Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №3

По дисциплине **«Операционные системы»**

Тема: **«Multi-file projects with pattern MVC»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Научиться грамотно проектировать и разрабатывать многофайловые проекты на языке программирования С/C++ с использованием архитектурного шаблона проектирования и разработки гибких и масштабируемых приложений Model-View-Controller (MVC).

**Требования:**

**1)** Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

**2)** Размерность матрицы в индивидуальном задании задаётся пользователем на стадии выполнения программы. Для повышения производительности программы использовать динамическое выделение памяти.

**3)** ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

**4)** Каждое задание оформить в виде отдельной бизнес-функции.

**5)** Все функции должны быть сгруппированы по соответствующим отдельным файлам.

**6)** Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

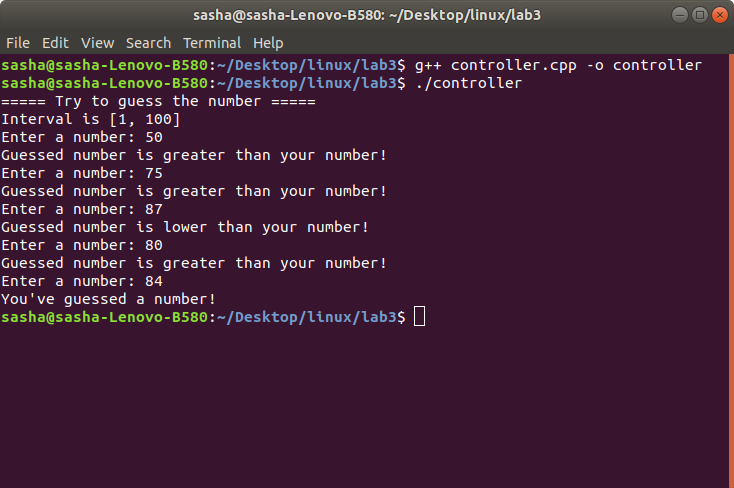
**7)** При выполнении задания запрещается использовать интегрированные средства разработки (Integrated Development Environment, IDE). Рекомендуется задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а для отладки – gdb-отладчик.

8) При разработке программ необходимо придерживаться соглашения по написанию кода на C/C++ (Code-Convention).

**Основное задание:**

Разработать интерактивную программу «Try to Guess the Number» («Попробуй угадать число»), которая эмулирует классическую игру на отгадывание числа. Суть программы (игры) сводиться к следующему: компьютер генерирует случайное число из диапазона, к примеру, от 1 до 100, а пользователь (далее игрок) пытается отгадать число за минимальное количество попыток. При каждой очередной попытке компьютер «подсказывает» игроку, как соизмеряется предложенный вариант игрока с действительным загаданным компьютером числом: загаданное число больше или меньше указанного (higher/lower). Как только игрок отгадывает число, компьютер должен «поздравить» его с выводом на экран угаданного числа и количество затраченных игроком попыток. Далее компьютер может «предложить» повторно сыграть игру или выйти из программы.

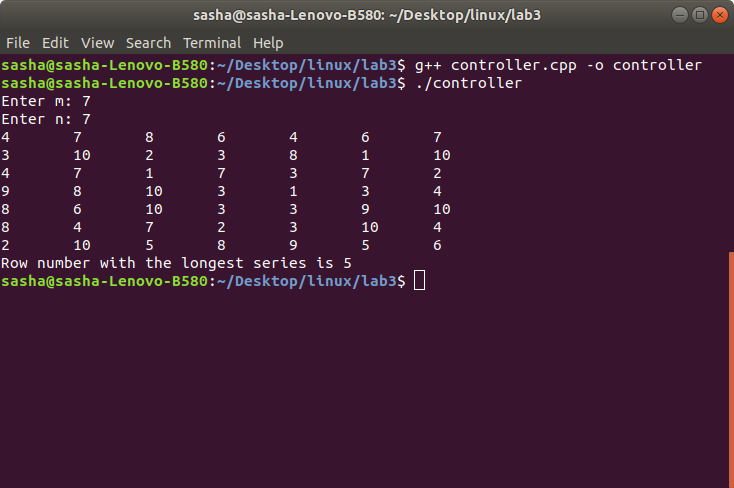
*Результаты выполнения:*



**Индивидуальное задание:**

8) Найти номер строки заданной матрицы, в которой находится самая длинная серия (последовательность одинаковых элементов).

*Результаты выполнения:*



**Контрольные вопросы:**

**1.** Объясните основные термины (объявление (declaration), определение (definition), вызов (call)), которые связаны с описанием и использованием функций в языке C/C++, приведите пример. Зачем нужно определение функции?

*Ответ:*

Объявление - Формальное объявление функции выглядит следующим образом:

тип имя\_функции(параметры)

Определение – определение добавляет логическую часть, ограниченную фигурными скобками.

*тип имя\_функции(параметры)*

{ instruction }

Вызов - Для выполнения функции ее необходимо вызвать. Вызов функции осуществляется в форме:

*имя\_функции(аргументы)*

**2.** Какие существенные преимущества имеют многофайловые проекты перед однофайловыми? Что такое сборка?

*Ответ:*

Преимущества: легко модифицируется, независимость одного файла от другого, хорошо реализованное ООП и SOLID принципы.

Сборка – это компоновка файлов одного проекта для его запуска и дальнейшей работы.

**3**. Почему в C/C++ раздельная компиляция? Опишите её преимущества и недостатки.

*Ответ:*

**Первый этап компиляции**: будут созданы объектные файлы main.o и объект.о, содержащие переносимый код. При этом main.o ничего не знает о таких функциях и обеъктах в надежде на линковщик и следующий этап компиляции.  
  
**Второй этап компиляции:** сборка единого исполняемого модуля, где должны присутствовать все необходимые функции.

**Преимущества и недостатки:**

Из-за необходимости интерпретации байт-код выполняется значительно медленнее машинного кода сравнимой функциональности, однако он более переносим (не зависит от операционной системы и модели процессора). Чтобы ускорить выполнение байт-кода, используется динамическая компиляция, когда виртуальная машина транслирует псевдокод в машинный код непосредственно перед его первым исполнением (и в при повторных обращениях к коду исполняется уже скомпилированный вариант).

**4**. Опишите общий алгоритм процесса сборки многофайлового проекта и стадии его компиляции и выполнения.

*Ответ:*

Обычно процесс сборки многофайлового проекта осуществляется по следующему алгоритму:

1. Создаются и подготавливаются исходные файлы. Здесь есть одно важное замечание: каждый файл должен быть целостным, т. е. не должен содержать незавершенных конструкций. Функции и структуры не должны разрываться. Если в рамках проекта предполагается создание исполняемой программы, то в одном из исходных файлов должна присутствовать функция main().
2. Создаются и подготавливаются заголовочные файлы. У заголовочных файлов особая роль: они устанавливают соглашения по использованию общих идентификаторов (имен) в различных частях программы. Если, например, функция func() реализована в файле a.c, а вызывается в файле b.c, то в оба файла требуется включить директивой #include заголовочный файл, содержащий объявление (прототип) нашей функции. Технически можно обойтись и без заголовочных файлов, но в этом случае функцию можно будет вызвать с произвольными аргументами, и компилятор, за отсутствием соглашений, не выведет ни одной ошибки. Подобный "слепой" подход потенциально опасен и в большинстве случаев свидетельствует о плохом стиле программирования.
3. Каждый исходный файл отдельно компилируется с опцией -c. В результате появляется набор объектных файлов.
4. Полученные объектные файлы соединяются компоновщиком в одну исполняемую программу. Если необходимо скомпоновать несколько объектных файлов (OBJ1.o, OBJ2.o и т. д.), то применяют следующий простой шаблон: $ gcc -o OUTPUT\_FILE OBJ1.o OBJ2.o ...

**5.** Что всегда будет присутствовать (сопровождать) в любой востребованной программе?

*Ответ:* Хороший КОД.

**6**. Что такое рефакторинг кода?

*Ответ:*

**Рефакторинг** –  процесс изменения внутренней структуры [программы](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%B0%D1%8F_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B0), не затрагивающий её внешнего поведения и имеющий целью облегчить понимание её работы.

**7.** Опишите концепцию проектирования приложений с использованием шаблона Model-View-Controller. Зачем он применяется? Какие преимущества даёт использование данного шаблона при проектировании и реализации программного проекта?

*Ответ:*

MVC – парадигма, предлагающая реализовывать программу с опредленными требованиями, выделяя логику, представление и управление программой в различные слои.

Данная парадигма помогает легче реализовывать принципы SOLID. Что дает хороший, качественный код, с возможностями быстрого и легкого рефакторинга, переработки, изменений, дополнений и т.п. действий.

**8.** Объясните принцип "Единственной ответственности" (Single Responsibility Principle, SRP), который используется бля проектирования и реализации независимого или слабосвязанного кода.

*Ответ:*

Один метод/функция должен выполнять только 1 действие.

**9.** Что такое нисходящее и восходящее программирование?

*Ответ:*

**Нисходящее программирование***—* систематический метод проек­тирования, кодирования и тестирования программ в виде последо­вательности шагов.

Метод **восходящего программирования** использует противопо­ложную стратегию. Программа образуется путем объединения про­стых программных компонент для создания компонент более высо­кого уровня.

**10.** В чём преимущество и для чего нужно использовать в C/C++ проектах файлызаголовки? Какие проблемы они решают? С помощью какой программной конструкции языка можно предотвратить бесконечное (рекурсивное) включение заголовочных файлов друг в друга?

*Ответ:*

Целью заголовочных файлов является удобное хранение набора объявлений объектов для их последующего использования в других программах.

1. Решает следующие проблемы: для улучшения времени сборки.
2. чтобы связать с кодом, не имея источника для определений.
3. чтобы не отмечать все "inline".

**Листинг:**

1. **model.cpp**

#include <cstdlib>

#include <iostream>

int generateNumber(int begin, int end)

{

return begin + rand() % end;

}

int tryGuess(int number, int guessedNumber)

{

if(number == guessedNumber) return 0;

else if(guessedNumber < number) return -1;

else return 1;

}

int getLongestSeries(int\* row, int n)

{

int longestSeries = 1;

int tempSeries = 1;

for(int i = 1; i < n; i++)

{

if(row[i] == row[i-1])

tempSeries++;

else if(tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

if(i == n - 1 && tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

}

return longestSeries;

}

int getRowNumberWithLongestSeries(int\*\* matrix, int m, int n)

{

int rowNumber = 1;

int greatestSeries = 1;

for(int i = 0; i < m; i++)

{

int series = getLongestSeries(matrix[i], n);

if(series > greatestSeries)

{

greatestSeries = series;

rowNumber = i + 1;

}

}

return rowNumber;

}

1. **model.h**

#include "model.cpp"

int generateNumber(int begin, int end);

int tryGuess(int number, int guessedNumber);

int getRowNumberWithLongestSeries(int\* matrix, int size);

1. **view.cpp**

#include <iostream>

using namespace std;

void printGuessResult(int result)

{

if(result == -1)

cout << "Guessed number is lower than your number!" << endl;

else if(result == 1)

cout << "Guessed number is greater than your number!" << endl;

else

cout << "You've guessed a number!" << endl;

}

void printMatrix(int\*\* matrix, int m, int n)

{

for(int i = 0; i < m; i++)

{

for(int j = 0; j < n; j++)

{

cout << matrix[i][j] << "\t";

}

cout << endl;

}

}

void printMatrixResult(int result)

{

cout << "Row number with the longest series is " << result << endl;

}

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №4

По дисциплине **«Операционные системы»**

Тема: **«Makefiles»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Научиться эффективно использовать специальные средства для автоматизации процесса компиляции, сборки и запуска многофайловых проектов.

**Требования:**

1) Для автоматизации сборки проекта необходимо использовать стандартную оболочку сценариев Linux bash (/bin/sh), а в качестве утилиты автосборщика – GNU make.

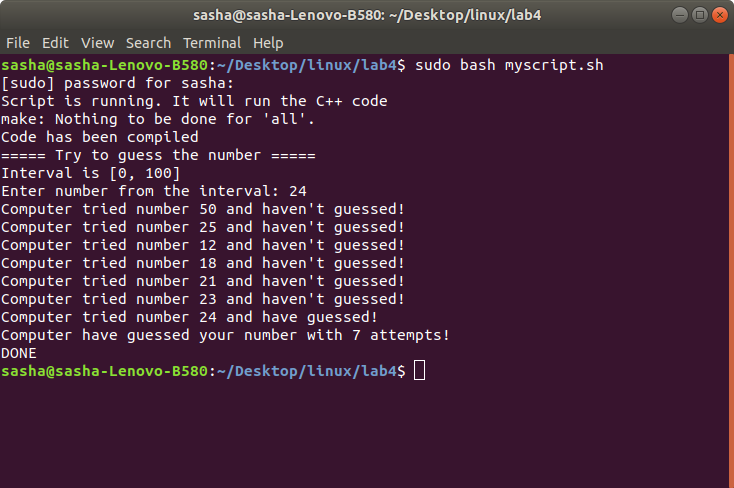
2) Для автосборки проекта с использованием GNU make необходимо описать два типа make-файлов: простой и параметризированный.

3) При выполнении задания запрещается использовать IDE. Рекомендуется задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а для отладки – gdb-отладчик.

**Основное задание:**

Разработать интерактивную программу «Try to Guess the Number» («Попробуй угадать число»), которая эмулирует классическую игру на отгадывание числа. Суть программы (игры) сводиться к следующему: компьютер генерирует случайное число из диапазона, к примеру, от 1 до 100, а пользователь (далее игрок) пытается отгадать число за минимальное количество попыток. При каждой очередной попытке компьютер «подсказывает» игроку, как соизмеряется предложенный вариант игрока с действительным загаданным компьютером числом: загаданное число больше или меньше указанного (higher/lower). Как только игрок отгадывает число, компьютер должен «поздравить» его с выводом на экран угаданного числа и количество затраченных игроком попыток. Далее компьютер может «предложить» повторно сыграть игру или выйти из программы.

*Результаты выполнения:*



**Индивидуальное задание:**

8) Найти номер строки заданной матрицы, в которой находится самая длинная серия (последовательность одинаковых элементов).

*Результаты выполнения:*

(также как и выше)

**Контрольные вопросы:**

**1**. Какие существуют способы и средства для автоматической сборки многофайловых проектов?

*Ответ:***make** *- система сборки групп программ, управляемая файлом Makefile*

**Autoconf** *- Конфигуратор для системы сборки программ*

**Automake** *- утилита для автоматического создания файлов Makefile.in*

**2.** Как осуществить сборку и запуск проекта с использованием исполняемого файла стандартной скриптовой оболочки ОС Linux bash?

