



Paso de Vectores como Parámetros

Material Original: Prof. Flor Narciso
Modificaciones: Prof. Andrés Arcia
Departamento de Computación
Escuela de Ingeniería de Sistemas
Facultad de Ingeniería
Universidad de Los Andes

Paso de Vectores como Parámetros

Dado que un vector no tiene tamaño predefinido, pasar su contenido como un parámetro por valor es un costo innecesario que C/C++ no asume.

Un vector completo (a través de su referencia) se puede pasar a una función como parámetro.

Recordemos que la referencia a un vector o la dirección inicial, se especifica mediante su nombre, sin corchetes ni subíndices.

```
char  caracteres[80] = "esta cadena es constante";  
...  
foo(caracteres, longitud);  
...
```

El parámetro formal debe ser definido dentro de la función, se escribirá un par de corchetes vacíos, es decir, el tamaño del vector no se especifica.

```
void foo(char cadena_entrada[], int lng)
```

También puede especificarse un apuntador del tipo del vector:

```
void foo(char *cadena_entrada, int lng)
```

Paso de Vectores como Parámetros

En C/C++ los arreglos en general son pasados como parámetros por referencia. Esto es, el nombre del arreglo es la dirección del primer elemento del arreglo.

En C/C++ un elemento cualquiera de un arreglo puede ser pasado a una función por valor o por referencia, tal y como se hace con una variable simple.

Paso de Vectores como Parámetros: Ejemplo 1

```
float media(int a, float x[])    // Definición de la funcion
{                                // Note que se incluyen los corchetes
    // calculo de la media
    // Fijese que el tamaño del vector 'x' es pasado en 'a'
}

void main ()
{
    int n;
    float med;
    float lista[100];
    ...
    med = media(n, lista); // Esta llamada pasa como parámetros
                           // actuales la longitud del vector y
                           // el vector. Note que no se incluyen los
                           // corchetes
    ...
}
```

Aritmética de Punteros

Suponga que se tiene el siguiente arreglo de enteros:

```
int array[10];
```

3	34	6	87	95	34	56	43	35	22
0	1	2	3	4	5	6	7	8	9

La posición del primer elemento puede ser extraída como:

```
int * p_array = &array[0];
```

El n-esimo elemento se extrae como `array[n-1]`, pero también puede utilizarse la aritmética de punteros: `*(p_array + n - 1)`.

Fíjese que `p_array` almacena la dirección de memoria del primer elemento y como el resto de los elementos son consecutivos, puede hacerse una simple suma o resta para llegar al elemento de interés. Note también que las unidades sumadas son relativas al tamaño.

Aritmética de Punteros

Recuerde que los apuntadores de memoria se mueven por bloques. Entonces, cuando hacemos alguna operación aritmética con algún apuntador de memoria, estamos sumando un bloque que esta formado por m bytes.

Ej:

```
float arr_fl[10];
```

```
int tamano_float = sizeof(float); // el tamaño de un float son 4 bytes.
```

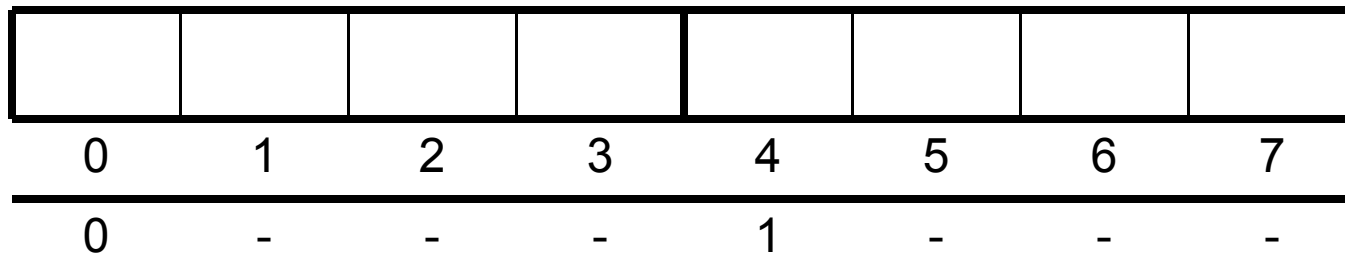
```
float * inicio_fl;
```

```
inicio_fl = arr_fl; // esta asignación también es valida. inicio_fl apunta al principio  
// del arreglo de flotantes.
```

```
*(inicio_fl + 5) = 11.2; // según esta asignación se esta modificando el 6to elemento  
// del arreglo.
```

Aritmética de Punteros

Suponga ahora que se desea hacer un recorrido por cada uno de los bytes que conforma a un arreglo.



