LUCRARE DE LABORATOR NR. 1
LA DISCIPLINA „PROGRAMAREA ÎN LIMBAJUL C++”

Tema: Structura – mecanism de abstractizare

Scopul lucrării:

- Studierea programării prin abstractizarea datelor;
- Studierea regulilor de definire şi utilizare a structurilor de date;
- Crearea variabilelor de tip structură, accesarea componentelor unei structuri.

Noţiuni de bază

Structura – este o mulţime de date grupate, conform unei ierarhii, de obicei de tipuri diferite. Structura poate fi comparată cu tabloul, însă tabloul conţine date de un acelaşi tip, iar elementele tabloului se accesează cu ajutorul indicilor, pe cînd accesarea componentelor structurii are o formă specifică.
Structura reprezintă un tip abstract de date, un tip definit de utilizator prin utilizarea tipurilor deja existente.
La stabilirea şi implementarea conceptelor necesare rezolvării unei probleme se realizează un proces de abstractizare care priveşte reprezentarea datelor şi prelucrarea acestora. Aceasta justifică termenul de *programare prin abstractizarea datelor*.
Fie definiţia de structură:

```
struct Book{
// definirea elementelor structurii
   char *author;
   char *title;
   int year;
   int pages;
};
```

înaintea cuvîntului cheie *struct* uneori se adaugă cuvîntul cheie
typedef. Însă această sintaxă este proprie limbajului C, şi nu C++. Elementele structurii pot fi de tip predefinit sau definit de utilizator, excepţie fiind tipul structurii date. Pentru definirea variabilelor este suficientă utilizarea numelui de structură la fel ca tipurile predefinite:

```
Book b1, b2, bs[10], *bptr;
```

În acest caz s-au definit două variabile de tip carte, un tablou din 10 cărţi şi un pointer spre carte.
Referirea la componentele unei structuri se face utilizînd atît numele structurii cît şi a componentei respective. Aceasta se realizează printr-o construcţie de forma:

```
b1.pages = 153;
bs[i].pages = 24;
```

În cazul pointerilor:

```
bptr = new Book;    // alocarea memoriei pentru o variabilă nouă
bptr->pages = 176;
```

Fie definiţia de structură:

```
struct Date{
 int day, month, year;
};
struct Student{
 char *name;
 Date birthDay;
 float media;
};
```

La definirea tipului de strutură este bine de a preciza funcţiile care definesc operaţii cu datele de tipul structurii. În acest scop variabilele se transmit funcţiilor prin parametri. Pentru modificarea variabilei ea se transmite prin adresa.

```
void setValues(Student* sptr, char* newN,float newM,Date d)
     {
sptr->name = new char[strlen(newN)+1];
strcpy(sptr->name,newN);
sptr->media = newM;
sptr->birthDay = d;
}
```

Funcţia de mai sus are un neajuns legat de alocarea memoriei. Deoarece funcţia este obişnuită, ea poate fi apelată de cîteva ori, ceea ce poate duce la pierderi de memorie. Pentru soluţionarea acestei probleme, mai întîi trebuie dezalocată memoria iniţială (sau să se înregistreze posibilitatea unui singur apel al funcţiei, ceea ce se reflectă în denumirea funcţiei).

```
void setValues(Student* sptr, char* newN,float newM,Date d)
{
if (sptr->name) delete[] sptr->name;
sptr->name = new char[strlen(newN)+1];
...
}
```

Dar şi această soluţie are neajunsurile sale, posibil mai grave decît în primul caz. Pericolul constă în încercarea de a elimina memoria nealocată la primul apel al funcţiei. Pot fi cîteva soluţii. Una constă în utilizarea unui parametru suplimentar, fie cu valoare implicită, care indică dacă funcţia se apelează pentru prima dată(pentru obiectul dat) sau nu. Această variantă nu este tocmai reuşită, este necesar de a urmări succesiunea apelurilor, sarcină realizată de către programator, în al doilea rînd, funcţia trebuie să conţină o condiţie ceea ce încetineşte îndeplinirea funcţiei.
Altă soluţie ar fi în setarea valorii iniţiale a pointerului. În acest scop, după crearea obiectului, componentelor pointer li se atribuie valoarea NULL;

```
Student s;
s.name = NULL;
```