*Ответ:  
sudo basj filename.sh*

**3**. Зачем и где используется утилита GNU make (автосборщик)?

*Ответ:*Почему стоит использовать утилиту make

* *она работает;*
* *легко настраивается как для новых, так и для существующих проектов;*
* *в большинстве ОС она предустановлена, если нет — её легко скачать;*
* *она крошечная и содержит мало зависимостей;*
* *make-файлы всё-таки могут быть короткими, ёмкими и красивыми;*
* *она не использует загадочные папки типа working или resource;*
* *да и вообще темной магией не занимается — всё на виду.*

**4**. Зачем нужен make-файл? Что он содержит?

*Ответ:*Для описания правил сборки проекта используются Makefile'ы. Создавая Makefile, мы [декларативно](https://ru.wikipedia.org/wiki/%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) описываем определенное состояние отношений между файлами*.*

Минимальное содержание:

цель: зависимости

[tab] команда

**5**. Опишите общий алгоритм процесса автоматической сборки программы.

*Ответ:*Компилятор берет файлы с исходным кодом и получает из них объектные файлы. Затем линковщик берет объектные файлы и получает из них исполняемый файл. Сборка = компиляция + линковка.

**6**. Опишите базовый синтаксис make-файла и общие правила определения целевых связок в make-файле: целей (targets), зависимостей (dependencies) и инструкций (instructions)? Какая цель будет обрабатываться самой первой при запуске утилиты make? Что такое «основная цель сборки» и что в неё входит?

*Ответ:*Make-файл состоит из набора зависимостей и правил. У*зависимости* есть цель или задание (выходной файл, который нужно создать) и набор исходных файлов, от которых она зависит. Правила или инструкции описывают, как создать выходной файл из зависимых файлов. Обычно цель — это единый исполняемый файл.

Make-файл читается командой

make

, определяющей выходной файл или файлы, которые должны быть сформированы, и затем сравнивающей даты и время исходных файлов, чтобы решить, какие инструкции нужно вызвать для формирования выходного файла. Часто следует выполнить другие промежуточные задания, прежде чем может быть сформирована заключительная цель. Команда

make

использует make-файл для определения порядка, в котором должны выполняться задания, и корректной последовательности запуска правил.

Если не указывать какой-либо цели в командной строке, то make выбирает в качестве главной первую, встреченную в make-файле цель.

Главная цель может быть прямо указана в командной строке при запуске make. В следующем примере make будет стремиться достичь цели iEdit (получить новую версию файла iEdit):

make iEdit

**7**. Какое необходимо дать стандартное имя make-файлу, чтобы его автоматически распознал автосборщик? А если в проекте используется нестандартное имя make-файла, как нужно вызывать утилиту make?

*Ответ:*Для make-файлов зарезервированы имена **GNUmakefile**, **makefile** и **Makefile**, причем поиск идет в указанном порядке. Если необходимо использовать нестандартное имя, то его можно передать явно через опцию -f. Когда make-файл уже написан, для сборки проекта достаточно выполнить в каталоге, где он находится, команду make.

**8**. Параметризация процесса сборки и переменные make-файла?

*Ответ:*Переменные в make-файлах. GNU Make поддерживает два способа задания переменных, которые несколько различаются по смыслу. Первый способ - традиционный, с помощью оператора '=': compile\_flags = -O3 -funroll-loops -fomit-frame-pointer. GNU Make поддерживает также и второй, новый способ задания переменной - с помощью оператора ':=': compile\_flags := -O3 -funroll-loops -fomit-frame-pointer. В этом случае переменная работает подобно ``обычным'' текстовым переменным в каком-нибудь из языков программирования

**9**. Зачем нужен рекурсивный вызов make-утилиты? Как его осуществить?

*Ответ:*«Рекурсивный make» это распространённый паттерн при задании правил makefile’а когда правило создаёт другую сессию make’а. Так как каждая сессия make’а только один раз читает makefile верхнего уровня, то это — естественный способ для описания makefile’а для проекта, состоящего из нескольких под-проектов.

«Рекурсивный make» создаёт много проблем. Но писать makefile’ы, которые не используют рекурсию — на самом деле сложная задача.

**10**. Как избежать перекомпиляции проекта и при этом сделать все файлы исходного кода программы последними или «свежими» (up to date), т.е. не нуждающимися в перекомпиляции?

*Ответ:  
-*

**11**. Как генерировать случайные значения с использованием стандартных функций rand() и srand(…) из стандартной C-библиотеки stdlib? Приведите пример.

*Ответ:***Функция srand()**устанавливает передаваемое пользователем значение в качестве стартового. srand() следует вызывать только один раз: в начале программы (обычно в верхней части функции main()).

**Функция rand()**генерирует следующее случайное число в последовательности. Оно будет находиться в диапазоне от 0 до RAND\_MAX ([**константа**](https://ravesli.com/urok-37-simvolnye-konstanty-const-constexpr/) в cstdlib, значением которой является 32 767).

**Листинг:**

* 1. **controller**

#include <iostream>

#include "model.h"

#include "view.h"

using namespace std;

void executeMainTask();

void executeIndividualTask();

int main()

{

executeMainTask();

//executeIndividualTask();

return 0;

}

void executeMainTask()

{

cout << "===== Try to guess the number =====" << endl;

cout << "Interval is [0, 100]" << endl;

int guessedNumber;

do{

cout << "Enter number from the interval: ";

cin >> guessedNumber;

if(guessedNumber < 0 || guessedNumber > 100)

{

cout << "Your number isn't from interval! Try again!" << endl;

}

} while(guessedNumber < 0 || guessedNumber > 100);

int result = tryGuess(guessedNumber);

printGuessResult(result);

}

void executeIndividualTask()

{

int m, n;

cout << "Enter m: ";

cin >> m;

cout << "Enter n: ";

cin >> n;

int\*\* matrix = new int\*[m];

for(int i = 0; i < m; i++)

{

matrix[i] = new int[n];

}

for(int i = 0; i < m; i++)

{

for(int j = 0; j < n; j++)

{

matrix[i][j] = generateNumber(1, 10);

}

}

//matrix[0][1] = 84;

//matrix[3][3] = 27;

//matrix[3][4] = 27;

printMatrix(matrix, m, n);

int result = getRowNumberWithLongestSeries(matrix, m, n);

printMatrixResult(result);

for(int i = 0; i < m; i++)

{

delete[] matrix[i];

}

delete[] matrix;

}

* 1. **model.cpp**

include <cstdlib>

#include <iostream>

#include "view.h"

#include "model.h"

int generateNumber(int begin, int end)

{

return begin + rand() % end;

}

int tryGuess(int guessedNumber)

{

int begin = 0;

int end = 100;

int number;

int attempts = 0;

do

{

number = begin + (end - begin) / 2;

if(number < guessedNumber)

{

begin = number;

printFailedGuessAttempt(number);

} else if(number > guessedNumber)

{

end = number;

printFailedGuessAttempt(number);

} else

{

printSuccessfulGuessAttempt(number);

}

attempts++;

} while(number != guessedNumber);

return attempts;

}

int getLongestSeries(int\* row, int n)

{

int longestSeries = 1;

int tempSeries = 1;

for(int i = 1; i < n; i++)

{

if(row[i] == row[i-1])

tempSeries++;

else if(tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

if(i == n - 1 && tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

}

return longestSeries;

}

int getRowNumberWithLongestSeries(int\*\* matrix, int m, int n)

{

int rowNumber = 1;

int greatestSeries = 1;

for(int i = 0; i < m; i++)

{

int series = getLongestSeries(matrix[i], n);

if(series > greatestSeries)

{

greatestSeries = series;

rowNumber = i + 1;

}

}

return rowNumber;

}

* 1. **model.h**

int generateNumber(int begin, int end);

int tryGuess(int guessedNumber);

int getRowNumberWithLongestSeries(int\*\* matrix, int m, int n);

* 1. **view.cpp**

#include <iostream>

#include "view.h"

using namespace std;

void printFailedGuessAttempt(int number)

{

cout << "Computer tried number " << number << " and haven't guessed!" << endl;

}

void printSuccessfulGuessAttempt(int number)

{

cout << "Computer tried number " << number << " and have guessed!" << endl;

}

void printGuessResult(int result)

{

cout << "Computer have guessed your number with " << result << " attempts!" << endl;

}

void printMatrix(int\*\* matrix, int m, int n)

{

for(int i = 0; i < m; i++)

{

for(int j = 0; j < n; j++)

{

cout << matrix[i][j] << "\t";

}

cout << endl;

}

}

void printMatrixResult(int result)

{

cout << "Row number with the longest series is " << result << endl;

}

* 1. **view.h**

#ifndef view

#define view

void printGuessResult(int);

void printMatrix(int\*\* matrix, int m, int n);

void printMatrixResult(int);

void printFailedGuessAttempt(int);

void printSuccessfulGuessAttempt(int number);

#else

#pragma once

#endif

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №5

По дисциплине **«Операционные системы»**

Тема: **«Memory and Args»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Закрепить знания и практические навыки эффективного использования памяти при проектировании и реализации интерактивных приложений на языке программирования C/C++.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Входные данные для программы должны вводиться в виде набора строк! Если в задании сказано про «слова» – это значит, что вводится должна одна строка, которая состоит из несколько слов. Если сказано про «строки», то должен вводиться набор строк, каждая из которых состоит минимум из двух трёх слов.

3) Предусмотреть два способа инициализации данных: с помощью аргументов командной строки и пользовательского ввода (при запуске программа должна проверять наличие параметров и использовать их для выполнения своего алгоритма при их наличии, в противном случае, запрашивать соответствующие данные у пользователя).

4) Программа должна выводить на консоль исходные данные и конечный результат.

5) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

6) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

7) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам.

8) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

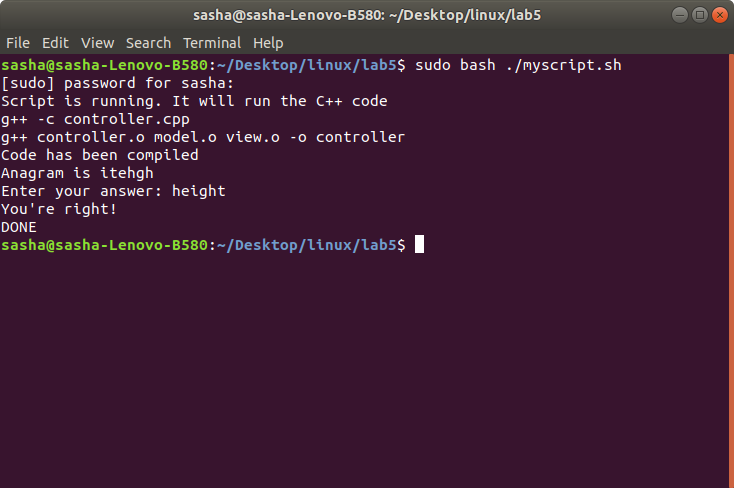
9) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++.

10) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make

**Основное задание:**

Написать программу «Анаграммы» (anagrams), которая реализует популярную словесную игру. Игра «Анаграммы» очень интересная и необычная игра в которой нет ни картинок ни слов. Суть игры (программы) заключается в следующем: формируется группа слов, компьютер случайным образом выбирает одно из слов и случайным образом переставляет в нём буквы, а затем представляет пользователю (игроку). Цель игрока – угадать выбранное компьютером слово.

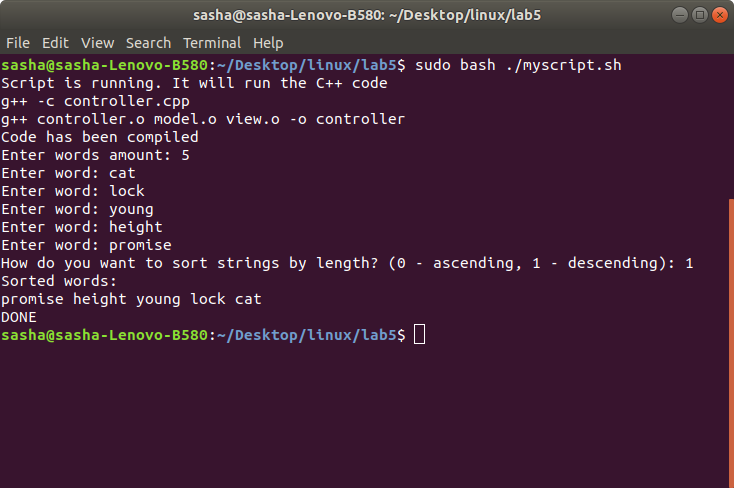
*Результаты выполнения*

**

**Индивидуальное задание:**

2) Упорядочить и вывести строки в порядке возрастания (убывания) значений их длины.

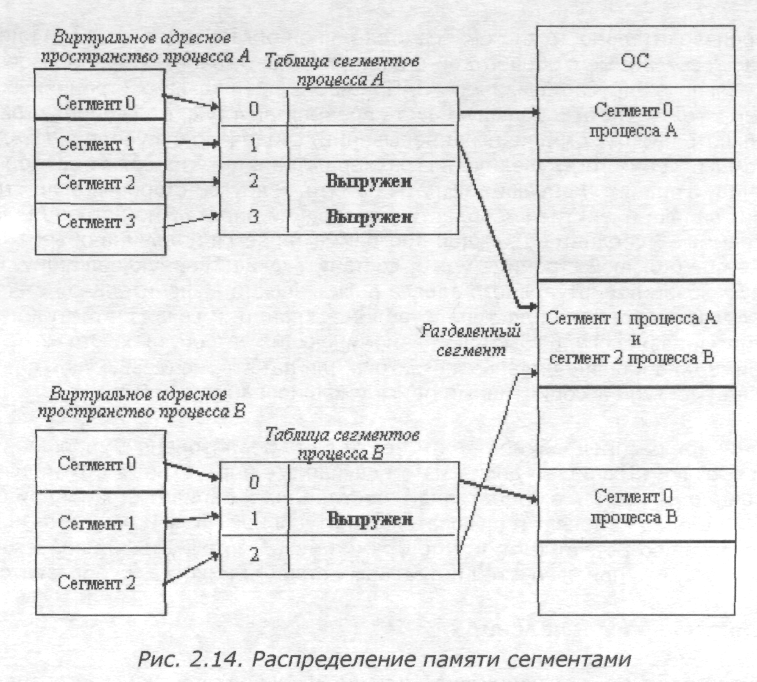
*Результаты выполнения*

**

**Контрольные вопросы:**

**1.** На какие укрупнённые блоки (области или сегменты) делиться виртуальное адресное пространство, которое выделяется операционной системой для каждой программы?

*Ответ:*



**2.** Для чего каждый из блоков нужен, что там хранится и в какой момент определяется его размер?

*Ответ:*

Размер пользовательского процесса ограничен объемом логического адресного пространства. Характерный размер логической памяти определяется разрядностью архитектуры и составляет для современных систем 232 (в недалеком будущем 264) *байт*. Эта величина обычно существенно превышает объем оперативной памяти, поэтому часть пользовательского процесса прозрачным образом может быть размещена во внешней памяти. Поэтому у пользователя создается иллюзия того, что он имеет дело с виртуальной памятью, отличной от реальной, размер которой потенциально больше, чем размер оперативной памяти. В дальнейшем наряду с термином "*логическая память*" будет употребляться термин "*виртуальная память*".

