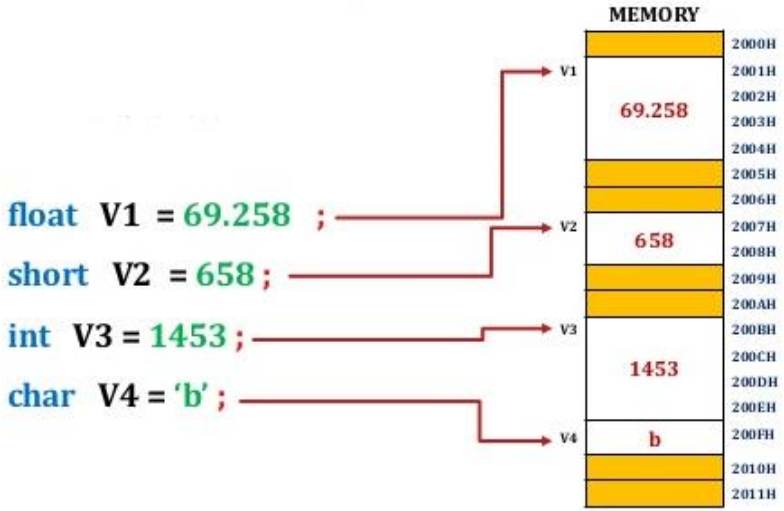
Análisis de Algoritmos

espacio de memoria

B.S. Rodolfo Mercado Gonzales
Universidad Nacional de Ingeniería - FIIS

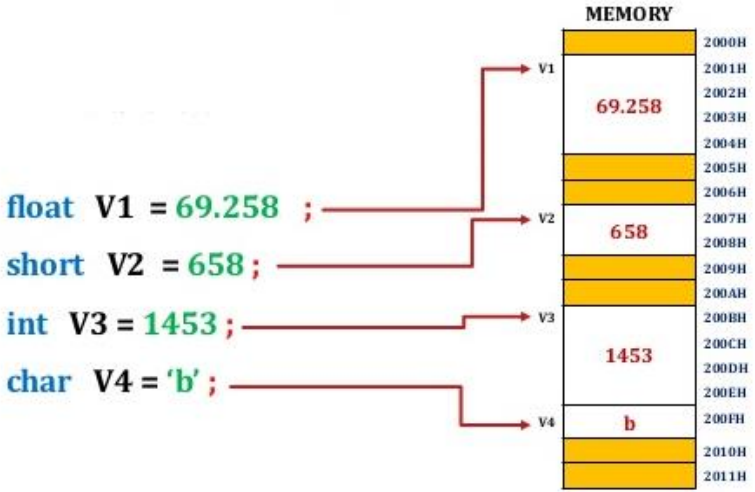
¿Qué es una variable?

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



Modelo de memoria

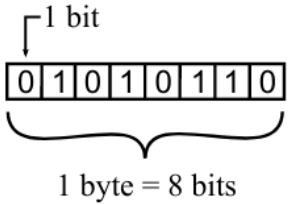
- La memoria (RAM) es como un arreglo muy grande de bytes.
- Al índice de cada posición lo llamamos **dirección de memoria** (hexadecimal).
- Las variables ocupan uno o más bytes (posiciones).
- La dirección es el índice del primer byte de la variable.



Bit

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

Byte (B)



- 1 byte (B) es una secuencia 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}

Representación de los números en un computador

- Los enteros son representados como una secuencia de bits.
- La cantidad de bits que se usan en su representación depende del tipo de dato.

Tipo de dato	Bits / Bytes en C++
int, unsigned int, float	32 bits / 4 B
long long, unsigned long long, double	64 bits / 8 B

Enteros sin signo

- Permite almacenar solo números enteros no negativos (unsigned).
- Guarda el número en base binaria, completando con ceros a la izquierda.

Decimal	Unsigned int (32 bits)	Unsigned long long (64 bits)
5	000…….00101	000000000…….00101
20	000…….10100	000000000…….10100