Arriba tenemos un arreglo de 2 enteros cuya declaración puede ser:

```
int a[2];
```

En la primera línea numerada tenemos la dirección de cada byte del del arreglo, que en total tiene 8 bytes (2 elem * 4 bytes_de_tamaño). En la segunda línea numerada tenemos las posiciones relativas al tipo de dato.

Aritmetica de Punteros

Si queremos navegar por todo el arreglo, *byte por byte*, podemos hacerlos de la siguiente manera:

```
#define tam_arr 10  
  
char * p_char;  
  
float arr_fl[tam_arr];  
  
p_char = (char *)arr_fl;  
  
for (int i=0; i<tam_arr*sizeof(float); i++)  
    printf("%c",*(p_char + i));
```

La línea que separa los arreglos y, los punteros y sus operaciones se ha difuminado!

Paso de Vectores como Parámetros: Ejemplo 3

```
#include <ctype.h>
#include <string.h>

void F1(char cadena[], int lng) {
    int i;

    for (i=0; i!=lng; i++)
        cadena[i] = toupper(cadena[i]);
}

int main () {
    char caracteres[80] = "Ejemplo";
    int longitud = strlen(caracteres);

    puts(caracteres);
    F1(caracteres, longitud);
    puts(caracteres);
    return 0;
}
```

Entrada = Ejemplo
Salida = EJEMPLO

Paso de Vectores como Parámetros: Ejemplo 4

```
// Paso de vector y elementos individuales del vector como
// parámetros

#include <stdio.h>
#define nroEltos 5

void modificarVector(int [], int); // Prototipo de la funcion
void modificarValor(int);          // Prototipo de la funcion
void modificarReferencia(int &);   // Prototipo de la funcion

int main() {
    int vector[nroEltos] = {0, 1, 2, 3, 4}, j;

    printf("Los valores del vector original son:\n");
    for (j = 0; j < nroEltos; j++)
        printf("%i\n", vector[j]);
    modificarVector(vector, nroEltos);
    printf("Los valores del vector modificado son:\n");
    for (j = 0; j <= nroEltos - 1; j++)
        printf("%i\n", vector[j]);
}
```

Paso de Vectores como Parámetros: Ejemplo 4

```
printf("Efectos de pasar un elemento de un vector como
parámetro por valor\n");

modificarValor(vector[3]);

printf("El valor del cuarto elemento del vector es: %i\n",
      vector[3]);

modificarReferencia(vector[3]);

printf("El valor del cuarto elemento del vector es: %i",vector
[3]);
}
```

Paso de Vectores como Parámetros: Ejemplo 4

```
void modificarVector (int b[], int num) {  
    int k;  
    for (k = 0; k <= num - 1; k ++)  
        b[k] *= 2;  
}
```

```
void modificarValor (int e) {  
    e *= 2;  
    printf("Valor modificado del elemento = %i\n", e);  
}
```

```
void modificarReferencia (int &e) {  
    e *= 2;  
    printf("Valor modificado del elemento = %i\n", e);  
}
```

Paso de Vectores como Parámetros: Ejemplo 5

Diseñar un programa que calcule y escriba el área y la circunferencia de n círculos, tal que todos n radios se deben almacenar en el vector R .

```
#include <stdio.h>
#define n 100

void leerRadios(float radios[], int &numEltos) {
    // Parametros de salida: vector radios su numero de
    // elementos numEltos

    int j;

    printf("¿Cuántos radios seran introducidos ?\n");
    scanf("%i", &numEltos);
    for (j = 0; j < numEltos; j++) {
        printf("radio %i = ");
        scanf("%i", &radios[j]);
    }
}
```

