Electrónica Digital II

Santiago Rúa Pérez, PhD.

18 de septiembre de 2022

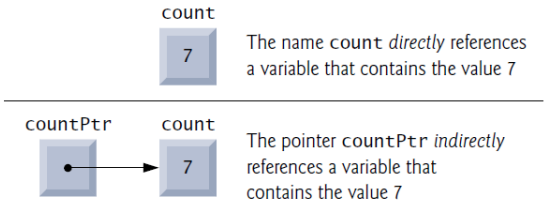
PUNTEROS EN C

Punteros en C - Objetivos

- Usar punteros y el operador puntero.
- Pasar argumentos de una función por referencia.
- Entender el identificador `const` y como puede afectar el comportamiento de una variable.
- Usar el operador `sizeof`.
- Usar el puntero para arreglos.

Punteros en C

En C, los punteros son variables cuyo valor son direcciones de memoria. Las variables especificamente apuntan a su valor.



La declaración se da con un asterisco.

```
1    int *countPtr;
2
```

El operador & sirve para asignar la dirección de una variable.

```
1    int y = 5;
2    int *yPtr;
3    yPtr = &y;
4
```

Punteros en C

El operador $*$ sirve para retornar el valor a donde apunta el puntero.

```
1   printf("\% d, *yPtr");
2
```


```
1   #include <stdio.h>
2
3   int main(void)
4   {
5     int a = 7;
6     int *aPtr = &a; // set aPtr to the address of a
7     printf("The address of a is % p"
8         "\nThe value of aPtr is % p", &a , aPtr);
9
10    printf("\n\nThe value of a is % d"
11        "\nThe value of *aPtr is % d", a, *aPtr);
12
13    printf("\n\nShowing that * and & are complements of "
14        "each other\n&*aPtr = % p"
15        "\n*&aPtr = % p\n", &*aPtr, *&aPtr);
16  }
17
```

Pasar argumentos a una función en C

En C se pueden usar punteros para pasar valor de una variable a una función. Cuando se realiza este tipo de operación se conoce como paso por referencia ya que se envia la dirección de memoria a modificar.

Paso por valor

```
1  #include <stdio.h>
2  int cubeByValue(int n); // prototype
3  int main(void)
4  {
5    int number = 5; // initialize number
6    printf("The original value of number is %d", number);
7
8    // pass number by value to cubeByValue
9    number = cubeByValue(number);
10   printf("\nThe new value of number is %d\n", number);
11 }
12
13 // calculate and return cube of integer argument
14 int cubeByValue(int n)
15 {
16   return n * n * n; // cube local variable n and return result
17 }
```

Pasar argumentos a una función en C

Paso por referencia

```
1  #include <stdio.h>
2  void cubeByReference(int *nPtr); // function prototype
3
4  int main(void)
5  {
6    int number = 5; // initialize number
7    printf("The original value of number is %d", number);
8
9    // pass address of number to cubeByReference
10   cubeByReference(&number);
11   printf("\nThe new value of number is %d\n", number);
12 }
13 // calculate cube of *nPtr; actually modifies number in main
14 void cubeByReference(int *nPtr)
15 {
16   *nPtr = *nPtr * *nPtr * *nPtr; // cube *nPtr
17 }
18
```

Pasar una cadena

```c
#include <stdio.h>
#include <ctype.h>
void convertToUppercase( char *sPtr ); // prototype

int main(void)
{
  char string[] = "cHaRaCters and $32.98"; // initialize char array

  printf("The string before conversion is: %s", string);
  convertToUppercase(string);
  printf("\nThe string after conversion is: %s\n", string);
}

// convert string to uppercase letters
void convertToUppercase( char *sPtr )
{
  while ( *sPtr != '\0' ) { // current character is not '\0'
    *sPtr = toupper(*sPtr); // convert to uppercase
    ++sPtr; // make sPtr point to the next character
  }
}

```

Usando el identificador const

El calificador const es usado para indicar al compilador que dicho valor no va a modificarse. Si una variable no cambia en el cuerpo de una función se debe declarar como constante.

```
1  #include <stdio.h>
2  void printCharacters( );
3
4  int main(void)
5  {
6    // initialize char array
7    char string[] = "print characters of
       a string";
8
9    puts("The string is:");
10   printCharacters(string);
11   puts("");
12 }
13
14 // sPtr is a "read-only" pointer
15 void printCharacters(const char *sPtr)
     {
16   // loop through entire string
17   for (; *sPtr != '\0'; ++sPtr) { //
       no initialization
18     printf("%c", *sPtr);
19   }
20 }
21
```

Tarea: implementar algoritmo de ordenamiento por burbuja utilizando paso por referencia.

