Programación I

Unidad 03

Punteros y memoria dinámica

Unidad 03
Punteros y memoria dinámica

Temario

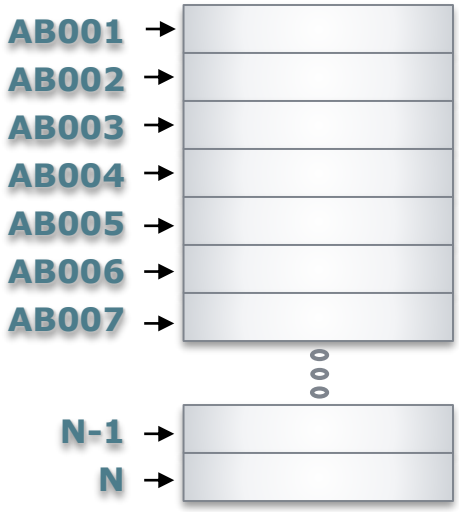
- Memoria del computador
- Asignación en C
- Punteros
- Comando de Indirección
- Ejercicios

Memoria de la computadora

Memoria del computador

La memoria del computador es el lugar donde se almacenan los valores de las variables que declaramos en nuestros programas.

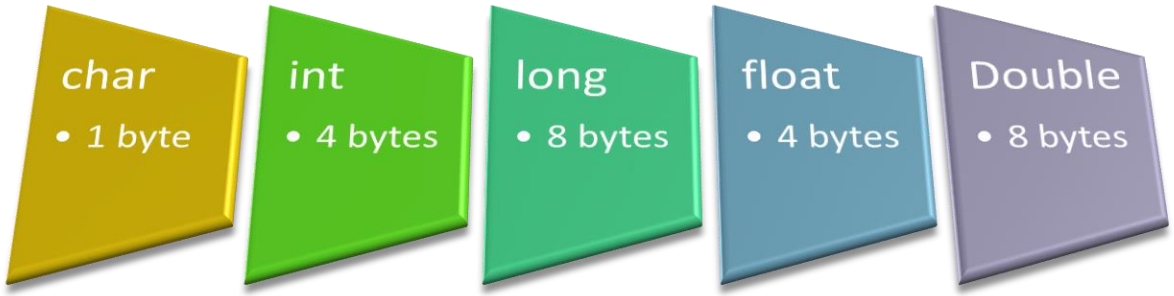
La memoria está formada por un conjunto de celdas continuas cada una de 1 byte de tamaño. Estas celdas están enumeradas de forma consecutiva de tal forma que cada celda tiene el mismo número que la anterior más 1.



Memoria del computador

Al declarar una variable el Sistema Operativo (S.O) reserva en la memoria el espacio necesario para almacenar dicha variable.

Las variables tienen diferente tamaño que pueden ser obtenidos utilizando la función **sizeof(tipo)**

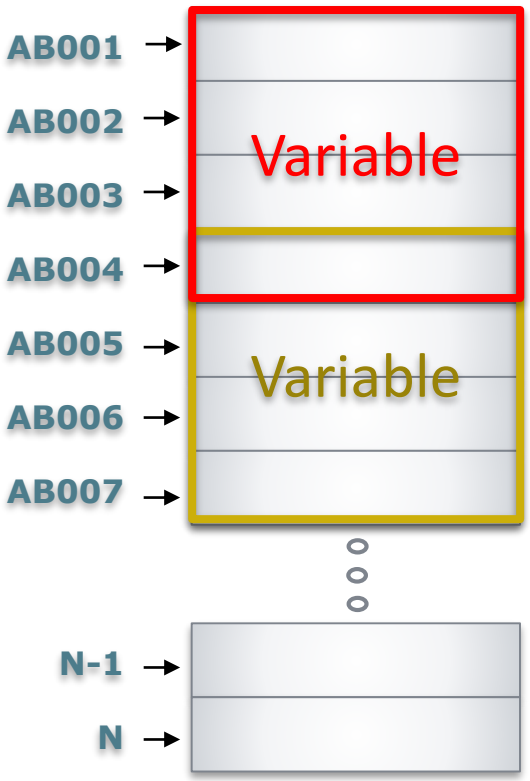


Memoria del computador

La ubicación de las variables dentro de la memoria es aleatoria y se calcula cada vez que se ejecuta el programa por lo que no podemos saber la posición exacta hasta que el programa se ejecute.

