

---

Ingresso e Uscita  
Variabili e costanti intere  
Processo risolutivo

# Programma e processo

---

- Definiamo *processo* un programma in esecuzione
- Sotto Linux si può vedere lo stato dei processi per esempio mediante il comando `top`

# Riepilogo ingresso e uscita

---

- **Input/Output (I/O)**
  - Ingresso di informazioni (da elaborare) all'interno di un processo
  - Uscita di informazioni (elaborate) da un processo
- **Esempio:** stampa di informazioni sullo schermo, lettura di valori da tastiera

# Flussi di caratteri 1/2

---

- L'I/O è spesso realizzato facendo entrare nel programma o uscire dal programma un flusso di caratteri
- Flusso di caratteri (stream):
  - successione di righe, ciascuna
    - costituita da zero o più caratteri, e
    - terminata dal carattere speciale *newline* '\n'

# Flussi di caratteri 2/2

---

- Esempio:

```
Rosso di sera buon tempo si spera\n\nChi domanda non fa errori\n
```

- Flusso di caratteri costituito da tre righe, di cui una vuota

# Input/output in C++ 1/2

---

- Il linguaggio C++ non prevede istruzioni per l'ingresso/uscita
- Implementato mediante *oggetti di libreria* chiamati esattamente *stream*
- *stream*: meccanismo generale per far entrare nel programma o uscire dal programma flussi di caratteri
- ...

# Input/output in C++ 2/2

---

- *ostream*: meccanismo per gestire un flusso di caratteri in uscita, output formattato
- *istream*: meccanismo per gestire un flusso di caratteri in ingresso, input formattato
- Per semplicità, d'ora in poi utilizzeremo il termine flusso di caratteri per indicare tanto il flusso stesso che il meccanismo per gestire tale flusso

# *cin, cout, cerr* 1/2

---

- Quando un programma inizia la propria esecuzione, ci sono tre flussi di caratteri già aperti
  - *cin*: flusso standard di ingresso
  - *cout*: flusso standard di uscita
  - *cerr*: flusso standard di uscita per comunicare messaggi di errore

# *cin, cout, cerr* 2/2

---

- Se il programma è invocato da una shell Unix (per chi conosce l'argomento: *senza redirectionamenti*)
  - Lettura da *cin*:
    - Lettura dei caratteri immessi dal terminale in cui è in esecuzione la shell (tastiera)
  - Scrittura su *cout* o su *cerr*:
    - Visualizzazione sul terminale in cui è in esecuzione la shell

# Operatore di uscita << 1/2

---

- Scrittura formattata su *cout*
- Forma più semplice
  - *cout<<stringa ;*
  - ove *stringa* è una sequenza di caratteri delimitata da doppi apici “
  - “*esempio di stringa*”

# Operatore di uscita <<

---

- Scrittura formattata su *cout*
- *cout<<obj1<<obj2<<...<<endl ;*
- Il generico oggetto da stampare può essere una stringa o un manipolatore
  - Ma anche qualcos'altro, come vedremo a breve

# Riepilogo sintassi

---

- 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

# Stampa variabili

---

- Stessa sintassi che si utilizza per stampare ad esempio una stringa
- Solo che stavolta si passa il nome della variabile

# Esercizio 1/2

---

- Scrivere un programma in cui si definisce una variabile intera inizializzata con il valore che si preferisce, e se ne stampa il valore sullo schermo

# Esercizio 2/2

---

```
#include <iostream>  
using namespace std;  
  
main()  
  
{  
  
    int i = 10 ;  
  
    cout<<i;  
  
}
```

# Ritorno valore finale

---

- Un processo ritorna un valore quando termina
- Può essere letto ed utilizzato dal programma che lo ha invocato
  - 0 indica tipicamente che tutto è andato bene
- La funzione *main* può (in effetti dovrebbe sempre) essere dichiarata ritornare un valore di tipo 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 ;
```

```
}
```

# Obbligo di ritorno valore finale

---

- Se non si dichiara il main di tipo `int` e/o non si fa ritornare esplicitamente un valore al *main* (mediante l'istruzione `return`), allora il compilatore può segnalare errori o warning
- In ogni caso, d'ora in poi, per brevità, nei nostri esempi trascureremo questo aspetto

# 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 e' 10.*

- E si va a capo

# Esercizio 2/2

---

```
#include <iostream>  
using namespace std;  
  
main()  
  
{  
  
int i = 10 ;  
  
cout<<"Il valore della variabile e' "  
  <<i<<". "<<endl ;  
  
