Esercizi su

Tipo booleano

Operatori relazionali e logici

Istruzioni di scelta: selezione

Esercizi precedenti

- Dubbi?
- Domande?

Tipo booleano

- Scrivere un programma che
  - definisca una variabile di tipo *bool*,
  - le assegni *true* e la stampi
  - successivamente assegni *false* **alla stessa variabile** e la stampi di nuovo
  - *stampa_bool.cc*

Precedenza

- L'operatore << ha *precedenza* maggiore di <, >, >=, ...
- Quindi se si scrive per esempio:
  cout<<2>3 ;
  è come se si fosse scritto:
  (cout<<2)>3 ;
  e si ottiene un errore in compilazione
- Soluzione per l'esempio precedente:
  cout<<(2<3) ;

Espr. logiche semplici 1/2

- Scrivere un programma che:
  - chieda all'utente di inserire due valori interi *a* e *b*;
  - stampi:
    - 1 se $a < b$
    - 0 altrimenti

Espr. logiche semplici 2/2

- Esempio di output:

Inserisci i valori di a e di b: 3 4

Valore di 3 < 4: 1

- *stampa_logica_semplice.cc*

Espressioni logiche composte

- Scrivere un programma che:
  - chieda all'utente di inserire tre valori interi *a*, *b* e *c*;
  - stampi 1 se $a < b$ oppure se $a < c$, 0 altrimenti
- Attenzione di nuovo alle precedenze
- *stampa_logica_composta.cc*

Errore logico 1/2

- Quanto vale l'espressione logica:

  1 < 3 < 2 ?

- Equivale a

  (1 < 3) < 2

- Ossia

  true < 2

Errore logico 2/2

- true è convertito ad 1, quindi
- *1 < 2*
- Quindi: *true* !!!!!!!!
- Problema: abbiamo confuso le regole di valutazione di una formula matematica con quelle di una espressione logica in C/C++

Messaggi di errore g++

- Ogni riga inizia con il nome del file sorgente
- Poi c'è
  - il nome della funzione
  - e/o il numero di riga e colonna

  in cui si è verificato l'errore
- Poi il termine *error* o *warning*
- Infine la descrizione del problema (può proseguire su più righe)

Consigli

- Progettare i programmi (almeno idea ed algoritmo) prima carta e penna (o almeno mentalmente nel caso degli esercizi più semplici)
- Rileggerli mettendosi nei panni
  - del compilatore prima
  - e del computer (esecutore) dopo
- Guardare le soluzioni solo quando si è sicuri di non essere in grado di risolvere l'esercizio da soli

Esercizio per casa

- Scrivere un programma che:
  - chieda all'utente di inserire tre valori interi *a*, *b* ed *x*;
  - stampi 1 se $a \leq x \leq b$, 0 altrimenti
- Scrivere e **controllare** nel modo migliore possibile **tutto** il programma prima della prima compilazione
- *stampa_1_se_in_intervallo.cc*

Esercizio 1/2

- Scrivere un programma che legge un numero intero da *standard input (cin, stdin)* che rappresenta il voto preso in Programmazione I e, se il voto è superiore a 27, stampa

  Programmazione I è veramente uno dei migliori corsi di Informatica!!

  Altrimenti non stampa nulla ed esce.

Esercizio 2/2

```
int main()
{
  int voto ;
  cin>>voto ;
  if (voto > 27)
    cout<<"Viva Programmazione I\n" ;

  return 0 ;
}
```

Esercizio 1/2

- Scrivere un programma che legge il voto preso in Programmazione I e, se il voto è superiore a 27 stampa

  Programmazione I è veramente uno dei migliori corsi di Informatica

  Altrimenti stampa

  Quel def … di docente di quel noioso …

  E variazioni a vostra discrezione …

Esercizio 2/2

```
int main()
{
  int voto ;
  cin>>voto ;
  if (voto > 27)
    cout<<"Viva Programmazione I\n" ;
  else
    cout<<"Quel def..." ;

  return 0 ;
}
```

Esercizio per casa 1/2

- Scrivere un programma che legge un numero intero da *standard input (cin)* e stampa

  Il numero inserito è positivo

  se il numero inserito è positivo. Altrimenti **non stampa nulla** ed esce.

