FUNDAMENTOS DE PROGRAMACIÓN

Inicio Clase 12

Profesor: Carlos Díaz

Clase 11: Punteros (Parte 2)

- Puntero a puntero (puntero doble)
- Punteros de cadenas
- Aritmética de punteros
- Puntero en los arrays de dos dimensiones
- Paso de puntero a función (ejemplo 4)

Puntero a puntero

- Un puntero puede apuntar a otra variable puntero.
- Para declarar un puntero a un puntero se hace preceder a la variable con dos asteriscos (**).
- En el siguiente código, ptr2 es un puntero a un puntero.

```
#include <iostream>
using namespace std;
int main(){
        int valor = 100;
        int *ptr1 = &valor;
        int **ptr2 = &ptr1;
        cout<<"Puntero a entero"<<endl;
        cout << "Valor: " <<valor<<endl;
        cout << "&valor: " <<&valor<<endl;
        cout << "ptr1: " <<ptr1<<endl;
        cout << "*ptr1: " <<*ptr1<<endl;
        cout<<"Puntero de puntero"<<endl;
        cout << "&ptr1: " <<&ptr1<<endl;
        cout << "ptr2: " <<ptr2<<endl;
        cout << "*ptr2: " <<*ptr2<<endl;
        cout << "**ptr2: " <<**ptr2<<endl;
}
```

```
Puntero a entero
Valor: 100
&valor: 0x6ffe34
ptr1: 0x6ffe34
*ptr1: 100
Puntero de puntero
&ptr1: 0x6ffe28
ptr2: 0x6ffe28
*ptr2: 0x6ffe34
**ptr2: 100
```

Punteros de cadenas

- Considérese la siguiente declaración de un array de caracteres que contiene las veintiséis letras del alfabeto internacional.

```
char alfabeto[27] = "abcdefghijklmnopqrstuvwxyz";
```

- Si p es un puntero a char. Se establece que p apunta al primer carácter de alfabeto escribiendo

```
char *p;

p = alfabeto; // o bien p = &alfabeto[0];
```

- Es posible, entonces, considerar dos tipos de definiciones de cadena

```
char cadena[]="Las continentes"; //array contiene una cadena

char *pCadena = "conocidos son 5:"; //puntero a cadena
```

- También es posible declarar un array de cadenas de caracteres:

```
char* Continentes[4] ={"America", "Europa", "Asia", " Africa", "Oceania"};

// array de punteros a cadena
```

Aritmética de punteros

- A un puntero se le puede sumar o restar un entero n; esto hace que apunte n posiciones adelante, o atrás de la actual.
- A una variable puntero se le puede aplicar el operador ++, o el operador --. Esta operación hace que el operador contenga la dirección del siguiente, o anterior elemento.
- Se pueden sumar o restar una constante puntero a o desde un puntero y sumar o restar un entero. Sin embargo, no tiene sentido sumar o restar una constante de coma flotante.
- Operaciones no válidas con punteros: no se pueden sumar dos punteros; no se pueden multiplicar dos punteros; no se pueden dividir dos punteros.

Ejemplo 1

- Modificación de una cadena con un puntero.
- El programa lee una cadena de caracteres, y mediante una variable puntero, inicializada a la primera posición del array de caracteres, se van cambiando las letras mayúsculas por minúsculas y recíprocamente.
- El bucle while itera hasta que se llegue al final de la cadena de caracteres.
- La sentencia *puntero = *puntero-32. Asigna al contenido del puntero el contenido del puntero menos el número ASCII 32 para que el carácter pase a letra minúscula.
- Posteriormente, el puntero avanza una posición (un byte por ser de tipo char).

```
#include <iostream>
using namespace std;
int main(){
	char *puntero;
	char Cadena[81];
	cout << "Introduzca cadena a convertir:\n";
	cin.getline(Cadena, 80);
	puntero = Cadena; // puntero apunta al primer carácter de la cadena
	while (*puntero){ // mientras puntero no apunte a \0
	if ((*puntero >= 'A') && (*puntero <= 'Z'))
		*puntero = *puntero+32; // sumar 32, para convertir en minúscula
	else if ((*puntero >= 'a') && (*puntero <= 'z'))
		*puntero = *puntero-32; // restar 32, para convertir en mayúscula
	else
		*puntero=*puntero;
	puntero++;
	}
	cout << "La cadena convertida es: " << endl;
	cout << Cadena << endl;
}
```

Puntero en los arrays de dos dimensiones

- Para apuntar a un array bidimensional como tal, o lo que es lo mismo, para apuntar a su inicio, el compilador de C++ considera que un array bidimensional es en realidad un array de punteros a los arrays que forman sus filas.
- Por tanto, será necesario un puntero doble o puntero a puntero, que contendrá la dirección del primer puntero del array de punteros a cada una de las filas del array bidimensional o matriz.
- Si a se ha definido como un array bidimensional, el nombre del array a es un puntero constante que apunta a la primera fila a[0]. El puntero a+1 apunta a la segunda fila a[1], etc. A su vez a[0] es un puntero que apunta al primer elemento de la fila 0 que es a[0][0]. El puntero a[1] es un puntero que apunta al primer elemento de la fila 1 que es a[1][0], etc.

