Introducción a la programación orientada a objetos (IPOO)

Memoria dinámica: Punteros y Arreglos dinámicos

carlos.andres.delgado@correounivalle.edu.co

Carlos Andrés Delgado S.

Facultad de Ingeniería. Universidad del Valle

Abril de 2017

Contenido

Contenido

Memoria dinámica

Introducción

- Cada vez que se ejecuta un programa, se carga primero en la memoria del computador
- En consecuencia, todos los elementos del programa también se cargan en memoria
- La memoria está organizada en series secuenciales de espacios de memoria
- Cada espacio se puede distinguir de los demás por su dirección.

Memoria dinámica

Introducción

Hasta el momento hemos manejado memoria estática, por ejemplo en la siguiente declaración:

```
int funcion(int a[]){
    int suma = 0;
    for(int i=0; i<5; i++){
        suma+=a[i];
    }
    return suma;
}

int main(){
    int arreglo[] = {2,4,6,8,10};
    int resultado = funcion(arreglo);
}
```

Memoria dinámica

Introducción

El anterior código en memoria podemos verlo así:

Memoria dinámica

Introducción

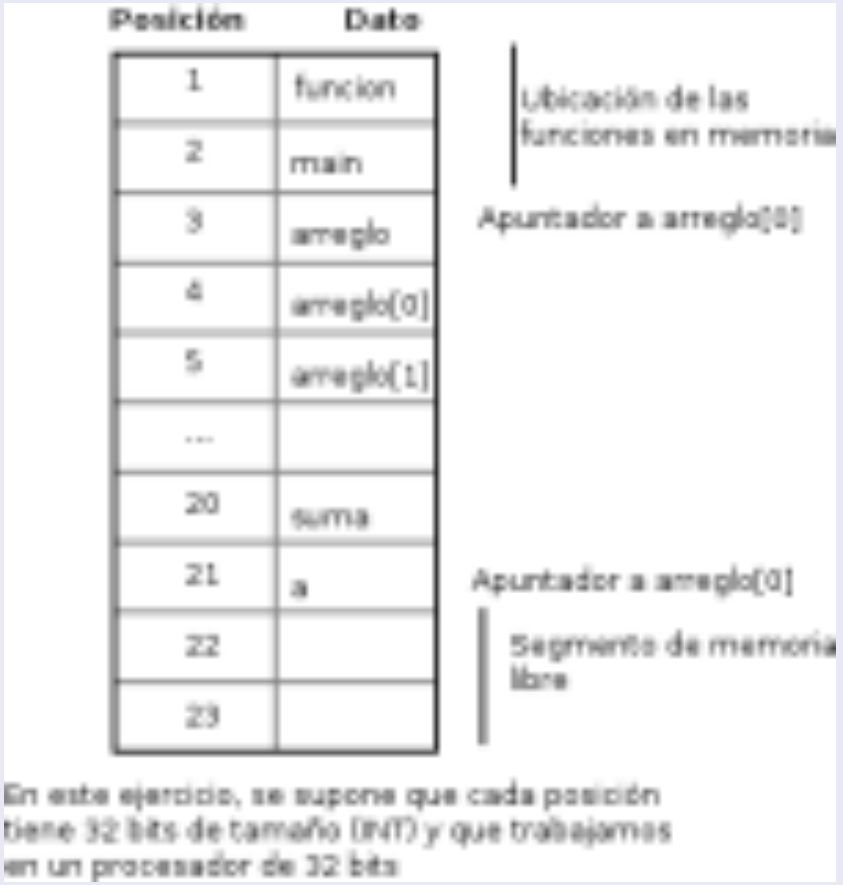
Para optimizar la memoria, usaremos referencias y punteros.

```
//Recibimos la referencia cómo puntero
//El puntero en este punto es tratado como referencia
    hacia un arreglo
int funcion(int * a){
    int suma = 0;
    for(int i=0; i<5; i++){
        suma+=a[i];
    }
    return suma;
}

int main(){
    int arreglo[] = {2,4,6,8,10};
    //Enviamos dirección del arreglo
    //De por sí, arreglo es una referencia hacia arreglo
        [0]
    int resultado = funcion(arreglo);
}
```

Introducción

Para el anterior código, la memoria se vería así:



Memoria dinámica

Definición

El ámbito de existencia de una variable puede ser:

1. **Estático:** Se crea en el momento de definirse y se destruyen al final del bloque }
2. **Dinámico:** Se crea con el operador **new** o **new[]** y se destruyen con el operador **delete** o **delete[]**

Memoria dinámica

Operadores

1. **new** Para crear un espacio de memoria para almacenar un tipo de dato u objeto
2. **new[]** para crear un espacio de memoria para almacenar uno o más tipos de datos u objetos del mismo tipo
3. **delete** Para liberar un espacio de memoria
4. **delete[]** Para liberar un espacio de memoria de uno o más tipos de datos u objetos

