Università degli Studi di Ferrara

DE: Department of Engineering Ferrara

Esercitazioni di

FONDAMENTI DI INFORMATICA MODULO B

ESERCITAZIONE 1

Azzolini Damiano: damiano.azzolini@unife.it
Bertagnon Alessandro: alessandro.bertagnon@unife.it

Passaggio di argomenti al programma

Quando si invoca un programma è possibile passare degli argomenti ad esso durante la chiamata.

Gli argomenti ricevuti sono di tipo stringa (array di char) quindi l'insieme di argomenti sarà contenuto in una matrice di char (array di stringhe).

Il programma deve poter leggere questi argomenti per poterli usare.

Modificare la funzione main in modo tale da considerare gli argomenti passati. Per fare ciò il main deve prendere in ingresso due parametri:

```
main(int argc, char *argv[])
```

Passaggio di argomenti al programma

```
main(int argc, char *argv[])
```

- **`int argc`** è un parametro di tipo **intero** che rappresenta il numero di argomenti passati al programma da linea di comando. Anche il nome del programma è considerato un argomento quindi argc vale sempre almeno 1.

- **`char *argv[]`** è un puntatore a un puntatore a carattere, ovvero un array di stringhe. Gli argomenti nel vettore di stringhe sono memorizzati nell'ordine con cui sono stati inseriti dall'utente sulla linea di comando.
  `argv[0]` contiene il nome del programma stesso.

Passaggio di argomenti al programma

prog.exe arg1 arg2 … argN (Windows)
./prog arg1 arg2 … argN (Linux)

argc vale N+1

argv[0] = "prog"

argv[1] = "arg1"

argv[2] = "arg2"

…

argv[N] = "argN"

argv[argc] = NULL

Esercizio 1. Passaggio di argomenti

Realizzare un programma C che, dato come argomento il nome di un file (di testo), ne visualizzi il contenuto a pagine (20 righe alla volta). Dopo la visualizzazione di ciascuna riga chiedere all'utente se vuole visualizzare le successive 20 (o le restanti nel file se il file ha meno di 20 righe non ancora lette) o fermarsi.

Se si utilizza un **percorso relativo**, il file col quale si vuole interagire (in questo caso leggere) deve trovarsi:

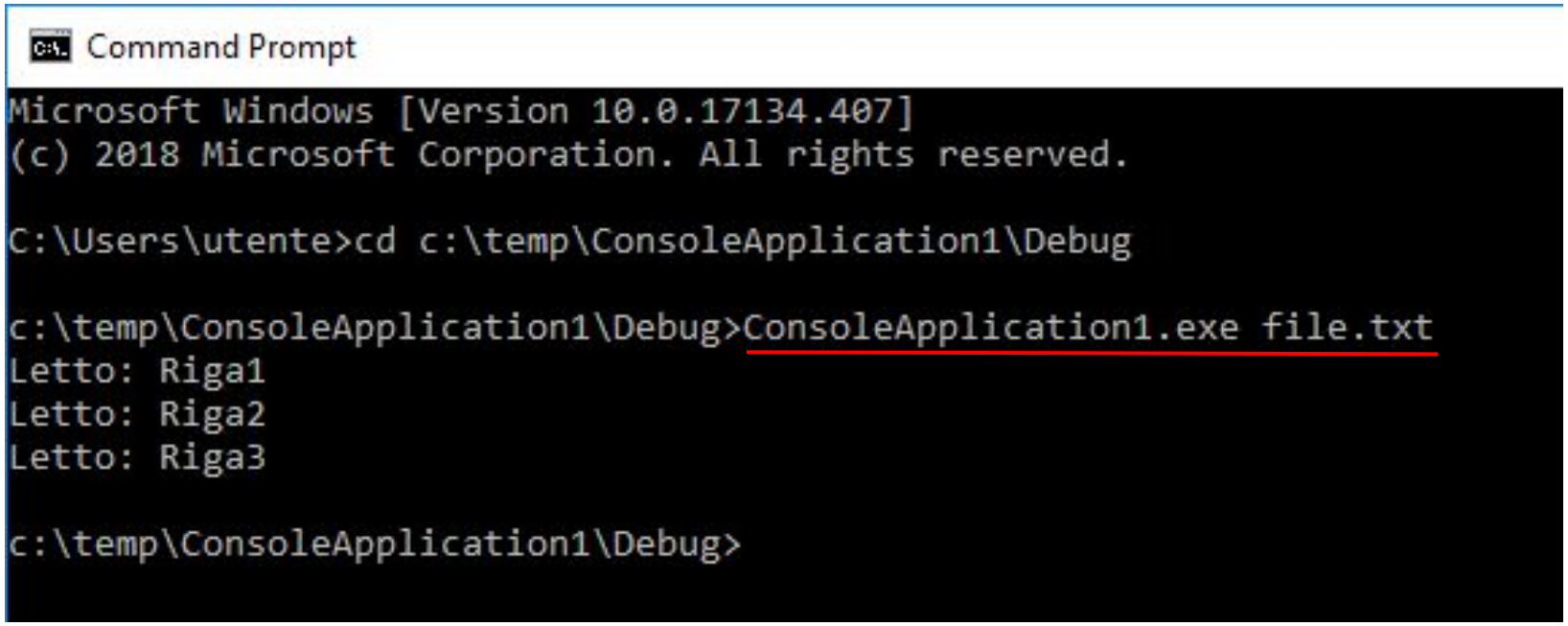
- nella stessa cartella dove si trovano i file .c e .h del progetto se si esegue il programma da Visual Studio.
- nella stessa cartella che contiene il file eseguibile <nome_progetto>.exe (solitamente la sottocartella Debug) se si esegue il programma da terminale.