Ejemplo 2

- Dada la declaración float A[5][3] que define un array bidimensional de cinco filas y tres columnas, se tiene la siguiente estructura:

Puntero a puntero fila		Puntero a fila		ARRAY BIDIMENSIONAL float A[4][3]		
A	→	A[0]	→	A[0][0]	A[0][1]	A[0][2]
A+1	→	A[1]	→	A[1][0]	A[1][1]	A[1][2]
A+2	→	A[2]	→	A[2][0]	A[2][1]	A[2][2]
A+3	→	A[3]	→	A[3][0]	A[3][1]	A[3][2]
A+4	→	A[5]	→	A[4][0]	A[4][1]	A[4][2]

- A es un puntero que apunta a un array de 5 punteros A[0], A[1], A[2], A[3], A[4].
- A[0] es un puntero que apunta a un array de 3 elementos A[0][0], A[0][1], A[0][2].
- A[1] es un puntero que apunta a un array de 3 elementos A[1][0], A[1][1], A[1][2].
- A[2] es un puntero que apunta a un array de 3 elementos A[2][0], A[2][1], A[2][2].
- A[3] es un puntero que apunta a un array de 3 elementos A[3][0], A[3][1], A[3][2].
- A[4] es un puntero que apunta a un array de 3 elementos A[4][0], A[4][1], A[4][2].

Ejemplo 2 (continuación)

- A[i][j] es equivalente a las siguientes expresiones:

*(A[i]+j) el contenido del puntero a la fila i más el número de columna.

(((A+i))+j). Si se cambia A[i] por *(A+i) se tiene la siguiente expresión anterior.

*(&A[0][0]+ 3*i+j).

- A es un puntero que apunta a A[0].
- A[0] es un puntero que apunta a A[0][0].
- Si A[0][0] se encuentra en la dirección de memoria 100 y teniendo en cuenta que un float ocupa 4 bytes, la siguiente tabla muestra un esquema de la memoria:

Contenido de puntero a puntero fila	Contenido de puntero a fila	Direciones del array bidimensional float A[4][3]		
*A = A[0]	A[0] = 100	&A[0][0] = 100	&A[0][1] = 104	&A[0][2] = 108
*(A+1) = A[1]	A[1] =112	&A[1][0] = 112	&A[1][1] = 116	&A[1][2] = 120
*(A+2) = A[2]	A[2] =124	&A[2][0] = 124	&A[2][1] = 128	&A[2][2] = 132
*(A+3) = A[3]	A[3] =136	&A[3][0] = 136	&A[3][1] = 140	&A[3][2] = 144
*(A +4) = A[4]	A[5] =148	&A[4][0] = 148	&A[4][1] = 152	&A[4][2] = 156

Ejemplo 3

- Direcciones ocupadas por punteros asociados a una matriz.
- El programa muestra las direcciones ocupadas por todos los elementos de una matriz de reales dobles de 5 filas y 4 columnas, así como las direcciones de los primeros elementos de cada una de las filas, accedidos por un puntero a fila.

```
#include <iostream>
using namespace std;
int main(){
        double A[5][4];
        int i,j;
        cout <<" direciones de todos lo elementos de la matriz\n";
        for (i = 0; i < 5; i++)  {
                for (j = 0; j < 4; j++)
                        cout << " &A[" << i << "][" << j << "]=" << &A[i][j];
        cout << "\n";
        }
        cout << " direcciones de comienzo de las filas de la matriz\n";
        for (i = 0; i < 5; i++)
        cout << " A["<<i<<"] = " << A[i]
        << " contiene direccion de &A[" << i << "][" << 0 << "]" << endl;
}
```

Ejemplo 4

- Lectura y escritura de matrices mediante punteros.
- Escribir un programa que lea y escriba matrices genéricas mediante punteros y funciones.
- Para poder tratar la lectura y escritura de matrices mediante funciones que reciban punteros como parámetros, basta con trasmitir un puntero a puntero.
- Además, se debe informar a cada una de las funciones el número de columnas y filas que tiene (aunque sólo es necesario el número de columnas), por lo que las funciones pueden ser declaradas de la siguiente forma:

`void escribir_matriz(int ** A, int f, int c)` y `void leer_matriz(int ** A, int f, int c)`

donde f y c son respectivamente el número de filas y el número de columnas de la matriz.

- Para tratar posteriormente la lectura y escritura de datos en cada una de las funciones hay que usar *(*A + c * i + j).
- Las llamadas a ambas funciones, deben ser con un tipo de dato compatible tal y como se hace en el programa principal.
- En el programa, además se declaran el número de filas F, y el número de columnas C como macros constantes.

```
using namespace std;
#define F 3
#define C 2
int A[F][C];
void escribir_matriz(int ** A, int f, int c){
	int i, j;
	for (i = 0; i < f; i++)
	{
		for(j = 0; j < c ; j++)
		cout << " " << *(*A + c*i+j);
	cout << endl;
	}
}
void leer_matriz(int** A, int f, int c){
	int i, j;
	cout<<"ingrese una matriz de 3 x 2\n";
	for (i = 0; i < f; i++)
		for(j = 0; j < c; j++)
			cin >> *(*A + c*i+j);
}
int main(){
	int * a = &A[0][0];
	leer_matriz(&a,F,C);
	escribir_matriz(&a,F,C);
}
```

FUNDAMENTOS DE PROGRAMACIÓN

Fin Clase 12

Profesor: Carlos Díaz