



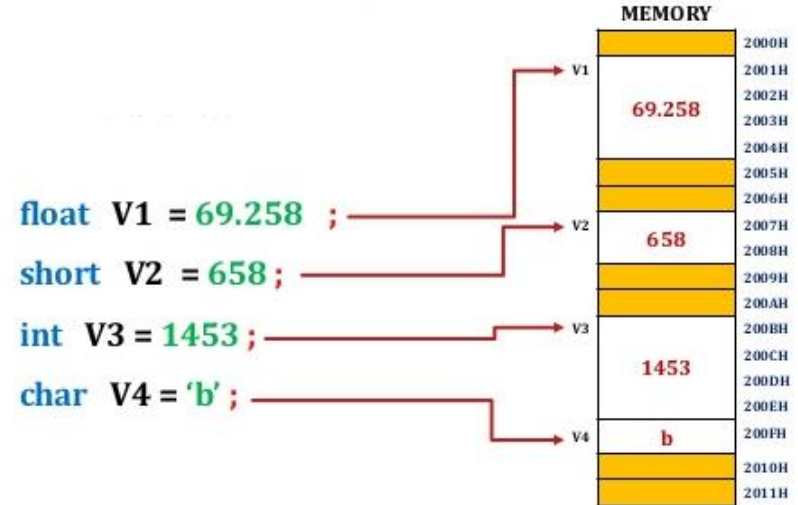
Análisis de Algoritmos

espacio de memoria

B.S. Rodolfo Mercado Gonzales
Universidad Nacional de Ingeniería

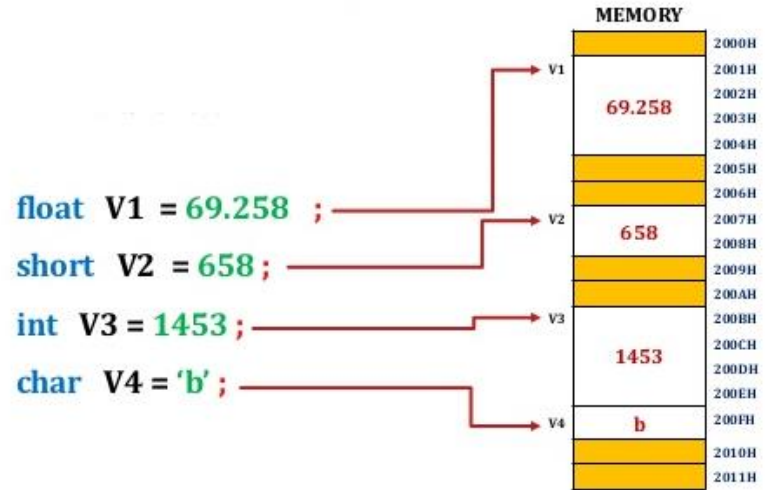
¿ Qué es una variable ?

- ❑ Espacio en la memoria de la computadora donde se almacenará un valor.
- ❑ Posee un nombre (identificador) asociado.



Modelo de memoria

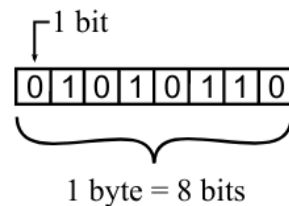
- ❑ La memoria (RAM) es como un arreglo muy grande de bytes.
- ❑ Al índice de cada posición lo llamamos **dirección de memoria** (hexadecimal).
- ❑ Las variables ocupan uno o más bytes (posiciones).
- ❑ La dirección es el índice del primer byte de la variable.



Bit (b)

- ❑ Unidad mínima de información.
- ❑ Es un dígito en el sistema binario, puede tomar dos valores: 0 (apagado) y 1 (prendido).
- ❑ La computadora solo “entiende” a nivel de bits.

Byte (B)



- ❑ Secuencia contigua de 8 bits.
- ❑ Unidad usada para medir capacidad de almacenamiento de una memoria.

