

TAREA 1

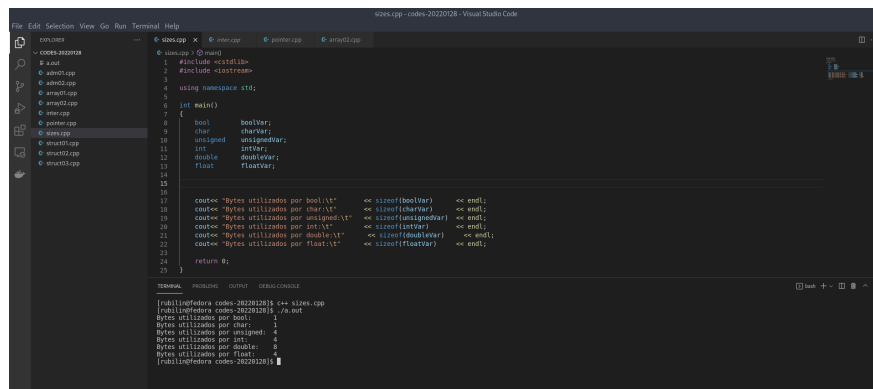
Problema 1

Compile, corra y analice el funcionamiento de los códigos:

- sizes.cpp
- inter.cpp

- **sizes.cpp:**

Compilar y correr:



```
1 #include <cstdlib>
2 #include <iostream>
3
4 using namespace std;
5
6 int main()
7 {
8     bool    boolVar;
9     char    charVar;
10    unsigned int    unsignedVar;
11    int        intVar;
12    double    doubleVar;
13    float     floatVar;
14
15
16    cout<< "Bytes utilizados por bool:\t" << sizeof(boolVar) << endl;
17    cout<< "Bytes utilizados por char:\t" << sizeof(charVar) << endl;
18    cout<< "Bytes utilizados por unsigned:\t" << sizeof(unsignedVar) << endl;
19    cout<< "Bytes utilizados por int:\t" << sizeof(intVar) << endl;
20    cout<< "Bytes utilizados por double:\t" << sizeof(doubleVar) << endl;
21    cout<< "Bytes utilizados por float:\t" << sizeof(floatVar) << endl;
22
23    return 0;
24 }
```

```
[rubí@fedora codes-20220128] $ g++ sizes.cpp
[rubí@fedora codes-20220128] $ ./a.out
Bytes utilizados por bool: 1
Bytes utilizados por char: 1
Bytes utilizados por unsigned: 4
Bytes utilizados por int: 4
Bytes utilizados por double: 8
Bytes utilizados por float: 4
[rubí@fedora codes-20220128] $
```

Figura 1: Resultado de correr sizes.cpp

Las primeras dos líneas se llaman a las librerías que se van a utilizar, la primer `cstdlib` contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras. Y `iostream` que es utilizado para operaciones de entrada/salida. Luego se utiliza los namespace `std` para dar acceso al espacio de nombres (namespace) `std`, donde se encuentra encerrada toda la librería estándar.

En el `main`, se define tipos de variables, una variable booleana, un caracter, un número sin signo, un entero, una variable `double` y un floatante o valor real. Y con la ayuda de `«cout»` se imprime en consola un string se deja una espacio tabular y luego se utiliza la función `«sizeof»` la cual indica el tamaño de la variable a la que se le aplica. En los casos de las variable booleanas y caracteres su tamaño es de 1 byte. Las variables sin signo, los enteros y los floatantes tienen un tamaño de 4 bytes. Y las variables `double` como su nombre lo indican tiene el doble que son 8 bytes.

- **inter.cpp**

Compilar y correr:

```

1 #include <cstdio>
2 #include <iostream>
3
4 using namespace std;
5
6 int main()
7 {
8     int Var1, Var2, Var3;
9
10    // Estas son tres formas de asignar valor a una variable.
11    // Notese que en este caso las variables tipo int son
12    // de 4 bytes o 32 bits.
13    // Binario
14    Var1 = 0b11001111001010101100110011010;
15    // Hexadecimal
16    Var2 = 0xc7ab33a;
17    // Decimal
18    Var3 = -819285190;
19
20    cout << Var1 << endl;
21    cout << Var2 << endl;
22    cout << Var3 << endl;
23
24    /* En el caso de variables enteras es posible reinterpretar el
25     * valor asignado de memoria a enteras con/sin signo de manera simple */
26    cout << (unsigned int) Var1 << endl;
27
28    /* En el caso de enteros a flotantes, con este mismo metodo se realiza
29     * una conversión de tipo, no una reinterpretación del valor asignado
30     * memoria como otro caso */
31
32    [Build]Procedura codes-2022012013 /> ./out
33    Bytes utilizados por total: 1
34    Bytes utilizados por Var1: 4
35    Bytes utilizados por Var2: 4
36    Bytes utilizados por Var3: 4
37    Bytes utilizados por float: 4
38    [Build]Procedura codes-2022012013 <<> inter.cpp
39    [Build]Procedura codes-2022012013 /> ./out
40    -819285190
41    0xc7ab33a
42    -819285190
43    3475682106
44    3475682106
45    3.47568e+09
46    Time
47    0.00s
48    [Build]Procedura codes-2022012013 />

