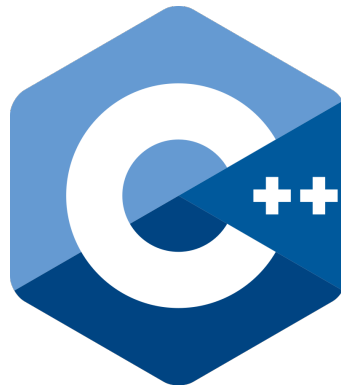




# Metodología de la Programación

DGIM

Curso 2021/2022



## Guion de prácticas

*Paso de argumentos a la función main()*

*Marzo de 2021*



# Índice

<b>1. Objetivos</b>	<b>5</b>
<b>2. Paso de parámetros a main() desde la línea de órdenes</b>	<b>5</b>
2.1. Validando el paso de argumentos a main() . . . . .	6
2.2. Un caso especial . . . . .	8
<b>3. Videotutoriales</b>	<b>8</b>





## 1. Objetivos

- Practicar el paso de parámetros a **main()** desde la línea de órdenes

## 2. Paso de parámetros a main() desde la línea de órdenes

Para poder pasar parámetros de cualquier tipo y en cualquier número a un programa desde la línea de comandos, la función **main** debe pasar de esta forma

```
int main() {
```

a esta otra

```
int main(int narg, char * args[]) {
```

De esta forma, todos los parámetros indicados en la línea de órdenes pasan a la función **main()** como un vector de cadenas-c, luego, si es necesario, se pueden convertir a otros tipos de datos, por ejemplo numéricos<sup>1</sup>). Cada componente del vector es una cadena-c que contiene el respectivo parámetro de la línea de órdenes, el primer componente del vector de parámetros es el nombre del binario que se ejecuta. El número total de parámetros viene indicado como el primer parámetro de main y debe ser de tipo **int**. Así una llamada como esta

```
$$> mi_binario 10 3.14 Pepe
```

Sería recibida en el main como

```
narg = 4  
args[0] = ``mi_binario``  
args[1] = ``10``  
args[2] = ``3.14``  
args[3] = ``Pepe``
```

---

<sup>1</sup>Manual de referencia de cstdlib para conversión de tipos entre cadenas-c y diferentes tipos numéricos: ([Abrir →](#))



## 2.1. Validando el paso de argumentos a main()

Se deben comprobar cuando los argumentos que se pasan son correctos. Vease el siguiente programa, para el que el número de argumentos es siempre el mismo.

```
#include <iostream>
#include <cstdlib>
using namespace std;

void help() {
    cout << endl << "-n<number1><number2> Positive numbers to calculate product" << endl
    << "-b|h|d|o base-h-exadecimal, d-decimal, o-ctal" << endl;
}

void errorArguments() {
    cerr << "Error in arguments" << endl;
    help();
    exit(1);
}

int main(int narg, char * args[]) {
    int n=-1, m=-1, res;
    char base='0';
    // First. Check number of arguments, if possible
    if (narg != 6)
        errorArguments();
    // Second. Get the arguments
    n = atoi(args[2]);
    m = atoi(args[3]);
    base = args[5][0];
    // Third check for bad values
    if (n<0 || m<0 || (base!='h' && base != 'd' && base != 'o'))
        errorArguments();
    // Fourth. Proceed
    res = n*m;
    cout << "Result=";
    switch (base) {
        case 'h': cout << std::hex << res << endl; break;
        case 'o': cout << std::oct << res << endl; break;
        case 'd': cout << res << endl; break;
    }
    return 0;
}
```

La siguiente versión cuando aparecen parámetros opcionales.

```
#include <iostream>
#include <cstdlib>
using namespace std;

void help() {
    cout << endl << "-n<number1><number2>[-b|h|d|o]" << endl
    << "-n<number1><number2> Positive numbers to calculate product" << endl
    << "-b|h|d|o base-h-exadecimal, d-decimal, o-ctal, default is h" << endl;
}

void errorArguments() {
    cerr << "Error in arguments" << endl;
    help();
    exit(1);
}

int main(int narg, char * args[]) {
    int n=-1, m=-1, res;
    char base='h';
    // First. Check number of arguments, if possible
    if (narg != 6 && narg != 4)
        errorArguments();
    // Second. Get the arguments
    n = atoi(args[2]);
    m = atoi(args[3]);
    if (narg == 6)
        base = args[5][0];
    // Third check for bad values
    if (n<0 || m<0 || (base!='h' && base != 'd' && base != 'o'))
        errorArguments();
    // Fourth. Proceed
    res = n*m;
    cout << "Result=";
    switch (base) {
        case 'h': cout << std::hex << res << endl; break;
        case 'o': cout << std::oct << res << endl; break;
        case 'd': cout << res << endl; break;
    }
    return 0;
}
```