Для определения схемы виртуальной памяти, реализованной в ОС *Windows*, лучше всего подходит термин "сегментно-*страничная виртуальная память*"

**3.** Что такое класс памяти (storage class) переменной и что он определяет?

*Ответ:*

Класс памяти переменной (англ. Storage class) — понятие в некоторых языках программирования. Он определяет область видимости переменной, а также как долго переменная находится в памяти.

**4**. Какие существуют разновидности классов памяти в языках программирования С/С++ и их основные отличия?

*Ответ:*

* Автоматический (automatic);
* Регистровый (register);
* Статический (static);
* Внешний (external).

АВТОМАТИЧЕСКИЕ

По умолчанию, локальные переменные имеют класс auto. Такие переменные располагаются на стеке, а их область видимости ограничена своим блоком. Спецификатор auto уведомляет компилятор о том, что локальная переменная, перед именем которой он стоит, создаётся при входе в блок и разрушается при выходе из блока. Все переменные, определённые внутри функции, являются автоматическими по умолчанию, и поэтому ключевое слово auto используется крайне редко.

РЕГИСТРОВЫЙ

Когда мы определяем регистровую переменную, то мы просим компилятор, чтобы переменная располагалась в регистре, а не в оперативной памяти. Компилятор может сделать переменную регистровой, если позволяют условия (регистры не заняты, и по мнению компилятора это не приведёт к увеличению издержек). Регистровые переменные определяются с помощью служебного слово register перед типом.

СТАТИЧЕСКИЙ

Спецификатор static указывает компилятору на хранение локальной переменной во время всего жизненного цикла программы вместо ее создания и разрушения при каждом входе в область действия и выходе из неё. Следовательно, возведение локальных переменных в ранг статистических позволяет поддерживать их значения между вызовами функций.

Модификатор static можно также применять к глобальным переменным. В этом случае область видимости такой переменной ограничивается файлом, в котором она объявлены, это означает, что переменная будет иметь внутреннюю привязку. Внутренняя привязка говорит о том, что индикатор известен только внутри своего файла. Если функция объявлена как static, то она видна только в своём файле. Из другого файла к static функции обратиться нельзя.

ВНЕШНИЙ

Внешние переменные вводятся как противоположные автоматическим. Это глобальные переменные и к ним можно обращаться именами из любой функции. Поскольку внешние переменные доступны везде, их можно использовать для связи между функциями, не пренебрегая механизму формальных параметров.

Класс памяти extern в Си используем в двух случаях:

* если переменная объявляется в программе ниже, чем ссылка на неё;
* если переменная объявлена в другом модуле.

**5**. Какие есть классы памяти у переменных? Какой из классов памяти и при каких условия используется по умолчанию (неявно) при объявлении переменной?

*Ответ:*

Классы памяти переменных. По умолчанию,локальные переменные имеют класс auto. Такие переменные располагаются на стеке а их область видимости ограничена своим блоком

* Автоматический (automatic);
* Регистровый (register);
* Статический (static);
* Внешний (external).

**6**. Какие есть классы памяти у функций? Какой из классов памяти и при каких условия используется по умолчанию (неявно) при описании функции?

*Ответ:*

Указатель – переменная, значением которой является адрес ячейки памяти. То есть указатель ссылается на блок данных  из области памяти, причём на самое его начало. Указатель может ссылаться на переменную или функцию.

В основном нужны для работы с динамическими массивами и для передачи данных в функции.

**7**. Что такое указатели и зачем они нужны при программировании?

*Ответ:*

**8**. Опишите основные операторы (операции), которые применяются для работы с указателями.

*Ответ:*

Ссылки – особый тип данных, являющийся скрытой формой указателя, который при использовании автоматически разименовывается.

1. Нельзя объявить массив ссылок.
2. У ссылки нет адреса.
3. Существует арифметика указателей, но нет арифметики ссылок.
4. Указатель может иметь «невалидное» значение с которым его можно сравнить перед использованием.

**9.** Что такое ссылка и чем указатели отличаются от ссылок в языке программирования С++?

*Ответ:*

**10.** Для чего необходимо использовать динамическое распределение памяти?

*Ответ:*

Динамическое распределение памяти — способ выделения оперативной памятикомпьютера для объектов в программе, при котором выделение памяти под объект осуществляется во время выполнения программы.

**11.** Опишите основные функции стандартной библиотеки языка программирования С и особенность их использования для работы с динамической памятью: malloc, calloc, realloc, free.

*Ответ:*

|  | |
| --- | --- |
| |  |  | | --- | --- | | 1 | void\* malloc(size\_t size); | |  |

Выделяет блок памяти размером size.

|  | |
| --- | --- |
| |  |  | | --- | --- | | 1 | void\* calloc(size\_t num, size\_t size); | |  |

Выделяет блок памяти для массива из num эелементов, где каждый элемент имеет размер равный size и выставляет все биты блока памяти в 0. То есть по сути выделяет проинициализированный нулем блок размера num \* size.

|  | |
| --- | --- |
| |  |  | | --- | --- | | 1 | void\* realloc(void\* ptr, size\_t size); | |  |

**12.** Опишите основные операторы языка программирования С++ для работы с динамической памятью: new и delete.

*Ответ:*

New – оператор выделяющий область в хип-памяти под объект, а также запускает конструктор объекта.

Delete – удаление объектов и чистка памяти.

**13.** Какие бывают ошибки при работе с динамической памятью?

*Ответ:*

1. ошибка связанна с возможным переполнением динамической области памяти
2. утечка памяти
3. повторный возврат памяти
4. нулевой указатель

**14**. Какие существуют способы передачи параметров в функцию в языках программирования С/C++?

*Ответ:*

- передача по значению

- передача по адресу(указатель)

- передача по ссылке

**15**. Что происходит с оригинальной переменной, передаваемой в качестве параметра в функцию, и памятью, выделенной для программы, в момент вызова соответствующей функции?

*Ответ:*

Создается копия в новом фрагменте памяти.

**16**. Что такое «утечка памяти» (memory leak)? Когда, где и по чьей вине это может возникнуть?

*Ответ:*

Утечка памяти (memory leak) - это процесс неконтролируемого уменьшения объёма свободной оперативной или виртуальной памяти компьютера, связанный с ошибками в работающих программах, вовремя не освобождающих ненужные уже участки памяти.

Происходит по вине программиста. Однако в более молодых языках присутствует автосборщик мусора, и утечка памяти практически невозможна.

**17**. Как создать одномерный массив конкретного типа данных? Как объявить массив, который может хранить элементы различных типов данных?

*Ответ:*

Type name[amount];

Хранение разных типов данных в массивах доступно не во всех языках, для этого лучше воспользоваться коллекциями. Например vector в с++.

**18**. Какие существуют способы для работать с одномерными динамическими массивами?

*Ответ:*

Для работы динамических массивов нам понадобится при инициализации [указатель](https://codelessons.ru/cplusplus/ukazateli-v-c-podrobnoe-rukovodstvo.html)(всего лишь при инициализации!) и уже знакомый конструктор new.

**19**. Опишите способы динамического выделения и освобождения памяти для многомерного массива (минимум три)? Какой из способов эффективней и почему?

*Ответ:*

int \*p;

int sizeof(тип);

// Выделение памяти под указатели на строки  
  a = (int\*\*)malloc(n \* sizeof(int\*));

// Выделение памяти  
  a = (int\*)malloc(n\*m \* sizeof(int));

a = (int\*\*)malloc(n \* sizeof(int\*));  
  m = (int\*)malloc(n \* sizeof(int)); // массив кол-ва элеменов в строках массива

**20**. Какие существуют способы для работать с многомерными динамическими массивами

*Ответ:*

Работа с динамическим многомерным массивом синтаксически полностью совпадает с работой с многомерным массивом.

**Листинг:**

1. **controler.cpp**

#include <iostream>

#include <string>

#include "model.h"

#include "view.h"

using namespace std;

void executeMainTask();

void executeIndividualTask();

int main()

{

//executeMainTask();

executeIndividualTask();

return 0;

}

void executeMainTask()

{

string anagram = generateAndShuffleRandomWord();

cout << "Anagram is " << anagram << endl;

string word;

cout << "Enter your answer: ";

cin >> word;

bool result = guessAnagram(word, anagram);

printAnagramResult(result);

}

void executeIndividualTask()

{

int n;

cout << "Enter words amount: ";

cin >> n;

string\* words = new string[n];

for(int i = 0; i < n; i++)

{

cout << "Enter word: ";

cin >> words[i];

}

cout << "How do you want to sort strings by length? (0 - ascending, 1 - descending): ";

int choice;

cin >> choice;

if(choice == 0)

{

sortArray(words, n, true);

} else

{

sortArray(words, n, false);

}

cout << "Sorted words:" << endl;

printStrings(words, n);

delete[] words;

}

1. **model.cpp**

#include <cstdlib>

#include <string>

#include <algorithm>

#include "model.h"

using namespace std;

string\* getWords()

{

return new string[5] {"cat", "dog", "computer", "height", "weight"};

}

string generateAndShuffleRandomWord()

{

int randomIndex = rand() % 4;

string\* words = getWords();

string word = words[randomIndex];

random\_shuffle(word.begin(), word.end());

delete[] words;

return word;

}

bool areAnagrams(string word, string anagram)

{

int n1 = word.length();

int n2 = anagram.length();

if (n1 != n2)

return false;

sort(word.begin(), word.end());

sort(anagram.begin(), anagram.end());

for (int i = 0; i < n1; i++)

if (word[i] != anagram[i])

return false;

return true;

}

bool guessAnagram(string word, string anagram)

{

string\* words = new string[5] {"cat", "dog", "computer", "height", "weight"};

string guessedWord;

for(int i = 0; i < 5; i++)

{

if(areAnagrams(words[i], anagram))

{

guessedWord = words[i];

}

}

delete[] words;

return guessedWord == word;

}

bool matches(string first, string second, bool ascending)

{

if(ascending)

{

return first.length() > second.length();

} else

{

return first.length() < second.length();

}

}

void sortArray(string\* arr, int size, bool ascending)

{

for(int i = 0; i < size - 1; i++)

{

for(int j = 0; j < size - i - 1; j++)

{

if(matches(arr[j], arr[j + 1], ascending))

{

string temp = arr[j];

arr[j] = arr[j + 1];

arr[j + 1] = temp;

}

}

}

}

1. **model.h**

#include <string>

using namespace std;

#pragma once

string generateAndShuffleRandomWord();

bool guessAnagram(string word, string anagram);

void sortArray(string\* arr, int size, bool ascending);

1. **view.cpp**

#include <iostream>

#include "view.h"

using namespace std;

void printAnagramResult(bool result)

{

if(result)

{

cout << "You're right!" << endl;

} else

{

cout << "You're wrong!" << endl;

}

}

void printStrings(string\* arr, int size)

{

for(int i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

cout << endl;

}

1. **view.h**

#include <string>

#pragma once

void printAnagramResult(bool);

void printStrings(std::string\* arr, int size);

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №6

По дисциплине **«Операционные системы»**

Тема: **«Libraries»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Изучить и закрепить на практике создание и использование статически и динамически подключаемых библиотек в операционных системах семейства Linux.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Входные данные для программы должны вводиться в виде набора строк! Если в задании сказано про «слова» – это значит, что вводится должна одна строка, которая состоит из несколько слов. Если сказано про «строки», то должен вводиться набор строк, каждая из которых состоит минимум из двух трёх слов.

3) Предусмотреть два способа инициализации данных: с помощью аргументов командной строки и пользовательского ввода (при запуске программа должна проверять наличие параметров и использовать их для выполнения своего алгоритма при их наличии, в противном случае, запрашивать соответствующие данные у пользователя).

4) Программа должна выводить на консоль исходные данные и конечный результат.

5) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

6) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

7) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

8) Предусмотреть создание двух типов библиотек: статической и динамической. 9) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

10) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

11) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

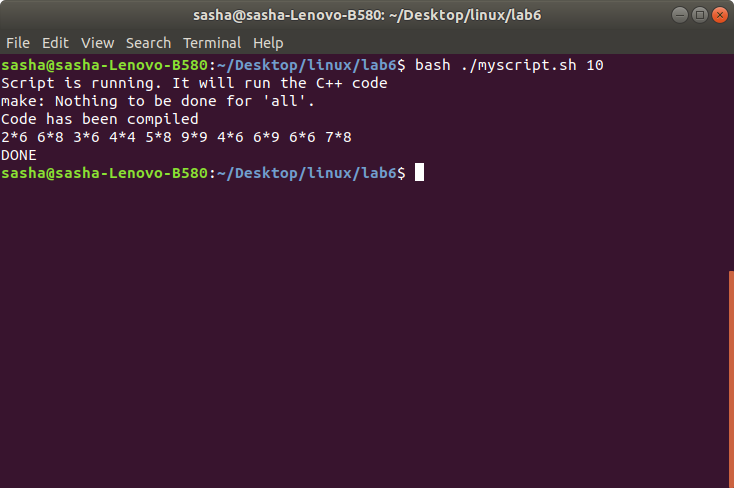
12) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

13) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Для проверки остаточных знаний учеников после рождественских каникул, учитель младших классов решил начинать каждый урок с того, чтобы задавать каждому ученику пример из таблицы умножения, но в классе, к примеру, 15 человек, а примеры среди них не должны повторяться. В помощь учителю разработайте универсальную библиотечную функцию, которая будет выводить на экран 15 (или N) случайных примеров из таблицы умножения (к примеру, от 2\*2 до 9\*9, потому что задания по умножению на 1 и на 10 – слишком просты). При этом среди 15 примеров не должно быть повторяющихся (примеры 2\*3 и 3\*2 и им подобные пары считать повторяющимися). Алгоритм поиска данных комбинаций должен быть эффективным!!! Напишите тестовую программу, которая бы демонстрировала работоспособность данной библиотечной функции.

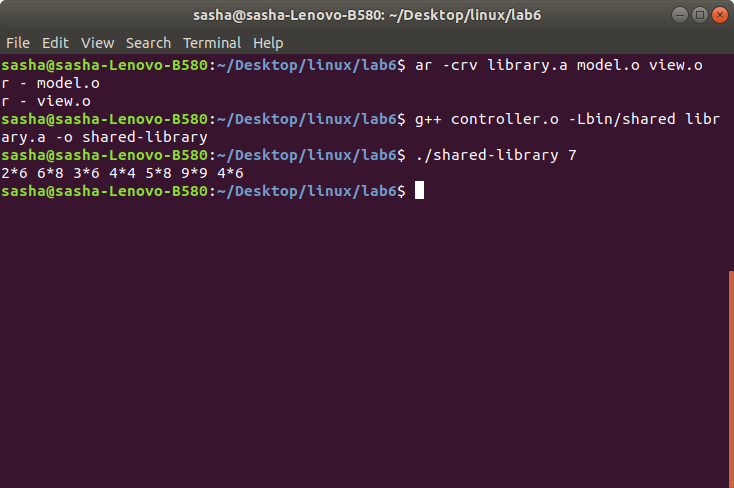
*Результаты выполнения*



**Индивидуальное задание:**

Произвести рефакторинг предыдущей лабораторной работы: вынесите код функций бизнес логики в отдельную библиотеку.