}
```

# Una soluzione alternativa

---

```
#include <iostream>  
using namespace std;  
  
main()  
  
{  
  
int i = 10 ;  
  
cout<<"Il valore della variabile e' "<<i ;  
  
cout<<". "<<endl ;  
  
}
```

# Esercizio 1/2

---

- Scrivere un programma in cui si definisce una variabile intera e se ne stampa il valore sullo schermo, quindi se ne cambia il valore e si stampa di nuovo il (nuovo) valore sullo schermo
- Utilizzare un numero maggiore di 255 come nuovo valore

# Esercizio 2/2

---

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

```
main()
```

```
{
```

```
    int i = 10 ;
```

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

```
    i = 1229 ;
```

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

```
}
```

# Letture 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 a video
- Provare anche ad immettere un numero maggiore di 255

# 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?
- La lettura **fallisce** e l'oggetto cin entra in stato di errore
- **Le successive letture falliranno**
- Vedremo in futuro come resettare lo stato dello *stream* per non fare più fallire le successive letture

# Fallimento lettura 1/2

---

- Cosa succede quando una lettura fallisce?
- Il risultato dipende dallo standard in uso
- Prima di procedere con la prossima slide tornare sulla lezione 2 per un excursus sull'ultima versione dello standard C++

# Fallimento lettura 2/2

---

- Cosa succede quando una lettura fallisce?
- Standard precedente al C++11
  - Il **valore del secondo argomento** dell'operatore di ingresso rimane **invariato** (non avviene alcuna memorizzazione)
- Standard C++11 o più recenti
  - Si memorizza il valore **0 nel secondo argomento**

# 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*

- 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*

- 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 (sfida nella prossima slide)

*Inserisci il primo numero: 5*

*Inserisci il secondo numero: 2*

*5 / 2 = 2 con resto 1*

- 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 di tipo **int**, 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 ...
- Fermiamoci un attimo e sfruttiamo questo semplice esercizio per fare un importantissimo passo per la nostra preparazione
  - **acquisire la mentalità giusta** per realizzare, partendo da un problema, un programma che lo risolva con successo

<http://linuxday.conoscerelinux.it>

# Sviluppo di una soluzione

---

- Un buon ordine con cui arrivare a risolvere, mediante un programma, un problema nuovo di cui non si conosce la soluzione è il seguente:
  - 1) Riflettere sul problema finché non si è sicuri di aver capito a sufficienza 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 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

# Proviamo ...

---

- ... ad applicare le precedenti fasi dello sviluppo al nostro problema dello scambio di variabili
- Dopo esserci assicurati di aver veramente capito il problema, ci vorrà un'idea ...
- Prima di discutere una possibile idea, provate da soli ad arrivare fino alla scrittura del programma
- Probabilmente vedremo un primissimo esempio delle conseguenze del mancato rispetto delle precedenti fasi

# Soluzione sbagliata 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 ;  
    cout<<"i = "<<j<<" , j = "<<i<<endl ;
```

# Soluzione sbagliata 2/2

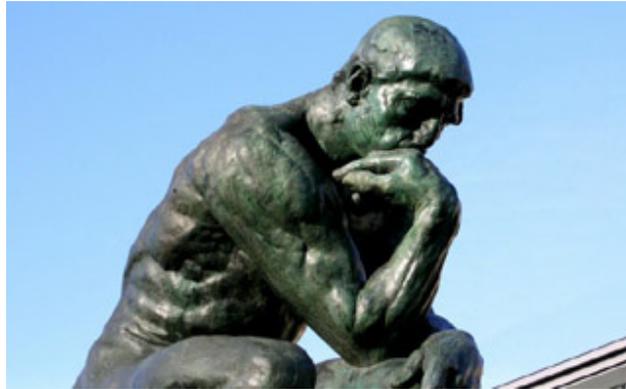
---

- Abbiamo semplicemente stampato il contenuto della variabili in ordine invertito
- Non abbiamo scambiato il contenuto delle variabili come richiesto
- Perché siamo andati così fuori strada?

# Errore al primo passo

---

- Perché non abbiamo eseguito il primo passo con la massima cura possibile: non abbiamo riflettuto abbastanza sul problema
- Se vogliamo, possiamo utilizzare questa lezione per migliorarci molto per il futuro
- Se invece preserviamo questo approccio superficiale, il nostro livello di risolutori di problemi rimarrà abbastanza basso
- Chiarito il primo passo, vediamo assieme un'idea per arrivare ad una soluzione



- 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  $i$*

- In alcuni sistemi, 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*