

# C++ vs C

Los códigos de C++ están copiados de la documentación de PRO1

# Primer programa: suma

```
#include <iostream>
using namespace std;
// This program reads two
// numbers and
// writes their sum
int main() {
    int x, y;
    cin >> x >> y;
    int s = x + y;
    cout << s << endl;
}
```

```
#include <stdio.h>
#include <string.h>
#define STDOUT 1
//This program receives two
//numbers //and writes their sum
int main(int argc,char *argv[])
{
    int x,y;
    char buff[128];
    x=atoi(argv[1]);
    y=atoi(argv[2]);
    int s=x+y;
    sprintf(buff,"%d\n",s);
    write(STDOUT,buff,strlen(buff));
}
```

# Como se utiliza

> suma  
3 4  
7

Return



Return



> suma  
5 6  
11

Return



Return



> suma 3 4  
7

Return



> suma 5 6  
11

Return





# Comparativa

- o Los includes llevan .h (hay que mirar cuales hacen falta para cada función)
  - o man strlen, man sprintf, etc
- o En C++ se leen los datos de la entrada std (lo haremos al final del curso)
- o En C para entrada de datos sencillos (argumentos del programa sencillos) se reciben como parámetros del main
- o La entrada/salida de datos hay que procesarla en C explícitamente
  - o Los parámetros siempre son strings → pasarlo a int
  - o La salida, si es la consola, hay que pasarla a string

# Decompose\_time

```
#include <iostream>
using namespace std;
// This program reads a natural number
that represents an amount
// of time in seconds and writes the
decomposition in hours,
// minutes and seconds
int main() {
    int N;
    cin >> N;
    int h = N / 3600;
    int m = (N % 3600) / 60;
    int s = N % 60;
    cout << h << " hours, " << m << " minutes
and "
<< s << " seconds" << endl;
}
```

```
#include <stdio.h>
#include <string.h>
#define STDOUT 1
// This program receives a natural number that
represents an amount
// of time in seconds and writes the
decomposition in hours,
// minutes and seconds
int main(int argc, char *argv[]) {
    int N;
    N=atoi(argv[1]);
    int h = N / 3600;
    int m = (N % 3600) / 60;
    int s = N % 60;
    char buff[128];
    sprintf(buff,"%d hours, %d minutes and %d
seconds\n",h,m,s);
    write(STDOUT,buff,strlen(buff));
}
```

# Tipos datos

o `int x,i,j;`

o Arithmetic

operators: +, -, \*, /,  
%

o `char a,b,c;`

o `bool A;`

o `string`

o `int x,i,j;`

o Arithmetic

operators: +, -, \*, /,  
%

o `char a,b,c;`

~~o `bool A;`~~ // no existe

~~o `String`~~ // No existe

No se pueden aplicar operadores básicos de string en C, hay que usar:

- `strlen` :para calcular la longitud “usada” de un string,  
(es diferente del tamaño)
- `strcmp`: para comparar dos strings



# Tipos datos

- o Conversión explícita tipos

- o `char(i), int('a')ç`

- o Visibilidad

- o Vectores

- o `vector<type>`  
`name(n);`

- o `vector<int> S(n);`

- o `int x=S[0];`

- o Conversión explícita tipos

- o `(char) i, (int)'a'`

- o Visibilidad (igual)

- o Vectores

- o `tipo name[n];`

- o `int S[n];`

- o `int x=S[0];`

No existe `vector.h` en C, sólo hay operaciones básicas.

Para conocer el tamaño en BYTES de cualquier variable tenemos la función `sizeof`

# Ejemplo: Min value of a vector

```
// Pre: A is a non-empty vector
// Post: returns the min value of
the vector
int minimum(const vector<int>&
A) {
    int n = A.size();
    int m = A[0]; // visits A[0]
    // loop to visit A[1..n-1]
    for (int i = 1; i < n; ++i) {
        if (A[i] < m) m = A[i];
    }
    return m;
}
```

```
// Pre: A is a non-empty vector
// Post: returns the min value of
the vector
int minimum(int *A, int size_A) {
    int n = size_A
    int m = A[0]; // visits A[0]
    // loop to visit A[1..n-1]
    for (int i = 1; i < n; ++i) {
        if (A[i] < m) m = A[i];
    }
    return m;
}
```



# Constantes

o `const tipo_dato  
nombre_variable=valor;`

o `#define nombre_variable  
valor`

# Operadores

- o Asignación =
- o Comparación >,<,>=,<=, ==
- o Logicas: && (AND) , || (OR)
  - o AND y OR no existen en C !!
  - o Para utilizar “booleanos” en C se usan normalmente enteros

# Estructuras condicionales

```
if (condicion)
    statements
else if (condicion)
    statements
else
    statements
```

Si hay más de un statement, se pone entre llaves {}



# Estructuras iterativas

- While → igual
  - While (condition) statements
- For → igual
  - for(S\_init;condition;S\_iter) S\_body;