UNIVERSITATEA TEHNICĂ A MOLDOVEI

PROGRAMAREA CALCULATOARELOR
OPERAȚII CU TABLOUL DE STRUCTURI FOLOSIND FUNCȚII ȘI POINTERI

Prelegere

Kulev **Mihail**, dr., conf. univ.

Stimați studenți și stimată audiență!

Mă numesc Kulev Mihail sunt doctor, conferențiar universitar la Universitatea Tehnică a Moldovei. Din cadrul cursului PROGRAMAREA CALCULATOARELOR Vă propun spre atenția Dumneavoastră prelegerea cu tema:

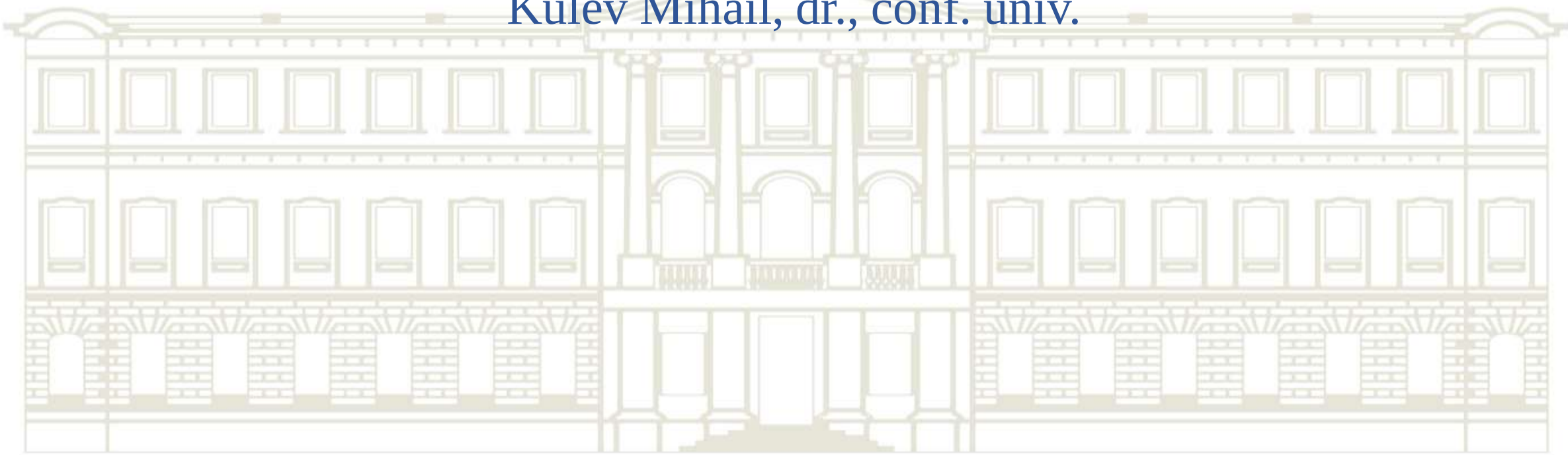
” OPERAȚII CU TABLOUL DE STRUCTURI FOLOSIND FUNCȚII ȘI POINTERI”

UNIVERSITATEA TEHNICĂ A MOLDOVEI

PROGRAMAREA CALCULATOARELOR
OPERAȚII CU TABLOUL DE STRUCTURI FOLOSIND FUNCȚII ȘI POINTERI

Prelegere

Kulev Mihail, dr., conf. univ.



UNIVERSITATEA TEHNICĂ A MOLDOVEI

Operații cu tabloul de structuri folosind funcții și pointeri

Conținutul prelegerii

1. Tablouri de structuri și structuri cu câmpuri de tip tablou.

Vom prezenta și analiza tablouri de structuri și structuri cu câmpuri de tip tablou cu demonstrarea aspectelor de implementare a funcțiilor, care definesc operații cu o structură.

2. Operații cu tablouri de structuri folosind funcții și pointeri.

Vom specifica operații cu tablouri de structuri și vom prezenta implementările funcțiilor pentru diferite operații cu tablouri.