*Результаты выполнения*

****

**Контрольные вопросы:**

**1.** Что такое библиотеки и для чего они используются?

*Ответ:*

Библиотеки - это часть готового функционала, который используется большим количеством программистов.  
Чтобы не писать тот функционал, который всем нужен и который уже реализован и протестирован

**2.** Опишите существующие типы библиотек и принципы их использования.

*Ответ:*

Библиотеки делятся на статические и динамические.

|  |
| --- |
| Ввод-вывод:   * [fstream](https://ru.wikipedia.org/wiki/Fstream) * iomanip * [ios](https://ru.wikipedia.org/w/index.php?title=Ios_(C%2B%2B)&action=edit&redlink=1) * iosfwd * [iostream](https://ru.wikipedia.org/wiki/Iostream) (ostream, istream) * [sstream](https://ru.wikipedia.org/wiki/Sstream) * streambuf   [Многопоточность](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D0%BF%D0%BE%D1%82%D0%BE%D1%87%D0%BD%D0%BE%D1%81%D1%82%D1%8C):   * atomic (C++11) * condition\_variable (C++11) * future (C++11) * mutex (C++11) * shared\_mutex (C++14) * thread (C++11)   Прочие:   * [algorithm](https://ru.wikipedia.org/wiki/Algorithm_(C%2B%2B)) * [bitset](https://ru.wikipedia.org/w/index.php?title=Bitset_(C%2B%2B)&action=edit&redlink=1) * chrono (C++11) * codecvt (C++11) * complex * exception * [functional](https://ru.wikipedia.org/wiki/Functional_(C%2B%2B)) * initializer\_list (C++11) * [iterator](https://ru.wikipedia.org/w/index.php?title=%D0%98%D1%82%D0%B5%D1%80%D0%B0%D1%82%D0%BE%D1%80%D1%8B_%D0%B2_C%2B%2B&action=edit&redlink=1) * limits * locale * memory * [new](https://ru.wikipedia.org/wiki/New_(C%2B%2B)) * numeric * random (C++11) * ratio (C++11) * regex (C++11) * stdexcept * [string](https://ru.wikipedia.org/wiki/String_(C%2B%2B)) * system\_error (C++11) * tuple (C++11) * typeindex (C++11) * typeinfo * type\_traits (C++11) * utility * valarray |
| [**Стандартная библиотека шаблонов**](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%88%D0%B0%D0%B1%D0%BB%D0%BE%D0%BD%D0%BE%D0%B2) |
| [Последовательные контейнеры](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D0%BB%D0%B5%D0%B4%D0%BE%D0%B2%D0%B0%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BD%D1%82%D0%B5%D0%B9%D0%BD%D0%B5%D1%80):   * [array](https://ru.wikipedia.org/w/index.php?title=Array_(C%2B%2B)&action=edit&redlink=1) (C++11) * [deque](https://ru.wikipedia.org/w/index.php?title=Deque_(C%2B%2B)&action=edit&redlink=1) * [forward\_list](https://ru.wikipedia.org/w/index.php?title=Forward_list&action=edit&redlink=1) (C++11) * [list](https://ru.wikipedia.org/w/index.php?title=List_(C%2B%2B)&action=edit&redlink=1) * [vector](https://ru.wikipedia.org/wiki/Vector_(C%2B%2B)), vector<bool>   Ассоциативные контейнеры:   * [map](https://ru.wikipedia.org/wiki/Map_(C%2B%2B)) * [set](https://ru.wikipedia.org/w/index.php?title=Set_(C%2B%2B)&action=edit&redlink=1) * [multimap](https://ru.wikipedia.org/w/index.php?title=Multimap_(C%2B%2B)&action=edit&redlink=1) * [multiset](https://ru.wikipedia.org/w/index.php?title=Set_(C%2B%2B)&action=edit&redlink=1) * [unordered\_map](https://ru.wikipedia.org/w/index.php?title=Unordered_map_(C%2B%2B)&action=edit&redlink=1) (C++11) * [unordered\_set](https://ru.wikipedia.org/w/index.php?title=Unordered_set&action=edit&redlink=1) (C++11) * [unordered\_multimap](https://ru.wikipedia.org/w/index.php?title=Unordered_map_(C%2B%2B)&action=edit&redlink=1) (C++11) * [unordered\_multiset](https://ru.wikipedia.org/w/index.php?title=Unordered_set&action=edit&redlink=1) (C++11)   Контейнеры-адаптеры:   * [stack](https://ru.wikipedia.org/w/index.php?title=Stack_(C%2B%2B)&action=edit&redlink=1) * priority\_queue * [queue](https://ru.wikipedia.org/w/index.php?title=Queue_(C%2B%2B)&action=edit&redlink=1) |
| [**Стандартная библиотека языка программирования Си**](https://ru.wikipedia.org/wiki/%D0%A1%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D0%B0%D1%8F_%D0%B1%D0%B8%D0%B1%D0%BB%D0%B8%D0%BE%D1%82%D0%B5%D0%BA%D0%B0_%D1%8F%D0%B7%D1%8B%D0%BA%D0%B0_%D0%A1%D0%B8) |
| * [cassert](https://ru.wikipedia.org/wiki/Assert.h) * [cctype](https://ru.wikipedia.org/wiki/Ctype.h) * [cerrno](https://ru.wikipedia.org/wiki/Errno.h) * [cfenv](https://ru.wikipedia.org/wiki/Fenv.h) (C++11) * [cfloat](https://ru.wikipedia.org/wiki/Float.h) * [cinttypes](https://ru.wikipedia.org/wiki/Inttypes.h) (C++11) * [ciso646](https://ru.wikipedia.org/wiki/Iso646.h) * [climits](https://ru.wikipedia.org/wiki/Limits.h) * [clocale](https://ru.wikipedia.org/wiki/Locale.h) * [cmath](https://ru.wikipedia.org/wiki/Math.h) * [csetjmp](https://ru.wikipedia.org/wiki/Setjmp.h) * [csignal](https://ru.wikipedia.org/wiki/Signal.h) * [cstdarg](https://ru.wikipedia.org/wiki/Stdarg.h) * [cstdbool](https://ru.wikipedia.org/wiki/Stdbool.h) (C++11) * [cstddef](https://ru.wikipedia.org/wiki/Stddef.h) * [cstdint](https://ru.wikipedia.org/wiki/Stdint.h) (C++11) * [cstdio](https://ru.wikipedia.org/wiki/Stdio.h) * [cstdlib](https://ru.wikipedia.org/wiki/Stdlib.h) * [cstring](https://ru.wikipedia.org/wiki/String.h) * [ctgmath](https://ru.wikipedia.org/w/index.php?title=Tgmath.h&action=edit&redlink=1) (C++11) * [ctime](https://ru.wikipedia.org/wiki/Time.h) * [cuchar](https://ru.wikipedia.org/w/index.php?title=Uchar.h&action=edit&redlink=1) (C++11) * [cwchar](https://ru.wikipedia.org/wiki/Wchar.h) * [cwctype](https://ru.wikipedia.org/wiki/Wctype.h) |

**3.** Опишите преимущества и недостатки каждого из типа библиотек.

*Ответ:*

Статическая библиотека это фактически архив объектных файлов, который используется в процессе статической линковки. В результате статической линковки из многих объектных файлов получается один исполняемый, запускается статическая линковка в момент создания исполняемого файла.  
  
Динамическая библиотека это фактически исполняемый файл (т.е. DLL и EXE в Windows имеют одинаковый формат). Динамическая линковка запускается в момент создания процесса (когда вы запускаете исполняемый файл на выполнение), линкуются между собой несколько исполняемых файлов каждый раз, когда создается новый процесс. Так же возможна динамическая линковка уже после запуска, т.е. новая библиотека может быть подгружена в адресное пространство ужа работающего процесса.

**4.** Как подключить библиотеку к программе? Опишите специфику подключения.

*Ответ:*

Так

#include "stdlib.h"

Или так

#include <iostream>

**5.** На какой стадии происходит подключение библиотек (объектных файлов) к основному модулю программы?

*Ответ:*

Подключение библиотекипроисходит на этапе компоновки исполняемого файла.

**6.** Чем отличается заголовочные файлы (файлы с расширением \*.h) от файлов библиотек?

*Ответ:*

Конкретно, .lib (.a), .dll - библиотеки: упакованные определённым образом реализации (функций, методов и пр.). Чтобы их использовать нужны заголовочные файлы, в которых, обычно, содержатся прототипы (интерфейсы) функций, методов и пр.

**7.** Как создать статическую (динамическую) библиотеку и подключить её к программе?

*Ответ:*

-ar –crv lib\_mae.a FILES.o

Link with main

Создание проекта **статической** библиотеки в Visual Studio 2019

1. В строке меню последовательно выберите файл > создать > проект , чтобы открыть диалоговое окно Создание нового проекта .
2. В верхней части диалогового окна для параметра Язык установите значение C++ , для параметра Платформа — значение Windows, а для параметра Тип проекта — значение Библиотека.
3. В отфильтрованном списке типов проектов выберите статическая библиотека , а затем нажмите кнопку Далее. На следующей странице введите MathFuncsLib в поле имя , чтобы указать имя проекта, и при необходимости укажите расположение проекта.
4. Нажмите кнопку Создать, чтобы создать клиентский проект.

### Создание проекта библиотеки **DLL** в Visual Studio 2019

1. В строке меню последовательно выберите **файл > создать > проект** , чтобы открыть диалоговое окно **Создание нового проекта** .
2. В верхней части диалогового окна для параметра **Язык** установите значение **C++** , для параметра **Платформа**— значение **Windows,** а для параметра **Тип проекта** — значение **Библиотека.**
3. В отфильтрованном списке типов проектов выберите **Библиотека динамической компоновки (DLL)**, а затем нажмите кнопку **Далее.**
4. На странице **Настройка нового проекта** введите маслибрари в поле **имя проекта** , чтобы указать имя проекта. Оставьте значения **расположения** и **имени решения** по умолчанию. Задайте **решение**для**создания нового решения**. Снимите флажок **размещать решение и проект в том же каталоге,** если он установлен.
5. Нажмите кнопку **Создать,** чтобы создать проект.

**8.** Что такое «позиционно-независимый код» (position independent code, PIC)?

*Ответ:*

**Позиционно-независимый код** (англ. position-independent code) — программа, которая может быть размещена в любой области памяти, так как все ссылки на ячейки памяти в ней относительные (например, относительно счётчика команд). Такую программу можно переместить в другую область памяти в любой момент, в отличие от перемещаемой программы, которая хотя и может быть загружена в любую область памяти, но после загрузки должна оставаться на том же месте.

**9**. Какие способы можно использовать для того, чтобы указать местоположение подключаемых динамических библиотек?

*Ответ:*

По имени библиотеке

По полному пути к библиотеке

**10**. Если в целевом каталоге присутствует два типа одной и той же библиотеки, какая из них всегда будет подключаться по умолчанию?

*Ответ:*

По умолчанию будет подключаться первая(новая) библиотека.

**Листинг:**

* 1. **Controller.cpp**

#include <iostream>

#include <string>

#include "model.h"

#include "view.h"

using namespace std;

int main(int argc, char \*argv[])

{

if(argc < 2)

cout << "Invalid args" << endl;

int amount = stoi(argv[1]);

string\* pairs = generatePairs(amount);

printStrings(pairs, amount);

delete[] pairs;

return 0;

}

* 1. **Model.cpp**

#include <string>

#include <algorithm>

#include <bits/stdc++.h>

#include <iostream>

#include "model.h"

using namespace std;

string\* generatePairs(int amount)

{

const int size = 36;

string\* allPairs = new string[size];

int counter = 0;

for(int first = 2; first < 10; first++)

{

for(int second = first; second < 10; second++)

{

allPairs[counter] = to\_string(first) + "\*" + to\_string(second);

counter++;

}

}

random\_shuffle(allPairs, allPairs + size);

string\* pairs = new string[amount];

for(int i = 0; i < amount; i++)

{

pairs[i] = allPairs[i];

}

delete[] allPairs;

return pairs;

}

* 1. **Model.h**

#include <string>

using namespace std;

#pragma once

string\* generatePairs(int amount);

* 1. **View.cpp**

#include <iostream>

#include "view.h"

using namespace std;

void printStrings(string\* arr, int size)

{

for(int i = 0; i < size; i++)

{

cout << arr[i] << " ";

}

cout << endl;

}

* 1. **View.h**

#include <string>

#pragma once

void printStrings(std::string\* arr, int size);

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №7

По дисциплине **«Операционные системы»**

Тема: **«IO with C»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Изучить фундаментальные основы и концепции файлового ввода-вывода в Linux, основные функции ввода-вывода стандартной библиотеки языка программирования С, а также закрепить навыки применения данной библиотеки при написании системных программ.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Предусмотреть ещё один способ инициализации данных и вывода результата работы программы – с помощью файлового ввода-вывода. Программа должна дополнительно выводить на консоль исходные данные и конечный результат.

3) Функции бизнес логики и ввода-вывода должны быть сгруппированы и вынесены в соответствующие библиотеки. Рекомендуется использовать динамические библиотеки.

4) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

5) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

6) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

7) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

8) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

9) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

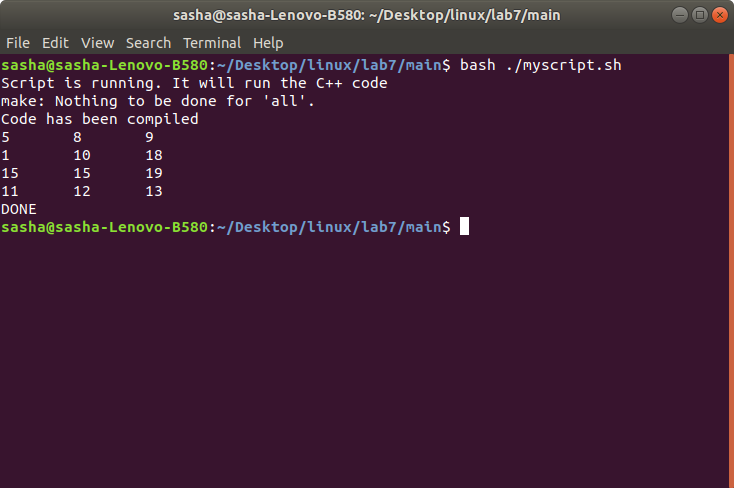
10) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

11) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Необходимо расширить функционал любой из программ, созданных в предыдущих лабораторных работах:  возможность задавать размеры матрицы и её элементы с помощью указанного файла;  возможность выводить результат в файл. Входной файл может содержать совокупность строк. Каждая строка файла содержит элементы матрицы. Необходимо предусмотреть как ручное создание файла и его заполнение, так и генерацию файла в автоматическом режиме.

