Clase 2 – Curso de preparación para la Olimpiada Informática Argentina

Pasaje de bases

Dado a que la computadora solamente entiende 0s y 1s, la idea es primero entender cómo podemos pasar un número a ese sistema. Este sistema es conocido como **binario**.

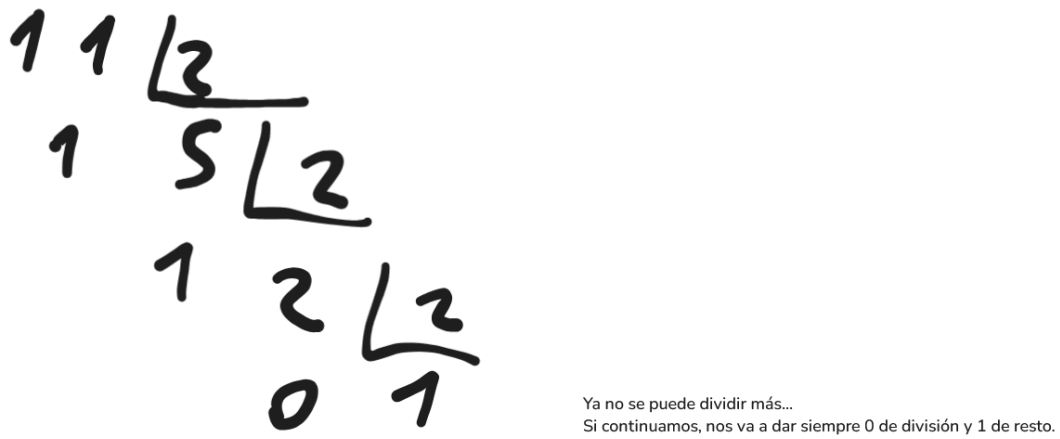
Existen otros más como el **octal** (que es de 0 a 7), **el hexadecimal** (que va así: 0123456789ABCDEF) o el sistema **ternario** (0,1,2).

Pero como solo nos interesa saber pasar a binario, la representación más fácil se da así:

Dividimos un número, supongamos 11, por 2.

Nos queda 5. Sí, pero no 5 fijo, nos sobra 1. Entonces vamos a ir tomando dichas "sobras" hasta no poder continuar dividiendo más.

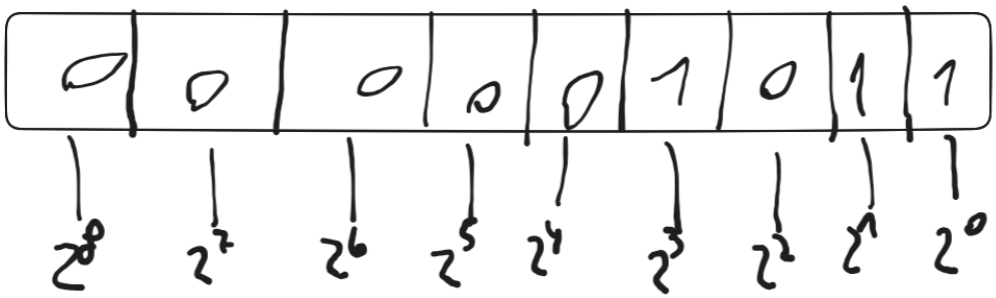
Sería algo así:



Entonces ahora vamos a pasarlo a una tabla para representar el cómo se podría entender este sistema.

Siempre marcamos de abajo para arriba. No importa mucho la tabla si nos pasamos, porque el resto de valores son 0.

Lo que sí nos interesa es saber que valores hay en dichos cuadrados. Y para eso usamos las potencias de 2



Y solo vamos a utilizar los valores que contengan un 1 en ellos y vamos a ir sumando. En este caso 2^3+2^1+2^0. Lo mismo que decir

8+2+1 => 11

Si nos da la misma suma que el valor que habíamos seleccionado, está perfecto.

TAREA: ¿Cómo podría pasarse a binario 511, 913 y 193?

Tipos de datos, variables y operaciones

Ahora, ¿Por qué nos interesa esto? Como bien dijimos arriba, nos sirve para ver un poquito a profundidad como se guarda en el programa. Y también porque, cuando creamos variables (De la definición de la clase presencial: "algo que varía"), tenemos un límite definido de qué es lo que guarda cada cosa.

