Corso “Programmazione 1”
Capitolo 01: Concetti Elementari

Docente:	Roberto Sebastiani - roberto.sebastiani@unitn.it
Esercitatori:	Mario Passamani - mario.passamani@unitn.it
	Alessandro Tomasi - alessandro.tomasi@unitn.it

C.D.L.:	Informatica (INF)
	Ing. Informatica, delle Comunicazioni ed Elettronica (ICE)
	Studenti con numero di matricola pari
A.A.:	2019-2020
Luogo:	DISI, Università di Trento
URL:	disi.unitn.it/rseba/DIDATTICA/prog1_2020/

Ultimo aggiornamento: 30 ottobre 2019, ore 11:29

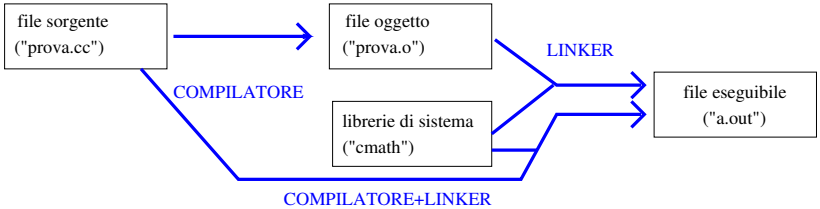
Scrittura di un programma

- scrittura di uno o più file di testo, chiamato/i sorgente
  - Scritto in un linguaggio di programmazione (es C++)
  - Stampabile, comprensibile ad un essere umano
- creazione e modifica tramite uno strumento chiamato editor
  - Es. Emacs sotto linux

in C++ tipicamente l'estensione del nome è "`.cc`", "`.cpp`" (o "`.h`")

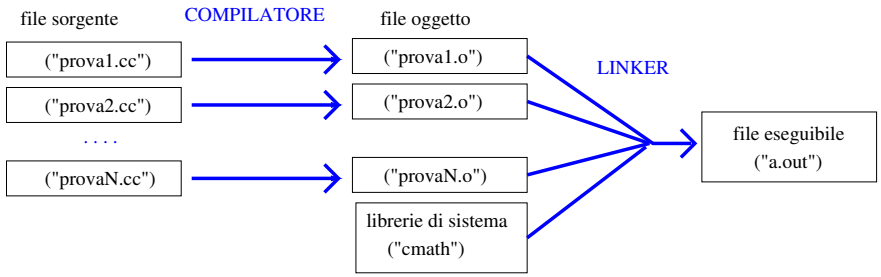
- Es. "`prova.cc`", "`prova.cpp`"

Compilazione (un unico file)



- file sorgente tradotto in un file oggetto dal compilatore
  - Es: `g++ -c prova.cc`
  - Imp: file oggetto illeggibile ad un essere umano e non stampabile!
- file oggetto collegato (linked) a librerie di sistema dal linker, generano un file eseguibile (default `a.out`, opzione `-o <nome>`)
  - Es: `g++ prova.o`
  - Es: `g++ prova.o -o prova`
  - File incomprensibili agli umani, ma eseguibili da una macchina
- Compilazione e linking possibile in un unica istruzione
  - Es: `g++ prova.cc`

Compilazione (su più files)



- file sorgente tradotti nei rispettivi file oggetto uno alla volta
- file oggetto collegato (linked) a librerie di sistema dal linker, generano un file eseguibile (default `a.out`)
  - Es: `g++ prova1.o prova2.o ... provaN.o`
- Compilazione e linking possibile in un unica istruzione
  - Es: `g++ prova1.cc prova2.cc ... provaN.cc`

ESEMPI

- esempio di com'è fatto un programma:
  { `ESEMPI_BASE/esempio_fattoriale.cc` }
- con compilazione separata:
  { `ESEMPI_BASE/fact.cc` }
  { `ESEMPI_BASE/esempio_fattoriale2.cc` }

Un “template” di programma C++

Componente “comune” a tutti i programmi di questo corso::
{ ESEMPI_BASE/template.cc }

```
using namespace std; // Definizione dello spazio
                     // dei nomi

#include <iostream>  // chiamata a libreria

int main ()          // funzione principale
{

return 0;            // chiusura funzione principale
}
```

Un programma C++ elementare

Output di una stringa predefinita:
{ ESEMPI_BASE/ciao.cc }

```
using namespace std;
#include <iostream> // chiamata a libreria

int main ()          // funzione principale
{
  cout << "Ciao a tutti\n";  // istruzione di output
return 0;
}
```

Variante con “endl”:
{ ESEMPI_BASE/ciao2.cc }

Elementi di base: sequenze di parole

Le parole di un programma C++ possono essere costituite da:

- lettere:
  `_ a b c d e f g h i j k l m n o p q r s t u v w x y z A B C D E F G H I J K L M N O P Q R S T U V W X Y Z`
- cifre (digit):
  `0 1 2 3 4 5 6 7 8 9`
- caratteri speciali:

```
!  ^ \& \* ( ) - + = { } | ~
[ ] \ ; ' : " < > ? , . /
```

