



Programación Avanzada



Pontificia Universidad
JAVERIANA
Colombia

Ing. Orlando Vasquez

Apuntadores

Concepto de apuntador: Un apuntador es una variable donde se puede asignar una dirección de memoria. Si se define una variable cualquiera el Lenguaje C++ localiza esa variable en la memoria del computador.

Ej: `int x;`
`char c;`

La variable x y c quedan en algún lugar desconocido de la memoria.

Utilizando un apuntador podemos almacenar en la dirección de memoria en la cual se encuentra la variable x o c.

Definición de un apuntador: Para definir un apuntador es necesario determinar inicialmente a que tipo de variable va a apuntar; si x es de tipo int, entonces el apuntador se debe definir como un apuntador a un tipo int de igual forma si c es de tipo char, entonces el apuntador se debe definir como un apuntador a char

Sintaxis:

Ej: Tipo de dato a apuntar * nombre_apuntador;

`int *pint;`

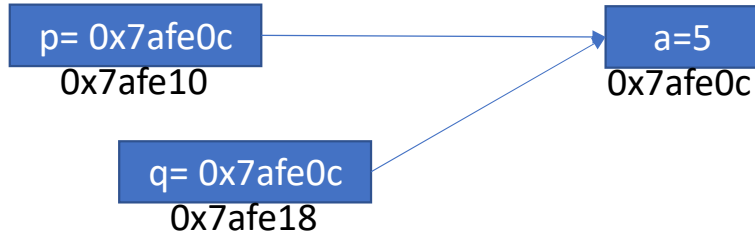
Apuntadores

Uso de un apuntador: La declaración de una variable apuntador no implica que quede apuntando a algún sitio. Para esto es necesario asignarle una dirección de memoria de alguna otra variable, esto se hace utilizando el carácter `&`.

Ej:

```
int a;  
int *p;  
int *q;  
a=5;  
p=&a;  
q=p;
```

Resultado:



Apuntadores

Uso de un apuntador: Para modificar o mostrar la información de la variable a la cual referencia un apuntador se utiliza *

Ejemplo: Siguiendo con el caso anterior

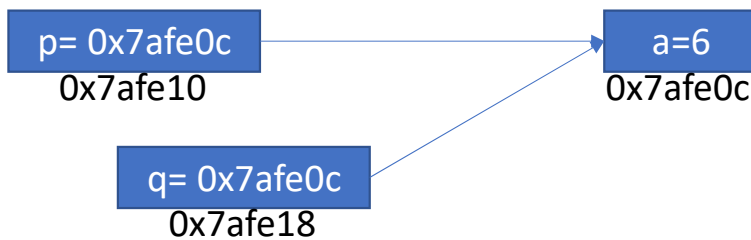
```
cout <<"*p:"<< *p; //Muestra por pantalla el valor 5, que es el valor de la dirección de memoria almacenado en p
cout <<"*q:"<< *q; //Muestra por pantalla el valor 5, que es el valor de la dirección de memoria almacenado en q
```

Si asignamos un valor a *p ó *q, modificamos automáticamente el valor de **a**

Ej:

```
*p=6;
```

Resultado



Resumen:

- Si **px** es un apuntador, entonces *px es el contenido del apuntador (el valor almacenado en la dirección de memoria)
- Si **x** es una variable **&x**, es la dirección de memoria donde esta almacenada la variable



Apuntadores1.cpp

Apuntadores

Errores comunes usando apuntadores:

1) Apuntador sin inicializar

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main(){
    int *z;
    cout << "z: " << z;
    *z=2;
    getchar();
}
```

D:\Orlando Vasquez\Javeri

z: 0x10