Exemple de cod.

Pe parcursul studierii întrebărilor menționate vom prezenta exemple de cod în limbajul C.

1. Tablouri de structuri și structuri cu câmpuri de tip tablou

În limbajul C pot fi utilizate tablouri cu elemente de tip structură și, de asemenea, o structură poate avea câmpuri de tip tablou alocat static sau de tip pointer pentru adresa tabloului alocat dinamic. De exemplu:

```
typedef struct student{
char nume[40];
int an;
float medie;
} Student;
```

sau

```
typedef struct student{
 char *nume;
 int an;
 float medie;
 } Student;
```

```
Student s, sarr[20]; sau Student s, *sarr; int n=20;
        sarr=(Student*)calloc(n, sizeof(Student));
        if (sarr == NULL) exit(1);
```

Exemplu:

```
typedef struct vector{
int vec[40];
int m;
} Vector;
```

sau

```
typedef struct vector{
int *vec;
int m;
} Vector;
```

```
Vector v={0}, varr[20]={0};
```

sau

```
Vector v={0}, *varr; int n=20;
varr = (Vector*)calloc(n, sizeof(Vector));
if (varr == NULL) exit(1);
```

Alocarea memoriei pentru câmpul vec de tip pointer:

```
v.vec = (int*)calloc(vec.m, sizeof(int));
if(v.vec == NULL) exit(1);
```

Tablouri de structuri

Pentru a accesa o anumită structură din tablou de structuri de tip **`Student`** poate fi utilizată indexarea sau aritmetica pointerilor. **`sarr[i]`** `sau` **`*(sarr+i)`** – permite accesul la elementul cu indicele **`i`** al tabloului de structuri.

Pentru a accesa un câmp al unui anumit element din tablou se poate utiliza operatorii de acces - punct sau săgeată. De exemplu: **`sarr[i].an`** `sau` **`(sarr+i)->an -`** permite accesul la câmpul **`an`** al elementului cu indicele **`i.`** De menționat, că trebuie alocată memoria dinamică pentru câmpurile structurii, care sunt pointeri înainte de utilizarea lor:

```
sarr[3].nume=(char*)calloc(strlen("Popescu Ion")+1, sizeof(char));
if (sarr[3].nume == NULL) exit(1);
strcpy(sarr[3].nume, "Popescu Ion");
```

Funcţiile care definesc operaţii cu o structură

Pentru prelucrarea tabloului de structuri o bună practică este implementarea funcțiilor pentru operațiile cu o singură structură – un singur element al tabloului de structuri.

- Funcția de setare a câmpurilor structurii :

```
void setStudent(Student* sptr, char *num, int a, float med){
// if (sptr->nume != NULL) free(sptr->nume);
// sptr->nume = (char*)calloc(strlen(num)+1, sizeof(char));
// if(sptr->nume == NULL) exit(1);
strcpy(sptr->nume, num);
sptr->an = a;
sptr->medie = med;
}  Ex.: setStudent( &sarr[i] , ”Popescu Ion”, 21, 9.5);
```

Funcţiile care definesc operaţii cu o structură

- Funcția de eliberare a memoriei dinamice alocate pentru câmpuri de tip pointer ale structurii :

```
void freeMem(Student* sptr){
if (sptr->nume) free(sptr->nume);
sptr->nume = NULL;
}
```

NULL i se atribuie pointerului **name** pentru asigurarea unei utilizări repetate reuşite a variabilei.

```
Ex.: freeMem( &sarr[i] );
```

Funcţiile care definesc operaţii cu o structură

- Funcția de citire (introducere) a informației în câmpurile structurii de la tastatură:

```
void readStudent(Student *sptr) {
char buf[40];
printf ("Dati numele studentului: "); fflush(stdin); gets(buf);
// if (sptr->nume!=NULL) free(sptr->nume)
// sptr->nume = (char*) calloc((strlen(buf)+1), sizeof(char));
// if(sptr->nume==NULL) exit(1);
strcpy(sptr->nume, buf);
printf("Dati anul: "); scanf("%d", &sptr->an);
printf("Dati nota medie: ");
scanf("%f", &sptr->medie); }  Ex.: readStudent( &sarr[i] );
```

