YADRO Task

v.1.0.0

Создано системой Doxygen 1.9.6

Глава 1

Сортировка ленты

Данный репозиторий создан в ходе работы над тестовым заданием на позицию стажера в компанию YADRO.

1.1 Детали сортировки

В качестве алгоритма сортировки я использовал external sort, модифицированный под тестовое задание. А именно с изменением
времени считывания, перемещения головки, записи под нее, и чтения. Как образец взял алгоритм из следующей статьи, только переписал его под
C++. Было желание распареллелить этот алгоритм, но, к сожалению, не хватило времени. Краткое описание алгоритма:

1. Разбитие. Разбиваем входную ленту постепенно на части, помещающиеся в выделенной нам памяти. Сортируем под-ленту в памяти обычным std::sort(). После каждого разбития освобождаем буфер под новую часть ленты.
2. Слияние. Первый элемент каждой под-ленты пушим в кучу (std::priority_queue) и ассоциируем элемент с номером под-ленты, из которой он взят. Далее, пока куча не останется пустой, берем минимальный элемент из кучи, записываем его в выходную ленту, передвигаем головки.

1.2 Клонирование и начало работы

```
git clone https://github.com/bashkirian/YADROTask
cd build
```

Если вы работете на Linux, то закомментируйте строки, связанные со сборкой Windows, и раскомментируйте строки, связанные со сборкой Linux

1.3 Компилирование в обычном режиме

```
make clean
make tapesort
```

1.4 Компилирование для юнит-тестирования

```
make clean
make tapesort
```

1.5 Запуск программы для юнит-тестирования

```
./unittests
```

В консоли вы увидите результаты выполнения четырех-юнит тестов и время их выполнения.

1.6 Запуск программы в обычном режиме

Для начала работы необходимо создать два файла входной и выходной ленты в текстовом формате в директории build и наполнить входной файл желаемыми числами.

```
./sorttape "inputfile" "outputfile"
```

Глава 2

Алфавитный указатель пространств имен

2.1 Пространства имен

Полный список документированных пространств имен.

Глава 3

Иерархический список классов

3.1 Иерархия классов

Иерархия классов.

Глава 4

Алфавитный указатель классов

4.1 Классы

Классы с их кратким описанием.

Глава 5

Список файлов

5.1 Файлы

Полный список документированных файлов.

Глава 6

Пространства имен

6.1 Пространство имен configuration

Пространство имен для конфигурации

Классы

- struct Configuration
  Структура конфигурации

Определения типов

- using configurations_map = std::map< std::string, std::map< std::string, std::string > >

Функции

- configurations_map ReadConfigurations ()
  Функция, необходимая для получения конфигурации, написанной в виде [section] name - value.

6.1.1 Подробное описание

Пространство имен для конфигурации

6.1.2 Функции

6.1.2.1 ReadConfigurations()

configurations_map configuration::ReadConfigurations ( )

Функция, необходимая для получения конфигурации, написанной в виде [section] name - value.

Возвращает

configurations_map конфигурации

Глава 7

Классы

7.1 Структура configuration::Configuration

Структура конфигурации

#include <configuration.h>

Открытые атрибуты

- int m_rewind_delay
- int m_shift_delay
- int m_read_delay
- int m_write_delay

7.1.1 Подробное описание

Структура конфигурации

Объявления и описания членов структуры находятся в файле:

- src/Configuration/configuration.h

7.2 Структура Configurations

Открытые атрибуты

- int rewind_delay
- int shift_delay
- int read_delay
- int write_delay

Объявления и описания членов структур находятся в файлах:

- src/main.cpp
- src/UnitTests/unit_tests.cpp

7.3 Класс LogDuration

Класс для логирования времени выполнения программы

#include <profiler.h>

Открытые члены

- LogDuration (const string &msg="")

7.3.1 Подробное описание

Класс для логирования времени выполнения программы

Объявления и описания членов класса находятся в файле:

