Corso sul linguaggio C++

Modulo 4

1 – I puntatori

Prerequisiti

- Corso base di programmazione

Introduzione

In questa Unità vediamo l'utilizzo di puntatori per gestire array, stringhe e funzioni.

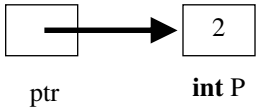
Argomenti

- I puntatori
- Dichiarazione di un puntatore
- Operazioni sui puntatori
- Uso di variabili tramite puntatori
- Passaggio parametri per indirizzo
- Array acceduti con puntatori
- Funzioni e puntatori
- Puntatori a funzione

I puntatori

Tramite il tipo di dato **puntatore** possiamo accedere indirettamente alle locazioni di memoria delle variabili senza fare uso del loro nome (indirizzo logico) e indipendentemente dal loro tipo (char, int, float, ecc)

Una **variabile puntatore** è una variabile (ptr) che può essere usata per accedere ad un'altra variabile (P) in memoria (**variabile puntata**). Cambiando il contenuto di ptr è possibile accedere a variabili **poste in qualunque altra locazione**



ptr **int** P

Dichiarazione di un puntatore

La sintassi per dichiarare una variabile puntatore è

*tipobase *identptr*;

Ad esempio:

int *ptr; /* puntatore a intero (4 byte) */

char *c; /* puntatore a carattere (1 byte) */

Lo **scalare** di ptr è **int** e la **dimensione scalare** è 4 byte.

Lo **scalare** di c è **char** e la **dimensione scalare** è 1 byte.

Il contenuto di ptr (*rvalue*) è l'indirizzo (*lvalue*) della variabile puntata.

Un puntatore consente di **puntare ad un dato di qualunque tipo**

Operazioni sui puntatori

Le **operazioni** sui puntatori sono:

Dichiarazione: Es.**int** *ptr1, *ptr2; /* ptr puntatore a intero */
Assegnazione: Es. ptr1=ptr2;
Indirizzo-di: Es: ptr1=**&**i; /* con i intera */
Indirezione: Es. *ptr=3; /* pone il valore 3 in i */
Confronto: Es. **if** (ptr1= =NULL) .. /* puntatore nullo */

Gli operatori & e * sono l'uno l'inverso dell'altro:

&*i= = i *& i = = i

Uso di variabili tramite puntatori

```
#include <iostream>
using namespace std;
int main()
{  int n, somma, i;
   int *pn, *psomma;
   pn=&n;  psomma=&somma;
   *psomma=0;
   for (i=0;i<5; i++)
   {  cin>>*pn;
      *psomma+=*pn;
   }
   cout<<*psomma<<endl;
   system("Pause");
   return EXIT_SUCCESS;
}
```

Dichiarazione puntatori

Inizializzazione puntatori

Utilizzo dell'indirezione per accedere alle variabili

L'istruzione **cin** può essere utilizzata per stampare gli indirizzi delle variabili (es: cin>>&n;)

Passaggio parametri per indirizzo

Passare per indirizzo un parametro ad una funzione ha due scopi principali:

- consentire che la funzione modifichi il valore di esso e lo riconsegni al programma chiamante modificato.
- evitare spreco di memoria **quando il dato da modificare è di dimensioni notevoli.** Infatti, in questo caso, il passaggio per valore, operando su una copia del dato, occupa evidentemente troppa memoria.

Passaggio parametri per indirizzo

```
#include <iostream>
using namespace std;
typedef struct dati
{  string cognome, nome;
   float altezza;
};
void leggidati(dati *d);
void stampadati(dati d);
int main()
{  dati d;   dati *dato;
   dato=&d;
   leggidati(dato);  stampadati(*dato);
   system("Pause");
   return EXIT_SUCCESS;
}
void leggidati (dati *d)
{  cin>>d->cognome;
   cin>>d->nome;
   cin>>d->altezza;
}
void stampadati (dati d)
{  cout<<d.cognome<<endl;
   cout<<d.nome<<endl;
   cout<<d.altezza<<endl;
}
```

Notare la sintassi "->" che consente la indirezione e l'accesso ad un campo di una struct

La funzione *leggidati*() usa passaggio per indirizzo per evitare di creare una copia della struct

Array acceduti con puntatori

```
#include <iostream>
using namespace std;
int main()
{   float f[5];
    cout<<"Vettore di float"<<endl;
    cout<<"Lettura vettore"<<endl;
    for (int i=0;i<5;i++)
    {     cout<<"i = "<<i*sizeof (float)<<": ";
          cin>>*(f+i);
    }
    cout<<"Stampa vettore"<<endl;
    for (int i=0;i<5;i++)
          cout<<*(f+i)<<endl;
    system("Pause");
    return EXIT_SUCCESS;
}
```

Esempio con array di interi

Notare come attraverso l'indirezione sia possibile l'accesso alle componenti del vettore

Array acceduti con puntatori

```
#include <iostream>
using namespace std;
const int MAX=10;
int main()
{   char c[MAX];
    char *p;
    int i;
    cout<<"Lettura stringa"<<endl;
    cin.get (c, 10, '\n');
    p=c;
    while (cin.get ()!='\n');
    while ((*p)!='\0')
        p++;
    while (p!=c)
    {  p--;
       cout<<":"<<*p<<endl;
    }
    system("Pause");
    return EXIT_SUCCESS;
}
```