```
#include <stdio.h>
#include <iostream>
using namespace std;
int main(){
    int *z;
    cout << "z: " << z;
    *z=2;
    getchar();
}
```

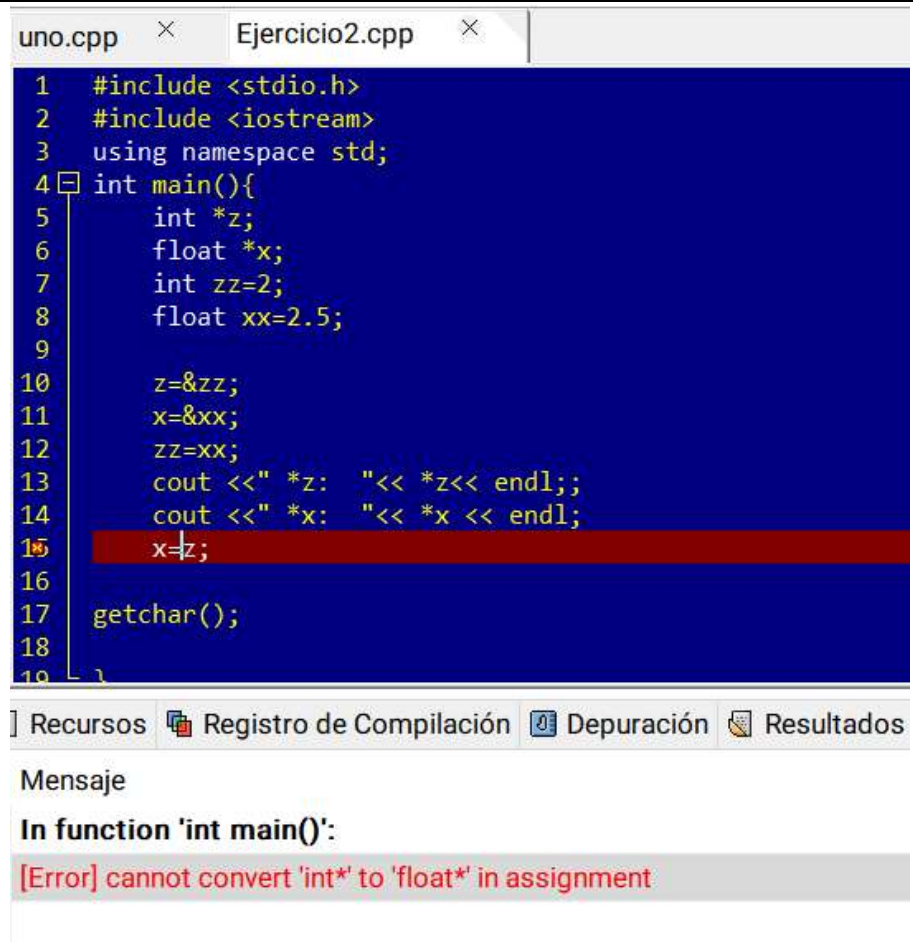
Excepción no controlada

Excepción no controlada en 0x0000000000401593 en Ejercicio2.exe:
0xC0000005: Infracción de acceso al escribir en la ubicación
0x0000000000000010.

Apuntadores

Errores comunes usando apuntadores:

2)Asignación incorrecta de tipos de datos



```
1  #include <stdio.h>
2  #include <iostream>
3  using namespace std;
4  int main(){
5      int *z;
6      float *x;
7      int zz=2;
8      float xx=2.5;
9
10     z=&zz;
11     x=&xx;
12     zz=xx;
13     cout <<" *z: " << *z<< endl;;
14     cout <<" *x: " << *x << endl;
15     x=z;
16
17     getchar();
18
19 }
```

Recursos Registro de Compilación Depuración Resultados

Mensaje

In function 'int main()':

[Error] cannot convert 'int*' to 'float*' in assignment

Apuntadores

Errores comunes usando apuntadores:

3) Asignación de valores de variables a un apuntador en vez de una dirección de memoria

```
1  #include <iostream>
2  using namespace std;
3  int main()
4  {
5      int x=5;
6      int *ptr;
7      ptr=x;
8      printf ("*ptr: %d",*ptr);
9
10
11 }
```

Compilador (2) Recursos Registro de Compilación Depuración Resultados Console Cerrar

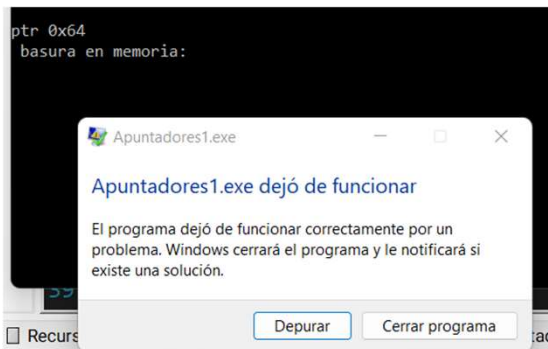
Lín...	Col.	Archivo	Mensaje
		D:\Orlando Vasquez\Javeriana\Programaci...	In function 'int main()':
6	6	D:\Orlando Vasquez\Javeriana\Programacion Av...	[Error] invalid conversion from 'int' to 'int*' [-fpermissive]

Apuntadores

Errores comunes usando apuntadores:

4) No inicializar la variable apuntador, nos lleva a trabajar con la *basura* que haya en la memoria.

