

**MINISTERUL EDUCAȚIEI, CULTURII ȘI CERCETĂRII**

**AL REPUBLICII MOLDOVA**

**Universitatea Tehnică a Moldovei**

**Facultatea Calculatoare, Informatică şi Microelectronică**

**Departamentul Informatică şi Ingineria Sistemelor**

**gr. IA-231, Chistol Maxim**

**Raport**

**pentru lucrarea de laborator Nr.1**

***la cursul de “Programarea Orientată pe Obiecte”***

Verificat:

Kulev Mihail dr., conf. univ.

Departamentul Informatică şi IS,

Facultatea FCIM, UTM

**Chișinău 2024**

Cuprins:

[**Tema:** 3](#_Toc179441511)

[**Scopul lucrării:** 3](#_Toc179441512)

[**Partea teoretică și explicativă:** 3](#_Toc179441513)

[**Sarcina Individual:** 3](#_Toc179441514)

[**Listingul programului:** 4](#_Toc179441515)

[**Testarea și verificarea programelor cu rezultatele obținute:** 13](#_Toc179441516)

[**Concluzii:** 14](#_Toc179441517)

[**Bibliografie:** 15](#_Toc179441518)

**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.

# **Partea teoretică și explicativă:**

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.

# **Sarcina Individual:**

**Varianta 7**

а) 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.

