Чумин Владимир Владимирович
КЭ-142
Лабораторная № 8

Задание:
Создайте систему на C++, которая читает текст из нескольких файлов, обрабатывает данные параллельно с использованием лямбда-выражений и многопоточности, а затем сохраняет результат в новых файлах. Каждый поток должен обрабатывать текст из своего файла, например, выполняя подсчет слов или модифицируя текст определенным образом. Используйте наследование для создания общего интерфейса обработчиков данных и мьютексы для синхронизации доступа к ресурсам, если это необходимо. Результатом работы каждого потока должен быть новый файл, в котором содержится обработанный текст. Это задание поможет вам понять, как эффективно использовать многопоточность и лямбда-выражения для выполнения параллельной обработки данных, а также даст практику в работе с файлами в C++.

Код:
```cpp
#include <iostream>
#include <vector>
#include <thread>
#include <functional>
#include <mutex>
#include <fstream>
#include <string>
#include <filesystem>
#include <cmath>
#include <valarray>
#include <locale>
#include <memory>
```

```
using namespace std;
namespace fs = filesystem;

vector<string> getFilesInDirectory(const string&
directoryPath) {
    vector<string> files;
    try {
        for (const auto& entry :
fs::directory_iterator(directoryPath)) {
            if (fs::is_regular_file(entry.path()))
{

files.push_back(entry.path().string());
            }
        }
    } catch (const fs::filesystem_error& e) {
        cerr << "Ошибка при доступе к директории: "
<< e.what() << endl;
    }
    return files;
}

class MiniFunctions {
protected:
    struct StringFunction {
        string name;
        function<string(string)> func;
    };
    struct IntFunction {
        string name;
        function<int(int)> func;
    };
    struct DoubleFunction {
        string name;
```

```
        function<double(double)> func;
    };

    static vector<StringFunction> stringFunctions;
    static vector<IntFunction> intFunctions;
    static vector<DoubleFunction> doubleFunctions;

    string lastFunction;

public:
    MiniFunctions () {};

    string returnLastFunction (){return
lastFunction;}

    function<string(string)>
getRandomStringFunction() {
        int randomInt = rand() %
stringFunctions.size();
        lastFunction =
stringFunctions[randomInt].name;
        return stringFunctions[randomInt].func;
    }

    function<int(int)> getRandomIntFunction() {
        int randomInt = rand() %
intFunctions.size();
        lastFunction =
intFunctions[randomInt].name;
        return intFunctions[randomInt].func;
    }

    function<double(double)>
getRandomDoubleFunction() {
```

```
        int randomInt = rand() %
doubleFunctions.size();
        lastFunction =
doubleFunctions[randomInt].name;
        return doubleFunctions[randomInt].func;
    }

};

// Инициализация статических членов класса внутри
класса
vector<MiniFunctions::StringFunction>
MiniFunctions::stringFunctions = {
    {"Добавление восклицательного знака в конце
слова.", [](string s) { return s + "!"; }},
    {"Преобразование строки в верхний регистр.",
[](string s) { transform(s.begin(), s.end(),
s.begin(), ::toupper); return s; }},
    {"Умножение длины строки на 2.", [](string s) {
return s + s; }},
    {"Удаление пробелов в начале и конце строки.",
[](string s) { s.erase(0, s.find_first_not_of('
')); s.erase(s.find_last_not_of(' ') + 1); return
s; }},
    {"Инвертирование строки", [] (string s)
{reverse(s.begin(), s.end()); return s; }}
};

vector<MiniFunctions::IntFunction>
MiniFunctions::intFunctions = {
    {"Удвоение числа.", [](int i) { return i * 2;
}},
    {"Утроение числа.", [](int i) { return i * 3;
}},
```

```
    {"Изменение числа на случайное.", [](int i) {
return rand() % 100; }},
    {"Добавление к числу 10.", [](int i) { return i
+ 10; }},
    {"Возведение числа в квадрат.", [](int i) {
return i * i; }}
};

vector<MiniFunctions::DoubleFunction>
MiniFunctions::doubleFunctions = {
    {"Удвоение числа.", [](double d) { return d *
2; }},
    {"Возведение числа в квадрат.", [](double d) {
return d * d; }},
    {"Умножение числа на 1,39.", [](double d) {
return d * 1.39; }},
    {"Добавление к числу 0,5.", [](double d) {
return d + 0.5; }},
    {"Умножение числа на 10", [](double d) { return
d * 10; }}
};

class Processor {
    protected:
    string filePath;
    vector<string> data;
    vector<string> oldData;

    public:
