Звіт

Автор: Богданов І.Ю. КІТ-119а Дата: 11 травня 2020

Лабораторна робота №9. Виключення

Тема. Виключення.

Мета: навчитись розробляти програми з реалізацією виключень.

1. Завдання до роботи **Індивідуальне завдання**:

Написати програму, що проводить операції з масивами цілих чисел, та файлами в які ці масиви записані. Реалізувати обробку виключень в програмі так, щоб вона не припинялась.

2. Опис класів, змінних, методів та функцій

2.1 Опис класів

Ця програма не має класів.

2.2 Опис змінних

`std::string str` – змінна використовується для вибору пункту меню.
`int n = 0` – змінна використовується для вибору пункту меню, значення цієї змінної отримується шляхом змінення типу даних минулої змінної на int.
`int** data = 0` – змінна необхідна для зберігання даних масивів.
`int i` – використовується для пошуку елемента по індексу, в цю змінну записується індекс масиву.
`int u` – використовується для пошуку елемента по індексу, в цю змінну записується індекс елемента.
`std::string nmout` – використовується для збереження назви файлу у який записуються дані.
`std::string nmin` – використовується для збереження назви файлу з якого беруться дані.

2.3 Опис методів

Оскільки ця програма не має класів, то вона не має і методів.

2.4 Опис функцій

`std::string get_str_from_user()` – функція зчитує строку з клавіатури.
`int** read_data_from_file(std::string name)` – функція зчитує масив з файлу.
`void print_data_to_file(int** data, std::string name)` – функція використовується для запису масиву в файл.

`int* find_max(int** data)` – функція використовується для формування нового масиву з найбільших елементів кожного масиву.
`int get_elem_by_ind(int** data, int ind1, int ind2)` – функція знаходить елемент за заданими індексами, де перший індекс – індекс масиву, а другий елемента в цьому масиві.
`void get_data_to_screen(int** data)` – функція що виводить на екран дані з двумірного масиву.
`void get_data_to_screen(int* data)` – функція що виводить на екран дані з одномірного масиву.
`void menu()` – функція меню.
`void clean_data(int** data)` – функція що видаляє двумірний масив з пам'яті.

3 Текст програми

Лабораторная работа 9.cpp

```
#define _CRTDBG_MAP_ALLOC
#include "func.h"

int main()
{
    setlocale(LC_ALL, "Russian");
    menu();
       if (_CrtDumpMemoryLeaks()) {
              std::cout << "Утечка памяти обнаружена." << "\n";
       }
       else {
              std::cout << "Утечка памяти не обнаружена." << "\n";
       }
}
```

func.h

```
#pragma once
#include <iostream>
#include <string.h>
#include <fstream>
#include <sstream>
#include <regex>

std::string get_str_from_user();
int** read_data_from_file(std::string name);
void print_data_to_file(int** data, std::string name);
int* find_max(int** data);
int get_elem_by_ind(int** data, int ind1, int ind2);
void get_data_to_screen(int** data);
void get_data_to_screen(int* data);
void menu();
void clean_data(int** data);
```

