

Corso di Laboratorio di Programmazione

Laboratorio 3

Puntatori e allocazione dinamica della memoria

2/11/2022

Nota: i quesiti sono tratti (ma non tradotti) dal libro di testo.

Discussione

A coppie, rispondete alle seguenti domande (Review, cap. 17, p. 623 sgg.):

1. What is a dereference operator and why do we need one?
2. What is an address? How are memory addresses manipulated in C++?
3. What information about a pointed-to object does a pointer have? What useful information does it lack?
4. What can a pointer point to?
5. When do we need a pointer (instead of a reference or a named object)?

Esercizi

Per lo svolgimento dei seguenti esercizi si raccomanda di **compilare molto frequentemente il codice**, senza aspettare di averlo scritto interamente.

1. Creare un programma che:
 1. definisce una funzione f(), chiamata dal main, che definisce un array stile C di 10 int come variabile locale automatica;
 2. in f(), crea un puntatore a uno degli elementi dell'array a scelta (non il primo), e scrive su tutto l'array usando l'operatore [] applicato al puntatore;
 3. definisce una funzione f_illegal() che definisce un array analogo a quello della funzione f(), ma scrive fuori dai limiti dell'array; verificare se questo causa un errore in esecuzione o meno in funzione di quanto dista la memoria a cui si accede illegalmente dall'array definito.
2. Creare un programma che:
 1. definisce nel main un array di double grande a piacere (la dimensione dell'array deve essere definita con un constexpr o variabile costante);
 2. stampa la dimensione dell'array usando sizeof;
 3. passa l'array come argomento a una funzione print_array_length(), opportunamente dichiarata e definita, che a sua volta stampa la dimensione dell'array usando sizeof. La funzione print_array_length() conosce la dimensione dell'array? Riesce ad accedere ai dati dell'array? L'accesso è lecito?

Commentare i risultati ottenuti con i vicini di banco.
3. Creare un programma che:
 1. Definisce una variabile int e una double nel main;
 2. le stampa da una funzione print_reference() a cui sono passate per const reference;
 3. le stampa da una funzione print_pointer() a cui sono passate per puntatore.

4. Implementare la classe MyVector, analoga alla classe vector discussa a lezione, che rappresenta vettori di double di lunghezza decisa in fase di costruzione (e non modificabile in seguito). Il buffer dove sono salvati i dati di MyVector deve essere allocato dinamicamente, poiché i dati da salvare potrebbero essere numerosi. Includere nella classe:
 1. un int che contiene la lunghezza del vettore;
 2. un costruttore che accetta un int che indica la lunghezza del vettore da costruire;
 3. l'overloading dell'operatore [] - sia in versione const che non const;
 4. le funzioni safe_set() e safe_get() che permettono l'accesso agli elementi del vettore effettuando il controllo di accesso entro i limiti del vettore;
 5. il distruttore.