    Processor(string fileP) : filePath(fileP) {
        loadData();
    }

    ~Processor() {
        saveData();
```

```
    }

    string serializeData (vector<string>
currentData) {
        string serializedData = "";
        for (string element : currentData) {
            serializedData += element + "|";
        }
        return serializedData;
    }

    string returnFilePath () {
        return filePath;
    }

vector<string> deserializeData(const string
&serializedLine) {
    if (serializedLine.empty())  return {};

    vector<string> deserializedData;
    size_t start = 0;
    size_t end = serializedLine.find('|', start);

    while (end != string::npos) {

deserializedData.push_back(serializedLine.substr(s
tart, end - start));
        start = end + 1;
        end = serializedLine.find('|', start);
    }

    if (start < serializedLine.size()) {

deserializedData.push_back(serializedLine.substr(s
tart));
```

```
    }

    return deserializedData;
}
    void saveData () {
        string serializedData =
serializeData(data);

        ofstream fileOut(filePath);
        fileOut << serializedData << "\n" <<
serializeData(oldData);
        fileOut.close();
    }

    void loadData () {
        ifstream fileIn(filePath);
        if (!fileIn.is_open()) {
            cout << "Ошибка при открытии файла: "
<< filePath << endl;
        }
        string line;
        getline(fileIn, line);

        for (string element :
deserializeData(line)) {
            data.push_back(element);
        }
        fileIn.close();
    }

    void printData () {
        for (string element : data) {
            cout << element << endl;
        }
    }
```

```
    virtual void useFunction () {}
};

class ProcessorForInt : public Processor {
    protected:
    function<int(int)> func;

    public:
    ProcessorForInt (string fileP, const
function<int(int)> &func) : Processor(fileP),
func(func) {}

    void useFunction () override {
        for (string &element : data) {
            oldData.push_back(element);
            int intElem = stoi(element);
            element = to_string(func(intElem));
        }
    }
};

class ProcessorForString : public Processor {
    protected:
    function<string(string)> func;

    public:
    ProcessorForString (string fileP, const
function<string(string)> &func) :
Processor(fileP), func(func) {}

    void useFunction () override {
        for (string &element : data) {
            oldData.push_back(element);
            element = func(element);
```

```
        }
    }
};

class ProcessorForDouble : public Processor {
    protected:
    function<double(double)> func;

    public:
    ProcessorForDouble (string fileP, const
function<double(double)> &func) :
Processor(fileP), func(func) {}

    void useFunction () override {
        for (string &element : data) {
            oldData.push_back(element);
            int intElem = stod(element);
            element = to_string(func(intElem));
        }
    }
};

int main () {
    vector<string> files =
getFilesInDirectory("Lab8/files/");
    vector<unique_ptr<Processor>> processors;
    vector<thread> threads;
    MiniFunctions miniFunc;

    for (string file : files) {
        bool isInt = file.find("int") !=
string::npos;
        bool isString = file.find("string") !=
string::npos;
```

```
        bool isDouble = file.find("double") !=
string::npos;

        if (isInt || isString || isDouble) {
            if (isInt) {
                auto processor =
make_unique<ProcessorForInt>(file,
miniFunc.getRandomIntFunction());

processors.push_back(move(processor));
            }
            else if (isString) {
                auto processor =
make_unique<ProcessorForString>(file,
miniFunc.getRandomStringFunction());

processors.push_back(move(processor));
            }
            else if (isDouble) {
                auto processor =
make_unique<ProcessorForDouble>(file,
miniFunc.getRandomDoubleFunction());

processors.push_back(move(processor));
            }
            cout << "К файлу " << file << "
передана функция " <<
miniFunc.returnLastFunction() << endl;
        }
    }

    for (int i = 0; i < processors.size(); i++) {
        threads.emplace_back([processor =
processors[i].get(), i]
{processor->useFunction();});
```

```
    }

    for (thread &t : threads) {
        t.join();
    }

    cout << "Работа завершена." << endl;
    return 0;
}
```