Memoria del computador

tipodato Variable;



La segunda vez que se ejecuta el programa

La primera vez que se ejecuta el programa

Analogía de la memoria

Analogía de la memoria

Un teatro	La memoria
• Una **persona** desea ingresar a un **teatro**.	• Una **variable** desea ingresar a la **memoria del computador**.
• El **acomodador** (personal encargado de la ubicación) busca un espacio disponible y ubica a la **persona** en dicho espacio.	• El **sistema operativo** busca un espacio disponible y ubica a la **variable** en dicho espacio.
• En la próxima **función de teatro**, el **acomodador** probablemente ubicará a la **persona** en un lugar distinto dependiendo de la disponibilidad en el **teatro**.	• En la próxima **ejecución**, el **Sistema Operativo** probablemente ubicará a la **variable** en un lugar distinto dependiendo de la disponibilidad en la **memoria**.

Asignación en C

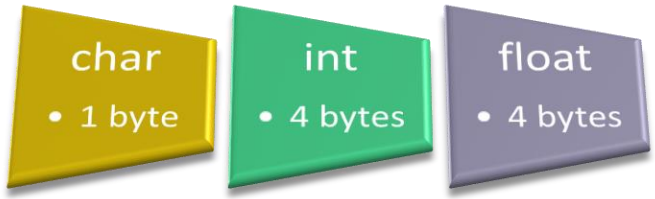
Asignación en C

Cuando hacemos una aplicación, declaramos diversas variables, por ejemplo:

- char x;
- int y;
- float z;

Pero recuerda que….

- Cada variable, según el tipo de dato, ocupa una cantidad distinta de bytes en memoria



Memoria del computador

Dirección	Valor
AB001	
AB002	
AB003	
AB004	
AB005	
AB006	
AB007	
AB008	
AB009	
AB00A	
AB00B	
AB00C	
AB00D	
AB00E	
AB00F	
AB010	
AB011	
AB012	
AB013	
AB014	
.	
.	
N	

Al declarar una variable, el S.O le asigna la mejor ubicación posible en memoria

- char x;
- int y;
- float z;

Asignación en C

	Dirección	Valor
	AB001	
	AB002	
z	AB003	
	AB004	
	AB005	
	AB006	
	AB007	
	AB008	
	AB009	
	AB00A	
	AB00B	
y	AB00C	
	AB00D	
	AB00E	
	AB00F	
	AB010	
	AB011	
	AB012	
x	AB013	
	AB014	
	.	
	.	
	N	

Luego de declarar las variables, recién podemos asignarle un valor

- x = **'A'**;
- y = **17**;
- z = **15.8**;

Asignación en C

	Dirección	Valor
	AB001	
	AB002	
z	AB003	15.8
	AB004	
	AB005	
	AB006	
	AB007	
	AB008	
	AB009	
	AB00A	
	AB00B	
y	AB00C	17
	AB00D	
	AB00E	
	AB00F	
	AB010	
	AB011	
	AB012	
x	AB013	'A'
	AB014	
	.	
	.	
	N	

En C++,

- Referenciamos a las variables por su nombre para imprimirlas
  - cout<< x;
  - cout<< y;
  - cout<< z;
- Referenciamos a las variables por su nombre cuando vamos a almacenar un valor en ellas
  - cin>> x;
  - cin>> y;
  - cin>> z;

Asignación en C

- Existen otro tipos de variables a los ya tradicionales
- Estas variables no almacenan un dato char o int o double o float, almacenan direcciones de memoria
- Las variables que almacenan direcciones de memoria, reciben el nombre de **puntero** o variable dinámica.

Punteros

Punteros

Un puntero es un tipo especial de variable, que almacena el valor de una dirección de memoria y es en la dirección de memoria, donde se encuentra en dato.

Para almacenar una dirección de memoria, las variables del tipo puntero, requieren de 2 bytes.

Punteros

Un puntero se define de la siguiente forma:

Tipo de dato * nombredelpuntero;

Ejemplo:

- int * ptrentero;
- char * cadena;
- float * numero;

Punteros

Imaginemos que hemos declarado dos variables:

- **int** y;
- **int *** ptrentero;

Recuerda que….

- Mientras la primera variable y almacena un dato entero la segunda ptrentero almacena una dirección de memoria.

Punteros

Dirección	Valor
AB001	
AB002	
AB003	
AB004	
AB005	
AB006	
AB007	
AB008	
AB009	
AB00A	
AB00B	
AB00C	
AB00D	
AB00E	
AB00F	
AB010	
AB011	
AB012	
AB013	
AB014	
.	
.	
N	

Al declarar una variable, el S.O le asigna la mejor ubicación posible en memoria

```
int y;
int *ptrentero;
```

Punteros

	Dirección	Valor
	AB001	
	AB002	
	AB003	
	AB004	
	AB005	
	AB006	
	AB007	
	AB008	
	AB009	
y	AB00A	
	AB00B	
	AB00C	
	AB00D	
	AB00E	
	AB00F	
	AB010	
	AB011	AB008
ptrentero	AB012	
	AB013	
	AB014	
	.	
	.	
	N	

Luego de declarar las variables, recién podemos asignarle un valor

- y = 14;

Pero….

