Автор: Білий Вадим, КІТ-119а

Дата: 25.05.2020

Лабораторна робота 9. ВИКЛЮЧЕННЯ

Тема. Виключення.

Мета – навчитись розробляти програми з реалізацією виключень.

Загальне завдання

У файлі розміщена інформація про N масивів. У першому рядку міститься інформація про кількість масивів, у кожній наступній – інформація про кількість елементів у кожному масиві та власне дані масиву. Необхідно реалізувати програму, що виконує перераховані нижче дії, причому кожна з них в окремій функції, поки користувач не введе замість назви файлу рядок \exit Дії, що має виконувати програма, такі:

- введення з клавіатури назви вхідного файлу з даними;
- читання даних з файлу;
- виконання індивідуального завдання;
- введення з клавіатури імені вихідного файлу;
- запис результату операції у файл;
- доступ до елемента за індексом слід винести в окрему функцію, що виконує перевірку на можливість виходу за межі масиву. Слід окремо звернути увагу, що при обробці виключення цикл не повинен перериватись.

Індивідуальні завдання

Підрахувати середнє значення елементів масиву. Результат операції – масив з середніх значень кожного із вхідних масивів.

Опис класів

Клас с основним завданням: CList

Опис змінних

int size1; -розмір головного масиву

int* size2; - розмір вкладених масивів

float* mass1; - масив результату операції

int** mass2; - масив даних с файлу

Опис методів

int getSize1(); - повертає розмір головного масиву

float* getMass1(); - повертає масив результатів

int** getMass2(); - повертає головний масив

string nameFile(); - читання ім’я файлу з консолі

void readFile(string fname); - читання файлу

void mean(); - середнє число

void writeFile(string fname); - запис у файл

int& operator[] (int i); - перевантаження оператора.

void cycle();

void end();

Текст програми

CList.cpp

```
include "CList.h"

int CList::getSize1()
{
	return size1;
}

float* CList::getMass1()
{
	return mass1;
}

int** CList::getMass2()
{
	return mass2;
}

string CList::nameFile()
{
	string fname;
	cout << "\nIf you want exit program, please enter \\exit\nEnter file name: ";
	cin >> fname;
```

```
	cout << endl;
	return fname;
}

void CList::readFile(string fname)
{
	stringstream ss;
	string line;
	ifstream file;
	int n;

	file.open(fname);
	if (!file.is_open()) {
		throw exception("file is not open");
	}
	getline(file,line);
	if (!file.is_open()) {
		return;
	}
	ss << line;
	ss >> size1;
	ss.clear();
	mass2 = new int* [size1];
	int i = 0;
	size2 = new int[size1];
	while (getline(file, line)&&i<size1)
	{
		ss << line;
		ss >> n;
		size2[i]=n;
		mass2[i] = new int[n];
		for (int j = 0; j < n; j++) {
			ss >> mass2[i][j];
		//	cout << mass2[i][j]<<" ";
		}
		ss.clear();
		i++;
	}
	if (i != size1) {
		throw exception("file is not correct");
	};

	file.close();
}

void CList::mean()
{

	mass1 = new float[size1];
	for (int i = 0; i < size1; i++)
	{
		mass1[i] = 0;
		for (int j = 0; j < size2[i]; j++) {

			mass1[i]+=mass2[i][j];
		}
		if (size2[i] <= 0)
			mass2[i] = 0;
		mass1[i] = mass1[i]/(float)size2[i];
		//cout << mass1[i]<< " ";
	}

}
void CList::writeFile(string fname)
{
```

```
	stringstream ss;
	ofstream file;
	file.open(fname);
	if (!mass2||!mass1||!size2)
	{
		throw exception("mass is NULL");
	}

	file.exceptions(ofstream::badbit | ofstream::failbit);
	if (!file.is_open()) {
		throw exception("file is not open");
	}
	if (!file.is_open()) {
		return;
	}
	for (int i = 0; i < size1; i++)
	{
		ss<<mass1[i]<<" ";
	}
	file << ss.str();
}

int& CList::operator[](const int i)
{
	if (i > size1 || i < 0)
	{
		throw exception("Error! There is no such element in array");
	}
	return *mass2[i];
}
void CList::end() {
	delete[] size2;
	delete[] mass1;
	for (size_t i = 0; i < size1; i++)
		delete[] mass2[i];
	delete[] mass2;
	size1 = 0;
}

void CList::cycle()
{
	string fname;
	string exit = "\\exit";
	bool flag = true;
	while (flag)
	{
		try
		{

			while (fname != exit) {
				if (size1>0) {
					end();
				}
				fname = nameFile();
				if (fname == exit)
					return;
				readFile(fname);
				mean();
				fname = nameFile();
				if (fname == exit)
					return;
				writeFile(fname);
			}
			end();
```

```
                        flag = false;
                }
                catch (const exception& ex)
                {
                        cout << ex.what();
                }

        }
}
CList::CList() :mass2(NULL), mass1(NULL), size1(0), size2(NULL)
{
}


CList::~CList()
{
}
```

CList.h

```
#pragma once
#define _CRT_SECURE_NO_WARNINGS
#include <iostream>
#include <fstream>
#include <sstream>
#include <iomanip>
#include <regex>
#include <assert.h>
using std::regex_match;
using std::regex;
using std::ifstream;
using std::ofstream;
using std::string;
using std::cout;
using std::cin;
using std::endl;
using std::setw;
using std::exception;
using std::istringstream;
using std::stringstream;


class CList
{
private:
        int size1;
        int* size2;
        float* mass1;
        int** mass2;

public:
        int getSize1();
        float* getMass1();
        int** getMass2();
        string nameFile();
        void readFile(string fname);
        void mean();
        void writeFile(string fname);
        int& operator[] (int i);
        void cycle();
        void end();
        CList();
        ~CList();
```

```
};
```

Sourse.cpp

```
#include <iostream>
#include "CList.h"
int main() {
	CList a;
	a.cycle();
	if (_CrtDumpMemoryLeaks())
		cout << "\nMemory leack deteckted\n";
	else
		cout << "\nMemory is not leack deteckted\n";


}
```

Test.cpp

```
#include "CList.h"
int main() {
	CList a;
	a.cycle();
	float* rez1 = a.getMass1();
	float test1[2] = { 2, 2.5 };
	for (int i = 0; i < 2; i++) {
		if (rez1[i] == test1[i])
		{
			cout << "test 1." << i<<": true"<< endl;
		}
		else {
			cout << "test 1." << i << ": false" << endl;
		}
	}
	a.cycle();
	float* rez2 = a.getMass1();
	float test2[3] = { 44, 23, 3 };
	for (int i = 0; i < 3; i++) {
		if (rez2[i] == test2[i])
		{
			cout << "test 2." << i << ": true" << endl;
		}
		else {
			cout << "test 2." << i << ": false" << endl;
		}
	}

	if (_CrtDumpMemoryLeaks())
		cout << "\nMemory leack deteckted\n";
	else
		cout << "\nMemory is not leack deteckted\n";

}
```

Висновок

При виконанні даної лабораторної роботи було набуто практичного досвіду роботи з виключеннями.

Було розроблено програму, що обробляє помилки за допомогою try catch,

Try catch – обробляє помилки під час виконання програми. Завжди можна дізнатись про помилку завдяки поліморфізму та наслідуванню класу exception.

Програма протестована, витоків пам’яті немає, виконується без помилок.