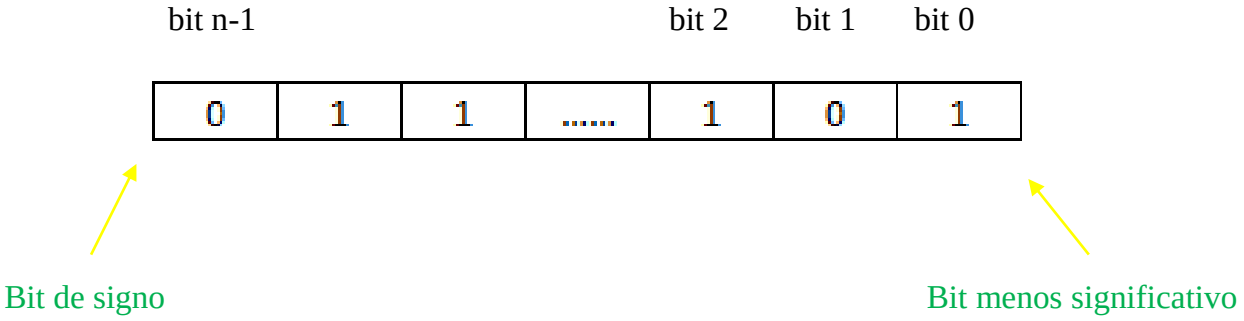
Enteros sin signo

- Con n bits podemos representar los enteros en el rango $[0, 2^n - 1]$.

Unsigned int (32 bits)	Unsigned long long (64 bits)
$[0, 2^{32} - 1]$	$[0, 2^{64} - 1]$

Enteros con signo

- Permite almacenar un número entero.
- Utiliza la representación complemento a dos.



Enteros con signo

Entero positivo o cero

El bit de signo es igual a 0 y los $n - 1$ bits restantes se completan con la representación binaria del número.

Decimal	int (32 bits)	long long (64 bits)
0	**0**00…….00000	**0**00000000…….00101
20	**0**00…….10100	**0**00000000…….10100

Enteros con signo

Entero negativo

El bit de signo es igual a 1 y los $n - 1$ bits restantes se completan con la representación binaria de $2^{n-1} - abs(x)$.

Decimal	int (32 bits)	long long (64 bits)
-1	**1**11…….11111	**1**11111111…….11111
-3	**1**11…….11101	**1**11111111…….11101

Enteros con signo

- Con n bits podemos representar los enteros en el rango $[-2^{n-1}, 2^{n-1} - 1]$.

int (32 bits)	long long (64 bits)
$[-2^{31}, 2^{31} - 1]$	$[-2^{63}, 2^{63} - 1]$

Observaciones

- Se utiliza la representación complemento a dos, debido a que aquí el 0 tiene una única forma (000…000), mientras que en otras representaciones podía tener dos formas (**0**00…000, **1**00..000).

- La representación de los enteros negativos en complemento a dos también se puede obtener como :

  - Hallamos la representación binaria de $abs(x)$ usando n bits.

  - Negamos todos los bits del número obtenido.

  - Finalmente le sumamos 1 al número obtenido.

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 elementales, ahora tenemos que contar la cantidad de espacio de memoria (bytes o celdas) que usaremos.

El uso de la memoria lo medimos como la mayor cantidad de memoria que usa un programa al mismo tiempo

Por ejemplo si nuestro programa usa 100 MB de memoria, liberamos unos 30 MB y luego ocupamos otros 20 MB; la memoria usada sería 100 MB.

Complejidad en espacio

int A[60000000] → $6 * 10^7 * 4\ bytes = 24 * 10^7$ B **= 240 MB**

long long A[60000000] → $6 * 10^7 * 8\ bytes = 48 * 10^7$ B **= 480 MB**

¿Es necesario saber el número exacto 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.

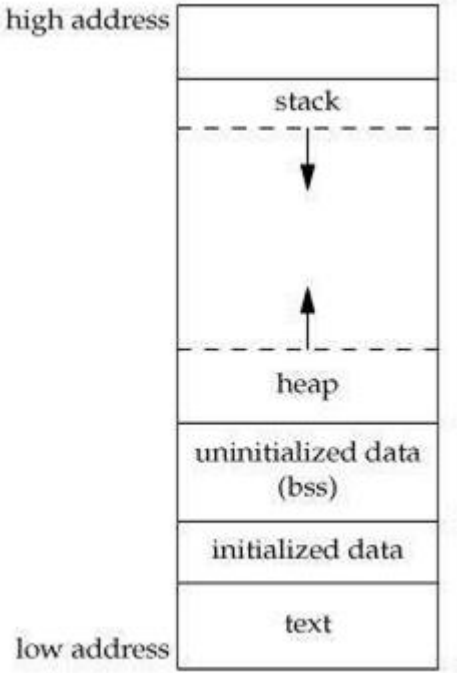
Complejidad en espacio

```
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 necesitemos.
- 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 !

Conclusiones

- Generalmente más nos preocuparemos por ser eficientes en tiempo de ejecución
- En los concursos online la memoria máxima varía entre 256M Y 512M

Problemas

Timus – Sequence Median

Referencias

- Hackerearth - **Memory Layout of C Program**

¡ Good luck and have fun !