Esercizio per casa 2/2

```
int main()
{
  int i ;
  cin>>i ;
  if (i > 0)
    cout<<"Il numero inserito è positivo\n" ;

  return 0 ;
}
```

Esercizio per casa 1/2

- Scrivere un programma che legge un numero intero da *stdin* e stampa

  Il numero inserito è non negativo

  se il numero inserito è positivo o nullo. Altrimenti stampa

  Il numero inserito è negativo

Esercizio per casa 2/2

```
int main()
{
  int i ;
  cin>>i ;

  if (i >= 0)
    cout<<"Il numero inserito è non negativo\n" ;
  else
    cout<<"Il numero inserito è negativo" ;

  return 0 ;
}
```

Esercizio per casa

- Scrivere un programma che stampi il massimo tra due numeri interi letti dallo *stdin*
- Esempio:

  Inserire i due numeri interi: 21 -3

  Il massimo tra 21 e -3 è 21
- Nota: non è necessario premere invio tra i due numeri
- Soluzione: *stampa_max.cc*

Esercizio 1/3

- Scrivere un programma che definisca ed inizializzi due costanti intere *a* e *b*, poi legga in ingresso un numero intero e scriva un messaggio se il numero non è compreso nell'intervallo *[a, b]*

Inserisci un numero intero: 101

101 non è in [1, 100]

Esercizio 2/3

- **Per puro esercizio**, il messaggio va stampato mediante due istruzioni e non una sola
- Esempio:

```
cout<<n<<” non e' incluso ” ;
cout<<” in [”... ;
```

e non:

```
cout<<n<<” non e' incluso in [”... ;
```

Esercizio 3/3

```
int main()
{
  int i, a = 1, b = 100 ;

  cout<<"Inserisci un numero intero: " ;
  cin>>i ;

  if (i < a || i > b) {
            cout<<i<<" non è in [“ ;
            cout<<a<<”, “<<b<<”]\n”;
   }

  return 0 ;
}
```

Domanda

- La precedente soluzione contiene una imprecisione rispetto alla traccia
  - Provate ad individuarla prima di guardare la risposta nella slide successiva

Risposta

- *a* e *b* sono variabili e non costanti

Esercizio: divisione intera 1/4

- Riprendiamo l'esercizio di calcolo della divisione intera tra due numeri interi
- Con i due numeri interi letti da *stdin*

  Inserisci i due numeri: 5 2

  5 / 2 = 2

Esercizio: divisione intera 2/4

```
int main()
{
  int i, j ;
  cout<<Inserisci due numeri interi: " ;
  cin>>i>>j ;

  cout<<i<<" / "<<j<<"="<<i/j<<endl;

  return 0 ;
}
```

Esercizio: divisione intera 3/4

- Proviamo ad inserire ad esempio 3 e 0

  Inserisci i due numeri: 3 0

  ????????

- Cosa è successo?

Terminazione forzata

- Abbiamo provato a far eseguire al processore una operazione illegale: una divisione per zero
- In questi casi il processore genera una **eccezione hardware**
- Nella fase di avvio iniziale, il sistema operativo associa ad ogni possibile eccezione hardware del codice di gestione dell'eccezione stessa
- Tra le altre cose, il codice eseguito a seguito della nostra eccezione di divisione per zero, **termina forzatamente** il processo che ha generato l'eccezione

Fallimento

- La terminazione forzata è una delle modalità (tra le più evidenti) di **fallimento** di un programma
- Un'altra modalità di fallimento è la restituzione di risultati errati

Alcune cause comuni ...

- ... di **fallimenti** a tempo di esecuzione:
  - Le variabili non sono inizializzate
  - I valori passati alle funzioni o immessi dall'esterno non sono quelli attesi e non si effettuano controlli
  - C'è stato un *overflow*
    - Lo vedremo meglio nella prossima esercitazione

Gestione delle eccezioni

- Spesso è necessario controllare il valore dei parametri passati alle funzioni o dei valori letti da *stdin* o da file e prendere contromisure
  - Segnalare errore ed uscire
  - Modificare i valori per riportarli forzatamente in intervalli validi

Correggiamo ...