Alternativamente è possibile utilizzare un **percorso assoluto** e il file può trovarsi in altre posizioni all'interno del file system.

Esercizio 1. Testare il programma

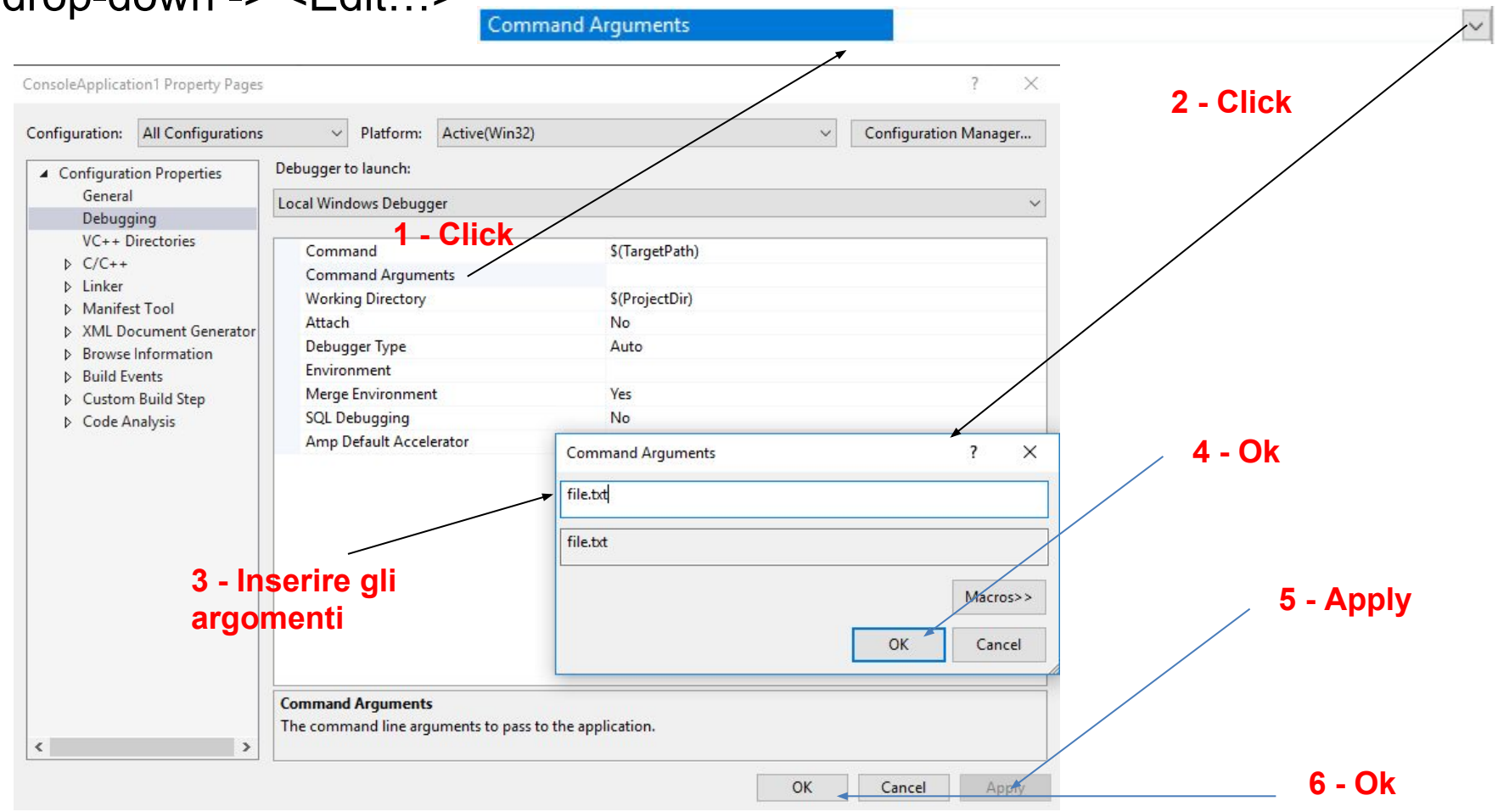
Per testare il programma bisogna passare il nome del file come argomento. Abbiamo due possibilità:

