UNIVERSIDAD DE GRANADA

ETSIIT Escuela Técnica Superior de Ingenierías Informática y de Telecomunicación

DECSAI

Metodología de la Programación

Curso 2025/2026

Guion de prácticas

Paso de argumentos a la función main()

Silvia Acid, Andrés Cano, Luis Castillo
Dpto. Ciencias de la Computación e Inteligencia Artificial
Universidad de Granada

Índice

1. Objetivos

- Practicar el paso de parámetros a la función **main()** desde la línea de órdenes al invocar un programa.

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 cambiar la declaración en su forma básica

```
int main() {
```

a esta otra declaración con dos argumentos. El primero un entero, el segundo un array de cadenas

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

o bien esta otra, equivalente con un entero y un array de cadenas

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

Supongamos, en primer lugar que tenemos un fuente que contiene una de las dos últimas declaraciones de main indicadas. Supongamos que disponemos del ejecutable `mi_binario` a través del flujo de trabajo. Podemos, hacer una llamada al ejecuble de la forma:

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

Con ello se consigue que, todos los parámetros de la línea de comandos `"mi_binario"` `"10"` `"3.14"` y `"Pepe"` se pasen a la función **main()** en el argumento `args`, vector de cadenas-c. Por otro lado, `narg`, contiene la longitud del vector de cadenas, a saber, 4, pues también se incluye el nombre del ejecutable.
Como se hace notar, todos ellos son cadenas, en ocasiones será necesario convertir algunos de ellos en otros tipos de datos, por ejemplo convertirlos en enteros o reales[1]). La llamada anterior se recibe en los parámetros de main() como

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

[1]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 debe comprobar si los argumentos que se pasan son correctos. Véase 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-ecimal,␣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-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 && 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;) {
        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 indeterminado de parámetros, hay que considerar si hay un mínimo como el caso que se muestra a continuación.

```
#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-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';
    string filename="";
    ifstream ifile;
    istream *input=&cin;

                        // First. Check number of arguments, if possible

                        // Second. Get the arguments
    for (int i=1; i<narg;) {
        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. Paso de argumentos. Configuración para depuración en VSCode

En el guion, *el depurador*, puede encontrar cómo proceder a la configuración de VSCode para ejecutar un programa en modo depuración con argumentos de main.