# **Listingul programului:**

**a)**

**#include <iostream>**

**#include <cstring>**

**#include <string>**

**using namespace std;**

**struct Monitor {**

**char\* firmaProducatoare;**

**float dimensiuniInch;**

**int numarCulori;**

**string rezolutie;**

**};**

**void setareDate(Monitor& monitor, const char\* firma, float dimensiune, int culori, const string& rez) {**

**monitor.firmaProducatoare = new char[strlen(firma) + 1];**

**strcpy(monitor.firmaProducatoare, firma);**

**monitor.dimensiuniInch = dimensiune;**

**monitor.numarCulori = culori;**

**monitor.rezolutie = rez;**

**}**

**void modificareDate(Monitor& monitor, const char\* firma, float dimensiune, int culori, const string& rez) {**

**delete[] monitor.firmaProducatoare;**

**monitor.firmaProducatoare = new char[strlen(firma) + 1];**

**strcpy(monitor.firmaProducatoare, firma);**

**monitor.dimensiuniInch = dimensiune;**

**monitor.numarCulori = culori;**

**monitor.rezolutie = rez;**

**}**

**bool compara(const Monitor& monitor, const Monitor& altMonitor) {**

**return (monitor.dimensiuniInch == altMonitor.dimensiuniInch &&**

**monitor.numarCulori == altMonitor.numarCulori &&**

**monitor.rezolutie == altMonitor.rezolutie);**

**}**

**void elibereazaMemoria(Monitor& monitor) {**

**delete[] monitor.firmaProducatoare;**

**}**

**int main() {**

**Monitor monitor1, monitor2;**

**// Setare date pentru monitor1 și monitor2**

**setareDate(monitor1, "Dell", 24.5, 16700000, "1920x1080");**

**setareDate(monitor2, "HP", 27.0, 16700000, "2560x1440");**

**if (compara(monitor1, monitor2)) {**

**cout << "Monitoarele sunt identice" << endl;**

**} else {**

**cout << "Monitoarele difera" << endl;**

**}**

**// Citirea noilor date de la tastatură pentru modificarea monitorului 2**

**char firma[50];**

**float dimensiune;**

**int culori;**

**string rezolutie;**

**cout << "Introduceti noua firma producatoare: ";**

**cin >> firma;**

**cout << "Introduceti noua dimensiune in inch: ";**

**cin >> dimensiune;**

**cout << "Introduceti noul numar de culori: ";**

**cin >> culori;**

**cout << "Introduceti noua rezolutie: ";**

**cin >> rezolutie;**

**// Modificare monitor2 cu datele introduse de utilizator**

**modificareDate(monitor2, firma, dimensiune, culori, rezolutie);**

**if (compara(monitor1, monitor2)) {**

**cout << "Monitoarele sunt identice" << endl;**

**} else {**

**cout << "Monitoarele difera" << endl;**

**}**

**// Eliberare memorie alocată**

**elibereazaMemoria(monitor1);**

**elibereazaMemoria(monitor2);**

**return 0;**

**}**

**b)**

**#include <iostream>**

**#include <cstring>**

**#include <cstdlib>**

**#include <ctime>**

**#include <algorithm>**

**using namespace std;**

**struct Vector {**

**int\* elements;**

**int count;**

**};**

**void init(Vector& vector, int dimensiune) {**

**vector.count = dimensiune;**

**vector.elements = new int[vector.count];**

**}**

**void freememor(Vector& vector) {**

**delete[] vector.elements;**

**vector.elements = nullptr;**

**vector.count = 0;**

**}**

**void moddimensiune(Vector& vector, int newcount) {**

**int\* newElements = new int[newcount];**

**memcpy(newElements, vector.elements, min(vector.count, newcount) \* sizeof(int));**

**delete[] vector.elements;**

**vector.elements = newElements;**

**vector.count = newcount;**

**}**

**int produsElementeNegative(const Vector& vector) {**

**int produs = 1;**

**bool hasNegatives = false;**

**for (int i = 0; i < vector.count; i++) {**

**if (vector.elements[i] < 0) {**

**produs \*= vector.elements[i];**

**hasNegatives = true;**

**}**

**}**

**return hasNegatives ? produs : 0;**

**}**

**void inmultireVectori(const Vector& v1, const Vector& v2, Vector& rezultat) {**

**if (v1.count != v2.count) {**

**cout << "Dimensiunile vectorilor nu sunt egale!" << endl;**

**return;**

**}**

**init(rezultat, v1.count);**

**for (int i = 0; i < v1.count; i++) {**

**rezultat.elements[i] = v1.elements[i] \* v2.elements[i];**

**}**

**}**

**int main() {**

**srand(time(0));**

**Vector vector1, vector2, rezultat;**

**int dimensiune;**

**cout << "Dimensiunea vectorilor: ";**

**cin >> dimensiune;**

**init(vector1, dimensiune);**

**init(vector2, dimensiune);**

**cout << "Vector 1: ";**

**for (int i = 0; i < dimensiune; i++) {**

**vector1.elements[i] = rand() % 21 -10;**

**cout << vector1.elements[i] << " ";**

**}**

**cout << endl;**

**cout << "Vector 2: ";**

**for (int i = 0; i < dimensiune; i++) {**

**vector2.elements[i] = rand() % 21 -10;**

**cout << vector2.elements[i] << " ";**

**}**

**cout << endl;**

**inmultireVectori(vector1, vector2, rezultat);**

**cout << "Rezultatul înmulțirii: ";**

**for (int i = 0; i < dimensiune; i++) {**

**cout << rezultat.elements[i] << " ";**

**}**

**cout << endl;**

**int produsNegativi = produsElementeNegative(vector1);**

**cout << "Produsul elementelor negative din Vector 1: " << produsNegativi << endl;**

**freememor(vector1);**

**freememor(vector2);**

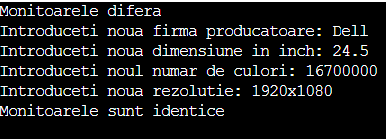
**freememor(rezultat);**

**return 0;**

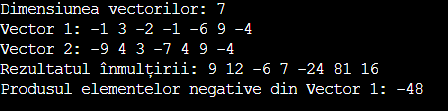
**}**

# **Testarea și verificarea programelor cu rezultatele obținute:**

**a)**

****

**b)**

****

# **Concluzii:**

În această lucrare am explorat utilizarea structurilor de date personalizate și gestionarea dinamică a memoriei în limbajul C++. Cele două coduri dezvoltate au demonstrat cum putem organiza și manipula eficient datele complexe, precum informațiile despre un monitor sau un vector de numere întregi.

Primul cod a ilustrat structura unui Monitor, unde am implementat funcții pentru setarea, modificarea și compararea datelor. Prin utilizarea memoriei dinamice pentru stocarea numelui firmei producătoare și gestionarea corectă a acesteia prin alocare și eliberare de memorie, am asigurat flexibilitatea și siguranța memoriei pe parcursul execuției programului.

Al doilea cod a evidențiat manipularea unui Vector de numere întregi, oferind funcții pentru redimensionarea acestuia, accesul la elemente și calcularea produsului elementelor negative. Am demonstrat și o operațiune utilă în matematică și algoritmi, și anume înmulțirea element cu element a doi vectori de aceeași dimensiune.

Aceste două exemple au subliniat importanța utilizării eficiente a memoriei dinamice și a creării de funcții care să asigure modularitatea și claritatea codului. Prin alocarea și eliberarea corectă a memoriei, am prevenit scurgerile de memorie și am gestionat corect resursele limitate ale sistemului.

Lucrarea de față a arătat cum structurile de date personalizate, alături de funcțiile asociate, permit o manipulare ușoară și eficientă a datelor în C++, oferind un model modular și reutilizabil pentru dezvoltarea programelor.

# **Bibliografie:**

1. Lucrare de laborator 2:<https://ocw.cs.pub.ro/courses/sd-ca/2019/laboratoare/lab-02> Acesat pe 9.10.2024
2. POO:<https://www.youtube.com/watch?v=rZcTaRU7AAw&list=PLQOaTSbfxUtBm7DxblJZShqBQnBAVzlXX> Acesat pe 9.10.2024