```

Figura 2: Resultado de correr inter.cpp

Las primeras dos líneas se llaman a las librerías que se van a utilizar, la primer `cstdio` contiene los prototipos de funciones de C para gestión de memoria dinámica, control de procesos y otras. Y `iostream` que es utilizado para operaciones de entrada/salida. Luego se utiliza los namespace `std` para dar acceso al espacio de nombres (namespace) `std`, donde se encuentra encerrada toda la librería estándar.

En la función `main` primero se declaran tres variables de tipo entero y luego se les asigna el mismo valor «-819285190» pero de tres formas diferentes una para cada variable, de forma binaria con «0b» en complemento a dos, de forma hexagesimal «0x», y de forma decimal. Y luego se imprimen en consola.

En la siguiente línea, en caso de variables enteras se hace una reinterpretación de la variable «Var1» a entero sin signo por lo que cambia a $2^{32} - 819285190 = 3475682106$ y se imprime. Luego en el caso de flotantes se hace una conversión de tipo flotante (no una reinterpretación del valor asignado) y de nuevo se imprime.

Luego, se utilizan las variables enteras y flotantes se pueden utilizar como variables booleanas utilizando la condición `if` y `else`. Si la variable es cero se interpretará como «false» si no se interpretará como «true». Como al principio la variable «Var1» tenía un valor no igual a cero se imprimió «true» y luego como se le asignó el valor 0 y por ende se imprimió «false».

Luego se declara la cuarta variable «Var4» de tipo flotante y se le asigna el mismo valor en binario que a la variable «Var1». Al imprimirlo se puede notar que se interpreta de forma diferente, sin signo, se interpreta como $2^{32} - 819285190 = 3475682106$ pero escrito de formato de coma flotante `3.47568e + 09`

Problema 2

Investigue la representación binaria con complemento a 2 para números enteros negativos. Calcule la representación a 32 bits de los siguientes números: -125, -4096, -1000000.

Complemento a 2: Es una operación matemática en números binarios, es usado como un método de cómputo en la representación de números con signo[1].

El complemento a dos de un número N que, expresado en el sistema binario con n dígitos, se define como:

$$C_2^N = 2^n - N$$

El total de números positivos será $2^{n-1} - 1$ y el de negativos 2^{n-1} , siendo n el número máximo de bits. El 0 contaría aparte[1].

Para obtener el complemento a dos de un número se puede primero pasar al complemento a uno (el valor obtenido al invertir todos los bits en la representación binaria del número, intercambiando 0 por 1 y viceversa) y luego sumar 1, debido a que el complemento a dos de un número binario es una unidad mayor que su complemento a uno (por la representación duplicada del 0) [1], [2].

$$C_2^N = C_1^N + 1$$

■ **-125:**

- Resolver para el entero sin signo: $(125)_{10} = (1111101)_2$
- Rellenar con ceros para formar un número de 32-bit:
0000 0000 0000 0000 0000 0000 0111 1101
- Para un número negativo debemos invertir todos los bits:
1111 1111 1111 1111 1111 1111 1000 0010
- Sumar uno:
1111 1111 1111 1111 1111 1111 1000 0011

■ **-4096:**

- Resolver para el entero sin signo: $(4096)_{10} = (1000000000000)_2$
- Rellenar con ceros para formar un número de 32-bit:
0000 0000 0000 0000 0001 0000 0000 0000
- Para un número negativo debemos invertir todos los bits:
1111 1111 1111 1111 1110 1111 1111 1111
- Sumar uno:
1111 1111 1111 1111 1111 0000 0000 0000

■ **-1000000:**

- Resolver para el entero sin signo: $(1000000)_{10} = (11110100001001000000)_2$
- Rellenar con ceros para formar un número de 32-bit:
0000 0000 0000 1111 0100 0010 0100 0000
- Para un número negativo debemos invertir todos los bits:
1111 1111 1111 0000 1011 1101 1011 1111
- Sumar uno:
1111 1111 1111 0000 1011 1101 1100 0000

Referencias

- [1] “Complemento a dos - wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Complemento_a_dos.
- [2] “Complemento a uno - wikipedia, la enciclopedia libre.” https://es.wikipedia.org/wiki/Complemento_a_uno.
- [3] “Binary number representations.” <http://www.mathcs.emory.edu/~cheung/Courses/255/others/BinNumReps.html>.