МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Центральноукраїнський національний технічний університет

Механіко-технологічний факультет

ЗВІТ

ПРО ВИКОНАННЯ ЛАБОРАТОРНОЇ РОБОТИ № 10

з навчальної дисципліни

“Базові методології та технології програмування”

Реалізація програмних модулів оброблення даних складових типів з файловим введенням/виведенням

ВИКОНАВ

Студент

академічної групи КН-23

Царенко Станіслав

ПЕРЕВІРИВ

викладач кафедри кібербезпеки

та програмного забезпечення

Ганна ДРЄЄВА

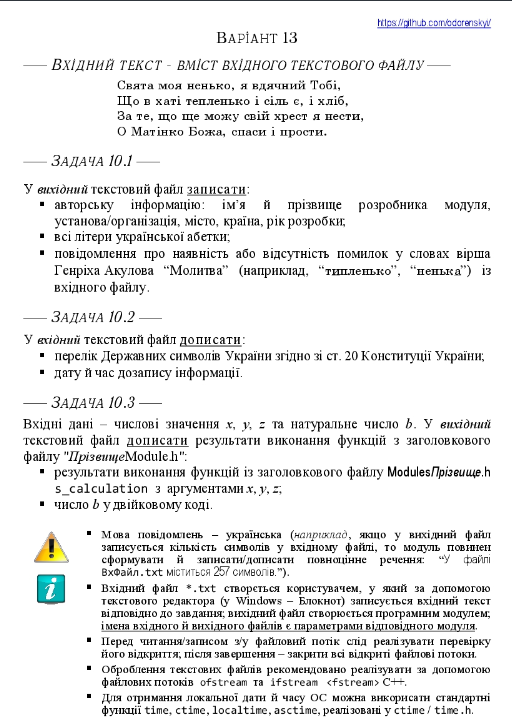
Кропивницький – 2024

**Тема:** Реалізація програмних модулів оброблення даних складових типів з файловим введенням/виведенням

**Завдання до лабораторної роботи:**

1. Реалізувати програмні модулі розв’язування задач 10.1–10.3 як складові статичної бібліотеки libModulesПрізвище.а (проект ModulesПрізвище лабораторних робіт №8–9).

2. Реалізувати тестовий драйвер автоматизованої перевірки програмних модулів розв’язування задач 10.1–10.3.



Задача 10.1

Вхідні дані: slova , alphabet,slovnik – назва вхідного та вихідного файлу та файлу для перевірки вірша.

Вихідні дані: вихідний файл з вказаною назвою та його зміст: авторська інформація; всі літери української абетки;

· повідомлення про наявність або відсутність помилок у словах вірша

Генріха Акулова “Молитва”

Для роботи з файлами потрібно використати файлові потоки ofstream та ifstream з <fstream>. Зміст вхідного файлу зчитується в змінну за допомогою функції getline, а потім обробляється відповідно до завдань. Спочатку обраховується кількість знаків за допомогою циклу, а потім в тому ж циклі перевіряється літера з переліку голосних літер. У випадку, якщо літера дійсно голосна – не додавати її в змінну з новим змістом файлу. В кінці подати інформацію українською мовою (за вимогою завдання), закрити файлові потоки.

**Задача 10.2**

Вхідні дані: slova – назва вхідного файлу

Вихідні дані: · перелік Державних символів України згідно зі ст. 20 Конституції України;

· дата й час дозапису інформації.

В цій задачі треба дописати дані у вхідний файл .Для реалізації цієї задачі можна використати стандартні бібліотеки.

**Задача 10.3**

Вхідні дані: x, y, z – числові значення; b – натуральне число; output – назва вихідного файлу

Вихідні дані: результат виконання функції s\_calculation з модуля ModulesChubenko та число b у двійковому коді

Використовується файловий потік ofstream для дозапису у файл. З модуля ModulesChubenko використовується функція s\_calculation. Для переводу числа b у двійковий код використовується клас bitset, що надається заголовковим файлом <bitset>.

*Реалізація завдань 10.1-10.3*

Результат виконання тестового драйверу

Файл відкритий

Усі дані записані у вихідний файл.

=== Тести завдання 10.1 ===

Тест-кейс №1

Вхідні дані:

inputFileName: slovnik.txt

outputFileName: output.txt

Файл відкритий

Усі дані записані у вихідний файл.

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №2

Вхідні дані:

inputFileName: slovnik.txt

outputFileName: output.txt

Файл відкритий

Усі дані записані у вихідний файл.

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №3

Вхідні дані:

inputFileName: slovnik.txt

outputFileName: output.txt

Файл відкритий

Усі дані записані у вихідний файл.