1. Eseguire da terminale il programma



Esercizio 1. Testare il programma

2. Settare l'argomento in Visual Studio. Project -> <nome_progetto> Properties -> Debugging -> Command Arguments -> Click sull'icona di drop-down -> <Edit…>



Esercizio 2. Funzioni come parametri

Si realizzi un programma che calcola l'integrale di una funzione **f : R -> R** in un intervallo dato [a,b] tramite la regola del rettangolo.

Definire una funzione che restituisca il valore dell'integrale, con la seguente interfaccia:

```
double rettangoli(double (*func)(double),
                  double a, double b, int N);
```

dove **func** è la funzione integranda, **a** e **b** gli estremi dell'intervallo e **N** il numero di sottointervalli.

Esercizio 2. Funzioni come parametri

Viene indicato di seguito come specificare il parametro formale di tipo funzione nella dichiarazione della funzione "**rettangoli**":

```
double (*func) (double)
```

(1) (2) (3)

(1) = **tipo** del parametro di ritorno della funzione func

(2) = **nome** formale della funzione referenziata (puntatore)

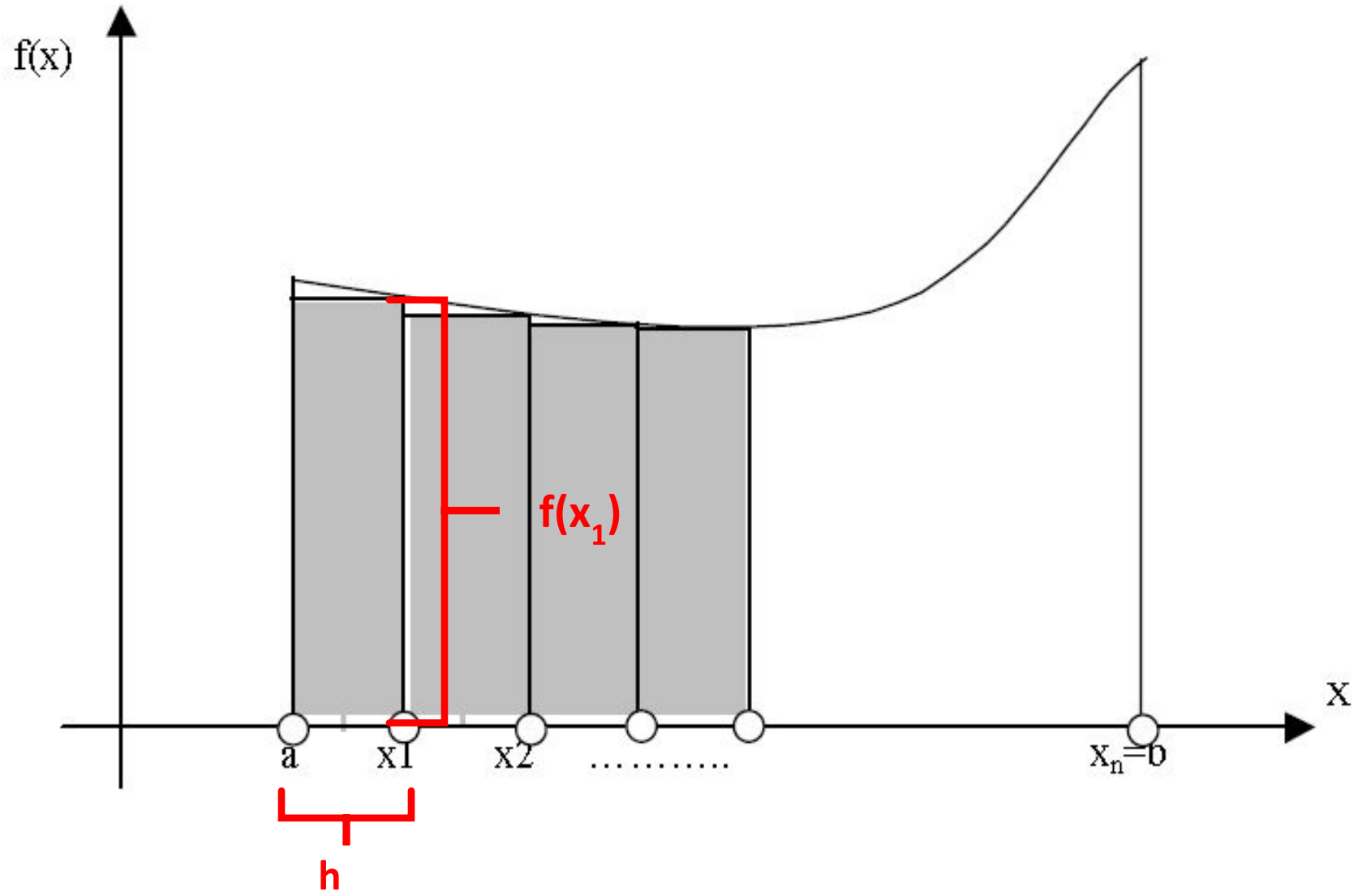
(3) = **lista dei tipi** di parametri di ingresso della funzione func

