Міністерство освіти і науки України
Центральноукраїнський національний технічний університет
Механіко-технологічний факультет

ЗВІТ
ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 11
з навчальної дисципліни
“Базові методології та технології програмування”

РЕАЛІЗАЦІЯ ПРОГРАМНИХ ЗАСОБІВ ОБРОБЛЕННЯ ДИНАМІЧНИХ СТРУКТУР ДАНИХ ТА БІНАРНИХ ФАЙЛІВ

ЗАВДАННЯ ВИДАВ
доцент кафедри кібербезпеки
та програмного забезпечення
Доренський О. П.
https://github.com/odorenskyi/

ВИКОНАЛИ
студенти академічної групи ТК-23-1
Шавлєнков П.О
Макодзеба П.П
студент академічної групи КН-23
Ситенкова В.В

ПЕРЕВІРИВ
доцент кафедри кібербезпеки
та програмного забезпечення
Доренський О. П.
https://github.com/odorenskyi/

Кропивницький – 2024

Тема: Реалізація програмних засобів оброблення динамічних структур даних та бінарних файлів

Мета роботи: Набуття ґрунтовних вмінь і практичних навичок командної (колективної) реалізації програмного забезпечення, розроблення функцій оброблення динамічних структур даних, використання стандартних засобів C++ для керування динамічною пам'яттю та бінарними файловими потоками.

Завдання до лабораторної роботи:
1. У складі команди ІТ-проекта розробити програмні модулі оброблення динамічної структури даних.
2. Реалізувати програмний засіб на основі розроблених командою ІТ-проекта модулів.

Варіант 8

—— ЗАВДАННЯ НА РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ——

Створити базу даних "**Деканат: облік студентів**".

За вибором користувача (диспетчера) додаток ОС Windows забезпечує:

- *виведення* всієї бази на екран або у текстовий файл;
- *додавання* нового запису в базу;
- *пошук* запису за введеним диспетчером прізвищем студента;
- *вилучення* заданого оператором запису з бази;
- *завершення* роботи програми з автоматичним записом бази у файл.

База даних *автоматично завантажується* з файлу під час запуску додатка (програми).

Склад команди ІТ-проекта: Макодзеба Павло, Ситенкова Вероніка, Шавлєнков Павло

Розроблений план виконання ІТ-проекта
«Деканат: облік студентів»

1. Аналіз задач ІТ-проекта та вимог до програмного забезпечення

● **Ціль проекту:**
- розробка бази даних «Деканат: облік студентів» та додатку для взаємодії з нею

● **Функціональні вимоги:**
1. ***Виведення*** всього реєстру на екран або у заданий текстовий файл
2. ***Додавання*** нового запису до реєстру студентів з такою інформацією: прізвище, ім'я, по батькові студента, дата народження, місце народження, громадянство, сімейний стан, місце проживання, факультет, спеціальність, спеціалізація.
3. ***Пошук*** запису в реєстрі за даним прізвищем;

4. ***Вилучення*** заданого запису з реєстру за даним прізвищем.
5. ***Завершення*** роботи програми з автоматичним записом реєстру у файл бази.

● Нефункціональні вимоги:

1. Програма повинна бути ефективною та швидкою у роботі;
2. Надійність та стійкість програми до помилок та відмов.

● Дані:

1. Прізвище, ім'я, по-батькові студента;
2. Дата народження;
3. Місце народження;
4. Громадянство;
5. Сімейний стан;
6. Місце проживання;
7. Факультет;
8. Спеціальність;
9. Спеціалізація;

Вимоги до програмного забезпечення

●Інтерфейс користувача:

- інтуїтивно зрозумілий та зручний для введення та виведення інформації.

● Продуктивність:

- швидке виконання запитів, оптимізація для великих обсягів даних.

● Кросплатформеність:

- підтримка різних операційних систем.

● Масштабованість:

- легкість оновлення та розширення функціоналу.

● Технічні обмеження:

- врахування можливостей використовуваного обладнання та програмного середовища.