Pentru finisarea exemplului, este necesar de a elibera memoria alocată iniţial. Pentru aceasta se creează încă o funcţie:

```
void freeMem(Student* sptr){
 if (sptr->name) delete[] sptr->name;
```

```
 sptr->name = NULL;
 ... // analogic pentru restul componentelor pointer
}
```

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

Întrebări de control:

1. Definiţi noţiunea – tip abstract de date.
2. Cum se defineşte o structură?
3. Prin ce se deosebeşte structura de alte tipuri de date?
4. Cum se defineşte o variabilă de tip structură?
5. Cînd se utilizează punctul, cînd - săgeata?
6. Care sunt deosebirile între structura din limbajul C şi C++?
7. O structură poate oare să conţină altă structură?
8. O structură poate oare să conţină pointer spre ea însăşi?
9. Poate oare să fie creată dinamic o variabilă de tip structură?
10. Explicaţi cum are loc apelul prin referinţă.

Sarcina
Varianta 1

a) Să se creeze; utilizînd structura, tipul abstract de date pentru reprezentarea numerelor complexe. Să se definească funcţiile de setare/citire a părţii reale şi imaginare a unui număr complex; de adunare, scădere, înmulţire, împărţire şi de comparaţie(mai mic, mai mare, etc.) a două numere complexe. Să se definească funcţia ce calculează şi returnează modulul unui număr complex.

b) Să se creeze tipul abstract de date (structura) – vector, care se compune dintr-un pointer spre *int* şi din numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a unui vector; de setare/ modificare a dimensiunii vectorului; de acces la elementele vectorului; de calcul a modulului unui vector. În funcţia *main* să se realizeze adunarea a doi vectori.

Varianta 2

a) Să se creeze tipul abstract de date (structura) – cîine, care se compune din rasă, nume şi vîrstă. Să se definească funcţiile de setare şi modificare a datelor şi de eliberare a memoriei alocate. Pentru iniţializarea componentelor textuale să se utilizeze operatorul de alocare dinamică a memoriei *new*. Să se definească funcţia pentru sortarea tabloului de tip cîine conform vîrstei şi numelui. Adică cîinii de aceeaşi vîrstă se sortează în ordine alfabetică.

b) Să se creeze tipul abstract de date (structura) – vector, care se compune dintr-un pointer spre *float* şi din numărul de elemente. Să se definească funcţiile: de iniţializare, de eliminare a unui vector; de setare/ modificare a dimensiunii vectorului; de acces la elementele vectorului; de calcul a sumei elementelor unui vector. În funcţia *main* să se exemplifice înmulţirea a doi vectori.

Varianta 3

a) Să se creeze tipul abstract de date (structura) – computer, care se compune din informaţia despre firma producătoare, procesor, frecvenţa de tact, etc. să se definească funcţiile de setare a datelor, de modificare a informaţiei, de comparaţie. Pentru iniţializarea componentelor textuale să se utilizeze operatorul de alocare dinamică a memoriei *new*. Să se elibereze memoria alocată. În funcţia *main* să se exemplifice sortarea tabloului de tip calculator după cîmpurile frecvenţa de tact + procesor.

b) Să se creeze tipul abstract de date (structura) – vector, care se compune dintr-un pointer spre *double* şi din numărul de elemente. Să se definească funcţiile: de iniţializare, de eliminare a unui vector; de setare/ modificare a dimensiunii vectorului; de acces la elementele vectorului; de calcul a sumei elementelor pozitive ale unui vector. În funcţia *main* să se exemplifice înmulţirea unui vector cu un număr.

Varianta 4

a) Să se creeze tipul abstract de date (structura) – ţara, care are denumire, continentul pe care se află şi numărul de locuitori. Să se definească funcţiile de setare a denumirii ţării şi a numărului de populaţie, de modificare a datelor, de comparare a ţărilor şi de eliberare a memoriei. Să se utilizeze operatorul *new* pentru setarea denumirii ţării. În *main* să se exemplifice căutarea ţărilor după denumire şi populaţie.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *int* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a mediei elementelor pozitive ale vectorului. Ca exemplu, în funcţia *main* să se realizeze compararea a doi vectori.