Al puntero debemos asignarle una dirección de memoria

ptrentero=&y;

Del ejemplo:

- La variable y ocupa 4 bytes (int) y tiene el valor 14
- La variable ptrentero ocupa 2 bytes (dirección de memoria) y tiene el valor de AB008
- En la dirección de AB008 se encuentra almacenado un dato entero cuyo valor es el de la variable y (14)
- Entonces, si se almacena la dirección de memoria, puedo obtener el dato allí almacenado ?????

SI!

Comando de indirección

Comando de Indirección

	Dirección	Valor
	AB001	
	AB002	
	AB003	
	AB004	
	AB005	
	AB006	
	AB007	
	AB008	14
	AB009	
y	AB00A	
	AB00B	
	AB00C	
	AB00D	
	AB00E	
	AB00F	
	AB010	
	AB011	AB008
ptrentero	AB012	
	AB013	
	AB014	
	.	
	.	
	N	

A partir de un puntero que almacena una dirección de memoria podemos usar el comando de indirección (*):

* ptrentero

Anteponer * al nombre del puntero significa: "El valor apuntado por "

Y cómo funciona????

Comando de Indirección

Dirección	Valor
AB001	
AB002	
AB003	
AB004	
AB005	
AB006	
AB007	
AB008	14
AB009	
AB00A	
AB00B	
AB00C	
AB00D	
AB00E	
AB00F	
AB010	
AB011	AB008
AB012	
AB013	
AB014	
.	
.	
N	

y

ptrentero

El comando es utilizado para, a partir, de una dirección de memoria obtener el valor allí almacenado .

Se puede usar para:

Imprimir el valor:

cout<< *ptrentero;

Comando de Indirección

Dirección	Valor
AB001	
AB002	
AB003	
AB004	
AB005	
AB006	
AB007	
AB008	14
AB009	
AB00A	
AB00B	
AB00C	
AB00D	
AB00E	
AB00F	
AB010	
AB011	AB008
AB012	
AB013	
AB014	
.	
.	
N	

y

ptrentero

El comando es utilizado para, a partir, de una dirección de memoria obtener el valor allí almacenado .

Se puede usar para:

Alterar el Valor :

15

***ptrentero= *ptrentero +1;**

Ejercicios

Muestra el uso de direcciones

Ejemplo 1

Escribir un programa en C++ que permita declarar dos variables enteras, asignarles un valor e imprimir sus direcciones de memoria.

```
#include <iostream>

using namespace std;

int main()
{
  int varA;
  int varB;

  varA = 90;
  varB = 120;

  cout<<"varA = " << varA << " y su direccion es: " << &varA << "\n";
  cout<<"varB = " << varB << " y su direccion es: " << &varB << "\n";
  return 0;
}
```