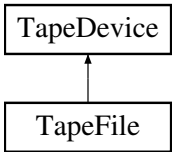
- src/UnitTests/profiler.h

7.4 Класс TapeDevice

Класс для устройства-ленты

#include <tape.h>

Граф наследования:TapeDevice:



Открытые члены

- virtual int read ()=0
- virtual void write (int value)=0
- virtual void rewind ()=0
- virtual void shift (int offset)=0
- virtual bool isended ()=0
- virtual configuration::Configuration getConfiguration ()=0
- virtual bool isOpen ()=0
- virtual void toOpen ()=0

7.4.1 Подробное описание

Класс для устройства-ленты

7.4.2 Методы

7.4.2.1 getConfiguration()

virtual configuration::Configuration TapeDevice::getConfiguration ( ) [pure virtual]

Замещается в TapeFile.

7.4.2.2 isended()

virtual bool TapeDevice::isended ( ) [pure virtual]

Замещается в TapeFile.

7.4.2.3 isOpen()

virtual bool TapeDevice::isOpen ( ) [pure virtual]

Замещается в TapeFile.

7.4.2.4 read()

virtual int TapeDevice::read ( ) [pure virtual]

Замещается в TapeFile.

7.4.2.5 rewind()

virtual void TapeDevice::rewind ( ) [pure virtual]

Замещается в TapeFile.

7.4.2.6 shift()

virtual void TapeDevice::shift (
int offset) [pure virtual]

Замещается в TapeFile.

7.4.2.7 toOpen()

virtual void TapeDevice::toOpen ( ) [pure virtual]

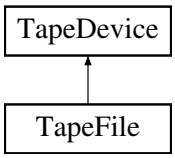
Замещается в TapeFile.

Объявления и описания членов класса находятся в файле:

- src/Tape/tape.h

7.5 Класс TapeFile

Граф наследования:TapeFile:



Открытые члены

- TapeFile (const TapeFile &other)
  Construct a new Tape File:: Tape File object.
- TapeFile (std::string filename, configuration::Configuration)
- int read () override
  Прочитать значение под головкой
- void write (int value) override
- void rewind () override
  Переместить головку в начало ленты
- void shift (int offset) override
  Переместить головку на offset позиций вперед
- configuration::Configuration getConfiguration () override
  Вернуть конфигурацию ленты
- bool isended () override
  Достигнут ли конец файла
- bool isOpen () override
  Проверить, открыт ли файл
- void toOpen () override
  Открыть файл

- virtual int read ()=0
- virtual void write (int value)=0
- virtual void rewind ()=0
- virtual void shift (int offset)=0
- virtual bool isended ()=0
- virtual configuration::Configuration getConfiguration ()=0
- virtual bool isOpen ()=0
- virtual void toOpen ()=0

7.5.1 Конструктор(ы)

7.5.1.1 TapeFile()

TapeFile::TapeFile (
const TapeFile & other)

Construct a new Tape File:: Tape File object.

Аргументы

other	

7.5.2 Методы

7.5.2.1 getConfiguration()

configuration::Configuration TapeFile::getConfiguration ( ) [override], [virtual]

Вернуть конфигурацию ленты

Возвращает

configuration::Configuration

Замещает TapeDevice.

7.5.2.2 isended()

bool TapeFile::isended ( ) [override], [virtual]

Достигнут ли конец файла

Возвращает

true
false

Замещает TapeDevice.

7.5.2.3 isOpen()

bool TapeFile::isOpen ( ) [override], [virtual]

Проверить, открыт ли файл

Возвращает

true
false

Замещает TapeDevice.

7.5.2.4 read()

int TapeFile::read ( ) [override], [virtual]

Прочитать значение под головкой

Возвращает

int

Замещает TapeDevice.

7.5.2.5 rewind()

void TapeFile::rewind ( ) [override], [virtual]

Переместить головку в начало ленты

Замещает TapeDevice.