Para escribir números en C++ utilizamos la palabra reservada **int**, que guarda hasta **32 bits (o 4 bytes)** en números **ENTEROS**, **no números con coma**. Y si no es suficiente 32 bits, existe el **long long, que soporta hasta 64 bits (o 8 bytes), una locura, 2^64.**

Ahora, guardar 32 bits implica que guarda hasta el número 2^32 (o 4 mil millones y algo de números), pero hay que acordarse que guarda también números negativos. Es decir, va de -2 mil millones hasta 2 mil millones. Fuera de ese rango, lo rompés y pasa a tirarte cosas medio extrañas que es conveniente **NO** saber.

Otro tipo de dato que tenemos para guardar números, es el **double o el float**, que nos permiten guardar números con coma. El primero soporta hasta 64 bits de tamaño, mientras que el segundo solo 32.

Para guardar texto, todavía no lo vamos a ver porque no es conveniente y aparte no quiero que Natalia si ve esto me mate, así que por el momento se van a conformar con guardar una letra, y para hacerlo tenemos la palabra reservada char.

¿Y cómo pepino funciona? Bueno, utiliza el número ASCII correspondiente, no hay mucha ciencia, existen 255 carácteres, por ejemplo el 'A' en números enteros sería el 65. Guarda 8 bits (o 1 byte)

Bueno, y el tipo de datos para saber si algo es verdadero o falso, es el bool, que literalmente solo guarda 0 si es falso o 1 si es verdadero. Aún así guarda 8 bits (o un byte)

```
#include<iostream>

Windsurf: Refactor | Explain | Generate Function Comment |
int main(){
    int num=5;
    double num_con_coma = 3.45;
    char letra = 'A';
    bool VoF = true;
    return 0;
}
```

Ahora, la posta es que esto es hiper mega triste, no nos sirve para hacer absolutamente nada con esto. Es por eso que existen palabras reservadas dentro de C++ como el **cout**, que implica una salida (o un mensaje) de consola. ¿Cómo se usa? Se tiene que poner std::cout<<variable;

El punto y coma no está puesto sin querer. PONGANLO. ¡Lo mismo que los puntos y comas de las variables! Es para indicar que ahí se termina. Si nosotros ponemos “std::cout<<num;”, en la consola va a aparecer un 5. ¿Por qué? Porque la variable “num” tiene un valor de “5”

También se puede poner std::cout<<”El número es: “<<num;” y de esta forma, te genera el mensaje correspondiente, para que sea más “lindo” (mentira, sigue siendo un programa de consola horrible)

```
#include<iostream>

Windsurf: Refactor | Explain | Generate Function Comment |
int main(){
    int num=5;
    double num_con_coma = 3.45;
    char letra = 'A';
    bool VoF = true;

    std::cout<<"El numero es: "<<num;
    return 0;
}
```

```
El numero es: 5
Process returned 0 (0x0)   execution time : 0.038 s
Press any key to continue.
```

Ahora… Solo podemos imprimir números… Pero esto no nos sirve, no estamos haciendo nada. Es por esto que existe otra función dentro de C++ que se llama **cin,** que implica una entrada de consola, donde vas a ingresar valores. ¿Cómo se usa? std::cin>>variable;

```cpp
#include<iostream>

Windsurf: Refactor | Explain | Generate Function Comment | X
int main(){
    int num=5;
    double num_con_coma = 3.45;
    char letra = 'A';
    bool VoF = true;

    std::cout<<"El numero es: "<<num;
    std::cout<<"ingrese nuevo valor para num: ";
    std::cin>>num;
    std::cout<<"El nuevo valor de num es: "<<num;
    return 0;
}
```

Bueno, venimos bien, compilamos, ejecutamos y entonces…

```
El numero es: 5ingrese nuevo valor para num:
```

Bueno, para solucionar este inconveniente que se juntan los cout, nos conviene poner al final de cada uno <<std::endl;

```cpp
#include<iostream>

Windsurf: Refactor | Explain | Generate Function Comment | X
int main(){
    int num=5;
    double num_con_coma = 3.45;
    char letra = 'A';
    bool VoF = true;

    std::cout<<"El numero es: "<<num<<std::endl;
    std::cout<<"ingrese nuevo valor para num: "<<std::endl;
    std::cin>>num;
    std::cout<<"El nuevo valor de num es: "<<num<<std::endl;
    return 0;
}
```

```
El numero es: 5
ingrese nuevo valor para num:
```

```
El numero es: 5
ingrese nuevo valor para num:
9
El nuevo valor de num es: 9
```