- ... l'esercizio sulla divisione intera
  - affinché il programma si accorga del problema e lo segnali senza essere forzatamente terminato dal sistema operativo

Esercizio: divisione intera 4/4

```
int main()
{
  int i, j ;
  cout<<Inserisci due numeri interi: " ;
  cin>>i>>j ;

  if (j == 0)
     cout<<"Attenzione: il divisore è nullo\n" ;
  else
     cout<<i<<" / "<<j<<"="<<i/j<<endl;

  return 0 ;
}
```

Esercizio per casa 1/2

- Scrivere un programma che definisca ed inizializzi due costanti intere *a* e *b*, poi legga in ingresso un numero intero e scriva un messaggio se il numero è compreso nell'intervallo *[a, b]*

  Inserisci un numero intero: 5

  5 è in [1, 100]

- **Non utilizzare l'operatore ||**

Esercizio per casa 2/2

```
int main()
{
  const int i, a = 1, b = 100 ;

  cout<<"Inserisci un numero intero: " ;
  cin>>i ;

  if (i >= a && i <= b)
    cout<<i<<" è in [“<<a<<”, “<<b<<”]\n”;

  return 0 ;
}
```

Domanda

- La precedente soluzione contiene un errore
  - Provate ad individuarlo prima di guardare la risposta nella slide successiva

Risposta

- *i* è erroneamente definita costante

Esercizio per casa 1/2

- Scrivere un programma che definisca ed inizializzi due costanti intere *a* e *b*, poi legga in ingresso un numero intero e scriva un messaggio se il numero è compreso nell'intervallo *[a, b]*

  Inserisci un numero intero: 5

  5 è in [1, 100]

- **Non utilizzare l'operatore &&**

Esercizio per casa 2/2

```
int main()
{
  const int a = 1, b = 100 ;

  cout<<"Inserisci un numero intero: " ;
  int i ; cin>>i ;

 if (!(i < a || i > b) )
    cout<<i<<" è in ["<<a<<", "<<b<<"]\n";

  return 0 ;
}
```

Indentazione 1/3

```
int main()
{
    int i ;
    ...
    if (...)
        cout<<"messaggio" ;

    return 0 ;
}
```

Indentazione 2/3

- Se C1 e' la colonna rispetto alla quale sono allineate
  - l'intestazione di una funzione,
  - una istruzione condizionale o una istruzione iterativa,
  - o l'inizio di una istruzione composta

Indentazione 3/3

- Tutte le istruzioni appartenenti al loro corpo, devono essere allineate a partire da una colonna C2,
- spostata a destra di un numero prefissato di spazi rispetto a C1.

Controversia ...

- https://stackoverflow.blog/2017/06/15/developers-use-spaces-make-money-use-tabs/

Indentazione 3/3

- Tutte le istruzioni appartenenti al loro corpo, devono essere allineate a partire da una colonna C2,
- spostata a destra di un numero prefissato di spazi rispetto a C1.

Esercizio per casa 1/3

- Scrivere un programma che definisca ed inizializzi due costanti intere *a* e *b*, poi legga in ingresso due numeri interi: *controllo* ed *i*
- L'intero *controllo* si utilizza per controllare il comportamento del programma
- In particolare,
  - se *controllo* è diverso da *0*,
    - il programma scrive un messaggio se *i* non è compreso nell'intervallo *[a, b]*
  - altrimenti segnala che non è stato effettuato nessun controllo

Esercizio per casa 2/3

- Esempi:

Inserisci il valore per controllo: 1

Inserisci un numero intero: 0

0 non è in [1, 100]

Inserisci il valore per controllo: 0

Inserisci un numero intero: 0

Nessun controllo effettuato

Esercizio per casa 3/3

```
int main()
{
  const int a = 1, b = 100 ;
  int i, controllo ;

  cout<<”Inserisci un numero intero: " ;
  cin>>i ;

  cin>>controllo ;

  if (controllo != 0 && (i < a || i > b) )
    cout<<i<<" non è in [“<<a<<”, “<<b<<”]\n”;
  else if (controllo == 0)
    cout<<”Nessun controllo effettuato”<<endl ;

  return 0 ;
}
```

Problema 1/2

- Dobbiamo scaricare **legalmente** un film non coperto da alcun **copyright**
- Ma per motivi puramente personali vogliamo effettuare l'operazione nel più breve tempo possibile ...
- Ci sono due server da cui scaricarlo
  - Il primo va ad una velocità di 200 KB/sec, ma solo per i primi 60 minuti di trasferimento, dopodiché la velocità scende a 50 KB/sec
  - Il secondo va costantemente ad una velocità di 100 KB/sec

Problema 2/2

- Il film è grande 644840 KB
- Quale dei due server ci conviene utilizzare ?

Risposta

- Sicuramente il primo, col quale ce la caviamo in poco meno di 54 minuti

E se ...

- ... il film occupava
  - 956982 KB,oppure
  - 1458874 KB ?
- Per essere in grado rispondere alla domanda in tutti i casi vi tocca inventare un algoritmo appropriato ...
- Incominciamo dal buttar giù un'idea di base per risolvere il problema
- Per semplicità arrotondiamo i tempi al secondo e per difetto
- Inoltre per leggibilità il nostro programma dovrà stampare il tempo minimo di trasferimento in minuti e secondi

Idea!