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №4

Вхідні дані:

inputFileName: slovnik.txt

outputFileName: output.txt

Файл відкритий

Усі дані записані у вихідний файл.

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №5

Вхідні дані:

inputFileName: slovnik.txt

outputFileName: output.txt

Файл відкритий

Усі дані записані у вихідний файл.

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

=== Тести завдання 10.2 ===

Тест-кейс №1

Вхідні дані:

inputFileName: slova.txt

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №2

Вхідні дані:

inputFileName: slova.txt

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №3

Вхідні дані:

inputFileName: slova.txt

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №4

Вхідні дані:

inputFileName: slova.txt

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

Тест-кейс №5

Вхідні дані:

inputFileName: slova.txt

Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n):

Статус тест-кейса: passed

=== Тести завдання 10.3 ===

Тест-кейс №1

Вхідні дані:

x: 5, y: 3, z: 10, b: 5676

outputFileName: output.txt

Відкрийте файл 'output.txt' та порівняйте результати з очікуваними результатами.

Співпадає? (y/n): y

Статус тест-кейса: passed

Тест-кейс №2

Вхідні дані:

x: 28, y: 22, z: 76, b: 4000

outputFileName: output.txt

Відкрийте файл 'output.txt' та порівняйте результати з очікуваними результатами.

Співпадає? (y/n): y

Статус тест-кейса: passed

Тест-кейс №3

Вхідні дані:

x: 11, y: 7, z: 28, b: 345

outputFileName: output.txt

Відкрийте файл 'output.txt' та порівняйте результати з очікуваними результатами.

Співпадає? (y/n): y

Статус тест-кейса: passed

Тест-кейс №4

Вхідні дані:

x: 14, y: 9, z: 40, b: 876

outputFileName: output.txt

Відкрийте файл 'output.txt' та порівняйте результати з очікуваними результатами.

Співпадає? (y/n): y

Статус тест-кейса: passed

Тест-кейс №5

Вхідні дані:

x: 12, y: 4, z: 30, b: 543

outputFileName: output.txt

Відкрийте файл 'output.txt' та порівняйте результати з очікуваними результатами.

Співпадає? (y/n): y

Статус тест-кейса: passed

Лістинг проекту ModulesChubenko

#include <iostream>

#include <cmath>

#include <iomanip>

#include <locale.h>

#include <Windows.h>

#include <fstream>

#include <ctime>

#include <bitset>

#include <fstream>

#include <string>

#include <set>

#include <sstream>

#include <chrono>

#include <iomanip> // Для форматування часу

#include <conio.h>

#include"ModulesChubenko.h"

using namespace std;

// Laboratory work №8

double s\_calculation(int x, int y, int z) {

double S;

if (x > y && y != 0) {

S = abs(abs(sqrt(z - 2 \* x)) - x \* sin(pow(x, 2 \* z)) \* y);

// cout << "S: " << setprecision(3) << fixed << S << endl;

}

else {

cout << "Результат обчислень невизначений або недійсний." << endl;

return 0.0;

}

// Повернення значення S

return S;

}

// Laboratory work №9

void calculateSalary(int day) {

double zar, p1, p2, p3, p4;

double sum;

int oplata = 450;

zar = day \* oplata;

cout << "Зарплатня: " << zar << endl;

p1 = 15.0 / 100 \* zar;

cout << "Податок 15 %: " << p1 << endl;

p2 = 2.0 / 100 \* zar;

cout << "Податок 2 % -: " << p2 << endl;

p3 = 0.6 / 100 \* zar;

cout << "Податок 0,6 % -: " << p3 << endl;

p4 = 1.0 / 100 \* zar;

cout << "Податок 1 %: " << p4 << endl;

sum = p1 + p2 + p3 + p4;

cout << "Сума до виплати: " << zar - sum << endl;

}

void printCopyright()

{

cout << " -----------------------------------------------\n\

| Chubenko Ihor, CNTU, https://github.com/IgorChubenkoKN-23 |\n\

| Chubenko Ihor, CNTU, https://github.com/IgorChubenkoKN-23 |\n\

------------ © All rights reserved ------------\n" << endl;

}

int countOnesAndZeros( int N) {

int zero\_count = 0;

int result=0;

int one\_count = 0;

for (int i = 0; i < 32; ++i) {

if (N & (1 << i)) {

++one\_count;

}

else {

++zero\_count;

}

}

(N == 0) ? result=zero\_count : result=one\_count;

return result;

}

string convertSockSize(int rozm) {

if (rozm == 23 || rozm == 24) {

return "В ЄС 37/38\nВ США 8\n";

