritmos, en la implementación **no utilizamos la posición 0** de los vectores. En su lugar, definimos un vector con **un elemento más**.
1.2.1) Vectores de tamaño fijo Vector de 10 números reales `V:real[10]`	`float V[11];`

1.2.2) Vectores de tamaño variable
 	
Vector cuyo tamaño depende del valor de una variable. El valor de la variable no lo sabemos hasta el momento de ejecución.
 	
`V:real[n]`
	Hay que reservar memoria.
	En C++ reservamos memoria con new. También se puede reservar con malloc, calloc (librería: stdlib.h).
	`// 1. Declarar variables`
	`float *V; int n;`
	
	`// 2. Asignar valores a n`
	
	`// 3. Reservar memoria`
	`V = new float [n+1];`
	`if (V == NULL)`
	`{`
	`  cout << "Error al reservar memoria" << endl;`
	`  return -1;  // exit(-1);`
	`}`
	En el caso de que no se pueda reservar memoria podemos salirnos de la función con return. Si utilizamos

	la función exit, finaliza el programa. La función exit se encuentra en la librería stdlib.h.
1.3) Matrices	
En los algoritmos en pseudocódigo se utilizan matrices que generalmente empiezan en la fila 1 y columna 1.	En C++ las matrices empiezan por la fila y columna 0. Para mantener el pseudocódigo de los algoritmos en la implementación **no utilizamos la posición 0.** En su lugar, definimos un vector con **un elemento más para las filas y un elemento más para las columnas**.
1.3.1) Matrices de tamaño fijo	
Matriz de 10 filas y 20 columnas	
`A:entero[10,20]`	`int A[11][21];`
1.3.2) Matrices de tamaño variable	
Matriz cuyas dimensiones solamente se saben en tiempo de ejecución del programa.	**Hay que reservar memoria.** En C++ reservamos memoria con new. También se puede reservar con malloc, calloc (librería: stdlib.h).

`A:entero[nfilas,ncolumnas]`	`// 1. Declarar variables`
	`int **A, nfilas, ncolumnas;`
	`int i;`
	
	`// 2. Asignar valores a nfilas y ncolumnas`
	
	`// 3. Reservar memoria`
	`A = new int* [nfilas+1];`
	`if (A == NULL)`
	`{`
	`  cout << "Error al reservar memoria" << endl;`
	`  return -1;`
	`}`
	
	`for (i=1 ; i<=nfilas ; i++)`
	`{`
	`  A[i] = new int [ncolumnas+1];`
	`  if (A[i] == NULL)`
	`  {`
	`  cout << "Error al reservar memoria";`
	`  cout << endl;`
	`  return -1;`
	`  }`
	`}`

2) Escribir valores por pantalla

Pseudocódigo	Implementación en C++
	En C++ utilizamos el operador de salida **cout**, que se encuentra en la librería iostream. También se puede utilizar la función printf de la librería stdio.h.
`escribir(nombrevariable)`	`cout << nombrevariable;`

2.1) Escribir un valor

```
x:entero  y:real  z:carácter

escribir(x)
escribir(y)
escribir(z)

V:real[10]
escribir(V3);

A:entero[10,20]
escribir(A3,4)
```

```
int x; float y; char z;

cout << x;
cout << y;
cout << z;

float V[11];
cout << V[3];

int A[11][21];
cout << A[3][4];
```

2.2) Escribir un vector

Hay que implementar un bucle para imprimir **cada uno** de los elementos del vector.

```
V:real[10]

escribir(V)
```

```
float V[11];

int i;
for (i=1 ; i<=10 ; i++)
   cout << V[i] << " ";
cout << endl;  // Imprimir un salto de línea
```

2.2) Escribir una matriz

Hay que implementar un bucle anidado.
Para cada una de las filas de la matriz se imprimen los elementos de sus columnas.

```
A:entero[10,20]

escribir(A)
```

```
int A[11][21];

int i,j;
for (i=1 ; i<=10 ; i++)
{
   for (j=1 ; j<=20 ; j++)
      cout << A[i][j] << " ";
   cout << endl;
}
```

3) Leer valores por teclado

Pseudocódigo	Implementación en C++
	En C++ utilizamos el operador de entrada **cin**, que se encuentra en la librería iostream. También se puede utilizar la función printf de la librería stdio.h.
`leer(nombrevariable)`	`cin >> nombrevariable;`
3.1) Leer un valor	
`x:entero  y:real  z:carácter`	`int x; float y; char z;`

`leer(x)`
`leer(y)`
`leer(z)`
	`cin >> x;`
	`cin >> y;`
	`cin >> z;`

`V:real[10]` `leer(V3);`	`float V[11];` `cin >> V[3];`
`A:entero[10,20]` `leer(A3,4)`	`int A[11][21];` `cin >> A[3][4];`

3.2) Leer un vector

Hay que implementar un bucle para leer **cada uno** de los elementos del vector.

```
V:real[10]

leer(V)
```

```
float V[11];

int i;
for (i=1 ; i<=10 ; i++)
   cin >> V[i];
```

3.3) Leer una matriz

Hay que leer elemento a elemento.

```
A:entero[10,20]

leer(A)
```

```
int A[11][21];

int i,j;
for (i=1 ; i<=10 ; i++)
   for (j=1 ; j<=20 ; j++)
      cin >> A[i][j];
```

4) Instrucción condicional "si"

Pseudocódigo

```
si a = b

   imprimir("Son iguales")

si no

   si a < b
      imprimir("a menor que b")
   si no
      imprimir("a mayor que b")
   fsi
fsi
```

Implementación en C++

```
if ( a == b )
{
   cout << "Son iguales";
}
else
{
   if ( a < b )
      cout << "a menor que b";
   else
      cout << "a mayor que b";

}
```

En este ejemplo en el segundo if se realiza una única instrucción tanto si la condición a<b es verdadera como si es falsa y no se han incluido las llaves. Si hubiera más de una instrucción hay que utilizar llaves para englobar todas las instrucciones dentro de cada caso.

