Ingresso e Uscita
Variabili e costanti intere

Riepilogo iostream

- Ogni file sorgente che contenga riferimenti ad oggetti della libreria di ingresso/uscita deve contenere le direttive

  #include <iostream>
  using namespace std;

  - Devono precedere il primo punto in cui viene utilizzato uno stream di ingresso/uscita

Riepilogo primo esercizio

```
#include <iostream>

using namespace std;

main()

{

  cout<<"Ciao mondo!\n" ;

}
```

Riepilogo secondo esercizio

```
#include <iostream>

using namespace std;

main()

{

  cout<<"Ciao mondo!"<<endl ;

}
```

Riepilogo compilazione

- Sintassi più semplice per generare un programma eseguibile da un file sorgente:
- *g++ nome_sorgente.cc*
  - Assegna un nome predefinito al programma eseguibile (a.out)
- *g++ -o nome_eseguibile nome_sorgente.cc*
  - Permette di scegliere il nome del programma eseguibile

Operatore di uscita <<

- Scrittura formattata su *cout*
- *cout<<obj1<<obj2<<...<<endl ;*
- Il generico oggetto da stampare può anche essere una *variabile* o una *costante* con nome

Esercizio 1/2

- Scrivere un programma in cui si definisce una variabile intera e se ne stampa il valore sullo schermo

Esercizio 2/2

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

main()

{

  int i = 10 ;

  cout<<i;

}
```

Domanda

- Cosa stampa su *stdout* il seguente programma?

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

main()

{

  int i = 10 ;

  cout<<"i";

}
```

Risposta

- Stampa `i`
- Non stampa `10`!
- Come mai?

Risposta

- Perché ciò che racchiudiamo tra doppi apici è una costante stringa
- Se passiamo una costante stringa al *cout*, chiediamo di stampare esattamente i caratteri contenuti nella costante stringa
- Per far stampare invece il contenuto di una variabile dobbiamo passare **solo** il nome della variabile (non racchiuso tra doppi apici)

Esercizio 1/2

- Scrivere un programma in cui si definisce una variabile intera e se ne stampa il valore sullo schermo col seguente formato:

Il valore della variabile è 10.

- E si va a capo

Esercizio 2/2

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

main()

{

  int i = 10 ;

    cout<<"Il  valore  della  variabile  è
  "<<i<<"."<<endl ;

}
```

Una soluzione alternativa

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

main()

{

  int i = 10 ;

  cout<<"Il valore della variabile è "<<i ;

  cout<<"."<<endl ;

}
```

Ritorno valore finale 1/3

- Spesso si scrivono delle applicazioni complesse all'interno delle quali vengono invocati diversi programmi
- Si pone spesso il bisogno, tra le altre cose, di controllare se l'esecuzione di un programma è andata a buon fine
- I processi passano un valore numerico intero al sistema operativo quando terminano
- Una delle convenzioni per controllare l'esito dell'esecuzione di un programma è proprio controllare tale valore di ritorno

Ritorno valore finale 2/3

- Vedrete in Programmazione II come si legge tale valore, per ora dobbiamo solo sapere che molti compilatori C/C++ pretendono che nel programma si dichiari esplicitamente quale valore si deve ritornare alla terminazione
- La funzione *main* può (in effetti dovrebbe sempre) essere dichiarata ritornare un valore di tipo intero

Ritorno valore finale 3/3

- E deve terminare con una istruzione **`return`**, con la quale si ritorna un calore intero
- Il valore di ritorno della funzione *main* è il valore ritornato dal processo stesso

Esempio

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

int main()

{

  int i = 10 ;

  cout<<"Il valore della variabile è "<<i ;

  cout<<"."<<endl;

  return 0 ;

}
```

così si dice che la funzione ritorna un valore intero

0 è il tipico valore usato per comunicare che è andato tutt

Esercizio 1/2

- Scrivere un programma in cui:
  - si definisce una variabile intera
  - se ne stampa il valore sullo schermo
  - si cambia il valore della variabile (non si definisce una nuova variabile)
  - si stampa il (nuovo) valore sullo schermo

Esercizio 2/2

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

main()

{

  int i = 10 ;

  cout<<”Il valore della variabile è "<<i<<endl ;

  i = 12 ;

  cout<<”Il nuovo valore è "<<i<<endl ;

}
```

Esercizio 1/2

- Scrivere un programma in cui si definisce una costante intera e se ne stampa il valore sullo schermo col seguente formato:

Il valore della costante è 10.

