Департамент профессионального образования Томской области

Областное государственное бюджетное профессиональное образовательное учреждение среднего профессионального образования

**«ТОМСКИЙ ТЕХНИКУМ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ»**

Специальность 09.02.07 «Информационные системы и программирование»

Отчёт о лабораторно-практической работе №12 по дисциплине  
«Основы алгоритмизации и программирования»

Файлы

Студенты

«число» год                            Чуприков Д. Э.

Преподаватель

«число» месяц год оценка                            Жабин Д. И.

Томск — 2022

**Оглавление**

[1 Постановка задачи 3](#_Toc81760271)

[2 Теоретические сведения 4](#_Toc81760272)

[3 Ход работы 5](#_Toc81760273)

[4 Результаты 7](#_Toc81760274)

[Приложение А. Исходный текст 8](#_Toc81760275)

[Приложение Б. Ответы на вопросы 15](#_Toc81760276)

# Постановка задачи

Цель работы: научиться писать программы с использованием файловых потоков на языке C++ в среде разработки Visual Studio 2022

Задачи:

1. Создать проект в Visual Studio.
2. Написать программу с использованием структур.
3. Выполнить программу.

Задание №1

Записать в файл последовательного доступа N произвольных натуральных чисел. Переписать в другой файл последовательного доступа числа, кратные K. Вывести полученный файл на печать.

Задание №2

Дан файл, содержащий различные даты, включающие в себя число, месяц и год. Найти:

1. год с наименьшим номером
2. все внесенные даты
3. самую позднюю дату

**Теоретические сведения**

Лямбда-выражение в программировании — специальный синтаксис для определения функциональных объектов, заимствованный из λ-исчисления. Применяется как правило для объявления анонимных функций по месту их использования, и обычно допускает замыкание на лексический контекст, в котором это выражение использовано.

Один из вариантов объявления лямбда-выражения в C++:

[](параметры){тело\_функции}

Указатель на функцию – одно из средств ЯП C++. Указатель на функцию хранит адрес функции. По сути указатель на функцию содержит адрес первого байта в памяти, по которому располагается выполняемый код функции. Самым распространенным указателем на функцию является её имя. С помощью имени функции можно вызвать её и получить результат её работы.

Но также указатель на функцию мы можем определить в виде отдельной переменной с помощью следующего синтаксиса:

тип\_возвращаемого\_значения(\*имя\_указателя)(параметры)

Указатель на функцию может быть параметром другой функции. Параметр функции типа указатель на функцию может принимать в качестве значения лямбда-выражение.

# Ход работы

Создаём проект в среде программирования Visual Studio 2022 «ОАиП ЛР – 12.1».

Пишем код первой программы (см. листинг А.1)

Результат работы первой программы:

|  |
| --- |
| Enter natural numbers(to end the sequence enter 0):  1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  0  K = 3  The multiples of K:  3  6  9  12  15 |

Создаём проект в среде программирования Visual Studio 2022 «ОАиП ЛР – 12.2».

Пишем код второй программы (см. листинг А.2)

Результат работы второй программы:

|  |
| --- |
| All dates:  26.05.2022  11.09.2003  18.06.2012  09.08.2010  11.11.2022  31.12.2020  17.04.1999  01.01.2000  The least year: 1999  The latest date: 11.11.2022 |

# Результаты

Мы научились писать программы с использованием файловых потоков на языке C++ в среде разработки Visual Studio 2022.