*Результаты выполнения*

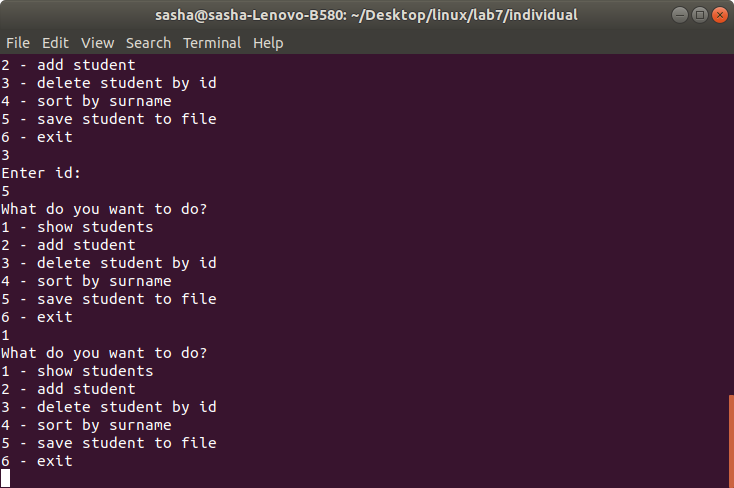


**Индивидуальное задание:**

В программе необходимо определить структуру сущностей согласно варианту задания (разрешается придумать свою структуру-сущность, предварительно согласовав её с преподавателем). Затем необходимо объявить динамическую структуру данных для хранения переменных данного структурного типа. Далее программа должна предоставлять возможность манипулировать данными элементами (к примеру, осуществить инициализацию контейнера, добавление или удаление элементов, отсортировать контейнер элементов по заданному критерию, осуществить необходимы поиск, подсчёт чего-то, сохранить данные контейнера и т.д.). Для сохранения состояния контейнера и его восстановления использовать файловый ввод-вывод.

1. Student: id, Фамилия, Имя, Отчество, Дата рождения, Адрес, Телефон, Факультет, Курс, Группа.

*Результаты выполнения*



**Контрольные вопросы:**

**1**. Устройство и структура жёсткого диска? Что такое главная загрузочная запись (Master Boot Record, MBR)? Что такое загрузочный сектор (Boot Record, BR)?

*Ответ:*

Любой жёсткий диск можно представить как огромный «чистый лист», на который можно записывать данные и откуда потом их можно считать. Чтобы ориентироваться на диске, всё его пространство разбивают на небольшие «клеточки» — **сектора.** Сектор — это минимальная единица хранения данных на диске, обычно его размер составляет 512 байт. Все сектора на диске нумеруются: каждый из n секторов получает номер от 0 до n–1. Благодаря этому любая информация, записанная на диск, получает точный адрес — номера соответствующих секторов. Так что диск ещё можно представить как очень длинную строчку (ленточку) из секторов. Можете посчитать, сколько секторов на вашем диске размером в N гигабайт.

Представлять жёсткий диск как единый «лист» не всегда бывает удобно: иногда полезно «разрезать» его на несколько независимых листов, на каждом из которых можно писать и стирать что угодно, не опасаясь повредить написанное на других листах. Логичнее всего записывать раздельно данные большей и меньшей важности или просто относящиеся к разным вещам.

Конечно, над жёстким диском следует производить не физическое, а логическое разрезание, для этого вводится понятие раздел (partition). Вся последовательность (очень длинная ленточка) секторов разрезается на несколько частей, каждая часть становится отдельным разделом. Фактически, нам не придётся ничего разрезать (да и вряд ли бы это удалось), достаточно объявить, после каких секторов на диске находятся границы разделов.

Главная загрузочная запись (англ. master boot record, MBR) — код и данные, необходимые для последующей загрузки операционной системы и расположенные в первых физических секторах (чаще всего в самом первом) на жёстком диске или другом устройстве хранения информации.

Загрузочный сектор — это особый сектор на жёстком диске, дискете или другом дисковом устройстве хранения информации. (Для дискеты это первый физический сектор, для жесткого диска — первый физический сектор для каждого раздела) В процессе загрузки компьютера с дискеты он загружается в память программой POST (в компьютерах архитектуры IBM PC обычно с адреса 0000:7c00), ему передается управление командой long jump.

**2.** Последовательность действий от включения компьютера до старта ОС?

*Ответ:*

После включения [персонального компьютера (ПК)](https://ru.wikipedia.org/wiki/%D0%9F%D0%B5%D1%80%D1%81%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80) на базе [процессора](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D1%81%D1%81%D0%BE%D1%80)семейства [Intel 80x86](https://ru.wikipedia.org/wiki/X86), его процессор начинает работу в [реальном режиме](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D1%80%D0%B5%D0%B6%D0%B8%D0%BC)адресации с сегментной организацией и выполнение инструкций процессора с адреса FFFF:0, инициализированного в паре регистров CS:IP (Code Segment : Instruction Pointer) после снятия сигнала RESET. В конце доступного процессору адресного пространства [оперативной памяти](https://ru.wikipedia.org/wiki/%D0%9E%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%B0%D1%8F_%D0%BF%D0%B0%D0%BC%D1%8F%D1%82%D1%8C) из [ПЗУ](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%BD%D0%BE%D0%B5_%D0%B7%D0%B0%D0%BF%D0%BE%D0%BC%D0%B8%D0%BD%D0%B0%D1%8E%D1%89%D0%B5%D0%B5_%D1%83%D1%81%D1%82%D1%80%D0%BE%D0%B9%D1%81%D1%82%D0%B2%D0%BE) [материнской платы](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D1%80%D0%B8%D0%BD%D1%81%D0%BA%D0%B0%D1%8F_%D0%BF%D0%BB%D0%B0%D1%82%D0%B0) загружается загрузчик базовой системы ввода-вывода ([BIOS](https://ru.wikipedia.org/wiki/BIOS)), на который передается выполнение по инструкции безусловного перехода, отображенной по адресу FFFF:0.

Программа инициализации [BIOS](https://ru.wikipedia.org/wiki/BIOS) с помощью программы [POST](https://ru.wikipedia.org/wiki/POST_(%D0%B0%D0%BF%D0%BF%D0%B0%D1%80%D0%B0%D1%82%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5)) проверяет, что устройства компьютера работают корректно и инициализирует их.

Затем BIOS опрашивает по порядку устройства из заранее сформированного (в консольной утилите CMOS SETUP) списка загрузочных устройств, пока не найдет загрузочное устройство. Если такое устройство не будет найдено, BIOS выведет на [консоль](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BD%D1%81%D0%BE%D0%BB%D1%8C) ПК сообщение об ошибке и процесс загрузки будет остановлен. При обнаружении BIOS’ом загрузочного устройства, с последнего будет произведена попытка чтения данных его нулевого сектора (512 байт, в которых должна находиться [загрузочная запись](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C)). После успешного завершения чтения данных нулевого сектора загрузочного устройства в оперативную память по начальному адресу 0000:7С00, BIOS передает на этот адрес дальнейшее исполнение инструкций процессора.

Нулевой сектор загрузочного [жесткого диска](https://ru.wikipedia.org/wiki/%D0%96%D0%B5%D1%81%D1%82%D0%BA%D0%B8%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA) содержит так называемую [«Главную загрузочную запись» (MBR — Master Boot Record)](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BF%D0%B8%D1%81%D1%8C), содержащую как данные первых четырёх записей [таблицы разделов](https://ru.wikipedia.org/wiki/%D0%A2%D0%B0%D0%B1%D0%BB%D0%B8%D1%86%D0%B0_%D1%80%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%BE%D0%B2), так и инструкции процессора Intel 80x86, обеспечивающие поиск по этим записям активного раздела, с которого, посредством одной из функций базовой системы ввода-вывода — [INT 13h](https://ru.wikipedia.org/w/index.php?title=INT_13h&action=edit&redlink=1) ([англ.](https://ru.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8%D0%B9_%D1%8F%D0%B7%D1%8B%D0%BA) [*Int 13h*](https://en.wikipedia.org/wiki/Int_13h))[[1]](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%87%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BA%D0%B0_%D0%BA%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%B0#cite_note-1), реализованной в виде обработчика [прерывания](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) процессора, будет считана [загрузочная запись операционной системы](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D1%87%D0%B8%D0%BA_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B). Этот загрузочный сектор, как правило, зависит от операционной системы и должен произвести загрузку в оперативную память [ядра операционной системы](https://ru.wikipedia.org/wiki/%D0%AF%D0%B4%D1%80%D0%BE_%D0%BE%D0%BF%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D0%BE%D0%B9_%D1%81%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D1%8B) с последующей передачей на него исполнения инструкций процессора. Если активного раздела не существует, или [загрузочный сектор](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D0%BE%D1%87%D0%BD%D1%8B%D0%B9_%D1%81%D0%B5%D0%BA%D1%82%D0%BE%D1%80) активного раздела некорректен, MBR может загрузить резервный начальный загрузчик и передать управление ему. Резервный начальный загрузчик должен выбрать раздел (зачастую с помощью пользователя), загрузить его загрузочный сектор и передать ему управление, который, в свою очередь, передаёт управление непосредственно [загрузчику ОС](https://ru.wikipedia.org/wiki/%D0%97%D0%B0%D0%B3%D1%80%D1%83%D0%B7%D1%87%D0%B8%D0%BA_%D0%9E%D0%A1) (дальнейшие этапы загрузки зависят от типа/версии установленной [ОС](https://ru.wikipedia.org/wiki/%D0%9E%D0%A1)).

**3.** Что такое файл (File)?

*Ответ:*

Файл (англ. file) — именованная область данных на носителе информации.

**4.** Что такое файловая система (File System) и зачем она нужна?

*Ответ:*

Файловая система - это часть ОС, которая отвечает за размещение, удаление, хранение информации на носителях (жестких дисках, дискетах, оптических носителях, flash-картах и т.д.), предоставление пользователям и приложениям доступа к этой информации, обеспечение ее безопасного совместного использования, а также защиту и восстановление при программных и аппаратных сбоях.

**5.** Основные цели файловой системы?

*Ответ:*

Целииспользования файла :

- *Долговременное и надежное хранение информации*. Долговременность достигается за счет использования запоминающих устройств, не зависящих от питания, а высокая надежность определяется средствами защиты доступа к файлам и общей организацией программного кода ОС, при которой сбои аппаратуры чаще всего не разрушают информацию, хранящуюся в файлах.

- *Совместное использование информации*. Файлы обеспечивают естественный и легкий способ разделения информации между приложениями и пользователями за счет наличия понятного человеку символьного имени и постоянства хранимой информации и расположения файла. Пользователь должен иметь удобные средства работы с файлами, включая каталоги-справочники, объеди­няющие файлы в группы, средства поиска файлов по признакам, набор команд для создания, модификации и удаления файлов. Файл может быть создан одним пользователем, а затем использоваться совсем другим пользователем, при этом создатель файла или администратор могут определить права доступа к нему других пользователей.

**6**. Классификация файловых систем?

*Ответ:*

Существуют файловые системы:

* для устройств произвольного доступа - жестких дисков;
* для устройств с последовательным доступом - магнитных лент;
* для оптических носителей - CD и DVD;
* сетевые ФС;
* виртуальные ФС.

**7.** Основная и главная философия системного программирования в Linux?

Системноепрограммирование — это практика написания системного ПО, низкоуровневый код которого взаимодействует непосредственно с ядром и основными системнымибиблиотеками. Иными словами, речь далее пойдет в основном о системных вызовах Linux и низкоуровневых функциях, в частности тех, которые определены в биб­ лиотеке C. Все доступно для редактирования.

**8**. Что такое файловый дескриптор (File Descriptor) и зачем он нужен?

*Ответ:*

Ко всем потокам ввода-вывода можно получить доступ через так называемые файловые дескрипторы. Файловый дескриптор — это неотрицательное целое число.

**9.** Обычные файлы (Regular Files) в Linux и их основные метаданные: позиция в файле (File Position), или смещение (File Offset), размер файла (File Length), имя файла (Filename), структура информационный узел (Information Node, inode)?

*Ответ:*

Это файлы, с которыми мы привыкли работать каждый день, они могут содержать текст, исполняемые инструкции для программ, изображения или другую информацию. Это самый распространенный тип файлов, которые вы можете найти в системе Linux. Рассмотрим небольшой список относящихся сюда файлов:

* Текстовые файлы
* Исполняемые файлы
* Файлы изображений
* Файлы архивов
* Файлы библиотек программ
* И другие подобные типы

**10**. Специальные файлы (Special Files) в Linux?

*Ответ:*

Специальные файлы намного интереснее, они предназначены для обмена информации с ядром, работы с устройствами или общения между программами. Такие файлы могут тоже быть нескольких типов, в зависимости от назначения.

**Блочные файлы -**это файлы устройств, которые обеспечивают буферизованный доступ к аппаратным компонентам. При записи данных на жесткий диск или на флешку нет смысла записывать данные сразу же после их поступления. Так мы будем только понапрасну расходовать ресурс устройства и энергию. Можно подождать пока наберется достаточное количество данных а потом записать их за один раз. Эти данные и собираются в буфере. С помощью таких файлов, файловая система и другие утилиты могут обращаться к драйверам аппаратных устройств. Такие файлы могут передать большой блок данных за небольшой один раз.

**11**. Что такое каталог (Directory)? Зачем нужен корневой каталог (Root Directory)?

*Ответ:*

**Каталог** (англ. **directory** справочник, указатель) — объект в файловой системе, упрощающий организацию файлов.

Root directory (корневой каталог) - основной каталог на жестком или гибком диске. Имеет фиксированный размер и местоположение для конкретного дискового тома и не может, подобно подкаталогу, динамически изменять размер. Корневой каталог, обозначаемый как “/”, является верхним элементом иерархии файловой системы. От него ответвляются все остальные каталоги, формируя “дерево файлов”. Корневой каталог (папка) - как правило - public.html корневой каталог для размешения файлов на сервере WebSphere Portal Content Publishing, применяемый для импорта и экспорта структурированной информации, а также информации на основе файлов

**12**. Что такое ссылка (Link)? Какие бывают типы ссылок?

*Ответ:*

Символические и жесткие ссылки - это особенность файловой системы Linux, которая позволяет размещать один и тот же файл в нескольких директориях.