5) Bucle "mientras"

Pseudocódigo

```
mientras j<=m  y  Ai,j>0 hacer

   valor ← valor +Ai,j
   j ← j+1
fmientras
```

Implementación en C++

```
while ( j<=m && A[i][j]>0 )
{
   valor = valor + A[i][j];
   j = j + 1;
}
```

6) Bucle "para"

Pseudocódigo

```
para i←1 hasta n hacer

   Vi ← 0
   x ← x + i
fpara
```

Implementación en C++

```
for (i=1 ; i<=n ; i++)
{
   V[i] = 0;
   x = x + i;
}
```

7) Definición de funciones

Pseudocódigo

```
función Ejemplo(x:entero,
                y:&real,
                z:carácter,
                V1:real[n1],
                V2:&real[n2],
                A:entero[nfil1,ncol1],
                B:&entero[nfil2,ncol2]
               ):entero

   valor:entero
   ...
   x ← x + 1
   y ← y / 2
   V1x ← x * y
   V2x ← x + y
   Ax,x ← x2
   Bx,x-1 ← x - Ax,x
   ...

   devolver valor
ffunción
```

Implementación en C++

```
int Ejemplo(int x,
            float &y,
            char   z,
            float *V1, int n1,
            float *V2, int n2,
            int   **A, int nfil1, int ncol1,
            int   **B, int nfil2, int ncol2
           )
{
   int valor;
   ...
   x = x + 1;
   y = y / 2;
   V1[x] = x * y;
   V2[x] = x + y;
   A[x][x] = x * x;
   B[x][x-1] = x - A[x][x];
   ...

   return valor;
}
```

8) Llamada a funciones

Pseudocódigo

Supongamos que queremos llamar a la función Ejemplo del apartado anterior y tenemos definidas las siguientes variables

```
a:entero
b:real
c:carácter
P1:real[tam1]
P2:real[tam2]
Q1:entero[nf1,nc1]
Q2:entero[nf2,nc2]
result:entero

...

result ← Ejemplo(a,b,c,P1,P2,Q1,Q2)
```

Implementación en C++

```
int a;
float b;
char c
float *P1; int tam1;
float *P2; int tam2;
float **Q1; int nf1, nc1;
float **Q2; int nf2, nc2;
int result;

/*
Aqui hay que reservar memoria para los vectores y
matrices porque no son de tamaño fijo
(ver apartados 1.2.2 y 1.3.2)
*/

...

result = Ejemplo(a,b,c,P1,tam1,P2,tam2,
                 Q1,nf1,nc1,Q2,nf2,nc2);
```

Actividad

1.- Dado el siguiente algoritmo en pseudocódigo que devuelve el número de veces que se encuentra un valor x dentro de un vector V de n elementos

```
función Contar (V:entero[n], x:entero):entero

   i,nveces:entero

   nveces ← 0
   para i←1 hasta n hacer
      si Vi = x
         nveces ← nveces + 1
      fsi
   fpara
   devolver nveces

ffunción
```

Implementa un programa en C++ que pida al usuario el elemento a buscar (x) y el tamaño del vector (n). A partir de estos valores el programa generará tres vectores para representar los casos peor, mejor y cualquier otro e imprimirá los vectores generados y el número de veces que encuentra x en los vectores.

A continuación, se detallan los pasos a seguir:

1.1.- Pide por teclado el valor a buscar x y el tamaño n del vector.

1.2.- Genera tres vectores llamados vpeor, vmejor y vcualquiera del tamaño introducido por el usuario. Cada uno de estos vectores representará los casos peor, mejor y cualquier otro, respectivamente.

IMPORTANTE: Recuerda que si no es el tamaño del vector hay que reservar memoria para **n+1** elementos y que se utilizan las posiciones del vector de 1 hasta n.

1.3.- Inicializa cada vector para que representen los casos posibles del algoritmo, imprime los vectores y verifica los datos que tienen.

- vpeor: todos los elementos iguales al valor x.
- vmejor: todos los elementos distintos del valor x.
- vcualquiera: cualquier otra distribución. Por ejemplo, la mitad de elementos del vector iguales a x y el resto diferentes.

1.4.- Implementa la función *Contar* según el pseudocódigo indicando anteriormente, siguiendo las reglas apropiadas de transformación del pseudocódigo.

Para implementar la función puedes hacerlo de dos formas:

a) Definiendo la función antes de la función *main*
b) Incluyendo el prototipo de la función al principio y definiendo la función *Contar* después de la función *main*.

1.5.- Llama a la función *Contar* desde el programa principal para los tres vectores e imprime en la función *main* el resultado que devuelve esta función para cada uno de los casos.

En la Figura 1 se muestra un ejemplo de ejecución para x=7 y n=10.

```
Introduce valor a buscar (x): 7
Introduce n.elementos del vector (n): 10

Vector vpeor:
7 7 7 7 7 7 7 7 7 7

Vector vmejor:
8 9 10 11 12 13 14 15 16 17

Vector vcualquiera:
7 7 7 7 7 13 14 15 16 17


CASO PEOR
Numero veces: 10

CASO MEJOR
Numero veces: 0

OTRO CASO
Numero veces: 5

Presione una tecla para continuar . . .
```

Figura 1: Ejemplo de ejecución para x=7 y n=10.