- E si va a capo

Esercizio 2/2

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

main()

{

  const int i = 10 ;

  cout<<"Il valore è “<<i<<”.”<<endl ;

}
```

Lettura dallo *stdin*

- Operatore di ingresso >> applicato ad un oggetto di tipo *istream*
- *Esempio: cin>>nome_variabile ;*
- Legge i caratteri in ingresso dallo *standard input* (abbreviato *stdin*)
- Li interpreta in base al tipo della variabile
- Assegna il valore letto alla variabile di nome *nome_variabile*

Esempio di interpretazione

- *cin>>a;*
- La variabile *a* è di tipo int
- Se l'utente scrive 23 e va a capo, Si leggono i caratteri 2, 3 e *\n*
- Vengono interpretati come le due cifre decimali del numero 23
- Il numero 23 viene memorizzato nella variabile *a*

Esercizio 1/2

- Si scriva un programma che legge un valore intero da tastiera e lo stampa su *stdout*

Esercizio 2/2

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

main()

{

  int i ;

  cin>>i ;

  cout<<"Il valore inserito è "<<i<<endl ;

}
```

Ingresso inconsistente

- Cosa accade se la sequenza di caratteri letta non rappresenta alcun numero in notazione decimale?
- Il **valore del secondo argomento** (ossia della variabile) rimane **invariato** (non avviene alcuna memorizzazione)
- Le **successive letture falliranno**
- Vedremo in futuro come resettare lo stato dello stream per non fare più fallire le successive letture

Esercizio 1/2

- Miglioriamo l'esercizio precedente
- Vogliamo stampare anche un messaggio di richiesta del numero da inserire:

Inserisci un valore intero: 13

Il valore inserito è: 13

Nota

- L'operatore di ingresso >> applicato al *cin* **non scrive sullo standard output (stdout)**

Esercizio 2/2

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

main()

{

  int i ;

  cout<<”Inserisci un valore intero " ;

  cin>>i ;

  cout<<"Il valore inserito è “<<i<<endl ;

}
```

Esercizio 1/3

- Scrivere un programma che legge in ingresso due valori interi e stampa il risultato della moltiplicazione tra i due numeri

  Inserisci il primo numero: 10

  Inserisci il secondo numero: 20

  *10 * 20 = 200*

Sfida

- Calcolare, usando il vostro programma, il valore di
  19312 * 7284

Esercizio 2/3

```
#include <iostream>
using namespace std;
main()
{
  int i, j, ris ;

  cout<<"Inserisci il primo numero " ;
  cin>>i ;
  cout<<"Inserisci il secondo numero " ;
  cin>>j ;

  ris = i * j ;
  cout<<i<<"*"<<j<<" = "<<ris<<endl ;
}
```

Esercizio 3/3

```
#include <iostream>   /* Soluzione alternativa: */
using namespace std;  /* senza variabile di     */
main()                /* appoggio               */
{
  int i, j ;

  cout<<"Inserisci il primo numero " ;
  cin>>i ;
  cout<<"Inserisci il secondo numero " ;
  cin>>j ;

  cout<<i<<"*"<<j<<" = "<<i*j<<endl ;
}
```

Esercizio 1/3

- Scrivere un programma che legge in ingresso due valori interi e stampa sia il risultato della divisione intera tra i due numeri che il resto della divisione stessa

  Inserisci il primo numero: 5

  Inserisci il secondo numero: 2

  5 / 2 = 2 con resto 1

Sfida

- Calcolare, usando il vostro programma, il valore della divisione intera e del resto di
  19312 / 7284

Esercizio 2/3

```
#include <iostream>
using namespace std;
main()
{
  int i, j, div, resto ;

  cout<<"Inserisci il primo numero " ;
  cin>>i ;
  cout<<"Inserisci il secondo numero " ;
  cin>>j ;

  div = i / j ;
  resto = i % j ;
    cout<<i<<" / "<<j<<" = "<<div<<" con  resto
     "<<resto<<endl ;
}
```

Esercizio 3/3