Символические ссылки более всего похожи на обычные ярлыки. Они содержат адрес нужного файла в вашей файловой системе. Когда вы пытаетесь открыть такую ссылку, то открывается целевой файл или папка. Главное ее отличие от жестких ссылок в том, что при удалении целевого файла ссылка останется, но она будет указывать в никуда, поскольку файла на самом деле больше нет.

Этот тип ссылок реализован на более низком уровне файловой системы. Файл размещен только в определенном месте жесткого диска. Но на это место могут ссылаться несколько ссылок из файловой системы. Каждая из ссылок - это отдельный файл, но ведут они к одному участку жесткого диска. Файл можно перемещать между каталогами, и все ссылки останутся рабочими, поскольку для них неважно имя. Рассмотрим особенности:

**13**. Понятие буфера, буферизованный и расширенный файловый ввод-вывод?

*Ответ:*

Буфер - это некое промежуточное хранилище, предоставляемое для хранения временных данных либо самой программой, либо операционной системой.

Буферизированный ввод-вывод – Ввод/вывод для дисков в операционной системе MS-DOS буферизован. Это означает, что данные не сразу записываются на диск, а накапливаются в специальном массиве (буфере). По мере заполнения буфер сбрасывается на диск. При чтении информация заполняет весь входной буфер, независимо от количества байтов, которые программа читает из файла. В дальнейшем, если программе потребуются данные, которые уже были считаны с диска и записаны во входной буфер, она получит данные непосредственно из этого буфера. Обращения к диску при этом не будет.

Расширенный ввод-вывод – Расширенный ввод/вывод иногда называют дежурным вводом/выводом (alertable I/O). О том, как использовать расширенные функции, рассказывается в последующих разделах.

**14.** На какие основные части условно делиться модель файлового ввода-вывода?

*Ответ:*

При работе с файлом можно выделить следующие этапы:

* создать объект класса fstream (возможно, ofstream или ifstream);
* связать объект класса fstream с файлом, который будет использоваться для операций ввода-вывода;
* осуществить операции ввода-вывода в файл;
* закрыть файл.

**15.** Абстракция файла в языке программирования С? Стандартные файловые указатели?

*Ответ:*

Ввод или вывод информации в файлы обеспечивается с помощью так называемого указателя на файл, который является указателем на файловую структуру в памяти. При записи информации в файл или при чтении из файла программа получает необходимую информацию из структуры. Указатель на файл определяется следующим образом:

FILE \*file\_pointer;

Имя структуры FILE говорит программе о том, что определяемая переменная является указателем именно на файловую структуру. Звездочка предписывает создать указатель с соответствующим именем переменной.

Если вы собираетесь использовать одновременно несколько файлов, вам нужны указатели для каждого из них. Например, если вы пишете программу, в которой содержимое одного файла копируется в другой, вам необходимы два указателя на файлы. Два указателя требуются и в том случае, если вы хотите прочитать информацию с диска и распечатать ее на принтере:

FILE \*infile, \*outfile;

Абстра́кция в [объектно-ориентированном программировании](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82%D0%BD%D0%BE-%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%BD%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5) — это использование только тех характеристик [объекта](https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%8A%D0%B5%D0%BA%D1%82_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5)), которые с достаточной точностью представляют его в данной системе. Основная идея состоит в том, чтобы представить объект минимальным набором полей и методов и при этом с достаточной точностью для решаемой задачи.

**16.** Функции стандартной библиотеки С (stdio) для открытия и закрытия файлов, а также чтения и записи данных: fopen (fdopen), fread, fwrite, fclose (fcloseall), fflush, fseek, fgetc (getc or getchar), fputc (putc и putchar), fgets, gets?

*Ответ:*

Функция fopen открывает файл, имя которого указано в параметре fname и связывает его с потоком, который может быть идентифицирован для выполнения различных операций с файлом.

Функция fread() возвращает количество прочитанных элементов

А функция fwrite()возвращает количество записанных элементов.

Функция fclose закрывает и разъединяет файл filestream, связанный с потоком.

fflush – проталкивание накопленных данных в потока данных.

**fseek**– установка позиции в потоке данных

fgetc – чтение одного байта из указанного потока данных.

fputc – вывод одного байта в указанный поток данных.

fgets – чтение строки из указанного потока данных.

gets – чтение строки из стандартного потока ввода.

**17.** Форматированный ввод-вывод: printf (fprintf и sprintf), scanf (fscanf и sscanf)?

*Ответ:*

Функция printf() возвращает число выведенных символов или отрицательное значение в случае ошибки.

Функция scanf() является процедурой ввода общего назначения, считывающей данные из пото­ка stdin.

**18.** Механизм произвольного доступа к данным с использованием языка С?

*Ответ:*

В информатике под **произвольным** **доступом** понимают возможность обратиться к любому элементу последовательности за равные промежутки времени, не зависящие от размеров последовательности (в отличие от последовательного **доступа**, когда чем дальше расположен элемент, тем больше требуется времени для **доступа**).

**19.** Функция определения конца файла?

*Ответ:*

[Функция feof проверяет наличие признака конца файла](http://all-ht.ru/inf/prog/c/func/feof.html" \t "_blank)

**20**. Функция стандартной библиотеки языка С для обработки ошибок файлового ввода-вывода?

*Ответ:*

Редко cat

Чаще stderr – exit.

**Листинг:**

* 1. **controller.cpp**

#include <stdio.h>

#include <iostream>

#include <string>

#include "model.h"

#include "view.h"

using namespace std;

int readMatrix(int rows, int cols, int\*\* a, const char\* filename)

{

FILE \*pf = fopen (filename, "r");

if (pf == NULL)

return 0;

for(int i = 0; i < rows; ++i)

for(int j = 0; j < cols; ++j)

fscanf(pf, "%d", a[i] + j);

fclose (pf);

return 1;

}

int main()

{

int m = 4;

int n = 3;

int\*\* matrix = new int\*[m];

for(int i = 0; i < m; i++)

{

matrix[i] = new int[n];

}

readMatrix(m, n, matrix, "matrix.txt");

printMatrix(matrix, m, n);

int result = getRowNumberWithLongestSeries(matrix, m, n);

printMatrixResult(result);

for(int i = 0; i < m; i++)

{

delete[] matrix[i];

}

delete[] matrix;;

return 0;

}

* 1. **model.cpp**

#include <cstdlib>

#include <iostream>

#include <string>

#include "model.h"

using namespace std;

int getLongestSeries(int\* row, int n)

{

int longestSeries = 1;

int tempSeries = 1;

for(int i = 1; i < n; i++)

{

if(row[i] == row[i - 1])

tempSeries++;

else if(tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

if(i == n - 1 && tempSeries > longestSeries)

{

longestSeries = tempSeries;

tempSeries = 1;

}

}

return longestSeries;

}

int getRowNumberWithLongestSeries(int\*\* matrix, int m, int n)

{

int rowNumber = 1;

int greatestSeries = 1;

for(int i = 0; i < m; i++)

{

int series = getLongestSeries(matrix[i], n);

if(series > greatestSeries)

{

greatestSeries = series;

rowNumber = i + 1;

}

}

return rowNumber;

}

* 1. **model.h**

#pragma once

int getRowNumberWithLongestSeries(int\*\* matrix, int m, int n);

* 1. **view.cpp**

#include <iostream>

#include "view.h"

using namespace std;

void printMatrix(int\*\* matrix, int m, int n)

{

for(int i = 0; i < m; i++)

{

for(int j = 0; j < n; j++)

{

cout << matrix[i][j] << "\t";

}

cout << endl;

}

}

void printMatrixResult(int result)

{

FILE \*pf = fopen ("matrix-result.txt", "w");

if (pf == NULL)

return ;

fprintf(pf, "%s", "Row number with the longest series is ");

fprintf(pf, "%d", result);

fclose (pf);

}

* 1. **view.h**

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

* 1. **controller.java**

package by.bntu.linux.lab7;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

import java.util.stream.Collectors;

public class Controller {

public static void main(String[] args) {

Input input = new Input();

Printer printer = new Printer();

List<Student> students = input.readFromFile();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("What do you want to do?\n" +

"1 - show students\n" +

"2 - add student\n" +

"3 - delete student by id\n" +

"4 - sort by surname\n" +

"5 - save student to file\n" +

"6 - exit");

choice = scanner.nextInt();

switch (choice) {

case 1:

students.forEach(System.out::println);

break;

case 2:

students.add(input.inputStudent());

break;

case 3:

System.out.println("Enter id: ");

int id = scanner.nextInt();

students = students

.stream()

.filter(s -> s.getId() != id)

.collect(Collectors.toList());

break;

case 4:

students.sort(Comparator.comparing(Student::getSurname));

break;

case 5:

printer.saveToFile(students);

break;

case 6:

break;

default:

System.out.println("Invalid number!");

}

} while (choice != 6);

}

}

* 1. **student.java**

package by.bntu.linux.lab7;

import java.util.Objects;

public class Student {

private int id;

private String surname;

private String name;

private String patronymic;

private String birthDate;

private String address;

private String phone;

private String faculty;

private int course;

private int group;

public Student() {}

public Student(int id, String surname, String name, String patronymic, String birthDate, String address,

String phone, String faculty, int course, int group) {

this.id = id;

this.surname = surname;

this.name = name;

this.patronymic = patronymic;

this.birthDate = birthDate;

this.address = address;

this.phone = phone;

this.faculty = faculty;

this.course = course;

this.group = group;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getBirthDate() {

return birthDate;

}

public void setBirthDate(String birthDate) {

this.birthDate = birthDate;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getFaculty() {

return faculty;

}

public void setFaculty(String faculty) {

this.faculty = faculty;

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public int getGroup() {

return group;

}

public void setGroup(int group) {

this.group = group;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return id == student.id &&

course == student.course &&

group == student.group &&

Objects.equals(surname, student.surname) &&

Objects.equals(name, student.name) &&

Objects.equals(patronymic, student.patronymic) &&

Objects.equals(birthDate, student.birthDate) &&

Objects.equals(address, student.address) &&

Objects.equals(phone, student.phone) &&

Objects.equals(faculty, student.faculty);

}

@Override

public int hashCode() {

return Objects.hash(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

@Override

public String toString() {

return "Student{" +

"id=" + id +

", surname='" + surname + '\'' +

", name='" + name + '\'' +

", patronymic='" + patronymic + '\'' +

", birthDate='" + birthDate + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", faculty='" + faculty + '\'' +

", course=" + course +

", group=" + group +

'}';

}

}

* 1. **printer.java**

package by.bntu.linux.lab7;

import java.io.\*;

import java.util.List;

public class Printer {

public void saveToFile(List<Student> students) {

try(BufferedWriter writer = new BufferedWriter(new FileWriter("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

for(Student student : students) {

String line = student.getId() + " "

+ student.getSurname() + " " + student.getName() + " " + student.getPatronymic() + " "

+ student.getBirthDate() + " " + student.getAddress() + " " + student.getPhone() + " "

+ student.getFaculty() + " " + student.getCourse() + " " + student.getGroup();

writer.write(line);

}

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

}

* 1. **input.java**

package by.bntu.linux.lab7;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Input {

public List<Student> readFromFile() {

try(BufferedReader reader = new BufferedReader(new FileReader("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

String line;

List<Student> students = new ArrayList<>();

while ((line = reader.readLine()) != null) {

String[] fields = line.split(" ");

Student student =

new Student(Integer.parseInt(fields[0]), fields[1], fields[2], fields[3], fields[4], fields[5],

fields[6], fields[7], Integer.parseInt(fields[8]), Integer.parseInt(fields[9]));

students.add(student);

}

return students;

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

public Student inputStudent() {

Scanner scanner = new Scanner(System.in);

System.out.print("Surname: ");

String surname = scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Patronymic: ");

String patronymic = scanner.nextLine();

System.out.print("Birth date: ");

String birthDate = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

System.out.print("Phone: ");

String phone = scanner.nextLine();

System.out.print("Faculty: ");

String faculty = scanner.nextLine();

System.out.print("Course: ");

int course = scanner.nextInt();

System.out.print("Group: ");

System.out.print("Id: ");

int id = scanner.nextInt();

int group = scanner.nextInt();

return new Student(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

}

* 1. **view.java**

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №8

По дисциплине **«Операционные системы»**

Тема: **«IO with C++»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Повторить фундаментальные основы и концепции файлового ввода-вывода в Linux, а также изучить основные функции ввода-вывода стандартной библиотеки ввода-вывода языка программирования С++ и закрепить навыки применения данной библиотеки при написании системных программ.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Предусмотреть ещё один способ инициализации данных и вывода результата работы программы – с помощью файлового ввода-вывода. Программа должна дополнительно выводить на консоль исходные данные и конечный результат.

3) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

4) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

5) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

6) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

7) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

8) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

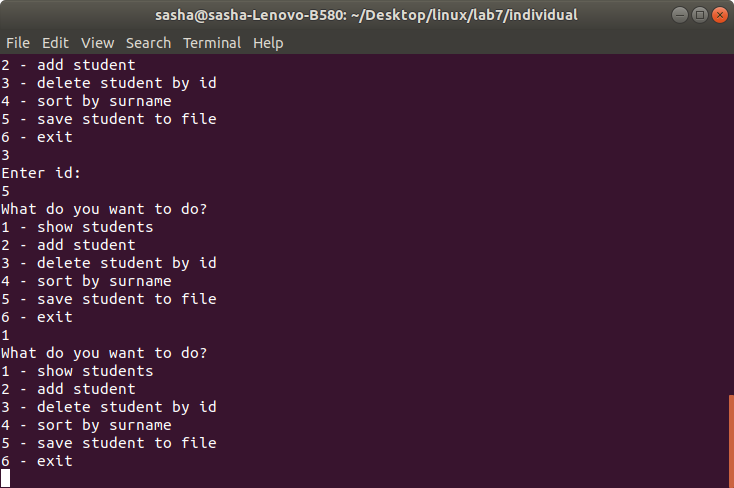
9) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

10) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Произвести рефакторинг индивидуального задания предыдущей лабораторной работы: пользовательскую библиотеку функций ввода-вывода переписать с использованием стандартной библиотеки С++.

