POINTERI. ALOCAREA DINAMICĂ A MEMORIEI

1. Pointeri

Un pointer este o variabilă care conţine adresa altei variabile sau a unei funcţii dacă este iniţializat cu această adresă.

2. Alocarea dinamică a memoriei

void *malloc(size_t n);

– alocă un bloc de memorie de **n** octeţi în heap (zonă de memorie folosită pentru alocări dinamice);

– în caz de succes, returnează un pointer la blocul de memorie alocat (căruia nu-i schimbă conţinutul)

– returnează valoarea 0 (zero) dacă nu s-a putut face alocarea de memorie (pentru că nu există spaţiu liber cu mărimea solicitată sau dacă se apelează cu n = 0).

Funcţia returnează un pointer generic de tip void, a cărui valoare poate fi asignată unui pointer de orice tip folosind o conversie explicită. Se recomandă să se testeze totdeauna dacă valoarea returnată este sau nu 0 (zero).

void *calloc(size_t nrElemente, size_t dimElement);

– alocă un bloc de memorie de mărime nrElemente * dimElement (care nu trebuie să depăşească 64 Ko octeţi din heap), conţinutul blocului fiind resetat (se scrie 0);

– în caz de succes, returnează un pointer la blocul de memorie alocat;

– returnează 0 dacă nu există spaţiu liber de mărimea solicitată (sau dacă se apelează cu valorea 0).

void *realloc(void* block, size_t marime);

– funcţia redimensionează (prim mărire sau micşorare) un bloc de memorie (alocat dinamic anterior) la numărul de octeţi specificaţi de parametrul **marime**;

– block trebuie să indice un bloc de memorie obţinut prin apelarea funcţiilor malloc() / calloc() / realloc() (altfel rezultatul este imprevizibil);

– dacă block este 0 (zero), lucrează exact ca malloc();

– funcţia ajustează mărimea blocului alocat la **marime**, copiind (dacă este cazul) conţinutul său la o nouă adresă;

– returnează adresa blocului realocat (poate diferi de block) sau 0 (zero), dacă nu se poate face realocarea sau marime = 0 (în acest ultim caz funcţia lucrează ca şi funcţia free()).

void free(void *block);

– eliberează un bloc de memorie alocat anterior de către malloc() / calloc() / realloc();

– rezultatul este dezastruos dacă parametrul nu este rezultatul unei alocări dinamice anterioare sau dacă se apelează de doua ori la rând cu acelaşi parametru.

TEMA

Problema nr. 1

Să se citească de la tastatură elementele a două tablouri unidimensionale (vectori) de **n** numere întregi **a** şi **b** alocate dinamic. Să se calculeze vectorul sumă şi să se afişeze.

Rezolvarea problemei presupune construirea unui proiect (cu fişierul header corespunzător) şi scrierea următoarelor funcţii:

- ✓ funcţie pentru: alocarea cu verificarea alocării (funcţia **xmalloc** din curs);
- ✓ funcţie pentru citirea unui vector de întregi cu **n** elemente. Funcţia trebuie să aibă următorul prototip:

  void citireVector(int *a, size_t n);
- ✓ funcţie pentru afişarea elementelor unui vector de întregi, cu prototipul:

  void afisareVector(int *a, size_t n);
- ✓ funcţie pentru calculul vectorului sumă a doi vectori, cu prototipul:

  int *sumaVectori(int *x, int *y, size_t n);

Problema nr. 2

Se citeşte de la tastatură un număr natural **n** şi un vector de numere naturale **v** pentru care se face alocare dinamică de memorie.

Se formează un nou vector **w** cu **n** elemente, de asemenea alocat dinamic, în care valoarea fiecărui element este suma cifrelor elementului corespunzător din vectorul **v**.

Să se afişeze elementul din vectorul **v** care are cea mai mare sumă a cifrelor sale.

Se vor scrie următoarele funcţii:

- ✓ Funcţie pentru citirea unui vector de numere naturale care are ca parametru numărul de elemente şi returnează un pointer.
- ✓ Funcţie pentru afişarea vectorului de numere naturale sub forma

A = (23, 543, 912)

Funcţia are ca parametri vectorul de afişat (exprimat ca un pointer) şi numărul de elemente.

✓ Funcţie pentru calculul sumei cifrelor unui număr natural. Funcţia are ca parametru un număr natural (numărul pentru care se calculează suma cifrelor) şi returnează un număr natural (suma calculată).

✓ Funcţie pentru determinarea elementelor vectorului **w**. Funcţia are ca parametri un pointer şi un număr natural şi returnează un pointer la un număr natural. Această funcţie face apel, pentru calculul sumei cifrelor unui număr natural, la funcţia definită la punctul anterior.

✓ Funcţie pentru determinarea maximului dintr-un şir de numere. Funcţia primeşte ca parametri un pointer la un număr natural şi un întreg şi returnează indexul elementului cu valoarea maximă.

