ALOCAREA SPAȚIULUI DE MEMORIE

CURS 3- 16.03.2021

Titular: Șef. Lucr. Dr. Mat. Cărbureanu Mădălina

Copyright@Departamentul de Automatică, Calculatoare și Electronică

Universitatea Petrol-Gaze din Ploiești

ALOCAREA SPAȚIULUI DE MEMORIE

- Noțiunea de alocare a spațiului de memorie;
- Alocarea statică a memoriei;
- Alocarea dinamică a memoriei. Pointeri;
- Operatori și operații cu pointeri;
- Aplicații.

Noțiunea de alocare a spațiului de memorie

- **alocarea spațiului de memorie pentru o variabilă** → asignarea/asocierea unui spațiu de memorie pentru variabila respectivă;
- **reprezentarea unei variabile în lb. C/C++** → tip date, valoare, adresa zonei de memorie în care variabila este memorată;
  - valoarea unei variabile este stocată în memoria internă a calculatorului;
  - fiecare locație de memorie are asociată o adresă unică, dimensiunea locației de memorie fiind dată de tipul de date al variabilei memorate;

Noţiunea de alocare a spaţiului de memorie

- **Declarare variabilă** → dimensiunea zonei de memorie alocate pentru variabila respectivă este dată de mărimea variabilei, respectiv i se alocă un număr de octeţi care depinde de mărimea variabilei;
- ***Observaţii:***
  - determinare dimensiune în octeţi a unei variabile: *sizeof(nume_variabilă)* sau *sizeof(tip_dată_variabilă);*
  - dimensiunile în octeţi ale principalelor tipuri de date sunt:
    - 2 octeţi pentru tipul *int*,
    - 4 octeţi pentru tipul *float*,
    - 8 octeţi pentru tipul *double,*
    - 4 octeţi pentru tipul *long;;*
    - 1 octet pentru tipul *char* ;
- ***Exemplu:*** *double x*; → determină alocarea unei zone de memorie de 8 octeţi pentru memorarea valorii variabilei *x*.

Alocarea statică a memoriei

- **Alocarea statică↔ alocare directă;**
- **<u>Alocarea statică:</u>**
  - **asocierea nume de variabilă, zonă de memorie (identificată prin adresa de memorie) este constantă pe toată durata de execuție a programului;**
  - presupune cunoașterea valorii curente a variabilei care va fi memorată de la începutul execuției programului, până la finalizarea acestuia;
  - alocarea memoriei se realizează în faza de compilare, spațiul de memorie alocat rămânând ocupat pe tot parcursul execuției programului, chiar dacă variabila, la un moment dat, nu mai este necesară.

Alocarea statică a memoriei

- ***Observații:***
  - dealocarea memoriei pe parcursul execuției programului este practic imposibilă;
  - valoarea unei variabile stocate în memoria calculatorului este accesată direct, adică este accesată prin numele de variabilă;
  - memoria este alocată în faza de compilare a programului și nu poate fi modificată pe parcursul execuției acestuia ;
  - **alocarea statică presupune lucrul cu vectori (șiruri) și matrice;**
  - alocarea memoriei este realizată static de către compilator.

Alocarea statică a memoriei

- **Exemplu:** asocierea nume de variabilă-zonă de memorie → **constantă pe toată durata de execuție a programului.**
- *int x;*

Tabel 1. Asociere nume_variabilă-zonă_memorie (alocare statică)

Nume_variabilă	Tip variabilă	Dimensiune zonă memorie alocată [octeți]	Adresă variabilă
x	int	2	00AA
x	float	4	00AA
x	double	8	00AA
x	char	1	00AA

Alocarea dinamică a memoriei. Pointeri

- **Alocare dinamică a memoriei ↔ alocare indirectă;**
- **<u>Alocare dinamică:</u>**
  - **asocierea nume de variabilă, zonă de memorie nu este constantă pe toată durata execuției programului;**
  - permite alocarea spațiului de memorie în mod dinamic în timpul execuției programului prin memorarea referinței, respectiv a adresei variabilelor;
  - se folosește tipul de date **pointer;**

Pointerii

- **Def.:**
  - variabile care au ca valoare adresa unei zone de memorie în care este stocată o altă variabilă;
  - variabile în care se pot memora adrese de memorie;
- **Utilitate:**
  - permit adresarea indirectă a adreselor de memorie;
  - oferă posibilitatea de a aloca dinamic memoria, adică pe parcursul execuției unui program se pot aloca sau dealoca zone de memorie asociate pointerilor.

Pointerii

- Sintaxă declarare **pointer**:

*tip *var_pointer;*

- ***Observații:***
- *tip* este tipul de date indicat de catre *var_pointer*;
- **!pointerii sunt singurele variabile care permit alocarea dinamică a memoriei;**
- **exemple:**
  - *float *x*; → *x* este un pointer către tipul de date *float*;
  - *int *y[10];* → *y* este un tablou de pointeri către tipul de date *int*;
  - *void *w*; → *w* este un pointer către nimic.

Pointerii

- **Alocarea dinamică** → diferite variante ale funcţiei *malloc()* (funcţia *calloc()*, funcţia *realloc(), etc.*), funcţie ce prezintă următoarea sintaxă:

  *void *malloc(unsigned int număr_octeţi);* → *alloc.h*

- ***Observaţii:***
  - funcţia *malloc()* întoarce un pointer de tip *void;*
  - dacă spaţiul de alocare este suficient, pointerul întors de funcţia *malloc()* conţine adresa primului octet al zonei de memorie alocate; în caz contrar (spaţiu insuficient), funcţia întoarce valoarea NULL.

