UNIVERSITÀ
DI TRENTO

Dipartimento di Ingegneria e
Scienza dell'Informazione
DISI - Trento

Programmazione 1

18 - Esercitazione

Matteo Franzil
matteo.franzil@unitn.it

Anno Accademico 2023/2024

Nelle puntate precedenti

- **Scope**
  - E’ la porzione di codice in cui è attiva una definizione, può essere **locale** o **globale**;

```
int a = 100;     // globale
int funzione()
{
  int b = 10;     // locale
  return b;
}
```

Nelle puntate precedenti

- **Visibilità di una definizione**

```
1.  const char numero = '1';
2.
3.  int f (int a)
4.  {
5.   int b = 10;
6.   int numero = 10;
7.  }
8.
9.  int main() {
10.  char numero = '5';
11.  }
```

Nelle puntate precedenti

- **Visibilità di una definizione**

```
1.  const char numero = '1';
2.
3.  int f (int a)
4.  {
5.   int b = 10;
6.   int numero = 10;
7.  }
8.
9.  int main() {
10.  char numero = '5';
11.  }
```

Nelle puntate precedenti

- **Visibilità di una definizione**

```
1.  const char numero = '1';
2.
3.  int f (int a)
4.  {
5.   int b = 10;
6.    int numero = 10;
7.  }
8.
9.  int main() {
10.   char numero = '5';
11.  }
```

Nelle puntate precedenti

- **Visibilità di una definizione**

```
1.  const char numero = '1';
2.
3.  int f (int a)
4.  {
5.   int b = 10;
6.   int numero = 10;
7.  }
8.
9.  int main() {
10.  char numero = '5';
11.  }
```

Nelle puntate precedenti

- **Visibilità di una definizione**

```
1.  const char numero = '1';
2.
3.  int f (int a)
4.  {
5.   int b = 10;
6.   int numero = 10;
7.  }
8.
9.  int main() {
10.  char numero = '5';
11.  }
```

Nelle puntate precedenti

- `static` **e** `extern`
  - `static`**:** permette di estendere la durata di una variabile locale oltre lo scope della funzione dove è definita. Se applicata invece ad oggetti di scope globale, restringe la visibilità dell'oggetto al file in cui occorre la definizione;
  - `extern`: ci consente di utilizzare oggetti globali che sono definiti in altri file;

Nelle puntate precedenti

- **Compilazione su più file**
  - Permette un'organizzazione **modulare** del vostro codice. Presenta inoltre molti altri vantaggi (riutilizzo di codice, facilità il lavoro in team, etc.);
  - Le funzionalità vengono raggruppate in due files, ad esempio **modulo.h** e **modulo.cc**. Il primo file conterrà gli header delle funzioni, il secondo le definizioni delle stesse;

Nelle puntate precedenti

- **Compilazione su più file**

```
g++ -Wall main.cc module.cc ... -o main
```

L’ordine in cui si compilano i file è importante! Prima il main, poi i moduli!

1 - Cifrario di Cesare

Scrivere un programma che implementi il Cifrario di Cesare. Il programma dovrà prendere in input il file contenente il messaggio da cifrare (assumete che il file contenga solo lettere minuscole e spazi fino ad un massimo di 255 lettere), il nome di un file in cui verrà salvato l'output e la chiave di cifratura. All'utente verrà inoltre chiesto se intende cifrare o decifrare il messaggio.
Utilizzare le funzioni della libreria `fstream`

Definire la funzione di cifratura e di decifratura in un file separato.

```
void crypt(char parola [], int chiave);
void decrypt(char parola [], int chiave);
```

2 - Estrazione di lettere

Scrivere la dichiarazione e definizione di una funzione ricorsiva `estrai` che, data una stringa lunga al massimo 80 caratteri senza spazi, estragga tutte le lettere maiuscole contenute e restituisca un'altra stringa contenente solo le lettere maiuscole.

Definire la funzione di estrazione in un file separato

```
"MarcoStefanoAndreaEnricoGiovannaMatteo"
=
"MSAEGM"
```

2 - Estrazione di lettere

Scrivere la dichiarazione e definizione di una funzione ricorsiva `estrai` che, data una stringa lunga al massimo 80 caratteri senza spazi, estragga tutte le lettere maiuscole contenute e restituisca un'altra stringa contenente solo le lettere maiuscole.

Nota: la funzione `estrai` deve essere ricorsiva. Al suo interno non devono essere presenti cicli o chiamate a funzioni che contengano cicli. Si può fare uso di eventuali funzioni ricorsive ausiliarie.

Nota 2: non è ammesso l'utilizzo di variabili globali, static o di funzioni di libreria

3 - Email

Scrivere un programma che, preso in input un file, legga le parole in esso contenute (massimo 255 lettere) e stampi a video le parole che possono essere potenzialmente un indirizzo email.
Utilizzare le funzioni della libreria `fstream`

Definire la funzione che controlla se una parola è potenzialmente un indirizzo email in un file separato

3 - Email

1. Deve contenere un solo simbolo @, che divida la parola in due parti, nella forma: `<utente>@<dominio>`

2. Entrambe le parole `<utente>` e `<dominio>` devono:
   a. Avere lunghezza >= 1;
   b. Essere formate solo da:
      i. lettere minuscole (a-z)
      ii. lettere maiuscole (A-Z)
      iii. cifre (0-9)
      iv. i simboli . (punto) e _ (underscore)

3. Non avere il simbolo . (punto) come prima o ultima lettera;

"C makes it easy to shoot yourself in the foot; C++ makes it harder, but when you do it blows your whole leg off"

Bjarne Stroustrup