Problema nr. 3

Să se definească tipul de dată MULTIME ca o structură care cuprinde:

- cardinalul mulţimii (numărul de elemente din mulţime) care este un număr întreg fără semn;
- elementele mulţimii (de tip real), stocate prin intermediul unui pointer

Să se scrie un program care pentru două mulţimi

1. citeşte de la tastatură cardinalul şi elementele mulţimii şi le stochează într-o structură de tip MULTIME.
2. afişează cele două mulţimi sub forma

   A = {3.14, 5.12, 3.00, 4.39}

   B = {34.29, 15.14, 3.14}

(se presupune că numele celor două mulţimi sunt **A** şi **B**, iar numerele reale se vor afişa cu două zecimale şi vor fi separate de o virgulă şi un spaţiu).

Observaţie (referitoare la punctele 1 şi 2):

Se scriu funcţii pentru o singură mulţime. Funcţiile se apelează de mai multe ori.

3. afişează un meniu care dă posibilitatea utilizatorului să aleagă una din următoarele prelucrări:

a) determinarea mulţimii intersecţie a celor două mulţimi (**A&B**) şi cardinalul acestei mulţimi.

b) determinarea mulţimii diferenţă simetrică a celor două mulţimi (**A-B**) (mulţimea diferenţă simetrică include elementele mulţimii **A** care nu sunt în mulţimea **B** şi elementele mulţimii **B** care nu sunt în mulţimea **A**) şi cardinalul acestei mulţimi.

c) determinarea mulţimii reuniune a celor două mulţimi (**A+B**) şi cardinalul acestei mulţimi.

4. Realizează prelucrarea dorită şi afişează mulţimea care rezultă în urma prelucrării.

Programul poate face o singură prelucrare în funcţie de opţiunea utilizatorului şi va trebui scris astfel încât să permită prelucrarea mai multor seturi de date.

Observaţii:

1) Se va citi cu atenţie tabelul următor care descrie în detaliu modul de rezolvare a problemei şi descrierea prototipurilor funcţiilor care trebuie scrise.

2) Dacă nu se folosesc funcţiile indicate codul nu se punctează.

3) Punctajul maxim se acordă pentru rezolvarea **CORECTĂ** a fiecărei subprobleme.

1. Construirea structurii MULTIME (funcţia primeşte ca parametru numele mulţimii şi returnează o structură de tip MULTIME)	1,0
2. Alocarea corectă (cu verificare) de spaţiu de memorie pentru pointeri	0,5
3. Citirea valorilor elementelor mulţimii (într-o funcţie care primeşte ca parametru un număr natural şi returnează un pointer la real)	1,0
4. Afişarea valorilor vectorului din structură conform modelului dat în problemă (funcţia are un parametru – o structură de tip MULTIME şi nu returnează nimic)	1,0
5. Scrierea meniului de prelucrare (funcţia nu are nici un parametru şi returnează un număr care reprezintă numărul opţiunii de prelucrare)	0,75
6. posibilitatea de reluare a programului (prelucrarea mai multor seturi de date)	0,5
7. Folosire proiect (corect)	0,5
8. Fişier header (corect şi complet)	0,5
9. Funcţia main (complet – inclusiv eliberarea corectă a zonelor de memorie folosite)	0,75
10. Funcţie care stabileşte dacă o valoare aparţine unei mulţimi. Funcţia are 2 parametri: valoarea şi mulţimea (reprezentată prin structură) şi returnează 1/0 după cum valoarea se găseşte sau nu în mulţime.	0,5
11. Calculul mulţimii intersecţie (funcţia primeşte ca parametri două structuri de tip MULTIME şi returnează o structură de tip MULTIME) – inclusiv alocarea corectă de memorie.	1,0

12. Calculul mulţimii diferenţă (funcţia primeşte ca parametri două structuri de tip MULTIME şi returnează o structură de tip MULTIME) – inclusiv alocarea corectă de memorie.	1,0
13. Calculul mulţimii reuniune (funcţia primeşte ca parametri doi pointeri la date de tip MULTIME şi returnează o structură de tip MULTIME) – inclusiv alocarea corectă de memorie.	1,0
TOTAL	**10 p**

Problema nr. 4

Se citeşte de la tastatură un număr întreg (poate fi pozitiv sau negativ). Citirea se face cu scanf. Se determină numărul de cifre ale numărului (prin logaritmare în baza 10).

Se face alocare dinamică de memorie pentru un şir de caractere capabil să memoreze cifrele numărului şi eventualul semn. Numărul citit se transformă în şir de caractere, fără a folosi funcţii de bibliotecă. Transformarea se face astfel încât fiecare cifră să se plaseze direct pe locul ei.

Se afişează şirul de caractere rezultat.

Rezolvarea problemei presupune realizarea unui proiect care, pe lângă funcţia **main**, să cuprindă şi funcţii pentru:

- ✓ alocare dinamică cu verificare (funcţia **xmalloc** din curs)
- ✓ transformarea numărului în şir de caractere. În acest caz funcţia are prototipul:

```
char *transformareToAscii(long int);
```