}

else if (rozm == 25 || rozm == 26) {

return "В ЄС 39/40\nВ США 9\n";

}

else if (rozm == 27 || rozm == 28) {

return "В ЄС 41/42\nВ США 10\n";

}

else if (rozm == 29 || rozm == 30) {

return "В ЄС 43/44\nВ США 11\n";

}

else if (rozm == 31) {

return "В ЄС 45/46\nВ США 12\n";

}

else {

return "Розмір не відповідає жодному варіанту.\n";

}

}

// Laboratory work №10.1-10.2

void taskTwoTen(const string& filePath, const string& outputPath) {

// Запис державних символів України до вхідного файлу slova.txt

ofstream fin\_out(filePath, ios::app); // Відкриття файлу slova.txt для запису

fin\_out << "\nПерелік державних символів України згідно зі ст. 20 Конституції України:\n";

fin\_out << " Державний прапор України\n";

fin\_out << " Державний герб України\n";

fin\_out << " Державний гімн України\n";

fin\_out.close(); // Закриття файлу після запису

// Відкриття вхідного файлу для додавання дати та часу дозапису

ofstream fout(filePath, ios::app);

if (!fout) {

cerr << "Не вдалося відкрити файл для запису" << endl;

exit(1);

}

// Отримання поточного часу та форматування його у потрібний формат

auto now = chrono::system\_clock::now();

auto nowTimeT = chrono::system\_clock::to\_time\_t(now);

tm localTime;

// Використання `localtime\_s` для отримання поточного часу

localtime\_s(&localTime, &nowTimeT);

// Запис дати та часу дозапису у вихідний файл

fout << "\nДата та час дозапису: " << put\_time(&localTime, "%Y-%m-%d %H:%M:%S") << endl;

fout.close(); // Закриття вихідного файлу

}

void taskOneTen(const string& dictPath, const string& filePath, const string& outputPath, const string& alphabetPath) {

// Зміна кодування консолі на 1251 (якщо потрібно)

// Відкриття вихідного файлу для додавання інформації

ofstream fout(outputPath, ios::app);

if (!fout) {

cerr << "Не вдалося відкрити файл для запису" << endl;

exit(1);

}

// Додавання авторської інформації у вихідний файл

fout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";

fout << "Ім'я розробника: Чубенко Ігор\n";

fout << "Установа/організація: Центральноукраїнський національний технічний університет\n";

fout << "Місто, країна: Кропивницький, Україна\n";

fout << "Рік розробки: 2024\n";

fout << "~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~\n";

string ukrainianAlphabet = "АБВГҐДЕЄЖЗИІЇЙКЛМНОПРСТУФХЦЧШЩЬЮЯабвгґдеєжзиіїйклмнопрстуфхцчшщьюя";

fout << "\nУсі літери української абетки:\n" << ukrainianAlphabet << endl;

string alphabetLine;

// Завантаження словника

set<string> dictionary;

ifstream dictFile(dictPath);

if (!dictFile) {

cerr << "Не вдалося відкрити файл словника" << endl;

exit(1);

}

string dictWord;

while (dictFile >> dictWord) {

dictionary.insert(dictWord);

}

dictFile.close();

// Перевірка слів у файлі та запис у вихідний файл

ifstream fin(filePath);

if (!fin) {

cerr << "Файл не відкритий" << endl;

exit(1);

}

cout << "Файл відкритий" << endl;

string line;

while (getline(fin, line)) {

istringstream iss(line);

string word;

while (iss >> word) {

if (dictionary.find(word) != dictionary.end()) {

fout << "Слово \"" << word << "\" правильно написано.\n";

}

else {

fout << "Слово \"" << word << "\" неправильно написано.\n";

}

}

}

fin.close();

cout << "Усі дані записані у вихідний файл." << endl;

}

void taskTenthPointThree(int x, int y, int z, int b, string outputFileName) {

// Відкриття файлу для додавання інформації

ofstream outputFile(outputFileName, ios::out | ios::app);

if (!outputFile) {

cerr << "Не вдалося відкрити файл для запису" << endl;

return;

}

// Виконання функції s\_calculation та збереження результату у змінну

double result = s\_calculation(x, y, z);

// Виведення результату функції s\_calculation у файл

outputFile << "Результат виконання функції s\_calculation: " << result << endl;

// Перетворення числа b у двійкове представлення за допомогою bitset<32>

outputFile << "Число b у двійковому коді: " << bitset<32>(b).to\_string() << endl << endl;

// Закриття файлу

outputFile.close();