Funcţiile care definesc operaţii cu o structură

- Funcția de scriere (afișare) a informației din câmpurile structurii la ecran:

```
void showStudent(Student s) {
printf("Nume: %s  An: %d  Nota medie: %.2f \n", s.nume, s.an, s.medie);}
Exemplu de apel: showStudent(sarr[i]);
```

- Funcția de comparare a două structuri după un anumit câmp al structurii:

```
int compNume(Student s1, Student s2){        // sau compMedie
if(stricmp(s1.nume, s2.nume) > 0) return 1; // sau if(s1.medie>s2.medie)
if(stricmp(s1.nume, s2.nume) < 0) return -1;// sau if(s1.medie<s2.medie)
return 0;
}
```

2. Operații cu tablouri de structuri folosind funcții și pointeri

De multe ori, tablourile de structuri se folosesc pentru a implementa baze de date. O bază simplă de date este constituită dintr-un tablou unidimensional de tip structură (adică tablou de structuri) cu câmpurile de tipul necesar și permite diverse operații pe baza de date și anume:

Alocarea dinamică a memoriei pentru tabloul de structuri. Introducerea informației despre elementele tabloului de la tastatură. Afişarea informației despre elementele tabloului la ecran. Căutarea elementului în tablou. Modificarea câmpurilor unui element din tablou. Interschimbarea a două elemente din tablou. Sortarea tabloului de structuri. Adăugarea unui element la sfârșitul tabloului. Adăugarea unui element la începutul tabloului. Înserarea unui element după elementul indicat al tabloului. Înserarea unui element înaintea elementului indicat al tabloului. Ştergerea elementului indicat al tabloului. Salvarea informației despre elementele tabloului în fişier. Citirea informației despre elementele tabloului din fişier. Eliberarea memoriei alocate pentru tabloul de structuri și alte operații privind determinarea unor informații (statistici) din baza de date.

Operații cu tablouri de structuri folosind funcții și pointeri

Toate aceste operații (opțiuni) pot fi implementate într-un program al limbajului C pentru prelucrarea unei baze de date a unei structuri prin crearea funcțiilor respective ale operațiilor și apoi prin apelarea lor din funcția principală **main ()** într-o anumită ordine. De obicei, ordinea și numărul acestor apeluri depind și sunt determinate de utilizator în timpul unei sesiuni de lucru cu baza de date. De aceea, este necesară dezvoltarea unei interfețe de utilizator adecvate pentru comunicarea interactivă dintre utilizator și program. Să luăm în considerare procesul de dezvoltare în limbajul C a unei interfețe simple pentru utilizator bazate pe utilizarea instrucțiunii de selecție „switch” și ciclului infinit, care oferă o posibilitate de a afișa repetat pe ecran a unui meniu simplu de operații (opțiuni) în timpul unei sesiuni de lucru cu baza de date.

Anterior, am implementat funcțiile care definesc operații cu un singur element al tabloului de structuri. Mai departe demonstrăm prototipurile funcțiilor, care definesc operațiile respective de prelucrare a întregului tablou de structuri (adică operații pentru lucrul cu baza de date).

Operații cu tablouri de structuri folosind funcții și pointeri

Pentru alocarea dinamică a tabloului utilizăm funcția **`calloc()`** din biblioteca standard așa cum a fost demonstrat anterior. Exemplu:

```
typedef struct student{
char nume[40]; // sau char *nume;
int an;
float medie;
} Student;
Student *s; int n;
. . .
s=(Student*)calloc(n, sizeof(Student));
if (s == NULL) exit(1);
```

Funcţiile care definesc operaţii cu tabloul de structuri

- **Funcția pentru introducerea informației despre elementele tabloului de la tastatură:**

```
void readInfo(Student *s, int n);
```