```
#include <iostream>
using namespace std;
int main(){
    int x=5;
    int *ptr;
    cout <<"ptr "<<ptr<<" basura en memoria: "<<*ptr<<endl;
    *ptr=*ptr+x; /*Aquí estoy trabajando con la basura en memoria*/
    cout<< "*ptr: "<<*ptr<<endl;
}
```



Aritmética de Apuntadores

Hasta este punto tenemos claro que un apuntador es un tipo de variable que almacena una dirección de memoria en la cual por lo general se encuentra almacenada información de una variable de nuestro programa. En C++ podemos sumar o restar números enteros a una variable apuntador.

```
Ej: int x=5;  
    int *ptr=&x;  
    ptr=ptr+1; /*es igual que ptr++*/
```

Este tipo de operaciones no suma o resta N al apuntador, sino que lo desplaza N cantidades de bytes dependiendo del tipo de dato.

Recordemos

string: 32 bytes
char: 1 byte
int: 4 bytes
float: 4 bytes
double: 8 bytes

Aritmética de Apuntadores

Ejemplos

```
string s="s";  
string *ptrs=&s;
```

ptrs:0x7afdb0

ptrs=ptrs+2; /*Esto es equivalente a sumar 64 bytes a la dirección de memoria original*/

64 bytes en decimal corresponde al numero **40 en hexadecimal**

Operación aritmética en memoria: **0x7afdb0+40=0x7afdf0**

ptrs:0x7afdf0

***ptrs: ?? /*Lo que hay en la dirección de ptrs, es un valor desconocido*/**

Como se ve esto en la memoria

0x7afdb0	(0x7afdd0+32bytes)=0x7afdf0
s	??

<https://calcuonline.com/calculadoras/calculadora-hexadecimal/>

<https://cual-es-mi-ip.online/herramientas/conversores-numericos/conversor-decimal-a-hexadecimal/>

Aritmética de Apuntadores

Ejemplos

```
int i=2;
int *ptri=&i;
ptri:0x7afe0c
ptri=ptri+3; /*Esto es equivalente a sumar 12 bytes a la dirección de memoria original*/
12 bytes en decimal corresponde al numero C en hexadecimal
Operación aritmética en memoria: 0x7afe0c+C= 0x7afe18
ptri:0x7afe18
*ptri:8060440 /*Lo que hay en la dirección de ptri, es un valor desconocido*/
```

Como se ve esto en la memoria

0x7afe0c	(0x7afe0c+12bytes)=0x7afe18
2	8060440 /*Vr. Desconocido*/



Aritmetica1.cpp

<https://calcuonline.com/calculadoras/calculadora-hexadecimal/>

<https://cual-es-mi-ip.online/herramientas/conversores-numericos/conversor-decimal-a-hexadecimal/>

Aritmética de Apuntadores

Nota: Las variables tipo arreglo por defecto son variables tipo apuntador. **Es decir para obtener la dirección de memoria de un arreglo a un apuntador no hay que colocar el &**

Ejemplo en arreglos: **Recordemos:** Las posiciones en memoria de un arreglo, son posiciones consecutivas

```
double datos[3]={20,30,40}
```

```
double *ptr=datos /*En este caso ptr queda apuntando a la dirección de la primera posición del arreglo/
```

```
-----
datos[0]: 20
Valor *ptr: 20
datos: 0x7afe00
Direccion ptr : 0x7afe00
```

0x7afe00	20
----------	----

```
-----
datos[1]: 30
ptr++;
Valor *ptr: 30
Direccion ptr: 0x7afe08
```

0x7afe00	20
0x7afe00+8bytes= 0x7afe08	30