Paso de Vectores como Parámetros: Ejemplo 5

```
void AreaCirc (float radio, float &area, float &cir)
// Parametros de entrada: radio
// Parametros de salida: area y cir
{
    float pi = 3.14159;
    area = pi * pow(radio, 2);
    cir = 2 * pi * radio;
}
```

```
void AreasCircs(float radios[], int numEltos, float
    areas[], float cirs[]) {
// Parametros de entrada: vector radios
// Parametros de salida: vectores areas y cirs

    int I;

    for (I = 0; I < numEltos; I++)
        AreaCirc(radios[I], areas[I], cirs[I]);
}
```

Paso de Vectores como Parámetros: Ejemplo 5

```
void escribirAreasCircs (float radios[], int numEltos, float
    areas[], float cirs[]) {
    int j;

    for (j = 0; j < numEltos; j++) {
        printf("Para la circunferencia de radio %i", radios[j]);
        printf("su area es: %i\n", areas[j]);
        printf("su circunferencia es: %i\n", cirs[j]);
    }
}

int main() {
    float R[n], A[n], C[n];
    int nRadios;

    leerRadios(R, nRadios);
    AreasCircs(R, nRadios, A, C);
    escribirAreasCircs (R, nRadios, A, C);
    return 0;
}
```

Ejercicios

Determinar la salida del siguiente programa si el dato de entrada es el entero 4.

```
#include <iostream.h>

int a (int y)
{
    y *= 2;
    return y;
}

int b (int x)
{
    x += 5;
    return x;
}
```

```
void main ()
{
    int x, x1;
    cout << "Introduzca un
numero" << endl;
    cin >> x;
    x1 = b(x);
    cout << x1 << endl;
    x1 = a(x);
    cout << x1 << endl;
}
```


Ejercicios

Determinar la salida del siguiente programa:

```
#include <iostream.h>

int reloj, num, temp;

void pascal (int num, int
&suma) {
    int reloj;
    temp = 2 * num + suma;
    reloj = temp * suma;
    num++;
    suma += num;
    cout << temp << reloj
<< num << suma << endl;
}
```

```
void main ()
{
    num = 1;
    reloj = 0;
    temp = 0;
    pascal (num, temp);
    cout << num << reloj
<< temp << endl;
    temp++;
    pascal (temp, num);
    cout << num << reloj
<< temp;
}
```

Ejercicios

Escribir una función llamada *digfinal* que reciba como parámetro un número entero positivo. La función dará como resultado el último dígito de su parámetro. Por ejemplo *digfinal* (2456) debe dar como resultado 6.

Escribir una función lógica llamada *multiplo* que tenga dos parámetros enteros m y n. La función puede producir el valor cierto si m es un múltiplo entero de n ó n es un múltiplo entero de m. Escribir un programa para probar esta función que lea cada pareja de enteros de una lista de n parejas y escriba cada pareja y el resultado.

Ejemplo de entrada

4	7
4	8
16	8

Ejemplo de salida

4	7	falso
4	8	cierto
16	8	cierto

Ejercicios

Escribir un procedimiento con tres parámetros formales reales (por referencia), a, b y c. El efecto del procedimiento debe ser girar los valores de los parámetros hacia la derecha de manera que, después de la ejecución, el valor que originalmente estaba en a quede en b, el que estaba en b quede en c y el que estaba en c quede en a. Pruebe el procedimiento con un programa que lea tres números reales de una lista de n triplas de números reales y exhiba cada tripla después de la rotación.

Ejemplo de entrada:

4.7	1.0003	7.5
-12.5	6.5	2.005

Ejemplo de salida

7.5	4.7	1.0003
2.005	-12.5	6.5

Ejercicios

Los registros mensuales de visitantes al parque Chorros de Milla durante el año 2000 son los siguientes:

Mes	Ene	Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov	Dic
#visitantes	100	90	120	150	210	250	300	310	280	230	160	120

Escribir un programa, estructurado en funciones, para realizar las siguientes tareas:

- Almacenar los datos en un vector y los nombres de los meses en otro vector.

- Calcular y escribir el promedio de visitantes durante el año 2000.

- Calcular y escribir los nombres de los meses con el mayor y el menor número de visitantes.