func.cpp

```
#include "func.h"
std::string get_str_from_user() {
       std::string str;
       std::cin >> str;
       return str;
}
int** read_data_from_file(std::string name) {
       std::ifstream in(name);
       if (!(in.is_open())) {
              throw 1;
       }
       else {
              int n;
              int t;
              int r;
              int** res;
              in >> n;
```

```
		res = new int* [n + 1];
		res[0] = new int;
		res[0][0] = n;
		for (int i = 1; i <= n; i++) {
			in >> t;
			res[i] = new int[t + 1];
			res[i][0] = t;
			for (int u = 1; u <= t; u++) {
				in >> r;
				res[i][u] = r;
			}
		}
		in.close();
		return res;
	}
}
void print_data_to_file(int** data, std::string name) {
	std::ofstream of(name);
	if (data == nullptr) {
		throw 'c';
	}
	else if (!(of.is_open())) {
		throw 1;
	}
	else {
		of << data[0][0] << "\n";
		for (int i = 1; i <= data[0][0]; i++) {
			of << data[i][0] << " ";
			for (int u = 1; u <= data[i][0]; u++) {
				of << data[i][u] << " ";
			}
			if (i != data[0][0]) {
				of << "\n";
			}
		}
		of.close();
	}
}
int* find_max(int** data) {
	if (data == nullptr) {
		throw 1;
	}
	else {
		int* res = new int[data[0][0] + 1];
		res[0] = data[0][0];
		for (int i = 1; i <= data[0][0]; i++) {
			res[i] = get_elem_by_ind(data, i - 1, 0);
			for (int u = 1; u <= data[i][0]; u++) {
				if (res[i] < get_elem_by_ind(data, i - 1, u - 1)) {
					res[i] = get_elem_by_ind(data, i - 1, u - 1);
				}
			}
		}
		return res;
	}
}
int get_elem_by_ind(int** data, int ind1, int ind2) {
	if (data == nullptr) {
		throw 1;
	}
	else {
		if (ind1 >= data[0][0]) {
			throw std::out_of_range("Out of range");
		}
		else if (ind2 >= data[ind1 + 1][0]) {
			throw std::out_of_range("Out of range");
		}
		else {
			return data[ind1 + 1][ind2 + 1];
```

```
		}
	}
}
void get_data_to_screen(int** data) {
	if (data == nullptr) {
		throw 1;
	}
	else {
		for (int i = 1; i <= data[0][0]; i++) {
			for (int u = 1; u <= data[i][0]; u++) {
				std::cout << get_elem_by_ind(data, i - 1, u - 1) << " ";
			}
			std::cout << "\n";
		}
	}
}
void get_data_to_screen(int* data) {
	for (int i = 1; i <= data[0]; i++) {
		std::cout << data[i] << " ";
	}
	std::cout << "\n";
}
void menu() {
	std::string str;
	int n = 0;
	int** data = 0;
	int i;
	int u;
	std::string nmout;
	std::string nmin;
	while (true) {
		std::cout << "Выберите желаемое действие: " << "\n";
		std::cout << "1 - ввести название файла (с расширением) для чтения информации." <<
"\n";
		std::cout << "2 - ввести название файла (с расширением) для записи информации." <<
"\n";
		std::cout << "3 - вывести массивы на экран." << "\n";
		std::cout << "4 - считать данные из файла." << "\n";
		std::cout << "5 - записать данные в файл." << "\n";
		std::cout << "6 - создать массив из максимальных элементов всех массивов." << "\n";
		std::cout << "7 - получить элемент по индексам." << "\n";
		std::cout << "Что бы завершить работу программы выберите первый пункт и напишите
\"\\exit\"." << "\n";
		std::cout << "Введите цифру, соответствующую нужному действию - ";
		std::cin >> str;
		try
		{
			n = std::stoi(str);
		}
		catch (std::invalid_argument const& e)
		{
			std::cout << "Введите число." << '\n';
		}
		catch (std::out_of_range const& e)
		{
			std::cout << "Введите число." << '\n';
		}
		if (n == 1) {
			std::cout << "Введите название файла: ";
			nmin = get_str_from_user();
			if (nmin == "\\exit") {
				break;
			}
		}
		else if (n == 2) {
			std::cout << "Введите название файла: ";
			nmout = get_str_from_user();
		}
		else if (n == 3) {
```

```
                        try {
                                std::cout << "Вот ваши массивы:" << "\n";
                                get_data_to_screen(data);
                        }
                        catch (std::out_of_range) {
                                std::cout << "Ошибка в данных массива по-пробуйте считать его заново." <<
"\n";
                        }
                        catch (int) {
                                std::cout << "Нет массива, попробуйте считать его с файла." << "\n";
                        }
                }
                else if (n == 4) {
                        try {
                                clean_data(data);
                                data = read_data_from_file(nmin);
                                std::cout << "Данные считаны из файла." << "\n";
                        }
                        catch (int) {
                                std::cout << "Файла с таким названием нет или вы не ввели его название,
попробуйте использовать другой файл." << "\n";
                        }
                }
                else if (n == 5) {
                        try {
                                print_data_to_file(data, nmout);
                                std::cout << "Данные записаны в файл." << "\n";
                        }
                        catch (int) {
                                std::cout << "Файла с таким названием нет или вы не ввели его название,
попробуйте использовать другой файл." << "\n";
                        }
                        catch (char) {
                                std::cout << "Нет массива, попробуйте считать его с файла." << "\n";
                        }
                }
                else if (n == 6) {
                        try {
                                int* maxmass;
                                maxmass = find_max(data);
                                std::cout << "Вот максимальные элементы массивов:" << "\n";
                                get_data_to_screen(maxmass);
                                delete[] maxmass;
                        }
                        catch (std::out_of_range) {
                                std::cout << "Ошибка в данных массива по-пробуйте считать его заново." <<
"\n";
                        }
                        catch (int) {
                                std::cout << "Нет массива, попробуйте считать его с файла." << "\n";
                        }
                }
                else if (n == 7) {
                        try {
                                std::cout << "Введите индекс массива и индекс элемента: ";
                                std::cin >> i >> u;
                                std::cout << "Вот элемент с нужными индексами: " << get_elem_by_ind(data,
i, u) << "\n";
                        }
                        catch (std::out_of_range) {
                                std::cout << "Ошибка в индексах попробуйте ввести их снова." << "\n";
                        }
                        catch (int) {
                                std::cout << "Нет массива, попробуйте считать его с файла." << "\n";
                        }
                }
        }
        clean_data(data);
}
```

```
void clean_data(int** data) {
	if (data != nullptr) {
		for (int i = 1; i <= data[0][0]; i++) {
			delete[] data[i];
		}
		delete[] data[0];
		delete[] data;
	}
}
```