A continuación cuando aparecen en cualquier orden.

```
#include <iostream>
#include <cstdlib>
using namespace std;

void help() {
    cout << endl << " _OPTIONS_ " << endl
         << "-n<number1><number2> _Positive numbers to calculate product" << endl
         << "-b_h|d|o_base_h-exadecimal, _d-ecimal, _o-ctal, _default-is_h" << endl;
}

void errorArguments() {
    cerr << "Error in arguments" << endl;
    help();
    exit(1);
}

int main(int narg, char * args[]) {
    int n=-1, m=-1, res;
    char base='h';
    // First. Check number of arguments, if possible
    if (narg != 6)
        errorArguments();
    // Second. Get the arguments
    for (int i=1; i<narg; i++) {
        string sarg=args[i++];
        if (sarg == "-n") {
            n = atoi(args[i++]);
            m = atoi(args[i++]);
        } else if (sarg == "-b") {
            base = args[i++][0];
        } else
            errorArguments();
    }
    // Third check for bad values
    if (n<0 || m<0 || (base!='h' && base != 'd' && base != 'o'))
        errorArguments();
    // Fourth. Proceed
    res = n*m;
    cout << "Result=_";
    switch (base) {
        case 'h': cout << std::hex << res << endl; break;
        case 'o': cout << std::oct << res << endl; break;
        case 'd': cout << res << endl; break;
    }
    return 0;
}
```

Y, finalmente, cuando aparecen un número no determinado de parámetros.

```
#include <iostream>
#include <cstdlib>
using namespace std;

void help() {
    cout << endl << "-b_h|d|o _n<number1><number2>[_<number1><number2>...]" << endl
         << "-n<number1><number2> _Positive numbers to calculate product" << endl
         << "-b_h|d|o_base_h-exadecimal, _d-ecimal, _o-ctal, _default-is_h" << endl;
}

void errorArguments() {
    cerr << "Error in arguments" << endl;
    help();
    exit(1);
}

int main(int narg, char * args[]) {
    int n=-1, m=-1, res;
    char base='h';
    // First. Check number of arguments, if possible
    if (narg < 3 || narg%2 == 1)
        errorArguments();
    // Second. Get the arguments
    base = args[2][0];

    // Third check for bad values
    if (base!='h' && base != 'd' && base != 'o')
        errorArguments();
    // Fourth. Proceed while processing remaining arguments
    for (int i=4; i<narg; i+=2) {
        n = atoi(args[i]);
        m = atoi(args[i+1]);
        // Fifth check arguments while processing them
        if (n<0 || m<0)
            errorArguments();
        res = n*m;
        cout << std::dec << n << "x"<<m<< "=_";
        switch (base) {
            case 'h': cout << std::hex << res << endl; break;
            case 'o': cout << std::oct << res << endl; break;
            case 'd': cout << std::dec << res << endl; break;
        }
    }
    return 0;
}
```

## 2.2. Un caso especial

A veces se pide leer (escribir) los datos desde teclado o desde un fichero indistintamente según un parámetro de main().

```
#include <iostream>
#include <cstdlib>
#include <fstream>
using namespace std;

void help() {
    cout << endl << "[_OPTIONS_]" << endl
    << "-i <filename> _Input_data_from_this_file ,_otherwise_from_keyboard" << endl
    << "-b <h|d|o> _base_h-exadecimal ,_d-decimal ,_o-octal ,_default_is_h" << endl;
}

void errorArguments() {
    cerr << "Error in arguments" << endl;
    help();
    exit(1);
}

int main(int narg, char * args[]) {
    int n=-1, m=-1, res;
    char base='h';
    string filename="";
    ifstream ifile;
    istream *input=&cin;

    // First. Check number of arguments, if possible

    // Second. Get the arguments
    for (int i=1; i<narg; i++) {
        string sarg=args[i++];
        if (sarg == "-i") {
            filename = args[i++];
            ifile.open(filename.c_str());
            if (!ifile) {
                cerr << "Error opening file " << filename << endl;
                exit(1);
            }
            input = &ifile;
        } else if (sarg=="-b") {
            base = args[i++][0];
        } else
            errorArguments();
    }


    // Third check for bad values
    if (base!='h' && base != 'd' && base != 'o')
        errorArguments();


    // Fourth. Proceed
    (*input) >> n >> m;
    if (input->eof()) {
        cerr << "Error reading data from " << filename << endl;
        exit(1);
    }

    if (n<0 || m<0)
        errorArguments();

    res = n*m;
    cout << "Result=";
    switch (base) {
        case 'h': cout << std::hex << res << endl; break;
        case 'o': cout << std::oct << res << endl; break;
        case 'd': cout << std::dec << res << endl; break;
    }
    if (input != &cin)
        ifile.close();
    return 0;
}
```

## 3. Videotutoriales

1. Videotutorial 1.  [\(Abrir →\)](#)

2. Videotutorial 2.  [\(Abrir →\)](#)