Corso di Laboratorio di Programmazione

Laboratorio 2 - Puntatori
25/10/2021

Nota: i quesiti e gli esercizi seguenti 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. nel main definisce una variabile int my_int e tre puntatori p1, p2 e p3 (definiti con tre istruzioni diverse), tutti inizializzati con l'indirizzo di my_int. Effettuare varie scritture e letture della variabile tramite my_int e i tre puntatori, verificando la coerenza;
    2. modificare il programma definendo i tre puntatori con un'unica istruzione.

Disegnare su carta la mappa della memoria.

2. 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. 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.

3. 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.

4. Creare un programma che:
    1. definisce nel main un array di interi my_array di dimensione a piacere (gestire opportunamente la dimensione dell'array: evitare i magic number);

2. definisce una funzione print_array() con due parametri: un puntatore void* e un int, che rappresentano il puntatore al primo elemento e il numero di elementi di un generico array;
3. passa my_array dal main a print_array() tramite un void*;
4. stampa gli elementi dell'array dentro print_array() – che conversione è necessaria per poter accedere ai dati dell'array?

5. 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.

6. Creare un programma che:
   1. definisce una stringa stile C nel main, inizializzata con un literal tra doppi apici;
   2. la stampa carattere per carattere usando un ciclo;
   3. stampa il contenuto del null-terminator convertito in int.