7.5.2.6 shift()

void TapeFile::shift (
int offset) [override], [virtual]

Переместить головку на offset позиций вперед

Аргументы

offset	

Замещает TapeDevice.

7.5.2.7 toOpen()

void TapeFile::toOpen ( ) [override], [virtual]

Открыть файл

Замещает TapeDevice.

7.5.2.8 write()

void TapeFile::write (
int value) [override], [virtual]

Замещает TapeDevice.

Объявления и описания членов классов находятся в файлах:

- src/Tape/tape.h
- src/Sorter/sorter.cpp
- src/Tape/tape.cpp

7.6 Класс TapeSorter

Класс для сортировки ленты

#include <sorter.h>

Открытые члены

- TapeSorter (size_t memory_limit)
  Construct a new Tape Sorter object.
- void tapesort (std::string input_filename, std::string output_filename, configuration::Configuration new_conf)
  Основная функция для сортировки ленты

7.6.1 Подробное описание

Класс для сортировки ленты

7.6.2 Конструктор(ы)

7.6.2.1 TapeSorter()

TapeSorter::TapeSorter (
size_t memory_limit) [inline]

Construct a new Tape Sorter object.

Аргументы

memory_limit	RAM

7.6.3 Методы

7.6.3.1 tapesort()

```
void TapeSorter::tapesort (
                std::string input_filename,
                std::string output_filename,
                configuration::Configuration new_conf )
```

Основная функция для сортировки ленты

Сортировка ленты

Аргументы

input_filename	Имя входного файла-ленты
output_filename	Имя выходного файла-ленты
new_conf	Конфигурация ленты

Аргументы

input_filename	
output_filename	
new_conf	

Объявления и описания членов классов находятся в файлах:

- src/Sorter/sorter.h
- src/Sorter/sorter.cpp

7.7 Класс TestRunner

Открытые члены

- template<class TestFunc >
  void RunTest (TestFunc func, const string &test_name)

Объявления и описания членов класса находятся в файле:

- src/UnitTests/test_runner.h

Глава 8

Файлы

8.1 Файл src/Configuration/conf_reader.h

Файл с читателем конфигурации ленты из файла config.cfg.

```
#include <iostream>
#include <fstream>
#include <string>
#include <map>
```

Пространства имен

- namespace configuration

  Пространство имен для конфигурации

Определения типов

- using configuration::configurations_map = std::map< std::string, std::map< std::string, std::string > >

Функции

- configurations_map configuration::ReadConfigurations ()

  Функция, необходимая для получения конфигурации, написанной в виде [section] name - value.

8.1.1 Подробное описание

Файл с читателем конфигурации ленты из файла config.cfg.

Автор

Almaz Shagiev (ash_bashkirian@gmail.com)

Версия

0.1

Дата

2023-03-19

Авторство

8.2 conf_reader.h

См. документацию.

```
00001
00011 #pragma once
00012
00013 #include <iostream>
00014 #include <fstream>
00015 #include <string>
00016 #include <map>
00017
00018
00022 namespace configuration
00023 {
00024     using configurations_map = std::map<std::string, std::map<std::string, std::string>>;
00025
00032     configurations_map ReadConfigurations()
00033     {
00034         std::string filename = "config.cfg";
00035
00036         std::ifstream file;
00037         file.open(filename);
00038         if (!file.is_open()) {
00039             throw std::runtime_error("Failed to open file with configurations " + filename);
00040         }
00041
00042         std::map<std::string, std::map<std::string, std::string>> sections;
00043         std::string current_section;
00044
00045         std::string line;
00046         while (std::getline(file, line)) {
00047             // Игнорируем комментарии и пустые строки
00048             if (line.empty() || line[0] == '#' || line[0] == ';') {
00049                 continue;
00050             }
00051
00052             // Проверяем, секция или нет
00053             if (line[0] == '[' && line[line.size()-1] == ']') {
00054                 current_section = line.substr(1, line.size()-2);
00055             }
00056             else {
00057                 // Парсинг имени параметра и его значения
00058                 size_t delimiter_pos = line.find('=');
00059                 if (delimiter_pos == std::string::npos) {
00060                     throw std::logic_error("Invalid configuration parameter: " + line);
00061                 }
00062
00063                 std::string name = line.substr(0, delimiter_pos - 1);
00064                 std::string value = line.substr(delimiter_pos + 2);
00065                 // Сохраняем параметр в текущей секции
00066                 sections[current_section][name] = value;
00067             }
00068         }
00069         return sections;
00070     }
00071 }
```