Ahora, la posta es que, escribir std:: cada 2x3 es una cagada MAL, así que vamos a usar un pequeño truquito. Debajo de nuestro #include<iostream>, vamos a poner un pequeño comandito para tener que ahorrarnos cada std.

using namespace std;

```cpp
#include<iostream>

using namespace std;

Windsurf: Refactor | Explain | Generate Function Comment | X
int main(){
    int num=5;
    double num_con_coma = 3.45;
    char letra = 'A';
    bool VoF = true;

    cout<<"El numero es: "<<num<<endl;
    cout<<"ingrese nuevo valor para num: "<<endl;
    cin>>num;
    cout<<"El nuevo valor de num es: "<<num<<endl;
    return 0;
}
```

Y listo. Solucionado. Ahora nos queda más bello y hermoso.

Ahora, bueno, sí, la posta es que tampoco podemos hacer casi nada con esto. Es por esto que los lenguajes de programación en general nos proveen de las siguientes operaciones para hacer: suma (+), resta (-), multiplicación (*), división (/) y módulo (%).

```cpp
#include<iostream>

using namespace std;

Windsurf: Refactor | Explain | Generate Function Comment | X
int main(){
    int num1, num2;
    cout<<"Ingresar dos valores: ";
    cin>>num1>>num2;
    cout<<num1+num2<<endl;
    cout<<num1-num2<<endl;
    cout<<num1*num2<<endl;
    cout<<num1/num2<<endl; //ojo con la división por 0!
    cout<<num1%num2<<endl;
    return 0;
}
```

```cpp
    int num1, num2;
    cout<<"Ingresar dos valores: ";
    cin>>num1>>num2;
    cout<<num1+num2<<endl;
    cout<<num1-num2<<endl;
    cout<<num1*num2<<endl;
    cout<<num1/num2<<endl; //ojo con la división por 0!
    cout<<num1%num2<<endl;
    return 0;
}
```

```
Ingresar dos valores: 10 4
14
6
40
2
2

Process returned 0 (0x0)   ex
Press any key to continue.
```

El signo de porcentaje lo que hace es tomarte el resto de la división. En este caso, como se observa, 10/4 es 2, sí, pero queda 2 suelto, por lo que siendo num1 = 10 y num2 = 4, num1%num2 = 2

Dato de color: el // y lo que escribí delante, es lo que se denomina comentario. Todo lo que esté escrito en esa línea, no lo detecta el programa. Es útil para recordarnos qué hace una función.

Condicionales y formas de comparar primitivas

Bueno, el truco de la programación es la comparación. Nos sirve para un montón de cosas, acá es donde se toman decisiones. Supongamos que tenemos que ir a votar *(sí, acabo de robarle el ejemplo a la profe, ¿y qué?)* Solamente pueden votar personas mayores a 16 años. Entonces, ¿Qué sucede si la persona tiene 15? Directamente no tiene posibilidad para votar. En cambio, el de 16 sí.

Las formas de comparar son las siguientes:

>(mayor)

<(menor)

>=(mayor o igual)

<=(menor o igual)

==(igual)

!=(distinto de)

Para estructurar un condicional hay que utilizar la palabra reservada if() y dentro de los paréntesis hay que escribir nuestra condición. If pregunta por si se cumple la condición interna, y normalmente viene acompañado de un else, que dice “si no se cumple, haz esto”

Vista por Visual Studio Code:

```
#include<iostream>

using namespace std;

Windsurf: Refactor | Explain | Generate Function Comment | X
int main(){
    int edad;
    cout<<"Ingresar la edad: ";
    cin>>edad;
    if(edad >= 16){
        cout<<"Pasa a votar amigo!!"<<endl;
    }
    else{
        cout<<"No podés votar"<<endl;
    }
    return 0;
}
```

Vista por Codeblocks:

```
#include<iostream>

using namespace std;

int main(){
    int edad;
    cout<<"Ingresar la edad: ";
    cin>>edad;
    if(edad>=16){
        cout<<"Pasa a votar amigo!!"<<endl;
    }
    else{
        cout<<"No podés votar"<<endl;
    }
    return 0;
}
```

```
Ingresar la edad: 17
Pasa a votar amigo!!
```

```
Ingresar la edad: 15
No podes votar
```

```
Ingresar la edad: 16
Pasa a votar amigo!!
```