}

void testThirdTask(int test\_case, int x , int y, int z, int b, string outputFileName) {

cout << "Тест-кейс №" << test\_case << endl;

cout << "Вхідні дані:\n";

cout << "x: " << x << ", y: " << y << ", z: " << z << ", b: " << b << "\n";

cout << "outputFileName: " << outputFileName << endl;

// Виклик функції taskTenthPointThree з вхідними параметрами

taskTenthPointThree(x, y, z, b, outputFileName);

// Попросіть користувача перевірити результат і порівняти з очікуваним результатом

cout << "Відкрийте файл '" << outputFileName << "' та порівняйте результати з очікуваними результатами." << endl;

cout << "Співпадає? (y/n): ";

// Замість getch використовуйте std::cin для введення користувача

char checkResult;

cin >> checkResult;

// Перевірте відповідь користувача

if (checkResult == 'y' || checkResult == 'Y') {

cout << endl << "Статус тест-кейса: passed" << endl;

}

else {

cout << endl << "Статус тест-кейса: failed" << endl;

}

cout << endl;

}

*Лістинг проекту TestDriver:*

#include <iostream>

#include<Windows.h>

#include"ModulesChubenko.h"

#include <fstream>

#include <string>

#include <set>

#include <sstream>

#include <chrono>

#include <iomanip>

#include <sstream>

#include <bitset>

#include <set>

#include <conio.h>

using namespace std;

struct TestCase {

int test\_case; // Номер тест-кейса

int x;

int y;

int z;

int b;

string outputFileName;

};

string dictPath = "slovnik.txt";

string filePath = "slova.txt";

string outputPath = "output.txt";

string alphabetPath = "alphabet.txt";

#define TEST\_CASES\_COUNT 5

void testFirstTask(int test\_case, string inputFileName, string outputFileName) {

cout << "Тест-кейс №" << test\_case << endl;

cout << "Вхідні дані:\ninputFileName: " << inputFileName << "\noutputFileName: " << outputFileName << endl;

taskOneTen(dictPath, filePath, outputPath, alphabetPath);

cout << "Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n): ";

char checkResult = \_getch();

if (checkResult == 'y' || checkResult == 'Y') {

cout << endl << "Статус тест-кейса: passed" << endl;

}

else {

cout << endl << "Статус тест-кейса: failed" << endl;

}

cout << endl;

}

void taskTwoTen(int test\_case, string inputFileName) {

cout << "Тест-кейс №" << test\_case << endl;

cout << "Вхідні дані:\ninputFileName: " << inputFileName << endl;

taskTwoTen(filePath, outputPath);

cout << "Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n): ";

char checkResult = \_getch();

if (checkResult == 'y' || checkResult == 'Y') {

cout << endl << "Статус тест-кейса: passed" << endl;

}

else {

cout << endl << "Статус тест-кейса: failed" << endl;

}

cout << endl;

}

void testThirdTask(int test\_case, int x, int y, int z, unsigned int b, string outputFileName) {

cout << "Тест-кейс №" << test\_case << endl;

cout << "Вхідні дані: " << endl;

cout << "x: " << x << endl;

cout << "y: " << y << endl;

cout << "z: " << z << endl;

cout << "b: " << b << endl;

cout << "outputFileName: " << outputFileName << endl;

taskTenthPointThree(x, y, z, b, outputFileName);

cout << "Відкрийте модифікований файл та порівняйте з текстом із поля Expected Result. Співпадає? (y/n): ";

char checkResult = \_getch();

if (checkResult == 'y' || checkResult == 'Y') {

cout << endl << "Статус тест-кейса: passed" << endl;

}

else {

cout << endl << "Статус тест-кейса: failed" << endl;

}

cout << endl;

}

int main() {

#define TEST\_CASES\_COUNT 5

system("chcp 65001 & cls");

// Визначення вхідних даних для завдань

string dictPath = "slovnik.txt";

string filePath = "slova.txt";

string outputPath = "output.txt";

string alphabetPath = "alphabet.txt";

taskOneTen(dictPath, filePath, outputPath, alphabetPath);

taskTwoTen(filePath, outputPath);

// Виклик тестової функції для завдання 10.1

cout << "=== Тести завдання 10.1 ===" << endl;

testFirstTask(1, dictPath, outputPath);

testFirstTask(2, dictPath, outputPath);

testFirstTask(3, dictPath, outputPath);

testFirstTask(4, dictPath, outputPath);

testFirstTask(5, dictPath, outputPath);

// Виклик тестової функції для завдання 10.2

cout << "=== Тести завдання 10.2 ===" << endl;

taskTwoTen(1, filePath);

taskTwoTen(2, filePath);

taskTwoTen(3, filePath);

taskTwoTen(4, filePath);

taskTwoTen(5, filePath);

