Estructuras de Datos y Algoritmos

CURSO 2009 – PRÁCTICO 1

SOLUCIÓN

1-
Observar las formas de pasar parámetros por referencia en cada uno de los dos ejemplos siguientes.

En c
(en C todos los parámetros se pasan por valor)

```
void promclase(float a[], int tam, float *prom) {
      float min, suma;
      suma=min=a[0];
      int i;
      for (i=1; i<tam; i++) {
            if (min>a[i]) {
                  min=a[i];
            }
            suma=suma + a[i];
      }
      (*prom)=(suma - min)/4;
}
```

Ejemplo de invocación:

```
int main(int argc, char *argv[]) {

      float b;
      float a[5]= { 10, 2, 4, 10, 8 };
      promclase(a,5,&b);

      printf("El promedio es %.2f \n", b);
      return 0;
}
```

En C/C++

```
void promclase(float a[],int tam ,float & prom) {

      float min, suma;
      suma=min=a[0];
      int i;
      for(i=1;i<tam;i++) {
            if(min>a[i]) {
                  min=a[i];
            }
            suma=suma + a[i];
      }
      prom=(suma - min)/4;
}
```

Ejemplo de invocación:

```
int main(int argc, char *argv[]) {

	float b;
	float a[5]= { 10, 2, 4, 10, 8 };
	promclase(a,5,b);

	printf("El promedio es %.2f \n", b);
	return 0;
}
```

3-

```
int esPrimo(int N) {
	int divisor, raiz;

	raiz = (int) sqrt(N);


	/*  N es un numero primo si sólo es divisible por N
	 o por la unidad. */
	for (divisor=2; N%divisor && divisor<=raiz; divisor++);


	/* Si se llego a dividir N entre todos los números menores
	 que su raíz cuadrada entonces es un primo */

	if (divisor == raiz+1)
		return 1;

	return 0;

	// otra forma
	//return (N % divisor) != 0;


}

void primos(int A, int B) {
	//PRE: B>=A
	for (int n=A; n<=B; n++) {
		if (esPrimo(n)) {
			printf("Es primo el valor : %d\n", n);
		}
	}
}
```

6-

```
// NOTA: ordena de forma descendente
void selectionSort(int A[], int largo) {
	int max, indiceMax;

	for (int i=0; i < largo-1; i++) {
		max = A[i];
		indiceMax = i;

		// buscamos el maximo
		for (int j=i+1; j < largo; j++) {
			if (A[j]> max) {
				max = A[j];
				indiceMax = j;
			}
		}
		// cambio de lugar
		A[indiceMax] = A[i];
		A[i] = max;
	}
}
```

A continuación, si bien no se pide en el ejercicio, se muestra como es el algoritmo de ordenación por inserción:

```
// NOTA: ordena en forma ascendente
void insertionSort(int numbers[], int array_size) {
	int i, j, index;
	for (i=1; i < array_size; i++) {
		index = numbers[i];
		j = i-1;
		while (j>=0 && numbers[j] > index) {
			numbers[j+1] = numbers[j];
			j--;
		}
		numbers[j+1] = index;
	}
}
```

7-

```
#define TAM_COL 3
#define TAM_FILA 3

void CAMBIO(int a[][TAM_COL], int M, int N) {
	int temp[TAM_COL];
	int i;
	for (i=0; i<TAM_COL; i++) {
		temp[i]=a[M][i];
		a[M][i]=a[N][i];
		a[N][i]=temp[i];
	}
}
```

8-

```c
#include <stdio.h>
#include <stdlib.h>

char* intToString(int num) {
	char* cadena=(char*)malloc(16*sizeof(char));
	sprintf(cadena, "%d", num);
	return cadena;
}

int stringToInteger(char* str) {
	int i = atoi(str);
	return i;
}
```

11-

```c
// funcion iterativa para fibonacci
int fibonacci_it(int n) {
	int f=0, f1=1, f2=1;
	if ((n==1) || (n==2))
		f=1;
	if (n==0)
		f=0;
	if (n>2) {
		for (int i=3; i<(n+1); i++) {
			f=f2+f1;
			f1=f2;
			f2=f;
		}
	}
	return (f);
}
```

12-

```c
// funcion iterativa para factorial
int factorial_it(int n) {
	int res = 1;

	for (int i=n; i>0; i--)
		res = res * i;
	return res;
}
```