Binary			Decimal		
Name	Symbol	Value (base 2)	Name	Symbol	Value (base 10)
kibibyte	KiB	2^{10}	kilobyte	KB	10^3
mebibyte	MiB	2^{20}	megabyte	MB	10^6
gibibyte	GiB	2^{30}	gigabyte	GB	10^9
tebibyte	TiB	2^{40}	terabyte	TB	10^{12}
pebibyte	PiB	2^{50}	petabyte	PB	10^{15}
exbibyte	EiB	2^{60}	exabyte	EB	10^{18}

Uso de memoria

Tipo de dato	Bytes en C++	Valores
bool	1 B	0, 1
char	1 B	−128 a 127
int	4 B	−2147483648 a 2147483647
long long	8 B	−9,223,372,036,854,775,808 a 9,223,372,036,854,775,807
float	4 B	$-3.4 * 10^{-38}$ a $3.4 * 10^{38}$
double	8 B	$-1.7 * 10^{-308}$ a $1.7 * 10^{308}$

Complejidad en espacio

Así como contábamos la cantidad de operaciones, ahora tenemos que contar la cantidad de espacio de memoria (bytes o celdas) que usaremos.

Complejidad en espacio

`int A[60000000]` $\rightarrow 6 * 10^7 * 4 \text{ bytes} = 24 * 10^7 \text{ B} = \mathbf{240 \text{ MB}}$

`long long A[60000000]` $\rightarrow 6 * 10^7 * 8 \text{ bytes} = 48 * 10^7 \text{ B} = \mathbf{480 \text{ MB}}$

¿Es necesario saber exactamente el número de bytes para comparar algoritmos?



Notación Big O

- ❑ Podemos usar la notación **Big O** y solo darnos una idea de la cantidad de elementos que tenemos que almacenar.
- ❑ Lógicamente debemos expresarla en función de la entrada.

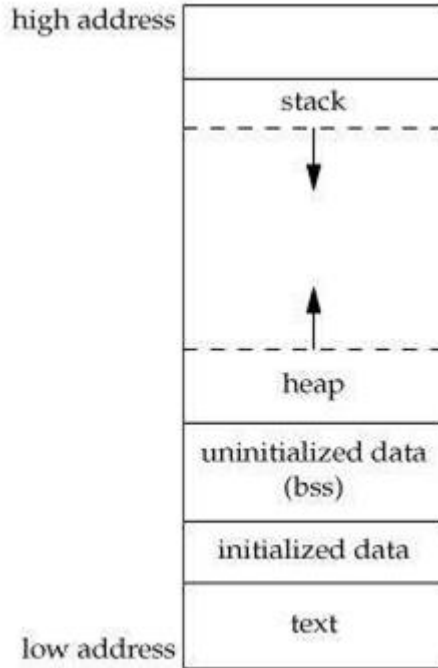
Complejidad en espacio

```
int A[ N ];  
  
for( int i = 0; i < N; ++i ){  
    for( int j = i + 1; j < N; ++j ){  
        if( A[ i ] > A[ j ] ) swap( A[ i ], A[ j ] );  
    }  
}
```

❑ Tiempo: $O(N^2)$

❑ Espacio: $O(N)$

Modelo de memoria en C++



- ❑ Segmento de código (text) : almacena el código en lenguaje máquina.
- ❑ **Segmento de datos** (initialized and uninitialized data) : almacena las variables globales .
- ❑ **Stack** : almacena variables locales y llamadas a funciones.
- ❑ **Heap** : reserva memoria dinámica.

Memoria Stack

- ❑ Región de la memoria que es gestionada eficientemente por el CPU.
- ❑ No necesitamos reservar ni liberar memoria manualmente.
- ❑ Tiene un tamaño límite pequeño.
- ❑ Almacena variables locales, parámetros y llamadas a funciones.

Memoria Stack

Que pasa si declaramos el siguiente arreglo de dentro de una función:

```
int main(){  
    int A[ 1000000 ]; //4MB  
}
```

hallemos el límite del stack !



Memoria Heap

- ❑ No es administrada automáticamente.
- ❑ Somos responsables de liberar la memoria cuando ya no la necesitamos.
- ❑ Su único límite de tamaño depende del hardware.
- ❑ La lectura y escritura es un poco más lenta que en el stack.

Memoria Heap

```
int main(){  
  
    int *numeros = new int [ 1000000000];  
    //delete[] numeros;  
}
```

el heap puede usar toda la RAM !



Segmento de datos

```
int A[ 1000000000 ];  
  
int main(){  
    return 0;  
}
```



hallemos el límite del
segmento de datos !

Problemas

[Timus – Sequence Median](#)

Referencias

- ❑ Steve Oualline - **Practical C++ programming**
- ❑ Daniel Jiménez - **CS 1723 Data Structures – UTSA**
- ❑ Hackerearth - **Memory Layout of C Program**

¡ Good luck and have fun !