Esempio con array di caratteri

Dichiarazione puntatore a carattare

Lettura stringa compresi eventuali spazi

Inizializzazione puntatore

Svuotamento buffer

Calcolo fine stringa

Funzioni e puntatori

L'utilizzo dei puntatori può essere esteso alle funzioni e possiamo avere:

- Funzioni che restituiscono un puntatore (**Funzioni puntatore**)
- **Puntatori a funzioni** (puntatori che indirizzano a funzioni)

Puntatori a funzione

Un **puntatore a funzione** rappresenta rappresenta **l'indirizzo iniziale** del codice che definisce la funzione.

In questo caso, il nome della funzione rappresenta l'indirizzo iniziale della funzione stessa

Un puntatore p a funzione si dichiara con

tipo (**p*) (*listaparametri*);

tipo indica il tipo di ritorno della funzione

p è il nome del puntatore a funzione

listaparametri è l'elenco dei parametri della funzione

Puntatori a funzione

Ad esempio data la funzione

```
double Quadrato (double x)
{
  return x*x:
}
```

Si ha:

```
double (*p) (double);// definizione di puntatore a funzione
p = Quadrato;        // assegnazione puntatore a funzione
cout<<(*p) (4);      // istanza di Quadrato() e stampa di 16
```


Puntatori a funzione

```
#include <iostream>
#include <math.h>
using namespace std;
int main()
{  double (*v[6])(double)={sin, cos, tan, exp, log, log10};
   int scelta;
   float x;
   cout<<"1 - sin"<<endl;
   cout<<"2 - cos"<<endl;
   cout<<"3 - tan"<<endl;
   cout<<"4 - exp"<<endl;
   cout<<"5 - log"<<endl;
   cout<<"6 - log10"<<endl;
   cout<<"Scelta ->";
   cin>>scelta;
   cout<<"Valore di x: ";
   cin>>x;
   cout<<(*v[scelta-1])(x);
   system("Pause");
   return EXIT_SUCCESS;
}
```

Un esempio di calcolatrice scientifica.

Dichiarazione ed inizializzazione array di puntatori a funzione

Istanza della funzione scelta

Attività

- Provare i programmi esposti nell'Unità
- Scrivere e provare le seguenti funzioni:
  - **char** *TrovaCarattere (**char** **s,* **char** *c*); che fornisce il puntatore alla prima occorrenza di *c* in oppure 0 in caso di insuccesso
  - **char** *SottoStringa (**char** **s*, **int** *inizio*, **int** *numcar*); che estrae la sottostringa di *s*, formata da *numcar* caratteri a partire dal carattere in posizione *inizio*. **int** strcmp(**char** *s1, **char** *s2)
  - **char** *StrCopy (**char** **s1*, **char** **s2*)
  - **char** *StrConccat (**char** **s1*, **char** **s2*)
  - **int** Strlength (**char** **s*)

Attività

- Servendosi dei puntatori, implementare i seguenti algoritmi di ricerca su un array:
  - ricerca sequenziale di un elemento
  - ricerca con sentinella di un elemento
  - ricerca binaria di un elemento
- Servendosi dei puntatori, implementare i seguenti algoritmi di ordinamento su un array:
  - ordinamento secondo la tecnica di inserzione
  - ordinamento secondo la tecnica di selezione
  - ordinamento secondo la tecnica di scambio

Obiettivi

- Puntatori
- Operazioni sui puntatori
- Passaggio di parametri per indirizzo
- Puntatori e array
- Puntatori e stringhe
- Puntatori a funzione

Conoscenze (sapere)

- Tipo puntatore
- Operazioni sui puntatori
- Array come puntatori
- Stringhe come puntatori
- Puntatori a funzione
- Passaggio parametri per indirizzo

Competenze (saper fare)

- Dichiarare una variabile di tipo puntatore
- Usare una variabile puntatore
- Operare su array mediante puntatori
- Operare su stringhe mediante puntatori
- Utilizzare puntatori a funzione
- Utilizzare parametri passati per indirizzo

Cosa ho imparato

- Tipo puntatore
- Operazioni sui puntatori
- Array come puntatori
- Stringhe come puntatori
- Puntatori a funzione
- Passaggio parametri per indirizzo

Cosa ho imparato a fare

- Dichiarare una variabile di tipo puntatore
- Usare una variabile puntatore
- Operare su array mediante puntatori
- Operare su stringhe mediante puntatori
- Utilizzare puntatori a funzione
- Utilizzare parametri passati per indirizzo

Terminologia

- Scalare
- Dimensione scalare
- Tipo puntatore
- Operatore di indirezione
- Operatore indirizzo-di
- Puntatori a funzione

Altre fonti di informazione

- P.Gallo, F.Salerno – Informatica Generale 1, ed. Minerva Italica
- M.Romagnoli, P.Ventura – Linguaggio C/C++, ed. Petrini
- J. Purdum, C – Ed. Jackson
- A.Garavaglia et alii – Informatica vol. II – ed. Masson