Esercizio 2. Funzioni come parametri

Il **main** deve eseguire le seguenti operazioni:

1. Chiedere all'utente di inserire da tastiera gli estremi dell'intervallo di integrazione (`double a, double b`) e il numero di sottointervalli (`int N`).
2. Chiamare la funzione **rettangoli** passando come parametri:
   - una funzione chiamata **quadra.** La funzione **quadra** prende come parametro un numero (`double`) e restituisce il suo quadrato (`double`);
   - i due estremi dell'intervallo d'integrazione inseriti dall'utente;
   - il numero di sottointervalli in cui suddividere l'intervallo di integrazione;
3. Stampare a video il risultato della funzione **rettangoli** e ripetere il punto 2 sostituendo la funzione **quadra** con una funzione **cubo** che calcola il cubo del valore passato come parametro.

Esercizio 2. Regola dei rettangoli



Esercizio 2. Funzioni come parametri

L'algoritmo da implementare nella funzione **rettangoli** è il seguente:

- Calcolare la dimensione di ciascun sottointervallo (indicata con `h` nella figura sulla slide precedente) suddividendo l'intervallo ricevuto come parametro in `N` parti uguali come richiesto.
- Per ogni sottointervallo, calcolare l'area del rettangolo associato con la formula **base x altezza** dove:
  - **base** è la dimensione del sottointervallo (calcolata al punto precedente)
  - **altezza** è il valore della funzione `func` calcolata nel punto corrente (`f(x1)` nella figura sulla slide precedente)
- Sommare le aree di tutti i rettangoli costruiti sui vari sottointervalli e restituire il valore al chiamante.

Esercizio 3. Variabili globali

Si realizzi un ***componente dotato di stato interno*** che rappresenta un ***contatore***, con operazioni:

- `void reset( void )`
- `void inc( int k )`
- `int getValue( void )`

Il main, che utilizza il contatore, deve in successione:

- resettarlo
- acquisire da tastiera una sequenza di interi positivi (terminata da 0) e usarli come incremento del contatore
- stampare lo stato finale
- *La variabile contatore deve essere **globale**, locale al file contenente le funzioni del contatore*
- Strutturare il programma su più file

Esercizio 4. Variabili statiche

Si realizzi un ***componente dotato di stato interno*** che rappresenta un ***contatore*** (variabile globale con scope limitato), con operazioni:

- `void reset( void )`
- `void inc( int k )`
- `int getValue( void )`

Il main, che utilizza il contatore, deve in successione:

- resettarlo
- acquisire da tastiera una sequenza di interi positivi (terminata da 0) e usarli come incremento del contatore.
- stampare lo stato finale
- *La variabile contatore deve essere **statica**, locale al file contenente le funzioni del contatore*
- Strutturare il programma su più file

Esercizio 4. Variabili statiche

Una volta concluso si può anche…

- Verificare che lo *scope* della variabile sia solo il file contatore.c (“usare” la variabile nel main -> errore di compilazione: variabile non definita)