Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи №2

З дисципліни «Об’єктно-орієнтоване програмування »

На тему:

Потоковий ввід-вивід

Виконав: Куцевол В.С. KI-109

Прийняв: асис. каф. ЕОМ

Гузинець Н.В

Львів – 2024

**Тема роботи:** Потоковий ввід-вивід

**Мета:** познайомитися із потоковим вводом-виводом.

**Теоретичні відомості**

Система вводу-виводу в стандартній бібліотеці С++ реалізована у вигляді потоків. Потік вводу-виводу – це логічний пристрій, який приймає та видає інформацію користувача. Кожен потік зв‘язаний з фізичним пристроєм (клавіатура, монітор) або з файлом. Бібліотека потоків iostream реалізована 20 як ієрархія класів та забезпечує широкі можливості для виконання операцій вводу-виводу.

Щоб мати можливість використовувати стандартні потоки необхідно підключити заголовочний файл iostream або iostream.h. Загалом різниця між стандартними заголовочними файлами з розширенням \*.h і без нього полягає в тому, що файли з розширенням \*.h відносяться до мови С, а без розширення – до С++. Таким чином програмуючи на мові С++ безпечніше використовувати заголовочні файли без розширення \*.h, які орієнтовані на мову С++. Проте в цьому випадку може бути необхідним підключати додатково простори імен. При використанні стандартних бібліотек вводу/виводу таким простором імен є std.

Об‘єкт стандартного потоку вводу cin класу istream, зв‘язаний із стандартним пристроєм вводу, за звичай клавіатурою. Об‘єкт стандартного потоку виводу cout класу ostream, зв‘язаний із стандартним пристроєм виводу, за звичай монітором. Об‘єкт cerr класу ostream, зв‘язаний із 21 стандартним пристроєм виводу повідомлень про помилки. Потоки даних, що виводяться, для об‘єкту cerr являються небуферизованими. Тобто кожна операція помістити в cerr приводить до миттєвої появи повідомлень про помилки. Об‘єкт clog класу ostream, зв‘язаний із стандартним пристроєм виводу повідомлень про помилки. Потоки даних, що виводяться, для об‘єкту clog являються буферизованими. Тобто кожна операція помістити в clog може привести до того, що вивід буде зберігатися в буфері до тих пір, поки буфер повністю не заповниться або ж поки вмістиме буферу не буде виведене примусово.

Вивід в потік виконується за допомогою операції «помістити в потік», а саме перевантаженої операції <<< повертає посилання на об'єкт типу ostream, для якого вона викликана. Це дозволяє будувати ланцюжок викликів операції «помістити в потік», що виконуються зліва направо.

Робота з файлами в мові С++ як і у мові С передбачає 3 етапи: відкривання файлу (файлового потоку), обмін даними з файловим потоком, закривання файлового потоку. Для виконання операцій з файлами в мові С++ передбачено три класи: іfstream, ofstream і fstream. Ці класи є похідними від класів іstream, ostream і іostream. Всі функціональні можливості (перевантажені операції << та >> для вбудованих типів, функції і прапорці форматування, маніпулятори й ін.), що застосовуються до стандартного вводу та виводу, можуть застосовуватися і до файлів. Існує деяка відмінність між використанням стандартних та файлових потоків. Стандартні потоки можуть використовуватися відразу після запуску програми, тоді як файловий потік спочатку слід зв'язати з файлом. Для реалізації файлового вводу-виводу потрібно підключити заголовочний файл fstream, що знаходиться в просторі імен std.

Режими відкриття файлу являють собою бітові маски, тому можна задавати два або більш режими, поєднуючи їх побітовою операцією АБО. Слід звернути увагу, що по замовчуванню режим відкриття файлу відповідає типові файлового потоку. У потоці вводу або виводу прапорець режиму завжди встановлений неявно.

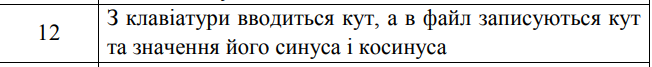
Між режимами відкриття файлу іos::ate та іos::app існує певна відмінність. Якщо файл відкривається в режимі додавання (іos::app), весь вивід у файл буде здійснюватися в позицію, що починається з поточного кінця файлу, безвідносно до операцій позиціонування у файлі. У режимі відкриття іos::ate (від англійського "at end") можна змінити позицію виводу у файл і здійснювати запис, починаючи з неї. Файли, які відкриваються для виводу, створюються, якщо вони ще не існують

Якщо при відкритті файлу не зазначений режим іos::bіnary, файл відкривається в текстовому режимі. Якщо відкриття файлу завершилося невдачею, об'єкт, що відповідає потокові, буде повертати нуль. Перевірити успішність відкриття файлу можна також за допомогою функції-члена іs\_open(). Дана функція повертає 1, якщо потік вдалося зв'язати з відкритим файлом.

Більшість прикладних програм передбачає інтерактивну взаємодію користувача з комп‘ютером. Для її реалізації з метою керування процесом роботи консольної програми часто використовується меню. Воно передбачає вивід на екран варіантів функціонування програми (включно з варіантом, що передбачає вихід з програми) з подальшим вибором одного з них користувачем.