```
#include <iostream>        /* Soluzione alternativa      */
using namespace std;       /* senza variabili di         */
main()                     /* appoggio                   */
{
  int i, j ;

  cout<<"Inserisci il primo numero " ;
  cin>>i ;
  cout<<"Inserisci il secondo numero " ;
  cin>>j ;

   cout<<i<<"  /  "<<j<<"  =  "<<i/j<<"  con  resto
    "<<i%j<<endl ;
}
```

Esercizio 1/4

- Scrivere un programma che legge in ingresso due valori interi e li memorizza in due variabili, quindi scambia il contenuto delle variabili e lo stampa sullo schermo

Inserisci il valore di i: 2

Inserisci il valore di j: 3

Dopo lo scambio: i = 3, j = 2

Esercizio 2/4

- Primo esercizio un po' più difficile
- Riflettiamo un po' sul problema: se assegniamo **i** a **j** abbiamo perso il valore di **i** e viceversa ...
- Sfruttiamo questo esercizio per capire in generale come **acquisire la mentalità giusta**

Sviluppo di una soluzione

- Un buon ordine con cui arrivare a risolvere un problema nuovo di cui non si conosce la soluzione è il seguente:

1) Analizzare il problema finché non si è sicuri di aver capito chiaramente tutti gli aspetti e le implicazioni
2) Cercare di farsi venire un'idea che sembri buona per risolvere il problema (o almeno per partire)
3) Provare a definire l'algoritmo e controllarlo per capire se è corretto (eventualmente modificarlo)
4) Quando si è sicuri dell'algoritmo, partire con la codifica
5) Collaudare il programma per verificare che faccia veramente quello che deve

Commenti

- A meno di problemi molto molto semplici, non rispettare il precedente ordine porta quasi sempre a risultati mediocri o pessimi
- Il tipico errore che si commette è quello di incominciare a scrivere il programma prima di aver chiaro l'algoritmo (se non addirittura prima di aver chiaro il problema stesso)
- Il passo 2 può essere quello più critico, perché richiede un atto creativo in mancanza del quale non si sa da dove partire
  - La capacità di compiere con successo tale passo si accresce con l'esercizio e l'esperienza

Esempio errore 1/2

- Vediamo un esempio delle conseguenze di una superficiale o errata analisi del problema
- Nell'esercizio che dobbiamo risolvere non capiamo chiaramente che bisogna scambiare il contenuto di due variabili
- E ci indirizziamo verso una soluzione errata in cui semplicemente ristampiamo il contenuto delle due variabili in ordine invertito, come mostrato nella prossima slide

Esempio errore 2/2

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

int main()
{
  int i, j ;

  cout<<"Inserisci il valore di i " ;
  cin>>i ;
  cout<<”Inserisci il valore di j " ;
   cin>>j ;
 cout<<”i: “<<j<<”j: “<<i<<endl;
  return 0;
}
```

Proviamo ...

- ... quindi ad applicare correttamente le precedenti fasi dello sviluppo al nostro problema dello scambio di variabili
- Se abbiamo finalmente chiaro il problema, allora adesso ci vuole innanzitutto un'idea ...

Idea

- Se memorizziamo il valore di una delle due variabili, per esempio di **i**, in una terza variabile d'appoggio, allora, quando assegnamo il valore di **j** ad **i**, non abbiamo perso il valore di **i** !
- Possiamo quindi assegnare a **j** il (precedente) valore di **i**, salvato nella variabile di appoggio

Algoritmo

1) Assegnare il valore contenuto in **i** ad una variabile d'appoggio **app**
2) Assegnare il valore contenuto in **j** ad **i**
3) Assegnare a **j** il valore contenuto nella variabile di appoggio (uguale al valore che **i** aveva prima del passo 2)

Se l'algoritmo ci è chiaro e ci sembra corretto, non ci resta che provare ad implementarlo ...