- Se col primo server ce la facciamo in meno di un'ora, sicuramente è quello il server che ci darà il tempo minimo
- Altrimenti, per il primo server, dobbiamo calcolare 1) quanta parte del film rimarrebbe ancora da trasferire dopo un'ora e 2) quanto si impiegherebbe per trasferire tale parte residua; infine dobbiamo sommare tale tempo residuo ad un'ora per ottenere il tempo totale
- Dobbiamo poi controllare se il tempo totale col primo server precedentemente calcolato è maggiore o minore di quello che impiegheremmo col secondo server

Altra possibilità

- Calcolare a priori quale sarebbe la lunghezza del film tale che entrambi i server impiegherebbero lo stesso tempo
  - Scegliere il primo server se la lunghezza effettiva è inferiore a tale valore
  - Scegliere il secondo server se la lunghezza effettiva è superiore a tale valore

Possibile algoritmo

- Calcolo quanti KB trasferisce il primo server in un'ora
- Se trasferisce più KB della lunghezza del film, allora seleziono il primo server come quello da utilizzare e calcolo il tempo necessario dividendo la lunghezza del film per la velocità iniziale del server (200 KB/sec)
- Se le cose non stanno così, calcolo i tempi sui due server nel modo seguente:
  - Sottraggo alla lunghezza del film i KB trasferiti dal primo server in un'ora per scoprire la lunghezza residua, calcolo quindi il tempo necessario a scaricare la parte residua a 50 KB/sec e lo sommo ai 60 minuti iniziali
  - Calcolo il tempo per il secondo server dividendo la lunghezza del film per 100 KB/sec
  - Seleziono come server da utilizzare quello che da il tempo minore ed assumo come tempo minimo tale tempo
- Stampo il server che conviene utilizzare ed il tempo minimo necessario per il trasferimento

Implementazione

- Implementiamo prima l'algoritmo utilizzando solo le nostre conoscenze attuali (minime) sulla qualità del codice
- Poi, attraverso i suggerimenti nelle prossime slide, proviamo a migliorare il codice
- Infine confrontiamoci con la soluzione suggerita

(Ri)cominciamo bene 1/3

- Approfittiamo di questo esercizio per applicare delle prime regole di buona programmazione
  - Modifichiamo il programma per rispettare tali regole
- **NON INSERIAMO NUMERI SENZA NOME NEL NOSTRO PROGRAMMA!**
  - Sono i cosiddetti **numeri magici**, perché appaiono magicamente in un programma
  - Chi legge il programma di norma non capisce da dove spuntano
- Al contrario, utilizziamo sempre costanti con nome
  - Il nome della costante fa capire a chi legge di cosa si tratta
  - Se dobbiamo cambiare un numero utilizzato più volte nel programma, basta cambiare valore alla costante una sola volta

Cominciamo bene 2/3

- **USIAMO NOMI SIGNIFICATIVI PER LE VARIABILI**
  - L'idea è che leggendo il nome di una variabile si dovrebbe capire cosa contiene e qual è il suo obiettivo
  - Il compromesso di norma è tra la lunghezza e la chiarezza del nome
    - Se mettiamo troppe informazioni nel nome, quest'ultimo diventerà molto lungo, rendendo faticosa la scrittura/modifica del programma

Cominciamo bene 3/3

- **NON REPLICHIAMO IL CODICE!**
  - Se in due punti del programma scriviamo due pezzi di codice (quasi) identici, cerchiamo il modo di ripensare la logica di quelle parti del programma in maniera tale da scrivere quel pezzo di codice una sola volta
    - Rendendolo magari un po' più generale per gestire in un sol colpo i due casi gestiti dai due pezzi di codice di partenza
  - Eliminare la replicazione non conviene solo nel caso in cui farlo comporterebbe complicazioni nel codice maggiori della replicazione stessa
- La replicazione rende il codice più lungo e spesso più difficile da capire, infine aumenta la probabilità di commettere errori e di dimenticare di correggerli in tutti i punti in cui le stesse istruzioni sono replicate

Soluzione

- *scegli_server.cc*

I tempi minimi erano ...

- 956982 KB → 138 min 59 sec, usando il primo server
- 1458874 KB → 243 min 8 sec, usando il secondo server

Compiti per casa

- *multiplo.cc*
- *tre_ordinati.cc*