Documentación

PROGRAMA#6: Paso por referencia y paso por valor.

Lenguajes de Programación, Grupo 04.

Abril 14, 2016.

Moreno Tagle Raphael Iván
Plata Martínez Jesus Alejandro
Rodríguez Bribiesca Isaac
Rodríguez García Alan Julián

Requerimientos

Windows, Mac OSX o cualquier otra plataforma con C y compilador GCC instalado

Objetivo

Desarrollar un programa que implemente lo siguiente:

1) Que reciba una lista con un número arbitrario de elementos y la devuelva invertida
2) Que reciba una lista de longitud arbitraria de puntos en el plano cartesiano y devuelva una lista de las distancias y los ángulos que existen entre cada uno de ellos

Pseudocódigos

invierteArreglo(arreglo[ ], tam)

desde i ← 1 hasta tam/2

temporal ← arreglo[i]

arreglo[i] ← arreglo [tam –i]

arreglo[tam – i] ← temporal

i ← i+1

distanciaAngulo(arreglo[ ][ ],tam)

resultados[ ][ ]

desde i ← 1 hasta tam

resultados[i][1] ← raíz((arreglo[i +1][1] – arreglo[i][1])^2 + (arreglo[i +1][2] – arreglo[i][2])^2)

resultados[i][2] ← angtan()

i ← i+1

Regresa resultados[][]

Modo de uso.

1. Abrir el programa e ingresar en primera instancia los valores del arreglo

2. Posteriormente se deben ingresar los puntos en el plano cartesiano que serán utilizados para calcular las distancia a la que se encuentra uno respecto de otro y el ángulo que forman entre ellos, para ello es necesario ingresar cada punto separando con una coma cada componente

    - Ejemplo:

      Si se desea ingresar el punto (4,9) sera suficiente con ingresar al programa 4,9

3. Una vez hecho lo anterior, el programa realiza las operaciones pertinentes y despliega en pantalla dos resultados

    - El primero de ellos corresponde con la lista invertida
    - El segundo corresponde a pares ordenados, cuyo primero elemento es la distancia entre los puntos ingresados, y el segundo el ángulo que forman entre ellos

    Cabe aclarar que se pueden meter tantos puntos como se haya seleccionado el tamaño del arreglo, la distancia y el ángulo se irán calculando tomando en cuenta pares de puntos

    - Ejemplo

      Si se ingresan los puntos {(0,0),(1,1),(2,1)}

      La salida del programa será: {(1.4142,45),(1,0)}

      Donde se puede observar que primero se tomaron los puntos (0,0) y (1,1) para el cálculo y después los puntos (1,1) y (2,1) para el siguiente

      De manera general a n puntos en el plano de entrada, se obtendrán n-1 vectores resultado

Pruebas

```
4 warnings generated.
[Ivans-MacBook-Pro:Programa6 Ivan$ ./PasoPar
Ingrese el tamano del arreglo -> 6

arreglo[0] -> 1
arreglo[1] -> 2
arreglo[2] -> 3
arreglo[3] -> 4
arreglo[4] -> 9
arreglo[5] -> 12


(x0, y0) -> 0, 0
(x1, y1) -> 1, 1
(x2, y2) -> 2, 1
(x3, y3) -> 3, 2
(x4, y4) -> 4, 8
(x5, y5) -> 1, 8

12
9
4
3
2
1

(1.4142, 45.0000)
(1.0000, 0.0000)
(1.4142, 45.0000)
(6.0828, 80.5377)
(3.0000, 180.0000)
```

Procesos en memoria.

```
void main(){
    int i, tam, *A;
    double **B;

    printf("Ingrese el tamano del arreglo -> ");
    scanf("%d", &tam);

    A = (int *)malloc(tam*sizeof(int));
    putchar('\n');
    for (i = 0; i < tam; ++i){
        printf("arreglo[%d] -> ", i);
        scanf("%d", A+i);
    }

    putchar('\n');
    B = (double **)malloc(tam*sizeof(double *));
    for (i=0; i<tam; i++)
        *(B + i) = (double *)malloc(2*sizeof(double));

    putchar('\n');
    for (i = 0; i < tam; ++i){
        printf("(x%d, y%d) -> ", i, i);
        scanf("%lf, %lf", *(B + i), *(B + i) + 1);
    }

    invierteLista(A, tam);
```

Se modifica y finaliza.

```
void invierteLista(int *arreglo, int tam){
    int i, aux;

    for(i = 0; i < (tam/2); i++){
        aux = *(arreglo + i);
        *(arreglo + i) = *(arreglo + (tam-1) - i);
        *(arreglo + (tam-1) - i) = aux;
    }
}
```

Se pasa la dirección de A

```c
void main(){
    int i, tam, *A;
    double **B;

    printf("Ingrese el tamano del arreglo -> ");
    scanf("%d", &tam);

    A = (int *)malloc(tam*sizeof(int));
    putchar('\n');
    for (i = 0; i < tam; ++i){
        printf("arreglo[%d] -> ", i);
        scanf("%d", A+i);
    }

    putchar('\n');
    B = (double **)malloc(tam*sizeof(double *));
    for (i=0; i<tam; i++)
        *(B + i) = (double *)malloc(2*sizeof(double));

    putchar('\n');
    for (i = 0; i < tam; ++i){
        printf("(x%d, y%d) -> ", i, i);
        scanf("%lf, %lf", *(B + i), *(B + i) + 1);
    }

    invierteLista(A, tam);

    putchar('\n');
    printf("Arreglo invertido: \n");
    for (i = 0; i < tam; ++i){
        printf("%d ", *(A+i));
    }
    printf("\n");

    B = distanciaAngulo(B, tam);

    putchar('\n');
    printf("Distancia entre angulos:\n");
    for (i = 0; i < tam-1; ++i){
        printf("(%.4lf, %.4lf)\n", *(*(B+i)), *(*(B+i) + 1));

    }
    free(A);
    free(B);
```

Se manda a llamar la función.

arreglo toma el valor de B

arreglo = B

```c
double **distanciaAngulo(double **arreglo, int tam){
    int i;
    double **copia = (double **)malloc((tam-1)*sizeof(double *));
    for (i=0; i<tam-1; i++)
        *(copia + i) = (double *)malloc(2*sizeof(double));

    for (i = 0; i < tam - 1; ++i){
        *(*(copia + i)) = sqrt(pow(abs(*(*(arreglo + i + 1))-*(*(arreglo + i))), 2) + pow(abs(*(*(arreglo + i + 1) + 1)
        *(*(copia + i) + 1) = atan2(*(*(arreglo + i + 1) + 1)-*(*(arreglo + i) + 1), *(*(arreglo + i + 1))-*(*(arreglo
    }

    return copia;
}
```

Al finalizar B toma el valor del arreglo “copia” de la función.