● Документація:

- повна документація коду

2. Специфікації ПЗ, концептуальні проектні рішення, архітектура програмного засобу, загальні алгоритми функціонування та інтерфейси модулів

Специфікація ПЗ

Обговорення щодо:

1) вимог до програмного забезпечення;
2) вибрання типу динамічної структури даних (список, стек, черга, дерево) для реалізації бази даних ПЗ;
3) створення заголовочного файлу;
4) розподілити підзадачі реалізації операції над динамічною структурою даних;
5) складання плану робіт проекту з урахуванням стандарту ISO/IEC 12207

Концептуальні проектні рішення

1) Модульна архітектура

• розподіл системи на незалежні модулі для спрощення розробки та тестування:

• модуль виведення реєстру - відповідає за виведення всього реєстру на екран або у заданий текстовий файл.
• модуль додавання нового запису - відповідає за додавання нового запису про студента до реєстру.
• модуль пошуку за студента за прізвищем - відповідає за пошук студента в реєстрі за його прізвищем. Якщо запис не буде знайдено, модуль виведе відповідне повідомлення.
• модуль вилучення запису - відповідає за вилучення заданого запису з реєстру.
• модуль автоматичного запису реєстру у файл - відповідає за автоматичний запис у файл під час завершення роботи програми.

2) Використання списку для реалізації електронного реєстру студентів
• оптимізація швидкості пошуку, вставки та видалення даних
• поліпшення впорядкувння та доступної інформації

Архітектура програмного засобу

1) Модуль управління даними
● Основною структурою даних, є структура **Student**
● Операції:
- виведення реєстру: функція для виведення всього реєстру на екран або у заданий текстовий файл. Ця операція дозволяє користувачу переглядати всі дані, що зберігаються у реєстрі.
- додавання нового запису: функція для додавання нового запису про студента до реєстру. Ця операція дозволяє користувачу ввести дані про нового студента та додати його до реєстру.
- пошук студента за прізвищем: функція для пошуку запису в реєстрі за заданим прізвищем. Ця операція дозволяє знайти інформацію про студента за його прізвищем.
- вилучення запису: функція для вилучення заданого запису з реєстру. Ця операція дозволяє користувачу вилучити інформацію про студента з реєстру.
- завершення: функція для завершення роботи програми з автоматичним записом реєстру у файл. Під час завершення роботи програми всі дані про студентів автоматично записуються у файл, щоб забезпечити збереження інформації.

2) ***Інтерфейс користувача***
• Консольний інтерфейс: простий для інтерфейс управліня реєстрами в БД;
• Взаємодія з користувачем: чіткі інструкції та запити для забезпечення правильного введення даних;

Загальні алгоритми функціонування

1. **Виведення всього реєстру на екран або у заданий текстовий файл:**
- проходження через всі записи в реєстрі;
- виведення кожного запису на екран або збереження у файл.

2. Додавання нового запису до реєстру:
- введення нових даних;
- збереження нового запису в реєстрі.

3. Пошук запису в реєстрі за даним державним номером:
- введення прізвища для пошуку;
- пошук в реєстрі за прізвищем;

- виведення результату пошуку або відповідного повідомлення, якщо запис не знайдено.

4. Вилучення заданого запису з реєстру:
- введення прізвища студента, якого потрібно вилучити з реєстру;
- пошук запису в реєстрі;
- вилучення знайденого запису.

5. Завершення роботи програми з автоматичним записом реєстру у файл:
- збереження всіх змін у реєстрі у файл.
-завершення виконання програми.

Обґрунтування цього вибору наступне:

1) Швидкий доступ:
Двонапрямлені списки дозволяють швидше переміщатися як вперед, так і назад по списку. Це важливо для операцій, таких як пошук та видалення записів за державним номером, де може знадобитися звернення до попередніх записів.

2) Ефективність додавання та вилучення:
Додавання та видалення елементів у двонапрямленому списку може бути ефективними, оскільки не потрібно переміщатися через весь список для зміни посилань.

3) Простота реалізації:
Двонапрямлені списки є достатньо простими для реалізації та керуються досить зрозумілими правилами, що полегшує їх розуміння та використання.

4) Зручність використання:
Для операцій, таких як виведення всього реєстру на екран або у заданий текстовий файл, двонапрямлений список також може бути зручним, оскільки його можна ітерувати у будь-якому напрямку.

5) Навчальний аспект:
Часто використовується у курсах зі структур даних. Відповідно, він ідеально підходить для проектів, спрямованих на вивчення студентами фундаментальних принципів роботи структур даних.