- **Funcția pentru afişarea informației despre elementele tabloului la ecran:**

```
void showInfo(Student *s, int n);
```

- **Funcția pentru căutarea elementului în tablou:**

```
int searchStud(Student *s, int n, char *num);
```

- **Funcția pentru modificarea câmpurilor unui element din tablou:**

```
void modifyInfo(Student *s, int k);
```

- **Funcția pentru interschimbarea a două elemente din tablou:**

```
void swapStud(Student *s, int k1, int k2);
```

Funcţiile care definesc operaţii cu tabloul de structuri

- **Funcția pentru sortarea tabloului de structuri:**

```
void sortStud(Student* s, int n);
```

- Funcția pentru adăugarea unui element la sfârșitul tabloului:

```
Student* appendStud (Student *s, int *pn, Student a);
```

- **Funcția pentru adăugarea unui element la începutul tabloului:**

```
Student* prependStud (Student *s, int *pn, Student b);
```

- **Funcția pentru înserarea unui element după elementul indicat al tabloului:**

```
Student* insertAfter(Student *s, int *pn, int k, Student a);
```

Funcţiile care definesc operaţii cu tabloul de structuri

- **Funcția pentru înserarea unui element înaintea elementului indicat al tabloului:**

```
Student* insertBefore(Student *s, int *pn, int k, Student a);
```

- **Funcția pentru ștergerea elementului indicat al tabloului:**

```
Student* deleteStud(Student *s, int *pn, int k);
```

- **Salvarea informației despre elementele tabloului în fişier:**

```
int saveInfo(Student* s, int n, char *fname);
```

- **Citirea informației despre elementele tabloului din fişier:**

```
int loadInfo(Student* s, int n, char *fname);
```

- **Eliberarea memoriei alocate pentru tabloul de structuri:**

```
void freeArray (Student *s, int n);
```

Funcţiile care definesc operaţii cu tabloul de structuri

- **Funcția pentru înserarea unui element înaintea elementului indicat al tabloului:**

```
Student* insertBefore(Student *s, int *pn, int k, Student a);
```

- **Funcția pentru ștergerea elementului indicat al tabloului:**

```
Student* deleteStud(Student *s, int *pn, int k);
```

- **Salvarea informației despre elementele tabloului în fişier:**

```
int saveInfo(Student* s, int n, char *fname);
```

- **Citirea informației despre elementele tabloului din fişier:**

```
int loadInfo(Student* s, int n, char *fname);
```

- **Eliberarea memoriei alocate pentru tabloul de structuri:**

```
void freeArray(Student *s, int n);
```

Problema propusă spre rezolvare pentru organizarea lucrului cu tabloul de structuri

Să se scrie un program pentru lucrul cu baza de date în formă unui tablou de structuri de tip dat cu elaborarea funcțiilor care definesc diferite operații cu tabloul de structuri.

Tutoriale online pentru tema prelegerii:

1. https://docs.google.com/document/d/1Z5GKvAlQC1KqR4F6uCocBUceOkn11mqCoXdPT4pY2wo/edit
2. https://www.pbinfo.ro/articole/7656/structuri-tablouri-si-functii
3. https://www.scritub.com/stiinta/informatica/c/Tablouri-de-structuri1837241914.php
4. http://www.eed.usv.ro/~pentiuc/cursul5.html
5. https://sites.google.com/site/razvanaciu/programarea-calculatoarelor/programarea-calculatoarelor---12---structuri
6. https://info64.ro/Structuri_de_date-Uniuni/
7. http://www.aut.upt.ro/~rraul/PC/2009-2010/PC_lab9.pdf
8. http://www.cs.ucv.ro/staff/gmarian/Programare/cap8_Structuri.pdf
9. http://old.elearning.usarb.md/moodle/pluginfile.php/11484/mod_resource/content/2/CLEC8_TDM.pdf
10. https://ocw.cs.pub.ro/courses/programare/laboratoare/lab11

VĂ MULȚUMESC PENTRU ATENȚIE!

MULTĂ SĂNĂTATE ȘI SUCCESE!