Elementi di base: spaziature

Le spaziature possono essere costituite da

- spazi: “ ”
- tabulazioni: “ ”
- caratteri di nuova riga: “
”
- commenti:
  - racchiusi fra “`/*`” e “`*/`” (come in C):

    ```
    /* Questo commento e`
       su piu` righe      */
    ```

  - tra “`//`” e fine riga:

    ```
    // Questo commento e` su una sola riga
    ```

Identificatori (Identifiers)

- le entità di un programma C++ devono avere dei nomi che ne consentano l'individuazione
- nel caso più semplice i nomi sono degli identificatori liberamente scelti
- un identificatore è una parola iniziante con una lettera

Nota

- il carattere "_" è una lettera
- il C++ distingue tra maiuscole e minuscole (si dice che è case sensitive).
  - Es: `Fact` è diverso da `fact`

Parole Chiave (keywords)

Le parole chiave del C++ sono un insieme di simboli il cui significato è stabilito dal linguaggio e non può essere ridefinito in quanto il loro uso è riservato.

Principali parole chiavi del C++

```
asm auto break case catch char class const continue
default delete do double else enum extern
float for friend goto if inline int long new
operator private protected public register return
short signed sizeof static struct switch
template this throw try typedef union virtual void
volatile while
```

Le Espressioni Letterali

- Denotano valori costanti, spesso sono chiamati solo “letterali” (o “costanti senza nome”, o “valori costanti”)
- Possono essere:
  - costanti carattere (denotano singoli caratteri)
  - costanti stringa (denotano sequenze di caratteri)
  - costanti numeriche intere
  - costanti numeriche reali

Rappresentazione costanti carattere

- una costante carattere viene rappresentata racchiudendo il corrispondente carattere fra apici: per esempio la costante 'a' rappresenta il carattere 'a'
- i caratteri di controllo vengono rappresentati da combinazioni speciali dette sequenze di escape che iniziano con una barra invertita (backslash '\')

Sequenze di Escape

Nome	Abbreviazione	Sequenza di Escape
nuova riga	NL(LF)	\n
tabulazione orizzontale	HT	\t
tabulazione verticale	VT	\v
spazio indietro	BS	\b
ritorno carrello	CR	\r
avanzamento modulo	FF	\f
segnale acustico	BEL	\a
barra invertita	\	\\
apice	'	\'
virgolette	"	\"

Esempio:

{ ESEMPI_BASE/escape.cc }

Rappresentazione di Costanti Stringa

- una costante stringa è una lista di caratteri compresa fra una coppia di virgolette
  - esempio: `"Hello!"`
  - possono includere anche spaziature, caratteri non alfanumerici e sequenze di escape: per esempio `"Hello, world!\n"`

Rappresentazione di Numeri

- un numero intero viene rappresentato da una sequenza di cifre, che vengono interpretate
  - in base decimale (default)
  - in base ottale se inizia con uno zero
  - in base esadecimale se inizia con '0x' (o '0X')
- un numero reale (in virgola mobile) viene rappresentato facendo uso del punto decimale e della lettera 'e' (o 'E') per separare la parte in virgola fissa dall'esponente;
  - ad esempio: -1235.6e-2 rappresenta -12,356

Nota:

In un numero reale, la virgola viene rappresentata secondo la notazione anglosassone con un punto ".".

Esempio di rappresentazione di numeri in C++:

{ `ESEMPI_BASE/rappr_numeri.cc` }

Operatori

Alcuni caratteri speciali e loro combinazioni sono usati come operatori, cioè servono a denotare certe operazioni nel calcolo delle espressioni.

- Esempio: `2*3+4`

Principali operatori del C++

```
+    -    *    /    %    ^    &    |    ~
!    =    <    >    +=   -=   *=   /=   %=
^=   &=   |=   <<   >>   >>=  <<=  ==   !=
<=   >=   &&   ||   ++   --   ,    ->*  ->
.*   ::   ()   []   ?:
```

Proprietà degli Operatori

- la posizione rispetto ai suoi operandi (o argomenti):
  un operatore si dice
  - prefisso se precede gli argomenti
  - postfisso se segue gli argomenti
  - infisso se sta fra gli argomenti
- il numero di argomenti (o arità)
- la precedenza (o priorità) nell'ordine di esecuzione;
  - ad esempio: l'espressione `1+2*3` viene calcolata come `1+(2*3)` e non come `(1+2)*3`
- l'associatività: l'ordine in cui vengono eseguiti operatori della stessa priorità
  - gli operatori possono essere associativi a destra o a sinistra

I Separatori

Principali Separatori del C++

```
(   )   ,   ;   :   {   }
```

- i separatori sono simboli di interpunzione, che indicano il termine di una istruzione, separano elementi di liste, raggruppano istruzioni o espressioni, ecc.
- alcuni caratteri, come la virgola, possono essere sia separatori che operatori, a seconda del contesto
- le parentesi devono essere sempre usate in coppia, come nel normale uso matematico