

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	<pre>int x; int x; float y; double y; char a,b;</pre>
1.2) Vectores En los algoritmos en pseudocódigo generalmente los vectores empiezan en la posición 1. NOTA: Cuando se utilice la posición 0 de los vectores en el pseudocódigo, se especificará en el algoritmo.	<p>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 <u>un elemento más</u>.</p>
1.2.1) Vectores de tamaño fijo Vector de 10 números reales V:real[10]	<pre>float V[<u>11</u>];</pre>
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]	<p>Hay que reservar memoria. En C++ reservamos memoria con new. También se puede reservar con malloc, calloc (librería: stdlib.h).</p> <pre>float *V; int n; V = new float [<u>n+1</u>]; if (V == NULL) { cout << "Error al reservar memoria" << endl; return -1; // exit(-1); }</pre>

<p>1.3) Matrices</p> <p>En los algoritmos en pseudocódigo se utilizan matrices que generalmente empiezan en la fila 1 y columna 1.</p> <p>1.3.1) Matrices de tamaño fijo</p> <p>Matriz de 10 filas y 20 columnas</p> <pre>A:entero[10,20]</pre> <p>1.3.2) Matrices de tamaño variable</p> <p>Matriz cuyas dimensiones solamente se saben en tiempo de ejecución del programa.</p> <pre>A:entero[nfilas,ncolumnas]</pre>	<p>En el caso de que no se pueda reservar memoria podemos salirnos de la función con return o con exit(-1). Si utilizamos la función exit, finaliza el programa. La función exit se encuentra en la librería stdlib.h.</p> <p>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 <u>un elemento más para las filas y un elemento más para las columnas</u>.</p> <pre>int A[11][21];</pre> <p>Hay que reservar memoria. En C++ reservamos memoria con new. También se puede reservar con malloc, calloc (librería: stdlib.h).</p> <pre>int **A, nfilas, ncolumnas; int i; 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" << endl; return -1; } }</pre>
--	--

2) Escribir valores por pantalla	
Pseudocódigo	Implementación en C++
<pre>escribir(nombrevariable)</pre>	<p>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.</p> <pre>cout << nombrevariable;</pre>

<p>2.1) Escribir un valor</p> <pre>x:entero y:real z:carácter escribir(x) escribir(y) escribir(z) V:real[10] escribir(V₃); A:entero[10,20] escribir(A_{3,4})</pre>	<pre>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];</pre>
<p>2.2) Escribir un vector</p> <pre>V:real[10] escribir(V)</pre>	<p>Hay que implementar un bucle para imprimir cada uno de los elementos del vector.</p> <pre>float V[11]; int i; for (i=1 ; i<=10 ; i++) cout << V[i] << " "; cout << endl; // Imprimir un salto de línea</pre>
<p>2.2) Escribir una matriz</p> <pre>A:entero[10,20] escribir(A)</pre>	<p>Hay que implementar un bucle anidado. Para cada una de las filas de la matriz se imprimen los elementos de sus columnas.</p> <pre>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; }</pre>

3) Leer valores por teclado	
Pseudocódigo	Implementación en C++
<pre>leer(nombrevariable)</pre> <p>3.1) Leer un valor</p> <pre>x:entero y:real z:carácter leer(x) leer(y) leer(z) V:real[10] leer(V₃); A:entero[10,20] leer(A_{3,4})</pre>	<p>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.</p> <pre>cin >> nombrevariable;</pre> <pre>int x; float y; char z; cin >> x; cin >> y; cin >> z; float V[11]; cin >> V[3]; int A[11][21]; cin >> A[3][4];</pre>

<p>3.2) Leer un vector</p> <pre>V:real[10] leer(V)</pre> <p>3.3) Leer una matriz</p> <pre>A:entero[10,20] leer(A)</pre>	<p>Hay que implementar un bucle para leer cada uno de los elementos del vector.</p> <pre>float V[11]; int i; for (<u>i=1</u> ; <u>i<=10</u> ; i++) cin >> V[i];</pre> <p>Hay que leer elemento a elemento.</p> <pre>int A[11][21]; int i,j; for (<u>i=1</u> ; <u>i<=10</u> ; i++) for (<u>j=1</u> ; <u>j<=20</u> ; j++) cin >> A[i][j];</pre>
---	---

