Corso sul linguaggio C++

Modulo 4

2 - Allocazione dinamica

Prerequisiti

- Strutture dinamiche
- Memoria heap

Introduzione

In questa Unità si illustra la tecnica dell'allocazione dinamica, che consiste nell'allocare spazio per i dati durante l'esecuzione del programma.

L'allocazione viene fatta prelevando celle dalla memoria **heap**.

Applichiamo la tecnica di allocazione dinamica per la creazione di **array dinamici**.

Argomenti

- Allocazione dinamica
- Deallocazione
- Array dinamico

Allocazione dinamica

Consiste nell'uso della memoria *heap*:

- l'area allocata non è identificata da un **nome**, ma è accessibile ***esclusivamente*** tramite un puntatore (indirizzo);
- la sua **visibilità** è legata a quella della variabile puntatore che contiene il suo indirizzo;
- il suo **tempo di vita** coincide con l'intera durata del programma, a meno che non venga esplicitamente deallocata;
- se il puntatore viene perduto (*out of the scope*), l'area non è più accessibile, ma continua a occupare memoria inutilmente.

Allocazione dinamica

L'**allocazione dinamica** si realizza nel seguente modo con l'operatore **new()**:

- **Stringhe**
  char *s = **new char**[*n*]; /* stringa di *n* caratteri */
- **Array di interi**
  int *v = **new int**[*dim*]; /* array di *dim* interi */
- **Interi**
  int x = **new int**(7); /* intero x inizializzato con 7 */

I primi due esempi dichiarano quello che si dice **array dinamico** che vedremo tra breve.

Allocazione dinamica

Ad esempio:

```
struct anagrafica
{
  .......
}
anagrafica * p;                /* puntatore a struct */
p = new anagrafica [100];
```

definisce una **struct** ***anagrafica*** e dichiara un **puntatore** a tale struttura, a cui **assegna** l'**indirizzo** del primo di 100 **oggetti** di tipo *anagrafica*, allocati nell'area **heap**.

Deallocazione

La **deallocazione** si esegue con l'operatore **delete**

- **Stringhe**
  delete [ ] *s*;
- **Array di interi**
  delete [ ] *v;*
- **Interi**
  delete *x*;

Se l'operando punta a un'area in cui sono stati allocati più oggetti, **delete** va specificato con una coppia di parentesi quadre (senza la *dimensione*, che il C++ é in grado di riconoscere automaticamente) altrimenti viene deallocato solo il primo della serie di oggetti

L'operatore delete consente di deallocare esclusivamente i dati allocati con l'operatore new.

Deallocazione

Se ad esempio, allochiamo 100 locazioni *float* nel seguente modo:

float *p = **new float** [100];

con l'istruzione

delete[ ] p;

deallochiamo *tutta* la memoria allocata, mentre l'istruzione

delete p;

dealloccherebbe solo la prima delle 100 locazioni e le altre 99 risulterebbero allocate, ma non più accessibili.

Array dinamico

L'array dinamico risulta utile in molte situazioni in cui non è noto a priori il numero di componenti da utilizzare.

Vediamo un esempio di applicazione dell'array dinamico al caricamento e stampa di una tabella (array di struct).

Le strutture dati sono le seguenti:

```
typedef struct persona
{
   string cognome;
   string indirizzo;
};
persona *tab = new persona[numel];  /* numel viene letto da input */
```

Array dinamico

```
#include<iostream>
using namespace std;
typedef struct persona
{  string cognome;
   string indirizzo;
};
void lettura(persona v[], int numel);
void stampa(persona v[], int numel);
int main()
{  int numel;
   cout<<"Immetti il numero di elementi: ";    cin>>numel;
   persona *tab = new persona[numel];
   lettura(tab, numel);
   stampa(tab, numel);
   delete[]tab;                /* dealloca l'array */
   system("Pause");
   return 0;
}
```

Il programma *SortingArrayDinamico.cpp* crea un vettore dinamico, lo ordina e lo stampa

Dichiarazione **array dinamico**

Array dinamico

```
void lettura(persona v[], int numel)
{  int indice;
   for (indice=0; indice<numel; indice++)
    {   cout<<"Inserisci "<<indice<<"\370 elemento: "<<endl;
        cout<<"Cognome: ";          cin>>v[0].cognome;
        cout<<"Nome...: ";          cin>>v[0].indirizzo;
    }
}

void stampa(persona v[], int numel)
{ int i;
  for (i=0; i<numel; i++)
  {  cout<<v[0].cognome<<" - ";
     cout<<v[0].indirizzo<<endl;
  }
}
```

ATTIVITA' :Scrivere la funzione lettura(…)

ATTIVITA' :Scrivere la funzione stampa(…)

Array dinamico

Alcune considerazioni….

- Nel caso il programma richieda una ulteriore componente, occorre:
  - creare un array dinamico di appoggio con un elemento in più:
  - copiare tab[] in temp[] e caricare il nuovo valore all'ultima posizione;
  - copiare di nuovo temp[] in tab[] per avere i dati aggiornati;
  - tenere presente che ora tab[] possiede un elemento in più;
  - deallocare temp[].
- Nel caso il programma richieda di eliminare una componente, occorre:
  - creare un array dinamico di appoggio con un elemento in meno:
  - copiare tab[] in temp[], saltando la posizione (o l'elemento) da cancellare;
  - copiare di nuovo temp[] in tab[] per avere i dati aggiornati;
  - tenere presente che ora tab[] possiede un elemento in meno;
  - deallocare temp[].

Cosa ho imparato

- Cosa vuol dire allocare dinamicamente la memoria
- Quali sono le istruzioni per allocare dinamicamente e per deallocare la memoria
- Il significato e l'utilità degli array dinamici

Cosa ho imparato a fare

- Allocare dinamicamente la memoria
- Deallocare la memoria
- Creare ed utilizzare array dinamici

Terminologia

- Allocazione dinamica
- Deallocazione
- Memopria heap
- Array dinamico
- **new**() e **delete** ()

Riepilogo rivedere

In questa Unità abbiamo mostrato le istruzioni per gestire l'allocazione dinamica della memoria (**new** () e **delete** ()) e le abbiamo applicate ad un primo esempio di struttura dinamica: l'**array dinamico**.

Tramite array dinamico abbiamo realizzato la costruzione della classe CStringa con i relativi metodi.

Altre fonti di informazione

- P.Gallo, F.Salerno – Informatica Generale 1, ed. Minerva Italica
- M.Romagnoli, P.Ventura – Linguaggio C/C++, ed. Petrini
- A.Garavaglia, F.Petracchi,S.Forte-INFORMATICA-ed. Masson