Operador sizeof

Es un operador que sirve para determinar el tamaño de bytes de una variable.

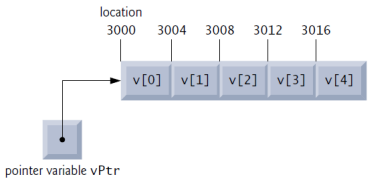
```
1  #include <stdio.h>
2  #define SIZE 20
3
4  size_t getSize(float *ptr); // prototype
5
6  int main(void)
7  {
8    float array[SIZE]; // create array
9
10   printf("The number of bytes in the array is %u"
11   "\nThe number of bytes returned by getSize is %u\n",
12   sizeof(array) , getSize(array) );
13 }
14 // return size of ptr
15 size_t getSize(float *ptr){
16   return sizeof(ptr);
17 }
18
```

Expresiones con punteros

Los punteros pueden ser usados para apuntar a los valores de un vector.

```
vPtr = v;
```

```
vPtr = &v[0];
```



Al operar con el puntero recuerde que esta operando sobre el valor de la direccion.

```
vPtr +=2;
```

El puntero por lo anterior estara posicionado en `v[2]`

Expresiones con punteros - Ejemplo

```c
#include <stdio.h>
#define ARRAY_SIZE 4

int main(void)
{
   int b[] = {10, 20, 30, 40};
   int *bPtr = b;

   // output array b using array index
      notation
   puts("Array b printed with:\nArray
      index notation");

   // loop through array b
   for (size_t i = 0; i < ARRAY_SIZE;
      ++i) {
      printf("b[%u] = %d\n", i, b[i]);
   }

   // output array b
   puts("\nPointer/offset notation
      where\n"
   "the pointer is the array name");
   for (size_t offset = 0; offset <
      ARRAY_SIZE; ++offset) {
      printf("*(b + %u) = %d\n", offset,
       *(b + offset));
   }

   // output array b using bPtr and
      array index notation
   puts("\nPointer index notation");
   for (size_t i = 0; i < ARRAY_SIZE;
      ++i) {
      printf("bPtr[%u] = %d\n", i, bPtr[
      i] );
   }

   // output array b
   puts("\nPointer/offset notation");
   for (size_t offset = 0; offset <
      ARRAY_SIZE; ++offset) {
      printf("*(bPtr + %u) = %d\n",
      offset, *(bPtr + offset));
   }
}
```

Encuentre el error

```
int *zPtr; // zPtr will reference array z
int *aPtr = NULL;
void *sPtr = NULL;
int number;
int z[5] = {1, 2, 3, 4, 5};
sPtr = z;
```

a)
```
++zptr;
```

b)
```
// use pointer to get first value of array; assume zPtr is initialized
number = zPtr;
```

c)
```
// assign array element 2 (the value 3) to number;
   assume zPtr is initialized
number = *zPtr[2];
```

d)
```
// print entire array z; assume zPtr is initialized
for (size_t i = 0; i <= 5; ++i) {
   printf("%d ", zPtr[i]);
}
```

e)
```
// assign the value pointed to by sPtr to number
number = *sPtr;
```

f)
```
++z;
```

Expresiones con punteros

Ejemplo: desarrolle un programa que realice operaciones elemento a elementos tales como exponencial.

Tarea: investigue como hacer punteros a funciones y haga un ejemplo.

Tarea: implementar un programa en C que baraje y entregue las cartas. Recuerde que una baraja tiene cuatro caras: espadas, corazones, treboles, y diamantes. Cada una de esas son 13 cartas.

PUNTEROS EN C

GRACIAS