Складові елементи структури

1. Прізвище

- Поле структури: last_name
- Тип даних: string

2. Ім’я
    - Поле структури: first_name
    - Тип даних: string
3. По-батькові
    - Поле структури: middle_name
    - Тип даних: string
4. Дата нароження

4. Місце народження
    - Поле структури: place_of_birth
    - Тип даних: string
5. Громадянство
    - Поле структури: citizenship
    - Тип даних: string
6. Сімейний стан
    - Поле структури: marital_status
    - Тип даних: string
7. Місце проживання
    - Поле структури: place_of_residence;
    - Тип даних: string
8. Факультет
    - Поле структури: faculty
    - Тип даних: string
9. Спеціальність
    - Поле структури: specialty
    - Тип даних: string
10. Спеціалізація
    - Поле структури: specialization
    - Тип даних: string

4. Створення заголовочного файлу

Задача: створити заголовочний файл struct_type_project_N.h (N — номер варіанта завдання) з описом елементів динамічної структури даних мовою C++

5. Розподіл завдань між учасниками команди підзадачі з реалізації операцій над динамічною структурою даних відповідно до розробленої архітектури програмного засобу.

Шавлєнков Павло

1. ***Виведення*** всього реєстру на екран або у заданий текстовий файл
2. ***Додавання*** нового запису до реєстру студентів з такою інформацією: прізвище, ім'я, по батькові студента, дата народження, місце народження,

громадянство, сімейний стан, місце проживання, факультет, спеціальність, спеціалізація

Ситенкова Вероніка

1. ***Пошук*** запису в реєстрі за даним прізвищем;
2. ***Вилучення*** заданого запису з реєстру.

Макодзеба Павло

1. ***Завершення*** роботи програми з автоматичним записом реєстру у файл бази.

Шавлєнков Павло

1. ***Виведення*** всього реєстру на екран або у заданий текстовий файл
2. ***Додавання*** нового запису до реєстру студентів з такою інформацією: прізвище, ім'я, по батькові студента, дата народження, місце народження, громадянство, сімейний стан, місце проживання, факультет, спеціальність, спеціалізація

Основні функції:

- getStudents(int status, string path = "") виведення всього реєстру на екран або у заданий текстовий файл.
- addStudent(Student student) додавання нового запису до реєстру студентів

Лістинг ModulesShavlienkov

```
#include <iostream>
#include <fstream>
#include <string>
#include <cctype>
#include <sstream>
#include <vector>

#include "struct_type_project_8.h"

using namespace std;

void getStudents(int status, string path = "") {
    try {
        if (status == 1) {
            ifstream file("temp.txt");
            string content;

            if (file.is_open()) {
                string line;
                while (getline(file, line)) {
                    content += line + "\n";
                }
            } else {
                throw runtime_error("Файл бази даних не знайдено!");
            }

            istringstream iss(content);
            string line;
            vector<string> labels = {
                "Прізвище", "Ім'я", "По-батькові", "Дата народження",
```

```
            "Місце народження", "Громадянство", "Сімейний стан",
            "Місце проживання", "Факультет", "Спеціальність", "Спеціалізація"
        };

        cout << "-----------------" << endl;

        while (getline(iss, line, '\n')) {
            istringstream lineStream(line);
            string word;
            int labelIndex = 0;

            while (lineStream >> word) {
                cout << labels[labelIndex] << ": " << word << endl;
                ++labelIndex;
                if (labelIndex >= labels.size()) {
                    labelIndex = 0;
                }
            }

            cout << "-----------------" << endl;
        }

    } else if (status == 2 && !path.empty()) {
        ifstream file("temp.txt");
        string content;

        if (file.is_open()) {
            string line;
            while (getline(file, line)) {
                content += line + "\n";
            }
        } else {
            throw runtime_error("Файл бази даних не знайдено!");
        }

        ofstream output(path);
        istringstream iss(content);
        string line;
        vector<string> labels = {
            "Прізвище", "Ім'я", "По-батькові", "Дата народження",
            "Місце народження", "Громадянство", "Сімейний стан",
            "Місце проживання", "Факультет", "Спеціальність", "Спеціалізація"
        };

        output << "-----------------" << endl;

        while (getline(iss, line, '\n')) {
            istringstream lineStream(line);
```

```
            string word;
            int labelIndex = 0;

            while (lineStream >> word) {
                output << labels[labelIndex] << ": " << word << endl;
                ++labelIndex;
                if (labelIndex >= labels.size()) {
                    labelIndex = 0;
                }
            }

            output << "-----------------" << endl;
        }
    }
  } catch (const exception& e) {
    cout << "Помилка: " << e.what() << endl;
  }
}

void addStudent(Student student) {
  ofstream output("temp.txt", ios_base::app);
  output << student.last_name << " " << student.first_name << " " <<
student.middle_name << " "
      << student.place_of_birth << " " << student.date_of_birth << " " <<
student.citizenship << " "
      << student.marital_status << " " << student.place_of_residence << " " <<
student.faculty << " "
      << student.specialty << " " << student.specialization << "\n";
  output.close();
}
```

