
Laboratorio di programmazione

Lezione V

Tatiana Zolo

`zolo@cs.unipr.it`

SWITCH

- Metodo alternativo ad istruzioni `if-else` annidate per scegliere fra un insieme di opzioni mutuamente esclusive.

```
switch (espressione) {  
  case C1:  
    S1;  
    break;  
  case C2:  
    S2;  
    break;  
  ...  
  default:  
    Sk;  
    break;  
}
```

- **Attenzione:** l'esecuzione *comincia* dall'etichetta `case` corrispondente allo `switch` e continua attraverso le successive \implies usare `break`!

SWITCH

- Se due o più valori devono essere gestiti dalla stessa sequenza di azioni, si può scrivere:

```
switch (espressione) {  
  case C1:  
  case C2:  
  case C3:  
    S1;  
    break;  
  ...  
  default:  
    Sk;  
    break;  
}
```

- es.mesi.cc.

STRINGHE

Definizione

- **Costanti stringa**: insiemi di caratteri racchiusi tra apici doppi.
- **Array di `char`**: per trattare come una sola unità un insieme di caratteri.

```
char str1[10];  
char str2[] = "come stai?";
```

`str1` è un array di 10 caratteri; `str2` ha un numero di elementi determinato dalla quantità di caratteri presenti tra doppi apici più uno: il carattere *null* (`\0`) che chiude la stringa.

`\0` \implies permette di usare le stringhe senza conoscerne a priori la dimensione.

es. `prova_str.cc`.

STRINGHE

Lettura di una stringa da tastiera

→ `cin`: attenzione ai caratteri di spaziatura (spazi, tab e newline)!

→ `getline()`: sintassi

```
cin.getline(string, size, delimiter = '\n');
```

- `string` è l'array di caratteri in cui sono posti i caratteri letti;
- `size` è il max numero di caratteri da leggere dallo stream;
- `delimiter` indica un carattere delimitatore (per default è l'invio).

Continua a leggere caratteri finché sono stati letti `size-1` caratteri oppure non si è incontrato il carattere delimitatore.

`gcount()`: restituisce il numero di caratteri letto dall'ultima chiamata di `getline()`.

→ es. `prova_str.cc`, `prova_str1.cc`, `prova_str2.cc`.

STRINGHE

Funzioni di libreria: includere il file header associato

`#include <string>` (attenzione `<cstring>` è un'altra cosa!).

Tra le funzioni più comuni:

- `strcpy(r, s)`: copia il contenuto della stringa `s` in `r`.
- `strcat(r, s)`: aggiunge `s` alla fine di `r`.
- `strcmp(r, s)`: confronta `s` e `r`. Se sono uguali restituisce 0; se `s` è maggiore di `r` lessicograficamente restituisce un valore positivo, altrimenti un valore negativo.
- `strlen(s)`: restituisce la lunghezza della stringa `s`.
- `es.copia.cc`.

ESERCIZI

1. **switch**: considerare due numeri complessi rappresentati come array di 2 elementi (prima cella per la parte reale e seconda per la parte immaginaria). Immettere da tastiera parte reale e parte immaginaria di ciascun numero ed il simbolo di operazione binaria che si vuole compiere con i due complessi. Stabilire quale operazione fare e visualizzarne il risultato (`operaz_complex.cc`).
2. **stringhe**: stabilire una password e memorizzarla in una stringa. Scrivere un programma che chieda all'utente di inserire la password e che ne confermi o meno l'esattezza. Permettere al piu' tre tentativi prima di negargli la possibilita' di riprovare (`password.cc`).

ESERCIZI

- 3. stringhe:** conversione stringhe → numeri interi. Scrivere un programma che legga una sequenza di caratteri terminata da "a capo" rappresentante una costante intera (eventualmente con segno '-'). Se la sequenza è sintatticamente corretta memorizzare il corrispondente numero intero in una variabile di tipo 'int' e visualizzarlo; altrimenti visualizzare un opportuno messaggio (`str_num.cc`). Es. L'utente inserisce i caratteri -, 1, 5 e il programma deve riconoscere che si tratta dell'intero -15.
- 4. stringhe:** conversione stringhe → numeri interi. Ripetere l'esercizio precedente dove, invece di considerare una sequenza di caratteri singoli, si considera in input una stringa rappresentante una costante intera (`str_num2.cc`). Es. L'utente inserisce la stringa ' '1082 ' ' e il programma deve riconoscere che si tratta dell'intero 1082.