Corso di Laboratorio di Programmazione

A.a. 2023/24

Laboratorio 1
Variabili e funzioni
22/10/2023

1.
Implementare “Hello, world”, poi compilarlo ed eseguirlo da riga di comando Linux. Per compilare fare riferimento al comando `g++` visto a lezione. Per eseguirlo: `./<nome eseguibile>`

2.
Considerare come base di partenza il seguente codice:

```
#include <iostream>

int main()
{
      return 0;
}
```

Evolvere questo software aggiungendo (leggere anche il punto successivo prima di procedere):
a. una variabile `int` locale automatica;
b. una variabile `int` locale statica inizializzata a un valore diverso da 0;
c. una variabile `int` locale statica non inizializzata;
d. una variabile `int` globale (in quanto globale, è anche statica senza bisogno di esplicitarlo) inizializzata a un valore diverso da 0;
e. una variabile `int` globale (in quanto globale, è anche statica senza bisogno di esplicitarlo) non inizializzata.

Per ciascun passaggio, analizzare le dimensioni di initialized data segment (AKA data segment) e uninitialized data segment (BSS) utilizzando la shell di linux nel seguente modo:
a. compilare il sorgente *senza linkare* – per farlo, utilizzate il comando `g++` con l’opzione `-c`, es:
`g++ -c <nome_sorgente>.cpp`
b. usare il comando `size <nome_sorgente>.o`

Rispondere alle seguenti domande (in autonomia o a gruppi):
a. in quale delle sezioni riportate da `size` compare la variabile locale automatica? Perché?
b. perché la variabile locale automatica è “automatica”?
c. cosa succede togliendo l’inizializzazione alle variabili statiche? Perché?
d. qual è lo scope di ciascuna delle variabili create?

3.
Scrivere una funzione `print()` che stampa uno `std::vector` di `int` su `cout`. Fornire due argomenti: una `std::string` per etichettare l’output (cioè definire cosa il programma deve stampare prima del vettore - il classico messaggio all'utente) e uno `std::vector<int>` da stampare.

4.
Creare uno `std::vector` di numeri di Fibonacci e stamparli usando la funzione dell'esercizio 3. Per creare lo `std::vector`, implementare una funzione `fibonacci(x, y, v, n)` dove `x` e `y` sono due `int`, `v` è uno `std::vector<int>` (output della funzione) che **non** si può assumere vuoto (è quindi necessario eliminare tutti gli elementi prima di scriverlo), e `n` è il numero di elementi da inserire in `v`. `v[0]` è `x` e `v[1]` è `y`.
Un numero di Fibonacci è tale se appartiene a una sequenza in cui ogni elemento è la somma dei due precedenti. Per esempio, partendo da 1 e 2, otteniamo 1, 2, 3, 5, 8, 13, 21, … La funzione `fibonacci()` deve generare tale sequenza partendo da `x` e `y`.
In questa funzione utilizzare il passaggio per copia o per riferimento come si ritiene più appropriato.

Nota: per aggiungere un nuovo elemento in coda a uno `std::vector`, usare la funzione membro `push_back()`. La funzione membro `clear()` elimina tutti gli elementi del vettore. L'accesso agli elementi in lettura e scrittura è gestito tramite l'operatore parentesi quadre `[]` - come è tipico per i vettori.

5.
Un `int` può contenere interi solo fino a un valore massimo. Trovare un'approssimazione di tale massimo usando `fibonacci()`.

6.
Implementare una funzione:

```
void call_counter(void);
```

che, a ogni chiamata, stampa su `cout` un numero progressivo che rappresenta il numero totale di chiamate effettuate a tale funzione. Che tipo di variabile è necessario utilizzare per risolvere questo problema? **Qual è la sua durata di memorizzazione? Qual è il suo scope?** Discutere con i vicini di banco.