# Приложение А. Исходный текст

Листинг А.1

|  |
| --- |
| #include <iostream>  #include <fstream>  void ErrorFileMessage(const char\* filename)  {  std::cerr << "Cannot open the file " << filename << std::endl;  }  int main()  {  const char\* file1Name = "file1.txt";  const char\* file2Name = "file2.txt";  std::ofstream file1Write(file1Name);  if (file1Write.is\_open() == false)  {  ErrorFileMessage(file1Name);  return -1;  }  std::cout << "Enter natural numbers(to end the sequence enter 0):"  << std::endl;  while (true)  {  int n = 0;  std::cin >> n;  if (n == 0)  {  break;  }  if (n < 0)  {  std::cerr << "Received non-natural number";  return -1;  }  file1Write << n << std::endl;  }  file1Write.close();  int k = 0;  std::cout << "\nK = ";  std::cin >> k;  std::ifstream file1Read(file1Name);  if (file1Read.is\_open() == false)  {  ErrorFileMessage(file1Name);  return -1;  }  std::fstream file2(file2Name, file2.trunc | file2.in | file2.out);  if (file2.is\_open() == false)  {  ErrorFileMessage(file2Name);  return -1;  }  int num = 0;  while (file1Read >> num)  {  if (num % k == 0)  {  file2 << num << std::endl;  }  }  file1Read.close();  std::cout << "\nThe multiples of K:" << std::endl;  file2.seekp(0);  while (file2 >> num)  {  std::cout << num << std::endl;  }  file2.close();  return 0;  } |