Ситенкова Вероніка

1. ***Пошук*** запису за введеним диспетчером прізвищем студента.
2. ***Вилучення*** заданого оператором запису з бази.

Основні функції:

- getStudentData(string last_name = "") пошук запису за введеним диспетчером прізвищем студента.
- deleteStudentData(string last_name = "") вилучення заданого оператором запису з бази.

Лістинг ModulesSytenkova

```
#include <iostream>
#include <fstream>
#include <string>
#include <cctype>
#include <sstream>
#include <vector>

#include "struct_type_project_8.h"

using namespace std;

void getStudentData(string last_name = "") {
    try {
        ifstream file("temp.txt");
        if (!file.is_open()) {
            throw runtime_error("Файл бази даних не знайдено!");
        }

        string line;
        bool found = false;

        cout << "-----------------" << endl;

        while (getline(file, line)) {
            vector<string> elements;
            stringstream ss(line);
            string item;
            while (ss >> item) {
                elements.push_back(item);
            }
            if (!elements.empty() && elements[0] == last_name) {
                cout << "Прізвище: " << elements[0] << endl;
```

```
            cout << "Ім'я: " << elements[1] << endl;
            cout << "По-батькові: " << elements[2] << endl;
            cout << "Місце народження: " << elements[3] << endl;
            cout << "Дата народження: " << elements[4] << endl;
            cout << "Громадянство: " << elements[5] << endl;
            cout << "Сімейний стан: " << elements[6] << endl;
            cout << "Місце проживання: " << elements[7] << endl;
            cout << "Факультет: " << elements[8] << endl;
            cout << "Спеціальність: " << elements[9] << endl;
            cout << "Спеціалізація: " << elements[10] << endl;
            found = true;
        }

        cout << "-----------------" << endl;
    }
    file.close();

    if (!found) {
        throw runtime_error("Студент відсутній у базі даних!");
    }
  } catch (const runtime_error& e) {
    cerr << "Помилка: " << e.what() << endl;
  }
}

void deleteStudentData(string last_name = "") {
  try {
    ifstream file("temp.txt");
    if (!file.is_open()) {
        throw runtime_error("Файл бази даних не знайдено!");
    }

    string line;
    ofstream temp_file("temp_tmp.txt");
    bool found = false;

    while (getline(file, line)) {
        vector<string> elements;
        stringstream ss(line);
        string item;
        while (ss >> item) {
            elements.push_back(item);
        }
        if (!elements.empty() && elements[0] == last_name) {
            found = true;
        } else {
            temp_file << line << endl;
        }
```

```
        }

        file.close();

        if (!found) {
            throw runtime_error("Студент відсутній у базі даних!");
        } else {
            cout << "Данні студента видалено!" << endl;
        }

        temp_file.close();

        remove("temp.txt");
        rename("temp_tmp.txt", "temp.txt");
    } catch (const runtime_error& e) {
        cerr << "Помилка: " << e.what() << endl;
    }
}
```

Макодзеба Павло

1. ***Завершення*** роботи програми з автоматичним записом реєстру у файл бази.

Основні функції:

- exitProgram() Завершення роботи програми з автоматичним записом реєстру у файл бази

Лістинг ModulesMakodzeba

```
#include <iostream>
#include <fstream>
#include <cstdlib>

using namespace std;

void exitProgram() {
    ifstream temp("temp.txt");
    ofstream database("database.txt");
    database << temp.rdbuf();
    temp.close();
    database.close();
    remove("temp.txt");
}
```

Висновок: У ході даної лабораторної роботи я здобув ґрунтовні вміння і практичні навички командної (колективної) реалізації програмного забезпечення. Я навчився розробляти функції для оброблення динамічних структур даних, а також використовувати стандартні засоби C++ для керування динамічною пам'яттю та бінарними файловими потоками.