

Fondamenti di Informatica T-1

modulo 2

Contenuti

- Parametri passati a linea di comando

Overview

- Il risultato del processo di compilazione/linking è un file eseguibile (un programma)
- Abbiamo visto almeno tre modi per mettere in “esecuzione” un programma
 - tramite l’IDE, durante la fase di debugging
 - tramite l’interfaccia grafica, come conseguenza di un evento particolare e specifico
 - tramite l’interprete dei comandi (“shell” in Unix-like/MacOs, “command prompt” in Windows)
 - (esistono anche altri modi...)

Interprete dei comandi

- Esegue ciclicamente:
 - legge il comando specificato da standard input
 - interpreta il comando specificato
 - la prima stringa come il nome di un file eseguibile
 - le stringhe successive come parametri/argomenti da passare al file specificato
 - mette in esecuzione tale file, passando gli argomenti
 - rimane bloccato in attesa della terminazione del programma

- Esempio (MS-DOS/Windows Command Prompt):

```
myprog <arg1> <arg2> ... <argN>  
c: \> copy    f1.txt  f2.txt  
c: \> xyz     12      accipicchia "str con spazio"
```

Interprete dei comandi

- Esempio (MS-DOS):

```
myprog <arg1> <arg2> ... <argN>
```

```
c:\> copy f1.txt f2.txt
```

```
c:\> xyz 12 accipicchia "str con spazio"
```

- NOTA: viene effettuato un "parsing" dei parametri, cioè:

- i parametri vengono riconosciuti come stringhe
- ogni spazio o tab è inteso come un separatore tra un parametro e il successivo

- Come si può passare un parametro che contenga anche spazi?

- Uso dei doppi apici
- In generale, è preferibile **EVITARE** parametri contenenti spazi...

Parametri nel main(...)

- In C gli argomenti sono passati come parametri della funzione main.
- Prototipo:

```
int main(int argc, char* argv[])  
{ ... }
```
- `argc` è il numero di argomenti passati; vale sempre almeno 1
- `argv` è un vettore di stringhe, ogni stringa un argomento
 - `argv[0]` è il nome del file in esecuzione; `argv[0]` esiste sempre
 - `argv[1] ... argv[argc-1]` sono gli argomenti

Esempio

Stampa di tutti gli argomenti

```
#include <stdio.h>

int main(int argc, char * argv[]) {
    int i;
    for (i=0; i<argc; i++)
        printf("%s\n", argv[i]);
    return 0;
}
```

Esercizio:

- Implementare e compilare il programma di cui sopra
- Eseguire il programma dall'interprete dei comandi, passando un elenco di parametri a piacere

Parametri nel main(...)

- Attenzione! Tutti i parametri sono passati come stringhe!!!
- Se un parametro rappresenta un tipo diverso dalla stringa, è necessario convertirlo!!!

- Diversi modi per convertire una stringa in altri tipi
 - `sscanf(...)`
 - `int atoi(char * s)`
 - `long atol(char * s)`
 - `double atof(char * s)`

Esercizio 1

(Parametri del main)

Una calcolatrice

- Realizzare un programma che effettui le quattro operazioni aritmetiche di base su numeri reali
- Gli operandi e l'operatore sono specificati tramite argomenti a linea di comando, secondo la seguente sintassi:
 - `-op1 operando1` specifica il primo operando
 - `-op2 operando2` specifica il secondo operando
 - `-op operatore` specifica l'operatore ("+", "-", "*", "/")
- I parametri speciali `-op1`, `-op2`, `-op` devono essere sempre seguiti da un altro argomento
- I parametri possono comparire in qualsiasi ordine

Esercizio 1 – Soluzione

(Parametri del main)

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define OK 0
#define WRONG_PARAM_NUMBER -1
#define WRONG_OPERATOR -2
#define WRONG_OPTIONS -3

double compute(char op, double num1, double num2);
void handleError(int resultCode);

...
```

Esercizio 1 – Soluzione

(Parametri del main)

```
...
int main(int argc, char **argv) {
    int resultCode;
    int i;
    char op;
    double num1, num2;
    double result;

    // test if the number of parameters is correct
    if (argc!=7)
        resultCode = WRONG_PARAM_NUMBER;
    else {
        resultCode = OK;
        ...
    }
}
```

Esercizio 1 – Soluzione

(Parametri del main)

```
i=1;
while (i<argc && resultCode==OK) {
    if (strcmp("-op",argv[i]) == 0) {
        i++;
        if (i>=argc || strlen(argv[i])!=1)
            resultCode = WRONG_OPERATOR;
        else op = argv[i][0];
    }
    else if (strcmp("-op1", argv[i])==0) {
        i++;
        if (i<argc)
            num1 = atof(argv[i]);
    }
    else if (strcmp("-op2", argv[i])==0) {
        i++;
        if (i<argc)
            num2 = atof(argv[i]);
    }
    else
        resultCode = WRONG_OPTIONS;
    i++;
}
}
```

Esercizio 1 – Soluzione

(Parametri del main)

...

```
if (resultCode == OK) {
    result = compute(op, num1, num2);
    printf("Result: %lf\n", result);
}
else
    handleError(resultCode);

return resultCode;
} // end main
```

Esercizio 1 – Soluzione

(Parametri del main)

```
double compute(char op, double num1, double num2) {
    double result;
    switch (op) {
        case '+': result = num1 + num2; break;
        case '-': result = num1 - num2; break;
        case '/': result = num1 / num2; break;
        case '*': result = num1 * num2; break;
        default: result = 0; // cosa fare???
    }
    return result;
}
```

Esercizio 1 – Soluzione

(Parametri del main)

```
void handleError(int resultCode) {
    switch (resultCode) {
        case WRONG_PARAM_NUMBER:
            printf("Numero di parametri sbagliato!\n"); break;
        case WRONG_OPERATOR:
            printf("L'operatore non viene riconosciuto...\n");
            break;
        case WRONG_OPTIONS:
            printf("Le opzioni specificate sono errate!\n");
            break;
        default:
            printf("Errore sconosciuto!\n");
    }
    return;
}
```