Pointerii

- **Eliberarea zonelor de memorie alocate dinamic** → funcţia *free(),* funcţie ce prezintă următoarea sintaxă:

free(var_pointer); → *alloc.h*

- ***Observaţii:***
  - sintaxă declarare masiv unidimensional (şir, vector) alocat dinamic: *tip *nume_şir;*
  - sintaxă declarare masiv bidimensional (matrice) alocare dinamică: *tip*nume_matrice[10]*; *tip**nume_matrice;*
  - memorarea constantelor şir la o adresă indicată de un pointer către tipul de date *char*, se foloseşte o sintaxă de forma:

*char*pointer_şir=”Proiectarea algoritmilor”;*

Alocarea dinamică a memoriei. Pointeri

- **Exemplu:** asocierea nume de variabilă-zonă de memorie→ **nu este constantă pe toată durata de execuție a programului;**
- *int x,*pointer;*
- *pointer=&x;*

Tabel 2. Asociere nume_variabilă-zonă_memorie (alocare dinamică)

Nume_variabilă	Tip variabilă	Adresă variabilă
x	int	00AC
x	float	00AE
x	double	00B2
x	char	00AB

Operatori și operații cu pointeri

- **Operatori pointeri:**
  - **Operatorul de referențiere** sau **de adresare &** → modalitate de a obține adresele unor variabile spre a fi memorate într-un pointer;
  - **Operatorul de dereferențiere** sau **de indirectare *** → permite accesul indirect la informația memorată în zonele de memorie indicate de pointeri.

Observaţii:

- secvenţa de instrucţiuni: **int *a, b; a=&b;** → corectă → adresa variabilei *b* este memorată în *a*;
- secvenţa de instrucţiuni: **int *a, b; b=5; a=&b;** → corectă → deoarece în pointerul *a* este memorată adresa variabilei *b*, variabila **a* va avea conţinutul variabilei *b*, adică **a=5;*
- pentru a afla valoarea unui pointer → descriptorul de format *%p*;
- valoarea NULL → pointer zero.

Operații cu pointeri

- **Adunarea, scăderea unui pointer cu un întreg;**
- **Exemplu** (*int *p*):
  - *p*+*n* va semnifică adresa aflată după *p* la o distanță de *n* pozitii;
  - *p*+*n* va conține adresa *p*+*n***sizeof(tip);*
  - *p*-*n* semnifică adresa aflată cu *n* poziții înaintea lui *p*;
  - *p*-*n* va conține adresa *p*-*n***sizeof(tip)*;

Aplicații – Alocarea dinamică a unui vector

- *#include<iostream.h>*
- *#include<conio.h>*
- *#include<alloc.h>*
- *void main()*

- *{int i, n,*a;*
- *cout<<"dimensiune vector n=";cin>>n;*
- ***//alocare memorie vector***
- ***a=(int*)malloc(sizeof(int));***
- *//citire vector*
- *for(i=1;i<=n;i++)*
- *{cout<<"a["<<i<<"]="; cin>>a[i];}*
- *//afisare vector*
- *cout<<"vectorul este:"<<endl;*
- *for(i=1;i<=n;i++) cout<<a[i];*
- *getch(); }*

Aplicații – Alocarea dinamică a unei matrice

- *#include<iostream.h>*
- *#include<conio.h>*
- *#include<alloc.h>*
- *void main()*
- *{int i, j, n,m, *a[10]; //* **matricea alocată dinamic a fost declarată ca vector de pointeri**
- *cout<<" nr linii=";cin>>n; cout<<" nr coloane=";cin>>m;*
- ***//alocare memorie matrice***
- ***for(i=0;i<n;i++) { a[i]=(int*)calloc(n,sizeof(int)); }***
- *//citire matrice*
- *for(i=0;i<n;i++)*
- *for(j=0;j<m;j++)*
- *{cout<<"a["<<i<<"]["<<j<<"]="; cin>>a[i][j]; }*
- *//afisare matrice*
- *cout<<"matricea este:"<<endl;*
- *for(i=0;i<n;i++)*
- *{for(j=0;j<m;j++) cout<<a[i][j]; cout<<endl;} getch(); }*

Observații:

- **int *a,*b,*c;** → vectori alocați dinamic;
- *a=(int*)malloc(n*sizeof(int))* → alocare memorie vector folosind *malloc()*;
- **int *a[10];** → matrice declarată ca vector de pointeri;
- **int **a;** → matrice declarată ca pointer la pointer;
- ***(a+i)*** semnifică valoarea elementului a[i];
- ***(b+j)*** semnifică valoarea elementului b[j];
- ***&*(a+i)*** are semnificația &a[i];
- ***&*(b+j)*** are semnificația &b[j].

Aplicații propuse alocare dinamică memorie:

- Sortarea prin interclasare (**Merge Sort**);
- Aplicațiile nr. 1, 4, 5, 7, 8, 9, 11, 12 și 14 de la pag. 52-53 → carte "*Elemente de proiectarea algoritmilor. Ghid teoretic și practic*", autor Șef lucr. dr. mat. Cărbureanu Mădălina, UPG, Ploiești, 2021.

Spor la lucru!