Ejemplo 2

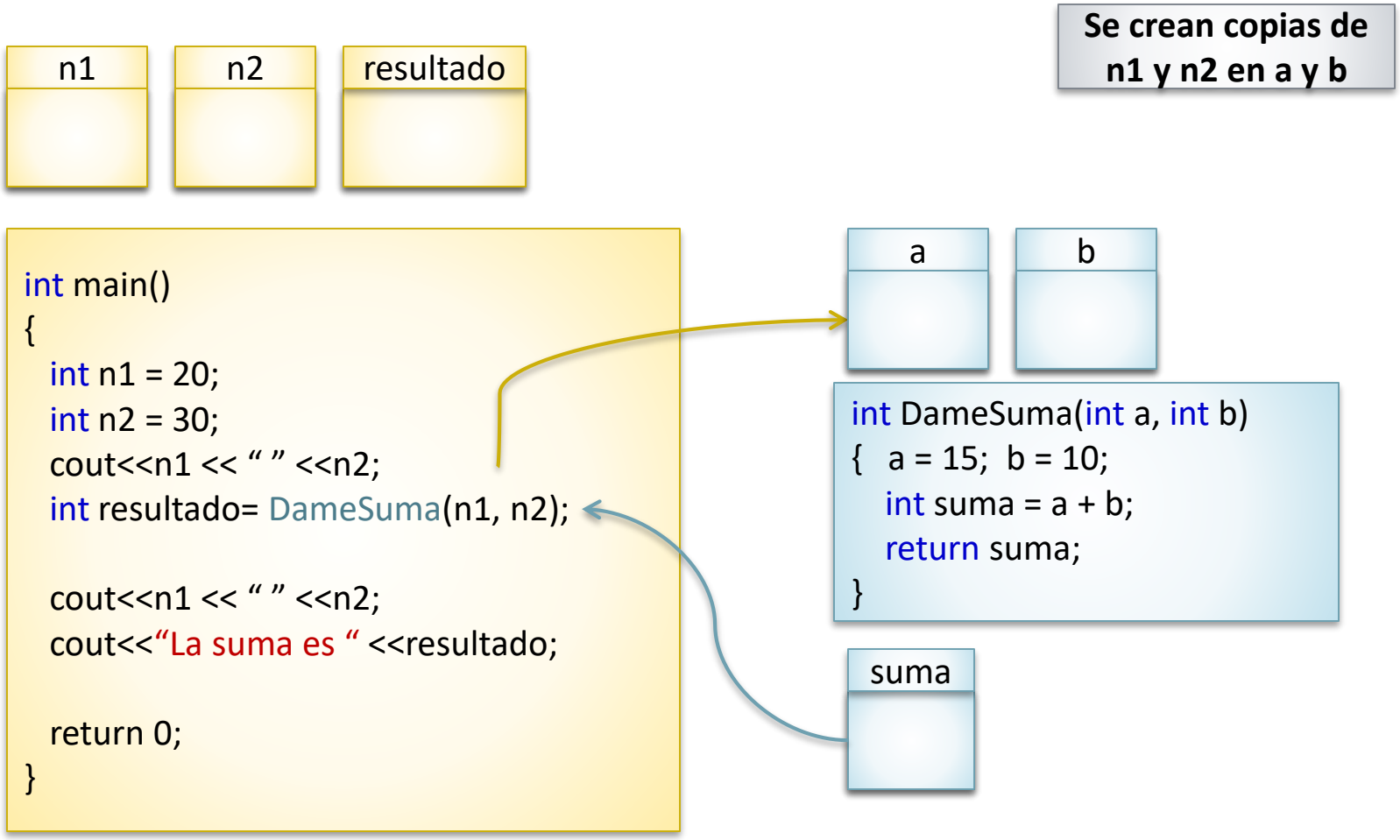
Escribir un programa en C++ que permita declarar dos variables enteras, asignarles un valor e imprimir sus direcciones de memoria. Luego, **declarar un puntero a una de ellas y manipular el dato a través del puntero**.

```
int main()
{
  int varA;
  int varB;

  varA = 90;
  varB = 120;

  cout<<"varA = " << varA << " y su direccion es: " << &varA << "\n";
  cout<<"varB = " << varB << " y su direccion es: " << &varB << "\n";

  int *pA = &varA;    // declara la variable puntero pA y le asigna un valor

  int *pB;            // declara la variable puntero pB
  pB = &varB;         // le asigna un valor
  *pB = 45;           // modifica el valor apuntado por pB

  cout<<"varA = " << *pA << " y su direccion es: " << pA << "\n";
  cout<<"varB = " << *pB << " y su direccion es: " << pB << "\n";

  return 0;
}
```

Parámetros de una función

Muestra el uso de direcciones

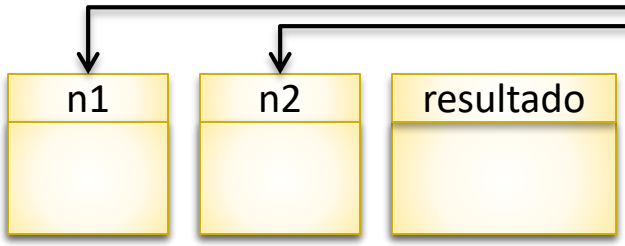
Parámetros por valor



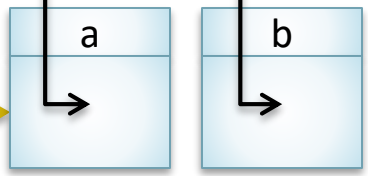
NOTA: Las variables en el main y los parámetros en DameSuma **pueden llamarse igual, pero son diferentes.**

Parámetros por referencia

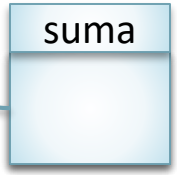
NO se crean copias, la función opera con n1 y con n2 directamente: mediante sus referencias.




```
int main()
{
  int n1 = 20;
  int n2 = 30;
  cout<<n1 << " " <<n2;
  int resultado = DameSuma(n1, n2);

  cout<<n1 << " " <<n2;
  cout<<"La suma es "<<resultado;

  return 0;

}
```




```
int DameSuma(int &a, int &b)
{  a = 25; b = 20;
   int suma = a + b;
   return suma;
}
```



Parámetros por referencia puntero