Листинг А.2

|  |
| --- |
| main.cpp  #include <iostream>  #include <fstream>  #include <string>  #include <stdexcept>  #include "Date.h"  #include "QueryingGeneric.h"  void FileErrorMessage(const char\* filename)  {  std::cerr << "Cannot open the file " << filename << std::endl;  }  void ExceptionErrorMessage(std::exception& ex)  {  std::cerr << "Error: " << ex.what() << std::endl;  }  int main()  {  try  {  const char\* filename = "dates.txt";  std::ifstream fileInput(filename);  if (fileInput.is\_open() == false)  {  FileErrorMessage(filename);  return -1;  }  std::vector<Date> dates(0);  std::string line;  while (fileInput >> line)  {  Date date = Date::Parse(line);  dates.push\_back(date);  }  fileInput.close();  std::cout << "All dates:\n" << std::endl;  for (int i = 0; i < dates.size(); ++i)  {  std::cout << dates[i] << std::endl;  }  int(\*yearSelector) (const Date&)  = [](const Date& date) {return date.Year(); };  int leastYear = qry::Min<Date, int>(dates, yearSelector).Year();  std::cout << "\nThe least year: " << leastYear << std::endl;  Date latestDate = qry::Max<Date>(dates);  std::cout << "\nThe latest date: " << latestDate << std::endl;  }  catch (std::exception& ex)  {  ExceptionErrorMessage(ex);  }  return 0;  }  Date.h  #pragma once  #include <vector>  #include <string>  #include <iostream>  struct Date  {  private:  int m\_year;  int m\_month;  int m\_day;  public:  static const int MIN\_YEAR;  static const int MAX\_YEAR;  static const int MIN\_MONTH;  static const int MAX\_MONTH;  static const int MIN\_DAY;  static const std::vector<int> MAX\_DAYS;  static Date Parse(std::string stringDate);  Date(int year, int month, int day);  Date();  int Year() const;  int Month() const;  int Day() const;  std::string ToString() const;  friend std::ostream& operator<<(std::ostream& stream, const Date& date);  bool operator>(const Date& other);  };  Date.cpp  #include "Date.h"  #include "StringUtil.h"  #include <stdexcept>  const int Date::MIN\_YEAR = 1;  const int Date::MAX\_YEAR = 9999;  const int Date::MIN\_MONTH = 1;  const int Date::MAX\_MONTH = 12;  const int Date::MIN\_DAY = 1;  const std::vector<int> Date::MAX\_DAYS  = {31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31};  Date Date::Parse(std::string stringDate)  {  const std::string FORMAT\_ERROR\_MESSAGE  = "Revieved string has invalid format";  if (stringDate.length() != 10)  {  throw std::invalid\_argument(FORMAT\_ERROR\_MESSAGE);  }  std::vector<std::string> stringComponents = Split(stringDate, '.');  if (stringComponents.size() != 3)  {  throw std::invalid\_argument(FORMAT\_ERROR\_MESSAGE);  }  int day = std::stoi(stringComponents[0]);  int month = std::stoi(stringComponents[1]);  int year = std::stoi(stringComponents[2]);  Date result(year, month, day);  return result;  }  Date::Date(int year, int month, int day)  {  const std::string YEAR\_OUT\_OF\_RANGE\_MESSAGE  = "Recieved year was out of range";  const std::string MONTH\_OUT\_OF\_RANGE\_MESSAGE  = "Recieved month was out of range";  const std::string DAY\_OUT\_OF\_RANGE\_MESSAGE  = "Recieved day was out of range";  if (year < MIN\_YEAR || year > MAX\_YEAR)  {  throw std::invalid\_argument(YEAR\_OUT\_OF\_RANGE\_MESSAGE);  }  if (month < MIN\_MONTH || month > MAX\_MONTH)  {  throw std::invalid\_argument(MONTH\_OUT\_OF\_RANGE\_MESSAGE);  }  if (day < MIN\_DAY || day > MAX\_DAYS[month - 1])  {  throw std::invalid\_argument(DAY\_OUT\_OF\_RANGE\_MESSAGE);  }  m\_year = year;  m\_month = month;  m\_day = day;  }  Date::Date() : Date(MIN\_YEAR, MIN\_MONTH, MIN\_DAY) {}  int Date::Year() const  {  return m\_year;  }  int Date::Month() const  {  return m\_month;  }  int Date::Day() const  {  return m\_day;  }  std::string Date::ToString() const  {  const std::string YEAR\_TEMPLATE = "0000";  const std::string MONTH\_TEMPLATE = "00";  const std::string DAY\_TEMPLATE = "00";  return ReplaceFromRight(DAY\_TEMPLATE, Day())  + '.' + ReplaceFromRight(MONTH\_TEMPLATE, Month())  + '.' + ReplaceFromRight(YEAR\_TEMPLATE, Year());  }  std::ostream& operator<<(std::ostream& stream, const Date& date)  {  stream << date.ToString();  return stream;  }  bool Date::operator>(const Date& other)  {  if (Year() != other.Year())  {  return Year() > other.Year();  }  else if (Month() != other.Month())  {  return Month() > other.Month();  }  else  {  return Day() > other.Day();  }  }  QueryingGeneric.h  #pragma once  #include <vector>  namespace qry  {  template<typename TSource, typename TSelector>  TSource& Min(std::vector<TSource>& source,  TSelector(\*selector) (const TSource&))  {  size\_t minIndex = 0;  for (size\_t i = 1; i < source.size(); ++i)  {  if (selector(source[i]) < selector(source[minIndex]))  {  minIndex = i;  }  }  return source[minIndex];  }  template<typename TSource, typename TSelector>  TSource& Max(std::vector<TSource>& source,  TSelector(\*selector) (const TSource&))  {  size\_t maxIndex = 0;  for (size\_t i = 1; i < source.size(); ++i)  {  if (selector(source[i]) > selector(source[maxIndex]))  {  maxIndex = i;  }  }  return source[maxIndex];  }  template<typename TSource>  TSource& Max(std::vector<TSource>& source)  {  return Max<TSource, TSource>(  source,  [](const TSource& element) {return element; });  }  } // namespace qry  StringUtil.h  #pragma once  #include <string>  #include <vector>  std::string ReplaceFromRight(const std::string& base,  const std::string& insertable);  std::string ReplaceFromRight(const std::string& base, int insertable);  std::vector<std::string> Split(const std::string base, char separator);  StringUtil.cpp  #include "StringUtil.h"  std::string ReplaceFromRight(const std::string& base,  const std::string& insertable)  {  std::string result = base;  int j = insertable.length() - 1;  for (size\_t i = base.length() - 1; i >= 0 && j >= 0; --i)  {  result[i] = insertable[j];  --j;  }  return result;  }  std::string ReplaceFromRight(const std::string& base, int insertable)  {  return ReplaceFromRight(base, std::to\_string(insertable));  }  std::vector<std::string> Split(const std::string base, char separator)  {  std::vector<std::string> result(0);    std::string current;  for (size\_t i = 0; i < base.length(); ++i)  {  if (base[i] == separator)  {  result.push\_back(current);  current.clear();  }  else  {  current += base[i];  }  }  result.push\_back(current);  return result;  } |

# Приложение Б. Ответы на вопросы

1. *Вопрос?*

Ответ.

1. *Вопрос?*

Ответ.

1. *Вопрос?*

Ответ.