4) Instrucción condicional “si”	
Pseudocódigo	Implementación en C++
<pre>si a = b imprimir("Son iguales") si no si a < b imprimir("a menor que b") si no imprimir("a mayor que b") fsi</pre>	<pre>if (a == b) { cout << "Son iguales"; } else { if (a < b) cout << "a menor que b"; else cout << "a mayor que b"; }</pre> <p>En este ejemplo se realiza una instrucción tanto si la condición $a < b$ es verdadera o falta. Si hubiera más de una instrucción hay que utilizar llaves.</p>

5) Bucle “mientras”	
Pseudocódigo	Implementación en C++
<pre>mientras j<=m y A_{i,j}>0 hacer valor ← valor + A_{i,j} j ← j + 1 fmientras</pre>	<pre>while (j<=m && A[i][j]>0) { valor = valor + A[i][j]; j = j + 1; }</pre>

6) Bucle “para”	
Pseudocódigo	Implementación en C++
<pre>para i←1 hasta n hacer V_i ← 0 x ← x + i fpara</pre>	<pre>for (i=1 ; i<=n ; i++) { V[i] = 0; x = x + i; }</pre>

7) Definición de funciones

Pseudocódigo	Implementación en C++
<pre> función Fun1(a: entero, b: &real, c: carácter, V1: real[n1], V2: &real[n2], W1: entero[nfil1,ncol1], W2: &entero[nfil2,ncol2]):entero valor:entero ... a ← a + 1 b ← b / 2 V1_a ← a * b V2_a ← a + b W1_{a,a} ← a² W2_{a,a-1} ← a - W1_{a,a} ... devolver valor ffunción </pre>	<pre> int Fun1(int a, float &b, char c, float *V1, int n1, float *V2, int n2, int **W1, int nfil1, int ncol1, int **W2, int nfil2, int ncol2) { int valor; ... a = a + 1; b = b / 2; V1[a] = a * b; V2[a] = a + b; W1[a][a] = a * a; W2[a][a-1] = a - W1[a][a]; ... return valor; } </pre>

8) Llamada a funciones

Pseudocódigo	Implementación en C++
<p>Supongamos que queremos llamar a la función Fun1 del apartado anterior y tenemos definidas las siguientes variables</p> <pre> x:entero y:real z:carácter P1:real[n] P2:real[n] Q1:entero[fil1,col1] Q2:entero[fil2,col2] resultado:entero ... resultado ← Fun1(x,y,z,P1,P2,Q1,Q2) </pre>	<pre> int x; float y; char z; float *P1; int n; float *P2; float **Q1; int fil1, col1; float **Q2; int fil2, col2; int resultado; /* 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) */ ... resultado = Fun1(x,y,z, P1,n, P2,n, Q1,fil1,col1, Q2,fil2,col2); </pre>

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  $V_i = x$ 
            nveces ← nveces + 1
        fsi
    fpara

    devolver nveces
ffunción
```

Implementa un programa en C++ que pida al usuario el tamaño del vector (n) y elemento a buscar (x), genere tres vectores para los casos peor, mejor y cualquier otro, imprima 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 v_{peor} , v_{mejor} y $v_{cualquiera}$ del tamaño introducido por el usuario. Cada uno de estos vectores representará los casos peor, mejor y cualquier otro, respectivamente.

IMPORTANTE: Recuerda que hay que reservar memoria para $n+1$ elementos y que se utilizan las posiciones del vector de 1 hasta n (siendo n el tamaño).

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

- v_{peor} : todos los elementos iguales al valor x .
- v_{mejor} : todos los elementos distintos del valor x .
- $v_{cualquiera}$: 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 que se da, 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.