8.3 Файл src/Configuration/configuration.h

Классы

- struct configuration::Configuration
  Структура конфигурации

Пространства имен

- namespace configuration
  Пространство имен для конфигурации

8.3.1 Подробное описание

Версия

0.1

Дата

2023-03-19

Авторство

Copyright (c) 2023

8.4 configuration.h

См. документацию.

```
00001
00010 #pragma once
00014 namespace configuration
00015 {
00019     struct Configuration
00020     {
00021         int m_rewind_delay; // задержка перемотки
00022         int m_shift_delay; // задержка движения на одну позицию
00023         int m_read_delay; // задержка чтения
00024         int m_write_delay; // задержка записывания
00025     };
00026 }
```

8.5 Файл src/RandomTapeMaker/randomtape.h

Хедер для наполнения пустого файла целыми числами случайным образом

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <random>
```

Функции

- void populateFile (const std::string &filename, const int N)

  Наполняет пустой файл случайными числами

8.5.1 Подробное описание

Хедер для наполнения пустого файла целыми числами случайным образом

Версия

0.1

Дата

2023-03-19

Авторство

Copyright (c) 2023

8.5.2 Функции

8.5.2.1 populateFile()

```
void populateFile (
            const std::string & filename,
            const int N )
```

Наполняет пустой файл случайными числами

Аргументы

filename	Имя файла для популяции
N	Количество чисел в итоговом файле

8.6 randomtape.h

См. документацию.

```
00001
00010 #pragma once
00011
00012 #include <iostream>
00013 #include <string>
00014 #include <fstream>
00015 #include <cstdlib>
```

```
00016 #include <random>
00017
00024 void populateFile(const std::string& filename, const int N) {
00025     std::random_device rd;
00026     std::mt19937 gen(rd());
00027     std::uniform_int_distribution<uint32_t> dis(0, INT_MAX);
00028     std::ofstream file(filename, std::ios::trunc);
00029     if (file.is_open()) {
00030         for (int i = 0; i < N; i++) {
00031             int randNum = dis(gen);
00032             file « " " « randNum;
00033         }
00034         file.close();
00035     } else {
00036         std::cerr « "Error opening file" « std::endl;
00037     }
00038 }
```

8.7 sorter.h

```
00001 #pragma once
00002
00003 #include "../Tape/tape.h"
00004 #include "../Configuration/configuration.h"
00005
00006 #include <string>
00007 #include <memory>
00008 #include <vector>
00009
00013 class TapeSorter {
00014 public:
00020     TapeSorter(size_t memory_limit) :
00021         m_memory_limit(memory_limit)
00022     {}
00023
00031     void tapesort(std::string input_filename, std::string output_filename, configuration::Configuration new_conf);
00032 private:
00040     std::vector<TapeFile> break_tape(TapeFile& m_input_tape, std::vector<std::string>& sub_tape_filenames);
00047     void merge_tapes(TapeFile& m_output_tape, std::vector<TapeFile>& sub_tapes);
00053     std::string create_temporary_file();
00054     size_t m_memory_limit;
00055     int m_temporary_file_counter = 0;
00056 };
```

8.8 tape.h