Programma 1/2

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

int main()
{
  int i, j ;

  cout<<"Inserisci il valore di i " ;
  cin>>i ;
  cout<<”Inserisci il valore di j " ;
  cin>>j ;
```

Programma 2/2

```
  int appoggio = i ;
  i = j ;
  j = appoggio ;

  cout<<"Dopo lo scambio: i = "<<i
        <<", j = "<<j ;
  return 0 ;
}
```

Esercizio più difficile

- Scrivere un programma che legge in ingresso due valori interi e li memorizza in due variabili, quindi **scambia il contenuto** delle variabili e lo stampa sullo schermo
- Ma senza utilizzare nessuna variabile d'appoggio!
- *scambia_senza_appoggio.cc*

Compiti per casa 1

- Scrivere i seguenti programmi
  - rispettando le fasi di sviluppo precedentemente viste
  - senza utilizzare istruzioni di controllo di flusso (niente istruzioni condizionali ed iterative, ma solo esecuzione sequenziale)
  - facendo uso solo di variabili di tipo *int* e dei relativi operatori
    - `+, -, *, /, %, abs()`
    - *Data una variabile intera* `i`*, l'operatore* `abs(i)` *ritorna il valore assoluto di* `i`

Nota

- Su alcune macchine, per usare la funzione **`abs()`** bisogna aggiungere la direttiva

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

  all'inizio del programma

Compiti per casa 2

- Scrivere un programma che legge in ingresso un numero intero, lo interpreta come un tempo espresso in secondi, e lo stampa in minuti e secondi (*da_sec_a_min_sec.cc*)

Tempo in secondi? 67

Equivalgono a 1 min, 7 sec

Compiti per casa 3

- Scrivere un programma che legge in ingresso due numeri, li interpreta come un tempo espresso in minuti e secondi, e lo stampa in secondi (*da_min_sec_a_sec.cc*)
  - Attenzione: per semplicità assumiamo come valido anche un ingresso in cui il secondo numero sia maggiore di 59

Minuti ? 3

Secondi ? 78

Equivalgono a 258 secondi

Compiti per casa 4

- Scrivere un programma che legge in ingresso quattro numeri, li interpreta come due tempi espressi in minuti e secondi, e stampa la differenza tra i due tempi, espressa in secondi (*soluzione non fornita*)
  - Attenzione: per semplicità assumiamo come valido anche un ingresso in cui il secondo numero sia maggiore di 59

Minuti e secondo primo tempo ? 3 45

Minuti e secondi secondo tempo ? 5 36

Differenza: 111

Compiti per casa 5

- Scrivere un programma che legge in ingresso quattro numeri, li interpreta come due tempi espressi in minuti e secondi, e stampa la differenza tra i due tempi, di nuovo espressa in minuti e secondi (*soluzione non fornita*)
  - Attenzione: per semplicità assumiamo come valido anche un ingresso in cui il secondo numero sia maggiore di 59

Minuti e secondo primo tempo ? 3 45

Minuti e secondi secondo tempo ? 5 36

Differenza: 1 51

Compiti per casa 6

- Scrivere un programma che legge in ingresso un numero intero e stampa 0 se il numero è pari, 1 altrimenti (*0_se_pari.cc*)

  Inserisci un numero intero: 23

  1

- Scrivere un programma che legge in ingresso un numero intero e stampa 1 se il numero è pari, 0 altrimenti (*1_se_pari.cc*)

Compiti per casa 7

- Scrivere un programma che legge in ingresso due numeri interi positivi, poi stampa 0 se il primo è multiplo dell'altro, 1 altrimenti (*0_se_multiplo.cc*)

Inserisci il primo numero intero positivo: 32

Inserisci il secondo numero intero positivo: 11

1

Compiti per casa 8

- Scrivere un programma che legge in ingresso due numeri interi positivi, poi stampa 1 se il primo è multiplo dell'altro, 0 altrimenti (*1_se_multiplo.cc*)

Inserisci il primo numero intero positivo: 32

Inserisci il secondo numero intero positivo: 11

0

Compiti per casa 9

- Scrivere un programma che legge in ingresso un numero intero diverso da 0, e stampa -1 se è negativo, 1 se è positivo (*1_se_pos-1_se_neg.cc*)

  Inserisci un numero intero: -3

  -1

Compiti per casa 10

- Scrivere un programma che legge in ingresso un numero intero diverso da 0, e stampa 0 se è negativo, 1 se è positivo (*0_se_neg_1_se_pos.cc*)

  Inserisci un numero intero: -3

  0

Compiti per casa 11

- Scrivere un programma che legge in ingresso un numero intero diverso da 0, e stampa 1 se è negativo, 0 se è positivo

  Inserisci un numero intero: -3

  1