Varianta 5

a) Să se creeze tipul abstract de date (structura) – colaborator, care are nume, specialitate, categorie şi salariu. Să se definească funcţiile de setare, modificare a datelor şi de comparare a colaboratorilor. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice sortarea colaboratorilor după diverse criterii.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *long* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a sumei elementelor negative ale vectorului. Ca exemplu, în funcţia *main* să se realizeze adunarea a doi vectori.

Varianta 6

a) Să se creeze tipul abstract de date (structura) – casă, care are denumirea firmei de construcţie, adresa, numărul etajelor şi al apartamentelor. Să se definească funcţiile de setare, de modificare a datelor, de comparare a caselor. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice sortarea caselor după numărul de etaje+adresa în ordine alfabetică.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *float* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a normei vectorului. Ca exemplu, în funcţia *main,* să se realizeze adunarea elementelor vectorului cu un număr.

Varianta 7

a) Să se creeze tipul abstract de date (structura) – monitor, care are denumirea firmei producătoare, dimensiunea în inch, numărul de culori şi rezoluţie. Să se definească funcţiile de setare, de modificare a datelor, de comparare a monitoarelor. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice căutarea monitorului potrivit după dimensiune şi alte caracteristici.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *char* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a produsului elementelor negative ale vectorului. Ca exemplu, în funcţia *main,* să se realizeze înmulţirea a doi vectori.

Varianta 8

a) Să se creeze tipul abstract de date (structura) – carte, care are denumire, autor, editura, volumul de pagini şi anul ediţiei. Să se definească funcţiile de setare, de modificare a datelor, de comparare. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice căutarea cărţii necesare după denumire şi după autor.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *short* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a sumei elementelor pare ale vectorului. Ca exemplu, în funcţia *main,* să se realizeze compararea a doi vectori.

Varianta 9

a) Să se creeze tipul abstract de date (structura) – student, care are nume, specialitate, anul de studii şi balul mediu. Să se definească funcţiile de setare, de modificare a datelor, de comparare. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice sortarea studenţilor după specialitate+reuşită.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *long* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a sumei elementelor pare ale vectorului. Ca exemplu, în funcţia *main*, să se realizeze căutarea poziţiei elementului maximal al vectorului.

Varianta 10

a) Să se creeze tipul abstract de date (structura) – produs soft (*software*), care are denumire, firma de producere, anul editării şi versiunea. Să se definească funcţiile de setare, de modificare a datelor, de comparare. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice căutarea produsului necesar după cîteva criterii.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *double* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a sumei elementelor pare ale vectorului. Ca exemplu, în funcţia *main*, să se realizeze adunarea unui vector cu un număr.

Varianta 11

a) Să se creeze tipul abstract de date (structura) – firma, care are denumire, forma de organizare, adresa şi anul fondării. Să se definească funcţiile de setare, de modificare a datelor, de comparare. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice căutarea după cîteva criterii a firmei necesare.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *int* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a sumei elementelor pare ale vectorului. Ca exemplu, în funcţia *main*, să se realizeze adunarea a doi vectori.

Varianta 12

a) Să se creeze tipul abstract de date (structura) – fişier, care are denumire, data şi timpul creării şi dimensiunea. Să se definească funcţiile de setare, de modificare a datelor, de comparare. Pentru crearea cîmpurilor textuale să se utilizeze operatorul *new*. Să se elibereze memoria. În *main* să se exemplifice sortarea fişierelor după cîteva criterii.

b) Să se creeze tipul abstract de date (structura) – vector, care are pointer spre *float* şi numărul de elemente. Să se definească funcţiile de iniţializare, de eliminare a vectorului, de setare/modificare a dimensiunii, de acces la elementele vectorului, de calcul a sumei elementelor pare ale vectorului. Ca exemplu, în funcţia *main*, să se realizeze căutarea poziţiei elementului minimal al vectorului.