```
00001 #pragma once
00002
00003 #include "../Configuration/configuration.h"
00004
00005 #include <string>
00006 #include <memory>
00007 #include <fstream>
00012 class TapeDevice {
00013 public:
00014     virtual int read() = 0;
00015     virtual void write(int value) = 0;
00016     virtual void rewind() = 0;
00017     virtual void shift(int offset) = 0;
00018     virtual bool isended() = 0;
00019     virtual configuration::Configuration getConfiguration() = 0;
00020     virtual bool isOpen() = 0;
00021     virtual void toOpen() = 0;
00022 };
00023
00024 class TapeFile : public TapeDevice {
00025 public:
00026     TapeFile() = default;
00027     TapeFile(const TapeFile& other);
00028     TapeFile(std::string filename, configuration::Configuration);
00029     int read() override;
00030     void write(int value) override;
00031     void rewind() override;
00032     void shift(int offset) override;
00033     configuration::Configuration getConfiguration() override;
00034     bool isended() override;
00035     bool isOpen() override;
00036     void toOpen() override;
00037     ~TapeFile();
00038 private:
```

```
00039     std::string filename;
00040     std::fstream m_file;
00041     configuration::Configuration conf;
00042 };
```

8.9 checker.h

```
00001 #pragma once
00002 #include <iostream>
00003 #include <fstream>
00004
00012 bool isFileSorted(const std::string& filename) {
00013     std::ifstream file(filename);
00014     int previousNumber, currentNumber;
00015     file » previousNumber;
00016     while (file » currentNumber) {
00017         if (currentNumber < previousNumber) {
00018             file.close();
00019             return false;
00020         }
00021         previousNumber = currentNumber;
00022     }
00023     file.close();
00024     return true;
00025 }
```

8.10 profiler.h

```
00001 #pragma once
00002
00003 #include <chrono>
00004 #include <iostream>
00005 #include <string>
00006
00007 using namespace std;
00008 using namespace std::chrono;
00009
00014 class LogDuration
00015 {
00016 public:
00017     explicit LogDuration(const string &msg = "")
00018         : message(msg + ": "), start(steady_clock::now())
00019     {
00020     }
00021
00022     ~LogDuration()
00023     {
00024         auto finish = steady_clock::now();
00025         auto dur = finish - start;
00026         cerr « message
00027              « duration_cast<milliseconds>(dur).count()
00028              « " ms" « endl;
00029     }
00030
00031 private:
00032     string message;
00033     steady_clock::time_point start;
00034 };
00035
00036 #define UNIQ_ID_IMPL(lineno) _a_local_var_##lineno
00037 #define UNIQ_ID(lineno) UNIQ_ID_IMPL(lineno)
00038
00039 #define LOG_DURATION(message) \
00040     LogDuration UNIQ_ID(__LINE__){message};
```

8.11 Файл src/UnitTests/test_runner.h

Описание простой юнит-тестирующей системы

#include <sstream>
#include <stdexcept>
#include <iostream>

```
#include <map>
#include <set>
#include <string>
#include <vector>
```

Классы

- class TestRunner

Макросы

- #define ASSERT_EQUAL(x, y)
- #define ASSERT(x)
- #define RUN_TEST(tr, func) tr.RunTest(func, #func)

Функции

- template<class T >
  ostream & operator<< (ostream &os, const vector< T > &s)
- template<class T >
  ostream & operator<< (ostream &os, const set< T > &s)
- template<class K , class V >
  ostream & operator<< (ostream &os, const map< K, V > &m)
- template<class T , class U >
  void AssertEqual (const T &t, const U &u, const string &hint={})
- void Assert (bool b, const string &hint)

8.11.1 Подробное описание

Описание простой юнит-тестирующей системы

Версия

0.1

Дата

2023-03-19

Авторство

Copyright (c) 2023

8.11.2 Макросы

8.11.2.1 ASSERT