```
-----
datos[2]: 40
ptr++;
Valor *ptr: 40
Direccion ptr: 0x7afe10
```

0x7afe00	20
0x7afe00+8bytes= 0x7afe08	30
0x7afe08+8bytes = 0x7afe10	40

Aritmética de Apuntadores

Recorrer Cadena

Aritmetica 1

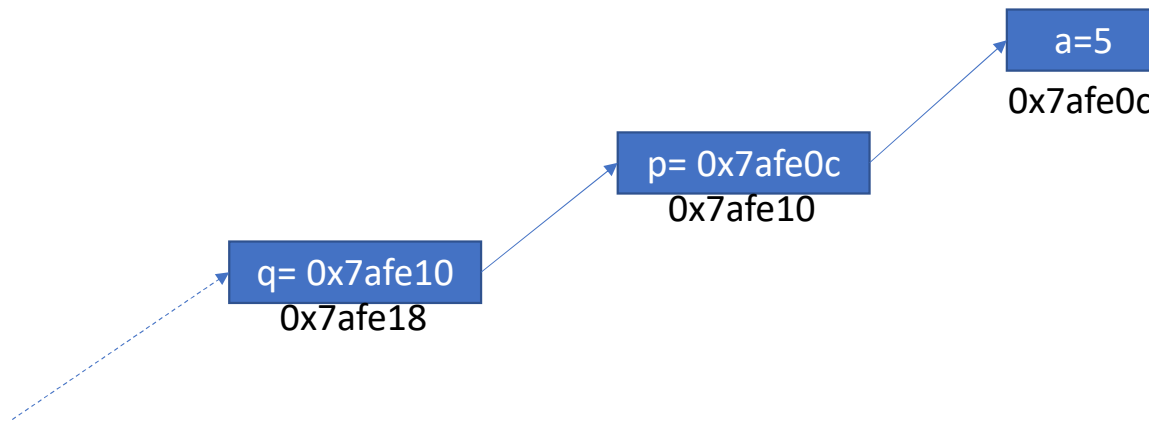
Aritmetica 2

<https://calcuonline.com/calculadoras/calculadora-hexadecimal/>

<https://cual-es-mi-ip.online/herramientas/conversores-numericos/conversor-decimal-a-hexadecimal/>

Apuntador a apuntador

Gráficamente un apuntador a un apuntador se puede representar así:



Así como un apuntador a una variable, almacena la dirección de memoria de dicha variable para poder manipular su valor, un apuntador a apuntador, es un apuntador que permite referenciar otra dirección de memoria.

Apuntador a apuntador

Apuntador_Apuntador 1

Apuntador Apuntador 2

Manejo de apuntadores en funciones

Paso de parámetros por valor: El paso de parámetros por valor significa que **a la función que recibe los parámetros se le pasa una copia** del valor que contiene el parámetro enviado.

Los valores de los parámetros de la llamada se copian en los parámetros de la cabecera de la función. **La función trabaja con una copia de los valores** por lo que cualquier modificación en estos valores no afecta al valor de las variables utilizadas en la llamada.

Aunque los parámetros (los que aparecen en la llamada a la función) y los parámetros definidos como entrada de la función tengan el mismo nombre **son variables distintas** por ende ocupan direcciones de memoria distinta

Parámetros por Valor 1

Parámetros por Valor 2

Manejo de apuntadores en funciones

Paso de parámetros por referencia: El paso de parámetros por referencia significa que **a la función que recibe los parámetros se le pasa la dirección de memoria de la variable** que contiene el parámetro enviado.

Las direcciones memoria de los parámetros de la llamada se copian en los parámetros de la cabecera de la función. **La función trabaja con las direcciones de memoria** por lo que cualquier modificación en estos valores afecta al valor de las variables utilizadas en la llamada.

Parámetros por Referencia 1

Parámetros por Referencia 2

Parámetros por Referencia 3

Manejo de arreglos en funciones

Paso de arreglos a funciones : Un arreglo puede ser pasado como parametro a una función, pero se debe tener en cuenta que los arreglos siempre se pasan por referencia, dado que la declaración de un arreglo es un apuntador en sí. El paso de un arreglo a una función se define de la siguiente manera:

```
tipo_retorno nombre_funcion (nombre_arreglo[]);  
tipo_retorno nombre_funcion (*nombre_arreglo);  
tipo_retorno nombre_funcion (nombre_arreglo[]...[]);
```

Arreglos en funciones 1

Arreglos en funciones 2

Arreglos en funciones 3

Ejercicios de ejemplo

Archivos como parámetros

Comparar Cadenas

Concatenar Cadenas

Convertir Cadena a Numero