// Визначення тестових випадків для завдання 10.3

const TestCase testCases[TEST\_CASES\_COUNT] = {

{1, 5, 3, 10, 5676, "output.txt"},

{2, 28,22, 76, 4000, "output.txt"},

{3, 11, 7, 28, 345, "output.txt"},

{4, 14, 9, 40,876, "output.txt"},

{5, 12, 4, 30,543, "output.txt"}

};

// Виклик тестової функції для завдання 10.3

cout << "=== Тести завдання 10.3 ===" << endl;

for (int i = 0; i < TEST\_CASES\_COUNT; i++) {

const auto& testCase = testCases[i];

testThirdTask(testCase.test\_case, testCase.x, testCase.y, testCase.z, testCase.b, testCase.outputFileName);

}

return 0; // Повернення 0 для вказання успішного завершення програми

}

**Висновок**

Виконуючи цю лабораторну роботу, я отримав ґрунтовні знання та практичний досвід роботи зі створенням та обробленням різноманітних типів даних, такі як масиви, структури, об'єднання, множини та переліки. Особливу увагу у цій лаборіторній роботі було приділено роботі з файловими потоками, що дало мені змогу попрацювати зі збереженням та зчитуванням даних з файлів.

Згідно методичних рекомендацій потрібно було почергово проаналізувати і постановити задачі 10.1, 10.2 та 10.3, проаналізувати вимоги до ПЗ та до вмісту вхідного файлу, спроектувати архітектуру та задокументувати результати у звіт. В цій лабораторній було додатково вказано, що під час проектування слід враховувати вхідні дані модуля, а саме ім’я вхідного та/або вихідного файлу.

Під час розробки тест-сьютів було обрано напівавтоматизоване unit-тестування програмного модуля. Як й завжди було розроблено три тестових наборів для задач 10.1-10.3, в кожному по п’ять тест-кейсів. На відміну від минулих лабораторних робіт в структуру тестового набору було додано колонку Preliminary Steps (Попередні кроки). В цій колонці описується назва вхідного/вихідного файлу та зміст файлу.

Під час реалізації задачі 10.1-10.3 мовою програмування C++ було опрацьовано роботу з файловим потоком (ofstream, ifstream), запис/дозапис/читання та відкриття/закриття файлу.

Задача 10.1 У ході роботи я працював з вхідними та вихідними файлами, використовуючи ifstream та ofstream з бібліотеки <fstream>. Вивчив техніку обробки тексту з файлу, обраховував кількість знаків та перевіряв голосні літери в тексті. Успішно обробив текст вірша Г. Акулова "Молитва" і сформував вихідний файл, який містить відповідну інформацію українською мовою. Ця задача сприяла вдосконаленню моїх навичок роботи з файлами та текстовими даними.

Задача 10.2 У цій задачі я працював з вхідним файлом та дописував до нього перелік Державних символів України згідно зі ст. 20 Конституції України, а також дату й час дозапису інформації. Ця робота дала мені можливість глибше ознайомитися з роботою файлових потоків та записом даних до файлів, що сприяло поглибленню моїх знань у цій галузі.

Задача 10.3 Ця задача вимагала виконання функції s\_calculation з модуля ModulesChubenko та переведення числа b у двійковий код. Під час її виконання я успішно використав різні математичні операції та клас bitset з заголовкового файлу <bitset> для конвертації числа у двійковий код. Це дозволило мені вдосконалити свої навички роботи з числовими даними та математичними функціями.

Завдяки цій лабораторній роботі я покращив свої практичні навички роботи з файлами, обробкою текстових та числових даних, а також дослідив нові підходи до виконання задач та використання стандартних бібліотек для обробки інформації. Загалом, ця робота сприяла зміцненню моїх знань у програмуванні та аналізі даних.

**Контрольні запитання і завдання**

1. Поясніть сутність понять області видимості об’єктів та

простору імен у мові програмування С++.

* Простір імен (Namespace):

Простір імен - це механізм управління ідентифікаторами (назвами) в програмі, який дозволяє уникнути конфліктів між іменами, що використовуються у різних частинах програми.

Використання просторів імен допомагає організувати код, зробити його більш зрозумілим та підтримуваним.

Приклад: namespace MyNamespace { int x; }.

* Область видимості об'єктів (Scope):

Область видимості визначає доступність ідентифікатора в коді програми.

Області видимості можуть бути глобальними, локальними (області функцій, блоків коду), класовими (для членів класу) та іншими.

