APRENDERAPROGRAMAR.COM

PASO DE PARÁMETROS O ARGUMENTOS EN C. POR DEFECTO, POR VALOR O POR REFERENCIA. SIZEOF MEMORIA (CU00550F)

Sección: Cursos
Categoría: Curso básico de programación en lenguaje C desde cero

Fecha revisión: 2031

Resumen: Entrega nº50 del curso básico "Programación C desde cero".

Autor: Mario Rodríguez Rancel

PASO POR VALOR EN C

En C la invocación de funciones implica que el paso de parámetros es, con carácter general y predefinido, por valor. De acuerdo con lo que indicamos cuando hablamos de pseudocódigo, esto impide que la variable transferida sea manipulada dentro de una función, ya que la función únicamente recibe un valor (no manipulable) copia del contenido de la variable.

Hay una excepción a este comportamiento: el paso de arrays como parámetros de función. Por motivos históricos, los arrays son transferidos por variable, de modo que cualquier manipulación del contenido del array que sea realizada dentro de la función implicará la modificación del array original pasado como parámetro.

Ejecuta este código y analiza los resultados para comprender lo expuesto:

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

int sumaUno (int numero) { numero = numero +1; return numero; }

int sumaUnoArray (int miArray[], int longitud) {
   int i=0; for(i=0; i<longitud; i++) {miArray[i] = miArray[i]+1;}
   return 1; //Ejemplos aprenderaprogramar.com
}

int main() {
   int n = 0;
   int i = 0;
   int num = 4;
   int unArray[3] = {2, 4, -3};
   n = sizeof(unArray)/sizeof(unArray[0]); //Numero de elementos en el array
   for(i=0; i<n; i++) {
            printf("Elemento %d del array antes: %d\n", i, unArray[i]);
   }
   printf("\n\nPasamos la variable num a la funcion\n");
   printf("La funcion devuelve: %d\n", sumaUno(num));
   printf("num no ha sido manipulada y sigue valiendo: %d\n\n", num);
   printf("\n\nPasamos el array unArray a la funcion\n");
   sumaUnoArray(unArray, n);
   for(i=0; i<n; i++) {
            printf("Elemento %d del array despues: %d\n", i, unArray[i]);
   }
   printf("unArray ha sido manipulado por la funcion\n\n");
   return 0;
}
```

El resultado esperado es el siguiente:

```
Elemento 0 del array antes: 2, Elemento 1 del array antes: 4, Elemento 2 del array antes: -3

Pasamos la variable num a la funcion, La funcion devuelve: 5, num no ha sido manipulada y sigue valiendo: 4

Pasamos el array unArray a la funcion

Elemento 0 del array despues: 3, Elemento 1 del array despues: 5, Elemento 2 del array despues: -2

unArray ha sido manipulado por la funcion
```

Comentarios: nos valemos del operador *sizeof*, que devuelve el espacio de memoria ocupado por una variable o tipo de dato (tamaño en bytes) para determinar el número de elementos (length o longitud) de que consta el array. En concreto usamos *sizeof(unArray)/sizeof(unArray[0])*, que equivale a plantear que si un elemento del array ocupa x bytes, n elementos del array ocuparán n*x. El número de elementos del array lo obtenemos como n*x/x.

Este operador no responde dentro de una función que recibe el array como parámetro debido a detalles del lenguaje en los que no vamos a entrar (en concreto, debido a que el paso como argumento de un array supone el paso de un puntero o referencia al array en lugar del array mismo). Por ello, para poder operar con la longitud del array dentro de la función, pasamos el dato de longitud como parámetro adicional en la llamada.

Aunque no vamos a profundizar en ello, con este ejemplo debemos comprender la diferencia entre paso por valor (el parámetro pasado no puede ser modificado por la función) y paso por variable (el parámetro pasado sí puede ser manipulado por la función).

Existen técnicas que permiten pasar (o simular el paso) de variables por referencia en C, pero no vamos a estudiarlas.

EJERCICIO RESUELTO Nº1: ENUNCIADO

Crear un programa en C que muestre un menú con dos opciones: 1. Cálculo y 2. Salir. Si se elige cálculo se ejecutarán 3 funciones: *entraDatos*, *proceso* y *resultados*. La función *entraDatos* debe pedir un entero entre 1 y 100 al usuario. La función *proceso* debe recibir el entero introducido por el usuario (a), calcular su raíz cuadrada (valores $\pm\sqrt{a}$) y calcular el sumatorio del tipo $\sqrt{a}$, $\sqrt{a-1}$, $\sqrt{a-2}$, ... , $\sqrt{0}$. Finalmente, la función *resultados* debe mostrar el dato base (introducido por el usuario), las raíces (*Raiz01* y *Raiz02*) y el valor de la suma de los términos de la sucesión.

EJERCICIO RESUELTO Nº1: SOLUCIÓN

