Laborator 2

Instructiuni generale

Acceptati invitatia pe GitHub Classroom pentru laborator, link:

- https://classroom.github.com/a/2qsWWVAV

Dupa accept, clonati repository-ul pe calculator:

- git clone <link_repository>

Creati fisierele main.cpp, functions.h si functions.cpp.
Toata logica programului trebuie implementata in functions.cpp, cu prototipurile in functions.h.
In main.cpp scrieti doar apelurile si testarea functiilor.
Fiecare functie din functions.cpp trebuie sa aiba un comentariu deasupra care explica ce face.
La final, urcati modificarile:

- git add .
- git commit -m "Laborator 2"
- git push

Task 0 – Creare fisiere si implementare functii (3 puncte)

Creati fisierele main.cpp, functions.cpp, functions.h si adaugati urmatoarele prototipuri in functions.h (aceste functii trebuie implementate in functions.cpp):

// Aloca dinamic un vector de dimensiune n
int* alocareVector(int n);

// Elibereaza memoria ocupata de vector
void eliberareVector(int* v);

// Calculeaza suma elementelor vectorului
int sumaVector(const int v[], int n);

// Realoca memoria vectorului, modificand vectorul si dimensiunea prin referinta
int* realocareVector(int*& v, int& dimVeche, int dimNoua);

// Afiseaza un vector de intregi
void afisareVector(const int v[], int n);

// Scrie vectorul intr-un fisier: primul rand = numarul de elemente, al doilea rand = elementele separate prin spatiu
void scriereVectorFisier(const int v[], int n, const char* filename);

// Citeste vectorul dintr-un fisier: primul rand = numarul de elemente, al doilea rand = elementele, returneaza vector alocat dinamic
int* citireVectorFisier(int& n, const char* filename);

Task 1 – Alocare si eliberare memorie (1.5 puncte)

---

1. In main.cpp, citeste n de la tastatura.
2. Apeleaza functia alocareVector(int n) pentru a crea vectorul.
3. Citeste n valori de la tastatura si salveaza-le in vector.
4. Afiseaza vectorul folosind afisareVector().
5. Elibereaza memoria cu eliberareVector().

Task 2 – Functie de realocare memorie (1 punct)

---

1. Foloseste functia realocareVector() pentru a mari dimensiunea vectorului.
2. Parametrii vector si dimensiune sunt transmisi prin referinta, astfel se modifica direct in main.
3. Citeste valorile pentru noile elemente adaugate.
4. Afiseaza vectorul complet dupa realocare.

Task 3 – Scriere si citire vector in/din fisier (1.5 puncte)

---

1. Foloseste functia scriereVectorFisier() pentru a salva vectorul intr-un fisier.
   Format fisier: primul rand = numarul de elemente, al doilea rand = elementele separate prin spatiu.
2. Foloseste functia citireVectorFisier() pentru a citi vectorul inapoi in program.
3. Afiseaza vectorul citit din fisier pentru verificare.

Task 4 – Scenariu de memorie ne-eliberata (Valgrind) (1 punct)

---

1. Comenteaza temporar apelul eliberareVector().
2. Ruleaza programul cu Valgrind si observa mesajul “definitely lost” care indica memorie ne-eliberata.

Task 5 – Scenariu de acces invalid (Valgrind) (1 punct)

---

1. Dupa ce eliberezi vectorul, incearca sa accesezi un element din el.
2. Ruleaza programul cu Valgrind si observa mesajul “Invalid read/write”.

Cum se compileaza si ruleaza

1. Compileaza programul:
   - g++ -g main.cpp functions.cpp -o lab2

2. Ruleaza:
   - ./lab2

3. Ruleaza cu Valgrind:
   - valgrind ./lab2

Exemple de mesaje:
- “definitely lost” → memorie ne-eliberata
- “Invalid read/write” → acces invalid dupa eliberare sau in afara vectorului

---

Punctaj:

- **Task 0:** 3 puncte
- **Task 1:** 1.5 puncte
- **Task 2:** 1 punct
- **Task 3:** 1.5 puncte
- **Task 4:** 1 punct
- **Task 5:** 1 punct

1 punct din oficiu

Total 10 puncte