Об'єкти, оголошені в межах однієї області видимості, можуть бути доступними лише в межах цієї області.

2. Яким є час життя локального об’єкта (змінної)? Наведіть

приклад оголошення локальної та глобальної змінних одного зі

складових типів С/С++.

Час життя локального об'єкта (змінної) визначається областю видимості, в якій вона оголошена:

* Для локальних змінних: Час життя локальної змінної розпочинається при вході у блок коду (наприклад, функцію чи іншу область видимості) та закінчується при виході з цього блоку.
* Для глобальних змінних: Час життя глобальної змінної розпочинається при початку виконання програми та триває до її завершення.

3. Якою є область видимості змінної, оголошеної у циклі С++?

Відповідь обґрунтуйте та доведіть на прикладі.

Область видимості змінної, оголошеної у циклі C++, обмежена межами блоку циклу (**for**, **while**, **do-while**) або іншої контрольної структури (**if**, **switch** і т.д.), у якій вона була оголошена.

Це означає, що змінна буде видимою і доступною тільки всередині блоку коду, де вона була оголошена, і не буде доступною за межами цього блоку.

#include <iostream>

int main() {

for (int i = 0; i < 5; ++i) {

std::cout << "Value of i: " << i << std::endl;

}

// Спроба вивести значення i за межами циклу призведе до помилки компіляції,

// оскільки i обмежена межами циклу for

// std::cout << i << std::endl; // Цей рядок не буде компільований

return 0;

}

4. Якими є призначення й синтаксис запису директиви extern?

Наведіть приклад її застосування під час реалізації

міжмодульних змінних або функцій.

Директива extern у C/C++ використовується для оголошення міжмодульних змінних або функцій, які визначені в іншому модулі (файлі). Її призначення полягає в тому, щоб вказати компілятору, що змінна або функція оголошена зовнішнім модулем, тобто визначена в іншому файлі, і тому не буде визначена у поточному модулі.

Синтаксис запису директиви extern для змінних та функцій виглядає наступним чином:

extern тип ім'я\_змінної\_або\_функції;

5. У чому полягає відмінність між масивом і рядком С/С++ з

погляду реалізації їх оброблення?

Відмінність між масивом і рядком полягає в тому, що рядок в є масивом символів, який завершується нуль-символом ('\0'), що служить позначенням кінця рядка. Це означає, що рядки обробляються спеціальним чином, особливо при роботі зі стандартними функціями для роботи з рядками, такими як strlen, strcpy тощо. В той час як масив може містити будь-які дані (не тільки символи) і не має вбудованого механізму для позначення кінця даних, окрім його розміру.

6. Яким є синтаксис запису та оголошення об’єднання у С/С++?

Об'єднання (union) в C/C++ — це спеціальний тип даних, який дозволяє зберігати різні типи даних у одній області пам'яті. Це означає, що всі члени об'єднання розташовуються в одній і тій же пам'яті, тому в будь-який момент часу може бути використаний тільки один із членів об'єднання.

union UnionName {

dataType1 member1;

dataType2 member2;

// і так далі...

};

Тут:

UnionName - ім'я об'єднання.

dataType1, dataType2, тощо - типи даних для кожного члена об'єднання.

member1, member2, тощо - імена членів об'єднання.

7. Дайте визначення винятку (exception). Як у С++ реалізовується

його оброблення?

Виняток (exception) в C++ — це механізм для обробки ситуацій, які виникають під час виконання програми та можуть призвести до неправильного виконання, або непередбачуваної поведінки програми. Виняток виникає, коли щось іде не так, наприклад, ділення на нуль, спроба звернутися до невірного індексу масиву, або проблеми з пам'яттю. В таких випадках програма може викинути (throw) виняток, який може бути перехоплений (caught) і оброблений.

8. Яким є синтаксис оголошення одно- й двовимірних масивів

даних у С/С++ та за допомогою якої операції і за якими

індексами здійснюється доступ до їх елементів?

Оголошення одновимірного масиву

тип\_даних ім'я\_масиву[розмір\_масиву];

Оголошення двовимірного масиву

тип\_даних ім'я\_масиву[рядків][стовпців];

Доступ до елементів масиву здійснюється за допомогою індексації: назва\_масиву[індекс] для одновимірного масиву та назва\_масиву[індекс1][індекс2] для двовимірного масиву. Індекси починаються з нуля.