Contenido

Punteros

Definición

Hasta ahora hemos creado y destruido de forma automática:

```
int main(){
vector<int> a; // Creamos a en memoria
vector<int> b; //Creamos b en memoria

    a = b; //Igualamos a a b (pero son distintos objetos)
a.push_back(3); //Usamos una función (con operador punto)
b.push_back(4);
    cout << a[0] << " " << b[0] << endl;

} // Destrucción de a y b
```

Punteros

Definición

Pero con uso de memoria dinámica podemos decidir cuando creamos y destruimos.

```
int main(){
    //Se crea el  punto tipo vector<int> b y con new creamos una instancia
        en memoria
vector<int> * b = new vector<int>;
  vector<int> *  a; //Puntero

a = b; //Igualamos punteros, es decir ambos apuntan al mismo objeto

a->push_back(3); //Usamos una función (con operador ->)
b->push_back(4);
    cout << (*a)[0] << " " << (*b)[0] << endl;

    // Destrucción de a y b
    delete a;
    delete b;
    //Ya que la memoria ha sido liberada y puede haber otra cosa ahi (otra
        variable)
    //Debemos aterrizar los punteros.
    a = 0;
    b = 0;
}
```

[3 4]

Punteros

Definición

Así mismo se puede usar el operador * para obtener el valor de la variable a la cual apunta:

```
int variable = 354;
int * ptrvariable = &variable;
cout << ptrvariable << endl;
cout << *ptrvariable << endl;
*ptrvariable = 866;
cout << variable << endl;
```

Punteros

Definición

Se pueden usar punteros para hacer pasos por referencia:

```
void funcion(int * p){
    cout << p << endl;
    //*p (se utiliza de esta forma para acceder al dato
        que apunta el puntero)
    *p = 5;
}

int main(){
    int x = 55;
    cout << x << endl;
    funcion(&x);
    cout << x << endl;

}
```

Contenido

Arreglos dinámicos

Definición

Un arreglo dinámico es un arreglo el cual podemos definir en ejecución:

```
int * arregloEnteros = 0;
int size;
cout << "Ingrese el número de elementos" << endl;
cin >> size;
arregloEnteros = new int[size];
for(int i=0; i<size; i++){
    cout << "Ingrese el valor del elemento "<<i+1<<endl;
    cin >> arregloEnteros[i];
}
```

El operador **new** le dice al compilador que debe reservar memoria para el arreglo u objeto.

Arreglos dinámicos

Definición

En el caso de arreglos bidimensionales, se especifican las dos dimensiones

```
int filas, columnas;
cout << "Ingrese el número de filas" << endl;
cin >> filas;
cout << "Ingrese el número de columnas" << endl;
cin >> columnas;

int ** arreglo2D = new int*[filas];

//Creamos las columnas
for(int i=0; i<filas; i++){
    arreglo2D[i] = new int[columnas];
}
```

Arreglo bidimensional dinámico,
tipo flotante (double)
5 Filas
2 columnas

Declarar e instanciar

Contenido

Paso de arreglos como parámetros

Definición

El siguiente código muestra un paso por referencia de un **arreglo estático**.

```
//Se pasa como una reserva de memoria
void pasoFunEstaticos(int* a){
    a[1]=10;
    cout << a[2] << endl;
}

int main(){
    int arreglo[5]={1,2,3,4,5};
    pasoFunEstaticos(arreglo);
    cout << arreglo[1] << endl;
}
```

Paso de arreglos como parámetros

Definición

El siguiente código muestra un paso por referencia de un **arreglo dinámico**.

```
void pasoFunDinamicos(int* & a){
    a = new int[2];
    a[1]=10;
    a[0]=3;
    cout << a[1] << endl;
}

int main(){
    int * arregloDinamico;
    pasoFunDinamicos(arregloDinamico);
    cout << arregloDinamico[1] << endl;
}
```

Paso de arreglos como parámetros

Ejercicio

1. Declare tres arreglos dinámicos (dos de enteros y uno de strings) para recolectar las edades, salarios y nombres de personas. Estos están declarados en el método **main**.
2. Diseñe una función llamada **recolectar**, la cual recibe estos dos arreglos dinámicos y un entero en paso por referencia, solicita al usuario el número de personas y solicita los datos correspondientes. Defina aquí ambos arreglos con **new**
3. Diseñe la función **imprimir**, que recibe estos dos arreglos y un entero en paso por referencia e imprime "La persona <nombre> tiene <edad> años".
4. **Todas las funciones deben ser invocadas desde el main**

¿Preguntas?

Y por fin llegamos a Objetos :)