```
//Programa SUC02 ejercicios resueltos aprenderaprogramar.com
#include <stdio.h>
#include <stdlib.h>
#include <math.h>

int dato=0;
double raiz01, raiz02;
double suce=0.0;
void entraDatos(); void proceso(int num); void resultados();

int main() {
   int opcionUsuario = 1;
   while (opcionUsuario!=2) {
      printf("\n\n1: Calculo\n2: Salir\n\nElija opcion: ");
      scanf("%d", &opcionUsuario);
      if (opcionUsuario==1) { entraDatos();   proceso(dato);   resultados();   }
   }
   return 0;
}

void entraDatos() {
   do {
      printf("\nPor favor introduzca numero entero entre 0 y 100: ");
      scanf("%d", &dato);
   } while (dato<0 || dato>100);
}

void proceso(int num) {
   suce=0.0; //Borrado dato valor suce
   raiz01=sqrt(dato);
   raiz02 = -raiz01;
   while(num>=0) {
      suce = suce + sqrt(num);
      num = num-1;
   }
}

void resultados() {
   printf("\nDato base: %d", dato);
   printf("\nRaiz01= %lf  Raiz02=%lf", raiz01, raiz02);
   printf("\nValor sucesion: %lf", suce);
}
```

Este podría ser un resultado de ejecución:

```
1: Calculo 2: Salir
Elija opcion: 1
Por favor introduzca numero entero entre 0 y 100: 85
Dato base: 85 Raiz01= 9.219544  Raiz02=-9.219544 Valor sucesion: 526.847258
Elija opcion: 1
Por favor introduzca numero entero entre 0 y 100: 1
Dato base: 1 Raiz01= 1.000000  Raiz02=-1.000000 Valor sucesion: 1.000000
Elija opcion: 1
Por favor introduzca numero entero entre 0 y 100: 7
Dato base: 7 Raiz01= 2.645751  Raiz02=-2.645751 Valor sucesion: 13.477573
Elija opcion: 2
```

El diseño – organización del código propuesto para el programa es uno de los posibles, no necesariamente el mejor. A modo de ejercicio, crea otro programa que tenga la siguiente organización del código: debe constar de tres funciones que serán la función *main*, la función *obtenerDatoEntrada* y otra función denominada *obtenerSuce*. En la función *main* se debe invocar a la función *obtenerDatoEntrada* que será una función que devuelva un tipo *int* con el dato de entrada introducido por el usuario. Fíjate que a diferencia del código anterior la función no tendrá tipo *void* ni trabajará con variables globales, además deberá incluir una sentencia *return*. El dato obtenido debe almacenarse en una variable y pasárselo a la función *obtenerSuce*. Esta función realizará el cálculo de la sucesión y devolverá su valor. Por tanto el tipo devuelto por la función será *double*. A diferencia del ejemplo anterior, esta función no trabajará con variables globales. Finalmente desde la propia función *main* se mostrarán los resultados y se dará opción a realizar otro cálculo o a salir del programa.

EJERCICIO Nº1

Refactoriza el código que hemos visto como ejercicio resuelto para que cumpla con estos requisitos:

a) La función entraDatos deberá tener como tipo de retorno int en lugar de void, de modo que devolverá el valor introducido por el usuario.

b) La función proceso ya no existirá y en su lugar tendremos dos funciones: calcularRaiz(int num) y calcularSumatorio (int num)

c) La función resultados ya no existirá y su cometido será realizado dentro del main.

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

EJERCICIO Nº 2

Estudia el siguiente código y responde a las preguntas:

```
#include <stdio.h>
#include <stdlib.h>
#include<conio.h>
// Ejercicios curso C aprenderaprogramar.com

//---------------------------------
void fun(int arr[]) {
int i;
for(i=0;i< 5;i++)
 arr[i] = arr[i] + 10;
}
//--------------------------------
void main() {
int arr[5],i;
clrscr();
printf("\nEnter the array elements : ");
for(i=0;i< 5;i++)
 scanf("%d",&arr[i]);

printf("\nPassing entire array .....");
fun(arr);  // Pass only name of array

for(i=0;i< 5;i++)
 printf("\nAfter Function call a[%d] : %d",i,arr[i]);

getch();
}
```

a) Busca información sobre conio.h, clrscr() y getch() ¿Forman parte del estándar de C?

b) Si el programa no te funciona, reescríbelo para que funcione.

c) ¿Qué objetivo tiene este programa? ¿Qué resultado debe devolver si se introducen los números 1, 2, 3, 4, 5?

d) ¿Es válido un bucle for sin llaves delimitadoras { ... } de las instrucciones a ejecutar?

Para comprobar si tus respuestas son correctas puedes consultar en los foros aprenderaprogramar.com.

Próxima entrega: CU00551F

Acceso al curso completo en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://www.aprenderaprogramar.com/index.php?option=com_content&view=category&id=82&Itemid=210