9. Яким є призначення наступних рядкових функцій з

заголовкового файлу cstring (string.h): strstr, strlen,

strcpy, strncpy, strcat, strncat, strcmp, strncmp, stricmp,

strnicmp, strchr, strcspn, strspn, strprbk, atof, atoi, atol?

strstr: Повертає вказівник на перше входження підрядка в рядку або nullptr якщо підрядок не знайдено.

strlen: Повертає довжину рядка (кількість символів до нульового символу).

strcpy: Копіює рядок з джерела в цільовий рядок, включаючи нульовий символ.

strncpy: Копіює до n символів із джерела в цільовий рядок, може залишити додаткові символи в цільовому рядку без змін.

strcat: Додає рядок джерела до кінця цільового рядка.

strncat: Додає до n символів з джерела до кінця цільового рядка.

strcmp: Порівнює два рядки; повертає 0, якщо вони однакові, негативне значення, якщо перший менший за другий, або позитивне значення, якщо перший більший за другий.

strncmp: Порівнює до n символів двох рядків і повертає результат, подібний до strcmp.

stricmp (або strcasecmp): Порівнює рядки, ігноруючи регістр символів.

strnicmp (або strncasecmp): Порівнює до n символів рядків, ігноруючи регістр.

strchr: Повертає вказівник на перше входження символу в рядку.

strcspn: Повертає кількість символів на початку рядка, які не входять у будь-який символ з вказаного набору.

strspn: Повертає довжину початкового відрізка рядка, який складається тільки з символів із вказаного набору.

strpbrk: Повертає вказівник на перший символ в рядку, який входить у вказаний набір.

atof: Перетворює рядок в float.

atoi: Перетворює рядок в int.

atol: Перетворює рядок в long.

10. Яке призначення, синтаксис опису і оголошення структур

(struct) С/С++? Чим вони відрізняються від масивів?

структура (struct) — це користувацький тип даних, який дозволяє згрупувати кілька змінних різних типів в один об'єкт. Структури є корисним засобом для моделювання складних даних, що складаються з декількох пов'язаних частин.

11. Що таке член (поле) структури struct С/С++ та який оператор

реалізовує доступ до нього?

Член (поле) структури - це змінна, оголошена всередині структури. Доступ до члена структури здійснюється за допомогою оператора крапки.

12. Що у С/С++ називають дескриптором структури (struct)?

За якими правилами він утворюється і задля чого

використовується програмістом?

"дескриптор структури" (struct descriptor) не є загальновживаним, проте, якщо розглядати його в контексті програмування, це може бути метадані структури або опис типу структури. Опис структури зазвичай включає в себе інформацію про її члени, їхні типи, розміри, порядок та іншу інформацію, яка визначає, як має бути організована пам'ять для об'єкта цієї структури.

13. Що слід виконати (записати у вихідному коді програми) для

оголошення змінної типу структура, якщо означений тип

описано у заголовковому файлі?

Для оголошення змінної типу структура, описаної у заголовковому файлі, потрібно включити цей заголовковий файл та оголосити змінну, використовуючи дескриптор структури.

Приклад

#include "ім'я\_заголовкового\_файлу.h"

14. Мовою програмування С/С++ наведіть приклад оголошення

структури (struct), один із членів якої ― структура.

#include <iostream>

// Оголошення структури Engine

struct Engine {

int horsepower; // Потужність двигуна у кінських силах

int cylinders; // Кількість циліндрів у двигуні

};

// Оголошення структури Car, яка містить член engine типу Engine

struct Car {

std::string make; // Виробник автомобіля

std::string model; // Модель автомобіля

Engine engine; // Двигун автомобіля (структура Engine)

int year; // Рік випуску автомобіля

};

int main() {

// Оголошення змінної car типу Car

Car car;

car.make = "Toyota";

car.model = "Camry";

car.year = 2021;

// Заповнення членів engine

car.engine.horsepower = 203;

car.engine.cylinders = 4;

// Виведення інформації про car

std::cout << "Car: " << car.make << " " << car.model << " (" << car.year << ")" << std::endl;

std::cout << "Engine: " << car.engine.horsepower << " HP, " << car.engine.cylinders << " cylinders" << std::endl;

return 0;

}

У цьому прикладі:

Ми оголошуємо структуру Engine, яка містить два члени: horsepower (потужність двигуна у кінських силах) і cylinders (кількість циліндрів у двигуні).

Потім ми оголошуємо структуру Car, яка містить кілька членів: make (виробник автомобіля), model (модель автомобіля), engine (член типу Engine для опису двигуна автомобіля) і year (рік випуску автомобіля).

У функції main() ми оголошуємо змінну car типу Car і встановлюємо значення для її членів.

Далі ми виводимо інформацію про змінну car, включаючи марку та модель автомобіля, рік випуску та характеристики двигуна.