**Хід виконання роботи**

**Варіант 12**

Завдання : 

//file AngleWork.h

#pragma once

#pragma once

#include<iostream>

#include<cmath>

#define P 3.1415926

using std::cout;

using std::cin;

using std::endl;

float CosTransformRad(double angle) {

return cos(angle);

}

float SinTransformRad(double angle) {

return sin(angle);

}

float CosTransformDeg(double angle) {

double cosin;

double trAngle = angle \* P / 180;

cosin = cos(trAngle);

if (cosin < 0.01) {

cosin = 0;

}

return cosin;

}

float SinTransformDeg(double angle) {

double sinn;

double trAngle = angle \* P / 180;

sinn = sin(trAngle);

if (sinn < 0.01) {

sinn = 0;

}

return sinn;

}

//file fileWork.h

#pragma once

#pragma once

#include<iostream>

#include<fstream>

#include<iomanip>

#include<string>

#include<string>

using std::ifstream;

using std::cout;

using std::cin;

using std::endl;

using std::setw;

using std::fstream;

using std::string;

void AngleInfoFileWriteRadiance(double cosin, double sin, double angle) {

fstream outfile("AngleInfo.txt", std::ios::app);

if (!outfile) {

cout << "Cannot open file" << endl;

}

outfile.setf(std::ios::left);

outfile << angle << "(radiance)" << endl;

outfile << sin << endl;

outfile << cosin << endl;

outfile.close();

}

void AngleInfoFileWriteDegree(double cosin, double sin, double angle) {

fstream outfile("AngleInfo.txt", std::ios::app);

if (!outfile) {

cout << "Cannot open file" << endl;

}

outfile.setf(std::ios::left);

outfile << angle << "(degree)" << endl;

outfile << sin << endl;

outfile << cosin << endl;

outfile.close();

}

int GetNumberOfFileStrings() {

fstream file("AngleInfo.txt", std::ios::in);

string line;

int numberOfLines = 0;

while (getline(file, line)) {

numberOfLines++;

}

file.close();

return numberOfLines;

}

void AngleInfoFileShow(int numberOfLines) {

fstream infile("AngleInfo.txt", std::ios::in);

string line;

if (!infile) {

cout << "Cannot open file" << endl;

}

cout.width(20);

cout << "Angle";

cout.width(20);

cout << "Sin";

cout.width(20);

cout << "Cos" << endl;

for (int i = 1; i <= numberOfLines; i++) {

cout.precision(3);

cout.width(20);

getline(infile, line);

cout << line;

if (i % 3 == 0) {

cout << endl;

}

}

infile.close();

}

//file ProgramComp.h

#pragma once

#include <iostream>

#include"AngleWork.h"

#include"FileWork.h"

using std::cout;

using std::cin;

using std::endl;

void AngleFileProgram() {

int choice;

cout << "This program shows you sin and cos of inputed angle." << endl;

cout << "Choose option:" << endl;

cout << "1.Add new angle in file.\n2.Show file with angles info." << endl;

cin >> choice;

switch (choice)

{

case 1:

double angle, sin, cos;

int formOfAngle;

cout << "Enter angle: ";

cin >> angle;

cout << "Form of angle:\n1.Radiant\n2.Deegres" << endl;

cin >> formOfAngle;

if (formOfAngle == 1) {

cos = CosTransformRad(angle);

sin = SinTransformRad(angle);

cout << "Cos of " << angle << " radiance = " << cos << endl;

cout << "Sin of " << angle << " radiance = " << sin << endl;

AngleInfoFileWriteRadiance(cos, sin, angle);

break;

}

if (formOfAngle == 2) {

cos = CosTransformDeg(angle);

sin = SinTransformDeg(angle);

cout << "Cos of " << angle << " degree = " << cos << endl;

cout << "Sin of " << angle << " degree = " << sin << endl;

AngleInfoFileWriteDegree(cos, sin, angle);

break;

}

case 2:

AngleInfoFileShow(GetNumberOfFileStrings());

break;

default:

break;

}

}

//file main.cpp

#include <iostream>

#include "ProgramComp.h"

using std::cout;

using std::cin;

using std::endl;

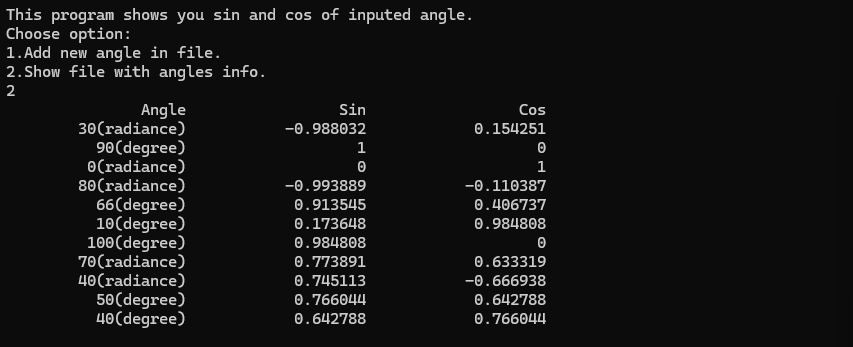
int main(void) {

AngleFileProgram();

return 0;

}





**Висновок**

У ході виконання лабораторної роботи я занурився в деталі потокового введення та виведення даних у мові програмування C++. Ця тема є важливою для будь-якого програміста, оскільки потоковий ввід та вивід є необхідною складовою взаємодії програм з користувачем та зовнішніми даними.

Вивчення потоків дозволило мені краще зрозуміти, як інформація переходить між програмою та її оточенням, будь то консольне введення, зчитування з файлу або вивід на екран. Я досліджував різні методи роботи з потоками у C++, включаючи використання стандартних об'єктів cin та cout, а також роботу з файловими потоками.

Ця робота дозволила мені освоїти основні прийоми роботи з потоковим введенням та виведенням, такі як форматування даних, обробка помилок та зберігання даних у різних форматах. Вивчення цих концепцій покращило мої навички програмування та зробило мене більш компетентним у створенні програм, які взаємодіють з користувачем та зовнішніми даними.