Результат:

```
vol@vol-All-Series:~/CLionProjects/C++_Labs$ g++ Lab8/lab_8.cpp && ./a.out
К файлу Lab8/files/int_1.txt передана функция Добавление к числу 10.
К файлу Lab8/files/string_4.txt передана функция Преобразование строки в верхний регистр.
К файлу Lab8/files/double_2.txt передана функция Умножение числа на 1,39.
К файлу Lab8/files/double_5.txt передана функция Удвоение числа.
К файлу Lab8/files/int_4.txt передана функция Добавление к числу 10.
К файлу Lab8/files/string_3.txt передана функция Добавление восклицательного знака в конце слова.
К файлу Lab8/files/int_5.txt передана функция Утроение числа.
К файлу Lab8/files/int_2.txt передана функция Изменение числа на случайное.
К файлу Lab8/files/int_3.txt передана функция Возведение числа в квадрат.
К файлу Lab8/files/double_4.txt передана функция Возведение числа в квадрат.
К файлу Lab8/files/string_5.txt передана функция Умножение длины строки на 2.
К файлу Lab8/files/string_1.txt передана функция Умножение длины строки на 2.
К файлу Lab8/files/string_2.txt передана функция Добавление восклицательного знака в конце слова.
К файлу Lab8/files/double_3.txt передана функция Умножение числа на 10
К файлу Lab8/files/double_1.txt передана функция Добавление к числу 0,5.
Работа завершена.
```

Данные в файлах:

```
vol@vol-All-Series:~/CLionProjects/C++_Labs$ cat Lab8/files/int_1.txt
27|60|-51|-79|109|43|28|-6|-33|
17|50|-61|-89|99|33|18|-16|-43|
vol@vol-All-Series:~/CLionProjects/C++_Labs$ cat Lab8/files/int_2.txt
26|40|26|72|36|11|68|
26|40|26|72|36|11|68|
vol@vol-All-Series:~/CLionProjects/C++_Labs$ cat Lab8/files/double_1.txt
-80.500000|-47.500000|-79.500000|-33.500000|-0.500000|5.500000|7.500000|31.500000|
-81.500000|-48.500000|-80.500000|-34.500000|-1.500000|5.500000|7.500000|31.500000|
vol@vol-All-Series:~/CLionProjects/C++_Labs$ cat Lab8/files/double_2.txt
-27.800000|-40.310000|8.340000|189.040000|-157.070000|173.750000|-119.540000|-180.700000|-173.750000
|100.080000|
-20.850000|-29.190000|6.950000|136.220000|-113.980000|125.100000|-86.180000|-130.660000|-125.100000|
72.280000|
vol@vol-All-Series:~/CLionProjects/C++_Labs$ cat Lab8/files/string_1.txt
peachpeachpeachpeach|pearpearpearpear|grapegrapegrapegrape|lemonlemonlemonlemon|grapegrapegrapegrape
|grapegrapegrapegrape|
peachpeach|pearpear|grapegrape|lemonlemon|grapegrape|grapegrape|
vol@vol-All-Series:~/CLionProjects/C++_Labs$ cat Lab8/files/string_2.txt
pear!!|melon!!|grape!!|banana!!|grape!!|cherry!!|apple!!|plum!!|orange!!|melon!!|
pear!|melon!|grape!|banana!|grape!|cherry!|apple!|plum!|orange!|melon!|
```

Вторая строчка - это данные до применения функции.