15. Наведіть синтаксис оголошення змінної множинного типу та

перелічіть допустимі операції над нею. Чим з погляду

використання відрізняються змінні типу масив і типу множина?

У мові програмування C/C++ можна оголосити змінну множинного типу за допомогою ключового слова typedef. Ось синтаксис оголошення змінної множинного типу:

typedef existingType newTypeName;

Допустимі операції над змінною множинного типу включають усі операції, які можна виконати над типом, що визначений existingType. Це означає, що допустимі операції залежать від типу, що визначений existingType.

Наприклад, якщо existingType є цілим числом (int), то змінна типу newTypeName буде також поводитися як ціле число, з такими операціями, як арифметичні операції, операції присвоєння тощо.

* Масив:

Масив - це послідовність елементів одного типу, які розташовані в пам'яті один після одного.

Елементи масиву мають однаковий тип і використовуються за допомогою індексів.

Розмір масиву визначається при його оголошенні і залишається постійним протягом його життєвого циклу.

* Множина:

Множина - це новий тип даних, який можна оголосити за допомогою typedef.

Змінна множинного типу може містити будь-яке значення, що належить типу existingType, в залежності від того, який тип визначений existingType.

Розмір множинного типу не визначається при його оголошенні, він може варіюватися в залежності від значень, що додаються або видаляються.

16. Що у С/С++ розуміють під явним і неявним перетворення типів

та яким чином реалізовується кожен з них?

У мовах C/C++ існують два типи перетворень типів: явні (явне приведення типів) і неявні (неявне приведення типів).

У мові програмування C і C++ перетворення типів (також відомі як кастинг) є процесом перетворення даних з одного типу в інший. Це важливий механізм, який дозволяє програмістам виконувати різні операції з даними різних типів. Існує два види перетворень типів: явне та неявне.

1. Неявне перетворення типів (Implicit Conversion):

Неявне перетворення відбувається автоматично компілятором, коли це дозволено правилами мови.

Такі перетворення зазвичай відбуваються, коли використовується операція з аргументами різних типів, або коли необхідно привести значення до потрібного типу для відповідності контексту.

Наприклад, перетворення цілого числа в дійсне (тип double) або короткого цілого (short) у ціле (int).

2. Явне перетворення типів (Explicit Conversion):

Явне перетворення також відоме як кастинг.

Явне перетворення здійснюється програмістом явно, використовуючи оператор кастингу або функції перетворення.

Цей тип перетворення потрібен, коли є необхідність у перетворенні даних до іншого типу, особливо коли компілятор не може зробити це автоматично.

Здійснюється за допомогою спеціальних синтаксичних конструкцій або функцій перетворення.

17. Наведіть приклад оголошення потокового об’єкта ofstream з

заголовкового файла fstream С++. Яким чином за його

допомогою здійснюється запис даних у файловий потік (файл)?

#include <fstream> // Підключення заголовкового файлу fstream

#include <iostream> // Підключення заголовкового файлу iostream для використання cout

int main() {

// Оголошення потокового об'єкта ofstream і відкриття файлу для запису

std::ofstream outFile("example.txt");

// Перевірка, чи вдалося відкрити файл

if (!outFile) {

std::cerr << "Не вдалося відкрити файл для запису!" << std::endl;

return 1; // Завершити програму з кодом помилки

}

// Запис даних у файл

outFile << "Це рядок, що буде записаний у файл." << std::endl;

outFile << "Ось ще один рядок." << std::endl;

// Закриття файлового потоку

outFile.close();

std::cout << "Дані успішно записані у файл." << std::endl;

return 0;

}

18. У якому режимі відкриються файлові потоки, об’єкти яких

оголошені типом fstream, ofstream, ifstream відповідно?

- fstream: відкриває файл для читання і запису;

- ofstream: відкриває файл для запису;

- ifstream: відкриває файл для читання.

19. Яке призначення наступних функцій-членів потокового об’єкта

ifstream С++: open(), eof(), close()?

- open(): відкриває файл для читання;

- eof(): перевіряє, чи досягнуто кінець файлу;

- close(): закриває файл

20. Поясніть призначення кожної з перелічених констант режимів

відкриття файлових потоків у С++: ios\_base::app,

ios\_base::ate, ios\_base::in, ios\_base::out, ios\_base::trunc

(простір імен – std).

- ios\_base::app: дописує дані в кінець файлу;

- ios\_base::ate: встановлює позицію запису/читання в кінець файлу;

- ios\_base::in: відкриває файл для читання;

- ios\_base::out: відкриває файл для запису;

- ios\_base::trunc: очищає вміст файлу при відкритті.