```
#define ASSERT(
                    x )
```

Макроопределение:

```
{                   \
ostringstream os;                   \
os « #x « " is false, "             \
« __FILE__ « ":" « __LINE__;     \
Assert(x, os.str());                \
}
```

8.11.2.2 ASSERT_EQUAL

```
#define ASSERT_EQUAL(
                    x,
                    y )
```

Макроопределение:

```
{           \
ostringstream os;                   \
os « #x « " != " « #y « ", "      \
« __FILE__ « ":" « __LINE__;     \
AssertEqual(x, y, os.str());        \
}
```

8.12 test_runner.h

См. документацию.

```
00001
00010 #pragma once
00011
00012 #include <sstream>
00013 #include <stdexcept>
00014 #include <iostream>
00015 #include <map>
00016 #include <set>
00017 #include <string>
00018 #include <vector>
00019
00020 using namespace std;
00021
00022 template <class T>
00023 ostream& operator « (ostream& os, const vector<T>& s) {
00024     os « "{";
00025     bool first = true;
00026     for (const auto& x : s) {
00027         if (!first) {
00028             os « ", ";
00029         }
00030         first = false;
00031         os « x;
00032     }
00033     return os « "}";
00034 }
00035
00036 template <class T>
00037 ostream& operator « (ostream& os, const set<T>& s) {
00038     os « "{";
00039     bool first = true;
00040     for (const auto& x : s) {
00041         if (!first) {
00042             os « ", ";
00043         }
00044         first = false;
00045         os « x;
00046     }
00047     return os « "}";
```

```
00048 }
00049
00050 template <class K, class V>
00051 ostream& operator « (ostream& os, const map<K, V>& m) {
00052     os « "{";
00053     bool first = true;
00054     for (const auto& kv : m) {
00055         if (!first) {
00056             os « ", ";
00057         }
00058         first = false;
00059         os « kv.first « ": " « kv.second;
00060     }
00061     return os « "}";
00062 }
00063
00064 template<class T, class U>
00065 void AssertEqual(const T& t, const U& u, const string& hint = {}) {
00066     if (!(t == u)) {
00067         ostringstream os;
00068         os « "Assertion failed: " « t « " != " « u;
00069         if (!hint.empty()) {
00070             os « " hint: " « hint;
00071         }
00072         throw runtime_error(os.str());
00073     }
00074 }
00075
00076 inline void Assert(bool b, const string& hint) {
00077     AssertEqual(b, true, hint);
00078 }
00079
00080 class TestRunner {
00081 public:
00082     template <class TestFunc>
00083     void RunTest(TestFunc func, const string& test_name) {
00084         try {
00085             func();
00086             cerr « test_name « " OK" « endl;
00087         } catch (exception& e) {
00088             ++fail_count;
00089             cerr « test_name « " fail: " « e.what() « endl;
00090         } catch (...) {
00091             ++fail_count;
00092             cerr « "Unknown exception caught" « endl;
00093         }
00094     }
00095
00096     ~TestRunner() {
00097         if (fail_count > 0) {
00098             cerr « fail_count « " unit tests failed. Terminate" « endl;
00099             exit(1);
00100         }
00101     }
00102
00103 private:
00104     int fail_count = 0;
00105 };
00106
00107 #define ASSERT_EQUAL(x, y) {            \
00108 ostringstream os;                       \
00109 os « #x « " != " « #y « ", "            \
00110 « __FILE__ « ":" « __LINE__;            \
00111 AssertEqual(x, y, os.str());            \
00112 }
00113
00114 #define ASSERT(x) {                     \
00115 ostringstream os;                       \
00116 os « #x « " is false, "                 \
00117 « __FILE__ « ":" « __LINE__;            \
00118 Assert(x, os.str());                    \
00119 }
00120
00121 #define RUN_TEST(tr, func) \
00122 tr.RunTest(func, #func)
00123
00124
```

Предметный указатель