tests.cpp

```
#include "func.h"
#define _CRTDBG_MAP_ALLOC

int main(){
	setlocale(LC_ALL, "Russian");
	std::cout << "Сейчас будет выведена информация от тестовом массиве и утечке памяти, если этого не произойдёт тест будет не пройден." << "\n";
	int** data;
	int n;
	std::string str = "test.txt";
	data = read_data_from_file(str);
	get_data_to_screen(data);
	clean_data(data);
	if (_CrtDumpMemoryLeaks()) {
		std::cout << "Утечка памяти обнаружена." << "\n";
	}
	else {
		std::cout << "Утечка памяти не обнаружена." << "\n";
	}
	std::cout << "Введите что угодно для завершения роботы программы: ";
	std::cin >> n;
}
```

data.txt

```
5
2 3 4
4 7 5 4 1
1 1
2 3 4
1 9
```

data2.txt

```
text.txt
9
3 3 2 2
6 4 5 3 1 0 9
10 11 22 33 44 55 66 77 88 99 0
1 1
2 2 2
3 3 3 3
1 9
4 0 0 0 0
6 5 4 7 8 9 1
```

text2.txt

```
test.txt
5
5 1 1 1 1 1
5 1 2 2 2 2
5 1 2 3 3 3
5 1 2 3 4 4
5 1 2 3 4 5
```

4. Результати роботи програми

Результати роботи програми:

```
Выберите желаемое действие:
1 - ввести название файла (с расширением) для чтения информации.
2 - ввести название файла (с расширением) для записи информации.
3 - вывести массивы на экран.
4 - считать данные из файла.
5 - записать данные в файл.
6 - создать массив из максимальных элементов всех массивов.
7 - получить элемент по индексам.
Что бы завершить работу программы выберите первый пункт и напишите "\exit".
Введите цифру, соответствующую нужному действию -
```

Результати тестів:

```
Сейчас будет выведена информация от тестовом массиве и утечке памяти, если этого не произойдёт тест будет не пройден.
1 1 1 1 1
1 2 2 2 2
1 2 3 3 3
1 2 3 4 4
1 2 3 4 5
Утечка памяти не обнаружена.
Введите что угодно для завершения роботы программы:
```

5. Висновки

В даній лабораторній роботі були використані блоки try, catch після вводу даних для вибору типу операції і в середині деяких блоків меню що виконують операції (блоки з 3 по 7).

Програма протестована, витоків пам’яті немає, всі виключення відловлються і не заважають роботі програми.