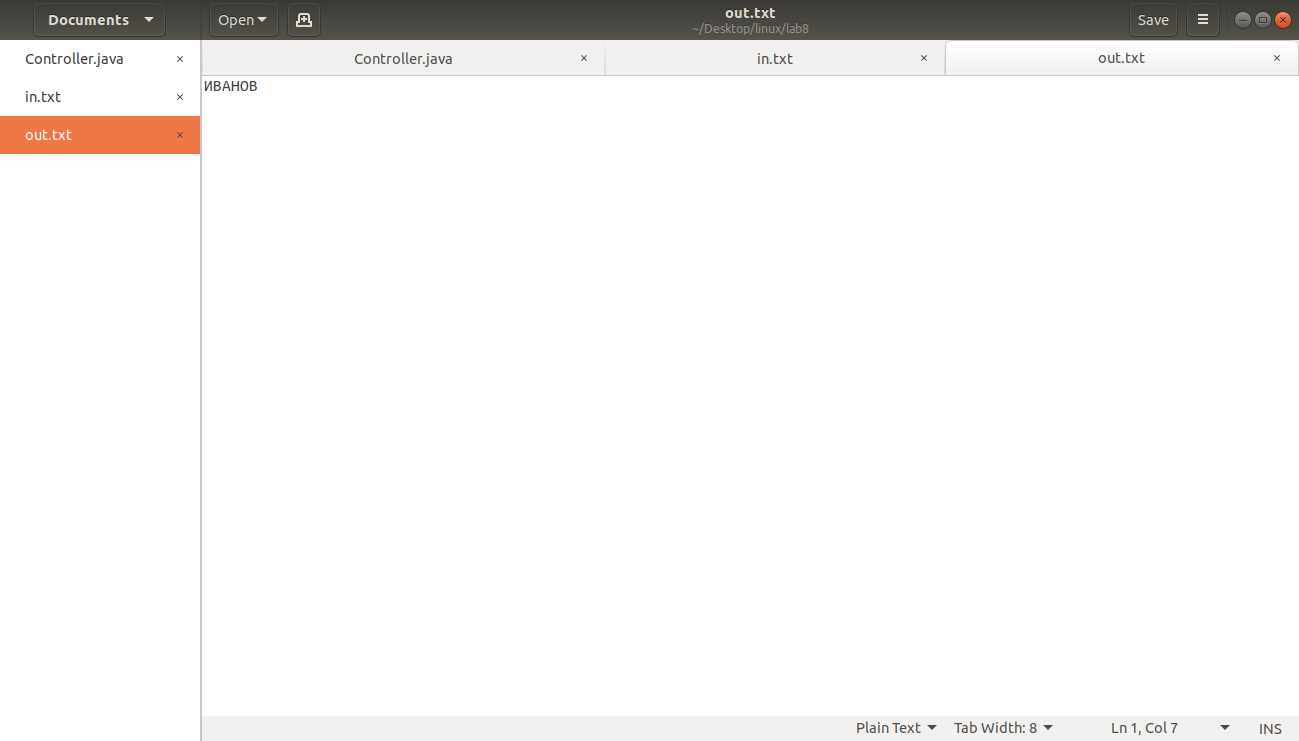
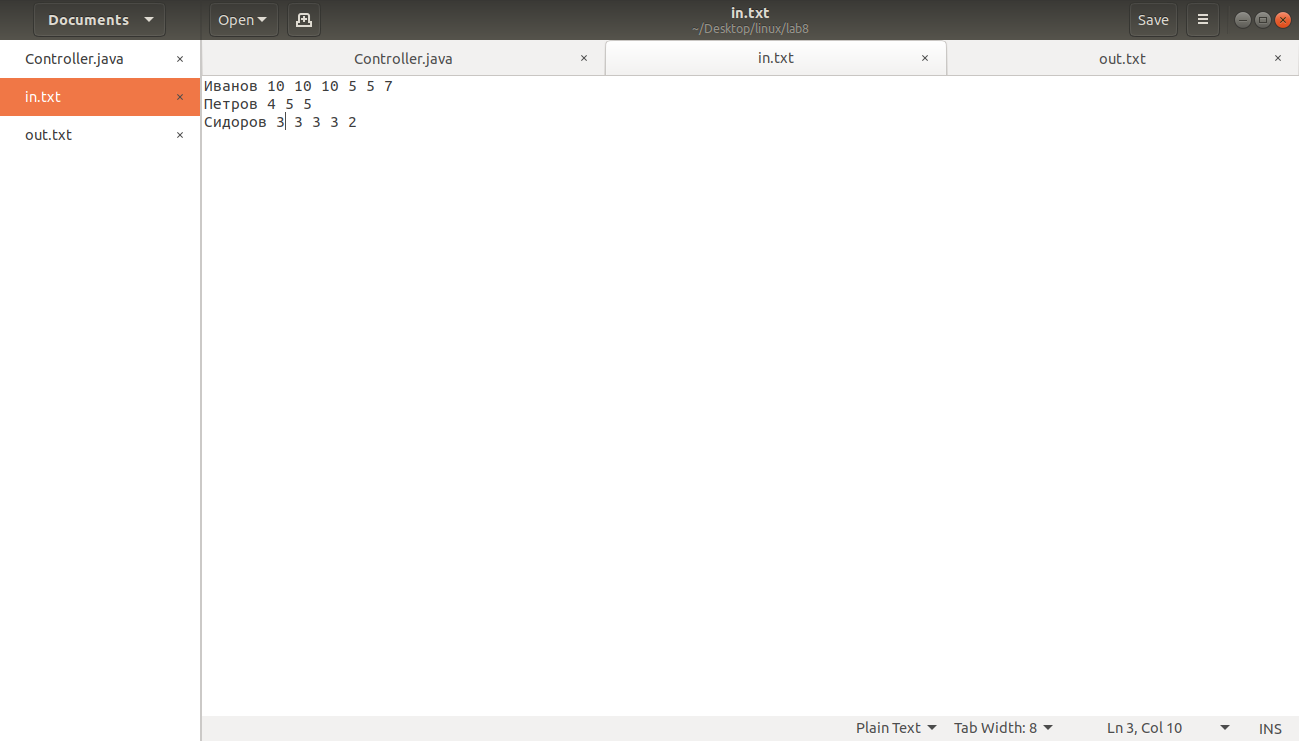
*Результаты выполнения*



**Индивидуальное задание:**

4) В файле, содержащем фамилии студентов и их оценки, записать прописными буквами фамилии тех студентов, которые имеют средний балл более “7”

*Результаты выполнения*



**Контрольные вопросы:**

**1**. Концепция файлового ввода-вывода в С++?

*Ответ:*

Есть три основных класса файлового ввода/вывода в C++:

   ifstream (является [дочерним](https://ravesli.com/urok-154-bazovoe-nasledovanie-v-c/) классу [istream](https://ravesli.com/urok-208-funktsional-klassa-istream/));

   ofstream (является дочерним классу [ostream](https://ravesli.com/urok-209-funktsional-klassov-ostream-i-ios-formatirovanie-vuvoda/));

   fstream (является дочерним классу iostream).

С помощью этих классов можно выполнять однонаправленный файловый ввод, однонаправленный файловый вывод и двунаправленный файловый ввод/вывод. Для их использования нужно всего лишь подключить [заголовочный файл](https://ravesli.com/urok-21-zagolovochnye-fajly/) fstream.

**2**. На какие основные части условно делиться модель файлового ввода-вывода в С++?

*Ответ:*

   ifstream (является [дочерним](https://ravesli.com/urok-154-bazovoe-nasledovanie-v-c/) классу [istream](https://ravesli.com/urok-208-funktsional-klassa-istream/));

   ofstream (является дочерним классу [ostream](https://ravesli.com/urok-209-funktsional-klassov-ostream-i-ios-formatirovanie-vuvoda/));

   fstream (является дочерним классу iostream).

**3**. Абстракция файла в языке программирования С++?

*Ответ:*

Абстракция данных означает предоставление только важной информации внешнему миру и скрытие их фоновых данных, т. е. Представление необходимой информации в программе без предоставления деталей.

Абстракция данных - это метод программирования (и проектирования), который основан на разделении интерфейса и реализации.

**4**. Стандартные файловые потоки?

*Ответ:*

Класс istream используется для работы с входными потоками. Оператор извлечения >> используется для извлечения значений из потока. Это имеет смысл: когда пользователь нажимает на клавишу клавиатуры, код этой клавиши помещается во входной поток. Затем программа извлекает это значение из потока и использует его.

Класс ostream используется для работы с выходными потоками. Оператор вставки << используется для помещения значений в поток. Это также имеет смысл: вы вставляете свои значения в поток, а затем потребитель данных (например, монитор) использует их.

Класс iostream может обрабатывать как ввод, так и вывод данных, что позволяет ему осуществлять двунаправленный ввод/вывод.

**5.** Функции стандартной библиотеки С++ для открытия и закрытия файлов?

*Ответ:*

.open

.close

**6**. Функции и перегруженные операторы стандартной библиотеки С++ для чтения и записи данных?

*Ответ:*

Read

Write

**7**. Форматированный ввод-вывод в С++?

*Ответ:*

Для управления вводом-выводом в C++  
используются :

* флаги форматированного ввода-вывода;
* манипуляторы форматирования.

Флаги позволяют установить параметры ввода-вывода, которые будут действовать во всех последующих операторах ввода-вывода до тех пор, пока не будут отменены. Манипуляторы вставляются в операторы **cin**(cout) и устанавливают параметры текущего оператора ввода-вывода.

**8**. Механизм произвольного доступа к данным с использованием языка С++?

*Ответ:*

В информатике под **произвольным** **доступом** понимают возможность обратиться к любому элементу последовательности за равные промежутки времени, не зависящие от размеров последовательности (в отличие от последовательного **доступа**, когда чем дальше расположен элемент, тем больше требуется времени для **доступа**).

**9.** Функция определения конца файла в С++?

*Ответ:*

eof

**10.** Функция стандартной библиотеки языка С++ для обработки ошибок файлового ввода-вывода?

*Ответ:*

Чаще stderr – exit.

**Листинг:**

* 1. **controller.java**

package by.bntu.linux.lab7;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

import java.util.stream.Collectors;

public class Controller {

public static void main(String[] args) {

Input input = new Input();

Printer printer = new Printer();

List<Student> students = input.readFromFile();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("What do you want to do?\n" +

"1 - show students\n" +

"2 - add student\n" +

"3 - delete student by id\n" +

"4 - sort by surname\n" +

"5 - save student to file\n" +

"6 - exit");

choice = scanner.nextInt();

switch (choice) {

case 1:

students.forEach(System.out::println);

break;

case 2:

students.add(input.inputStudent());

break;

case 3:

System.out.println("Enter id: ");

int id = scanner.nextInt();

students = students

.stream()

.filter(s -> s.getId() != id)

.collect(Collectors.toList());

break;

case 4:

students.sort(Comparator.comparing(Student::getSurname));

break;

case 5:

printer.saveToFile(students);

break;

case 6:

break;

default:

System.out.println("Invalid number!");

}

} while (choice != 6);

}

}

* 1. **student.java**

package by.bntu.linux.lab7;

import java.util.Objects;

public class Student {

private int id;

private String surname;

private String name;

private String patronymic;

private String birthDate;

private String address;

private String phone;

private String faculty;

private int course;

private int group;

public Student() {}

public Student(int id, String surname, String name, String patronymic, String birthDate, String address,

String phone, String faculty, int course, int group) {

this.id = id;

this.surname = surname;

this.name = name;

this.patronymic = patronymic;

this.birthDate = birthDate;

this.address = address;

this.phone = phone;

this.faculty = faculty;

this.course = course;

this.group = group;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getBirthDate() {

return birthDate;

}

public void setBirthDate(String birthDate) {

this.birthDate = birthDate;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getFaculty() {

return faculty;

}

public void setFaculty(String faculty) {

this.faculty = faculty;

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public int getGroup() {

return group;

}

public void setGroup(int group) {

this.group = group;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return id == student.id &&

course == student.course &&

group == student.group &&

Objects.equals(surname, student.surname) &&

Objects.equals(name, student.name) &&

Objects.equals(patronymic, student.patronymic) &&

Objects.equals(birthDate, student.birthDate) &&

Objects.equals(address, student.address) &&

Objects.equals(phone, student.phone) &&

Objects.equals(faculty, student.faculty);

}

@Override

public int hashCode() {

return Objects.hash(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

@Override

public String toString() {

return "Student{" +

"id=" + id +

", surname='" + surname + '\'' +

", name='" + name + '\'' +

", patronymic='" + patronymic + '\'' +

", birthDate='" + birthDate + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", faculty='" + faculty + '\'' +

", course=" + course +

", group=" + group +

'}';

}

}

* 1. **printer.java**

package by.bntu.linux.lab7;

import java.io.\*;

import java.util.List;

public class Printer {

public void saveToFile(List<Student> students) {

try(BufferedWriter writer = new BufferedWriter(new FileWriter("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

for(Student student : students) {

String line = student.getId() + " "

+ student.getSurname() + " " + student.getName() + " " + student.getPatronymic() + " "

+ student.getBirthDate() + " " + student.getAddress() + " " + student.getPhone() + " "

+ student.getFaculty() + " " + student.getCourse() + " " + student.getGroup();

writer.write(line);

}

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

}

* 1. **input.java**

package by.bntu.linux.lab7;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Input {

public List<Student> readFromFile() {

try(BufferedReader reader = new BufferedReader(new FileReader("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

String line;

List<Student> students = new ArrayList<>();

while ((line = reader.readLine()) != null) {

String[] fields = line.split(" ");

Student student =

new Student(Integer.parseInt(fields[0]), fields[1], fields[2], fields[3], fields[4], fields[5],

fields[6], fields[7], Integer.parseInt(fields[8]), Integer.parseInt(fields[9]));

students.add(student);

}

return students;

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

public Student inputStudent() {

Scanner scanner = new Scanner(System.in);

System.out.print("Surname: ");

String surname = scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Patronymic: ");

String patronymic = scanner.nextLine();

System.out.print("Birth date: ");

String birthDate = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

System.out.print("Phone: ");

String phone = scanner.nextLine();

System.out.print("Faculty: ");

String faculty = scanner.nextLine();

System.out.print("Course: ");

int course = scanner.nextInt();

System.out.print("Group: ");

System.out.print("Id: ");

int id = scanner.nextInt();

int group = scanner.nextInt();

return new Student(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

}

* 1. **view.java**

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

* 1. **controller.java**

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Scanner;

public class Controller {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(new File("in.txt"));

PrintWriter pw = new PrintWriter(new File("out.txt"));

String[] student;

while (sc.hasNextLine()) {

student = sc.nextLine().split("\\s+");

double sum = 0;

for (int i = 1; i < student.length; i ++) {

sum += Integer.parseInt(student[i]);

}

if (sum / (student.length - 1) > 7) {

pw.println(student[0].toUpperCase());

}

}

sc.close();

pw.close();

}

}

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №9

По дисциплине **«Операционные системы»**

Тема: **«IO with System Call»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Повторить фундаментальные основы и концепции файлового ввода-вывода в Linux, а также изучить системные вызовы (низкоуровневые функции) для взаимодействия с файловой системой и научиться использовать низкоуровневый ввод-вывод при написании системных программ.

**Требования:**

1) Разработать многофайловый консольный проект на С/С++ согласно варианту задания с использованием шаблона (паттерна) проектирования MVC.

2) Предусмотреть ещё один способ инициализации данных и вывода результата работы программы – с помощью файлового ввода-вывода. Программа должна дополнительно выводить на консоль исходные данные и конечный результат.

3) ЗАПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

4) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

5) Каждое задание оформить в виде отдельной бизнес-функции. Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

6) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

7) При выполнении задания разрешается использовать IDE, а также задействовать любой текстовый редактор (к примеру, gedit) и набор компиляторов GNU Compiler Collection (GCC), в частности, компиляторы языков программирования С/C++ gcc/g++, а также утилиту для создания файлов-архивов ar.

8) Для автоматизации сборки проекта необходимо использовать утилиту-автосборщик – GNU make.

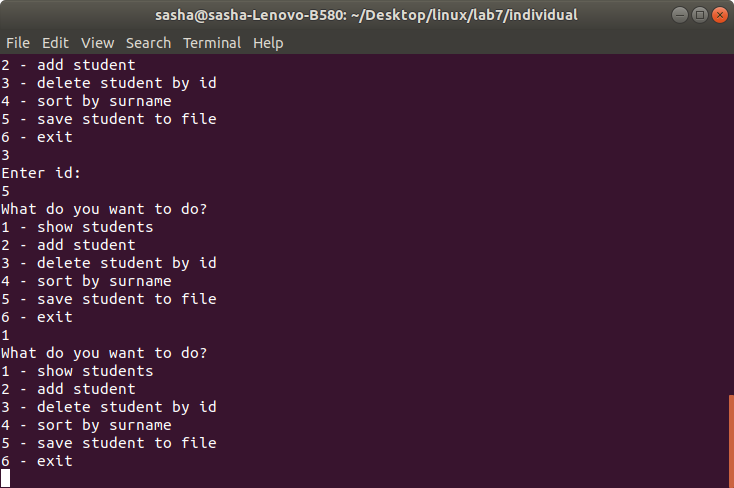
9) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

10) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Основное задание:**

Произвести рефакторинг основного задания предыдущей лабораторной работы: пользовательскую библиотеку функций ввода-вывода переписать с использованием низкоуровневых функций, которые представляет само ядро ОС.

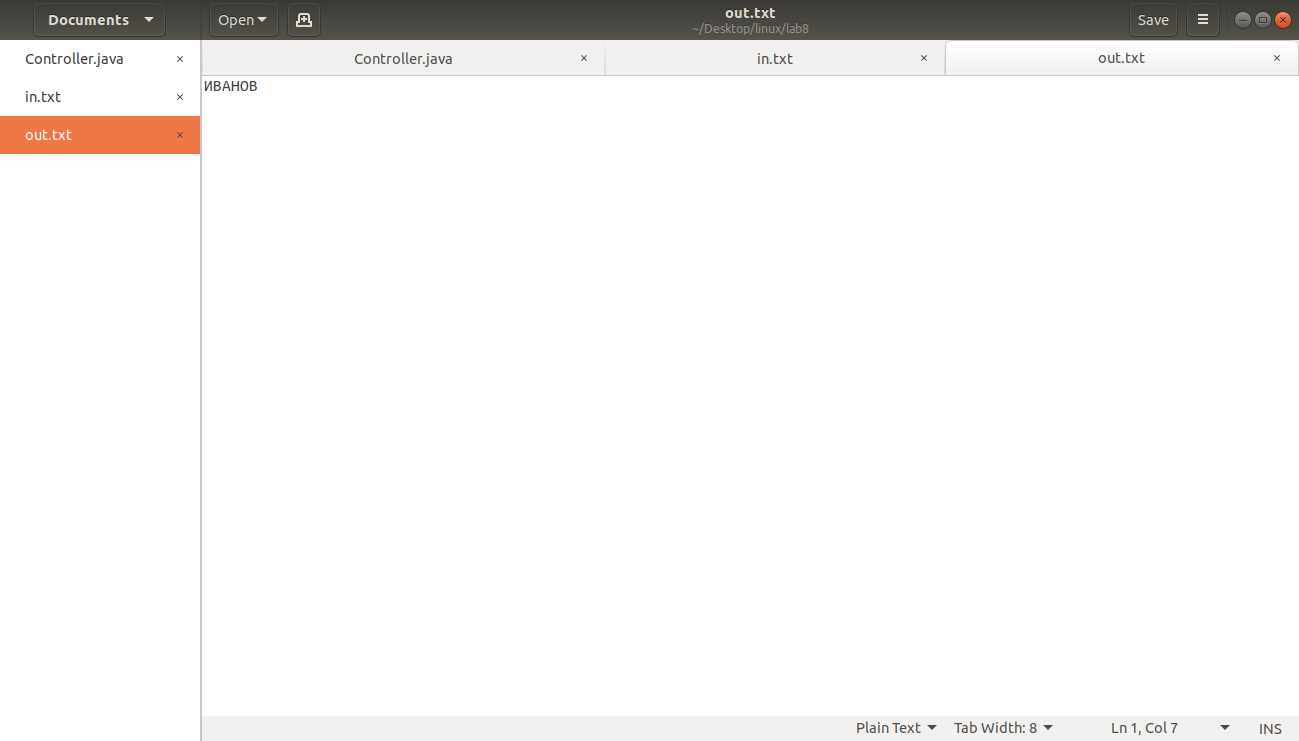
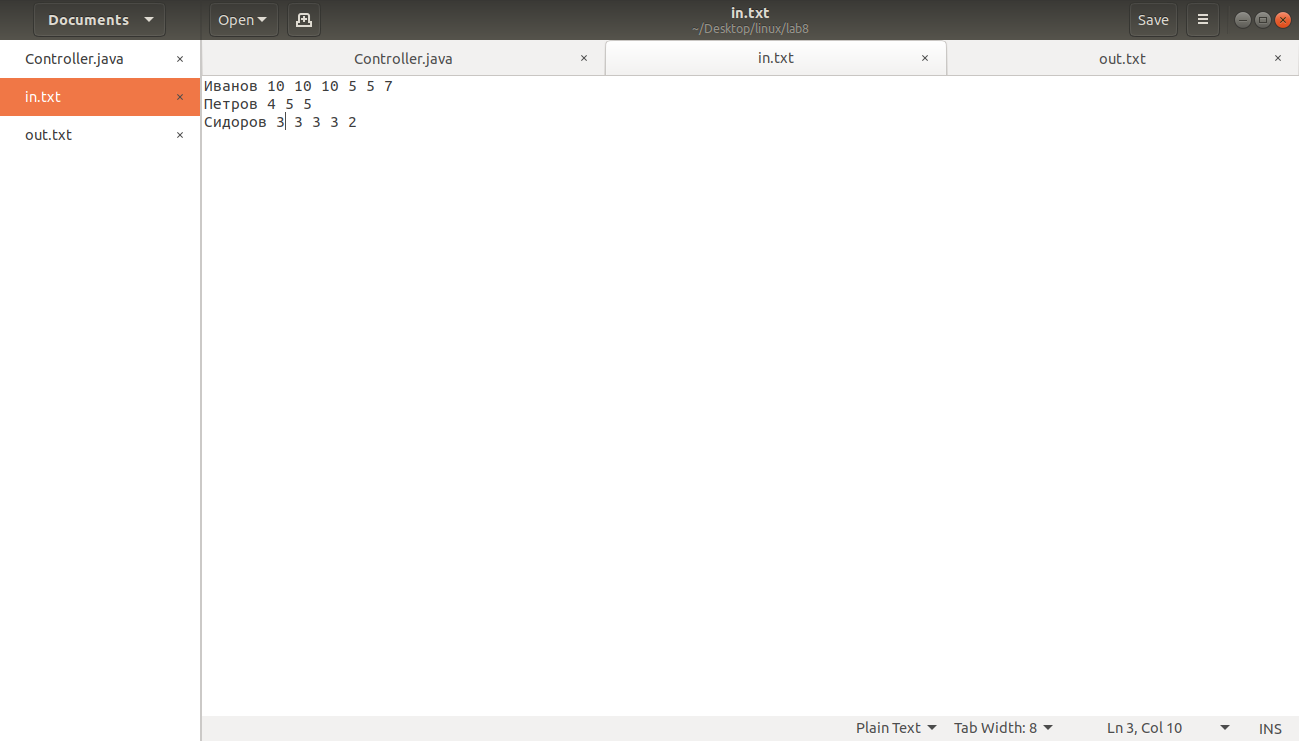
*Результаты выполнения*



**Индивидуальное задание:**

4) В файле, содержащем фамилии студентов и их оценки, записать прописными буквами фамилии тех студентов, которые имеют средний балл более “7”.

*Результаты выполнения*



**Контрольные вопросы:**

**1**. Основные концепции файлового ввода-вывода с точки зрения ОС Linux (таблица файлов, дескрипторы фалов (fd), обычные файлы, позиция в файле, смещение в файле, длина файла, усечение, информационный узел, номер inode)?

*Ответ:*

Обзор механизмов ввода-вывода в Linux. В языке C для осуществления файлового ввода-вывода используются механизмы стандартной библиотеки языка, объявленные в заголовочном файле stdio.h. Как вы вскоре узнаете консольный ввод-вывод - это не более чем частный случай файлового ввода-вывода. В C++ для ввода-вывода чаще всего используются потоковые типы данных. ... С точки зрениямодели КИС (Клиент-Интерфейс-Сервер), сервером стандартных механизмов вводавывода языка C (printf, scanf, FILE\*, fprintf, fputc и т. д.) является библиотека языка. А сервером низкоуровневого ввода-вывода в Linux, является само ядро операционной системы.

**2.** Абстракция файла и файлового ввода-вывода на уровне ядра ОС Linux (\*nix)?

*Ответ:*

Абстракция данных означает предоставление только важной информации внешнему миру и скрытие их фоновых данных, т. е. Представление необходимой информации в программе без предоставления деталей.

Абстракция данных - это метод программирования (и проектирования), который основан на разделении интерфейса и реализации.

**3**. Каталоги и ссылки в Linux (каталог, ссылка, корневой каталог, полный путь, относительный путь, записи каталога, разрешение полного пути, кэш dentry, абсолютный полный путь, рабочий каталог, жёсткие ссылки, символические ссылки, специальные файлы)?

*Ответ:*

*Каталог —* это особый файл, чье содержимое принимает форму таблицы из имен файлов в совокупности с указателями на соответствующие файлы. Эта связка из имени файла и указателя на него называется *ссылкой,* и у файлов в одном и том же или в разных ката­логах может быть несколько ссылок, а следовательно, и несколько имен.

Каталоги могут содержать ссылки как на файлы, так и на другие каталоги. С помощью ссылок между каталогами устанавливается иерархия каталогов, показанная на рис. 1.

Каждый каталог содержит как минимум две записи: . (точка), которая представляет собой ссылку на сам каталог, и .. (точка-точка), которая является ссылкой на его роди­тельский каталог — тот каталог, что расположен над ним в иерархии. Каждый каталог, за исключением корневого, имеет свой *родительский каталог.* Для корневого каталога запись .. является ссылкой на него самого (таким образом, обозначение /.. — то же самое, что и /).

Символьные ссылки

Подобно обычной ссылке, *символьная ссылка* предоставляет альтернативное имя для файла. Но, в отличие от обычной ссылки, представляющей собой в списке каталога запись вида «имя файла плюс указатель», символьная ссылка — это специально помеченный файл, содержащий имя другого файла. (Иными словами, у символьной ссылки в каталоге есть запись вида «имя файла плюс указатель», и файл, на который ссылается указатель, содержит строку с именем другого файла.) Этот последний файл часто называют *целью*символьной ссылки, и зачастую говорится, что символьная ссылка «указывает» или «ссылается» на целевой файл. Когда в системном вызове указывается путевое имя, в боль­шинстве случаев ядро автоматически снимает *косвенность* каждой символьной ссылки в путевом имени (также говорят *«следует* по ним»), заменяя ее именем того файла, на который она ведет. Этот процесс может происходить рекурсивно, если цель символьной ссылки сама по себе является символьной ссылкой. (Ядро накладывает ограничение на количество ссылок, чтобы предотвратить возможность появления замкнутых цепочек символьных ссылок.) Если символьная ссылка указывает на несуществующий файл, то говорится, что это *битая ссылка.*

В качестве альтернативных названий для обычной и символьной ссылки зачастую используются выражения *«жесткая ссылка»* и *«мягкая ссылка»*.

У каждого процесса есть свой *текущий рабочий каталог* (который иногда называют про­сто *рабочим* или *текущим).* Это «текущее местоположение» процесса в иерархии одного каталога, и именно с данного каталога для процесса интерпретируются относительные путевые имена.

Процесс наследует свой текущий рабочий каталог от родительского процесса. В случае входа в систему для оболочки рабочим каталогом является домашний каталог пользова­теля, который указан в его записи в файле паролей. Текущий рабочий каталог оболочки может быть изменен с помощью команды cd.

С каждым файлом связаны UID и GID, определяющие владельца этого файла и группу, к которой он принадлежит. Понятие «владение файлом» применяется для определения прав доступа пользователей к файлу.

**4**. Системные вызовы: ограничения системных вызовов, коды возврата системных вызовов, режим ядра, режим пользователя?

*Ответ:*

Систе́мный вы́зов (англ. system call) в программировании и вычислительной технике — обращение прикладной программы к ядруоперационной системы для выполнения какой-либо операции. Современные операционные системы (ОС) предусматривают разделение времени между выполняющимися вычислительными процессами (многозадачность) и разделение полномочий, препятствующее обращению исполняемых программ к данным других программ и оборудованию. Ядро ОС исполняется в привилегированном режиме работы процессора.

**5**. Использование системных вызовов для низкоуровневой работы с каталогами и файлами: mkdir, rmdir, chdir, opendir, closedir, readdir, readlink, link, unlink, stat, … ?

*Ответ:*

Mkdir - команда для создания новых каталогов

Rmdir - команда в операционных системах DOS, UNIX, Windows, которая удаляет каталог из файловой системы

Chdir - каманда каманднага радка, якая ўжываецца для змены бягучага каталога ў Unix, Linux, DOS, Windows, а таксама ў іншых аперацыйных сістэмах

Opendir -  открывает дескриптор каталога

Closedir – закрывает дескриптор каталога

Readdir - получить элемент каталога по его дескриптору

Readlink - считывает значение символьной ссылки

Link - команда Unix, создающая жёсткую ссылку или символьную ссылку на файл

Unlink – команда для удаления файл в системах GNU/Linux

Stat - утилита командной строки, которая отображает подробную информацию о данных файлах или файловых системах

**6**. Использование системных вызовов для низкоуровневой работы с файлом: open, creat, write, read, close, … ?

*Ответ:*

Open – открывают файл или устройство

Create –  открывают и создают файл или устройство

Write – unix-утилита, позволяющая отправить сообщение другому пользователю, который имеет сессию в этой же системе

Read – дну строку из стандартного входного потока и записывает ее содержимое в указанные переменные.

Close – закрывает файловый дескриптор, который после этого не ссылается ни на один и файл и может быть использован повторно.

**7**. Функции библиотеки С для получения описания возникших ошибок файлового ввода-вывода: perror, strerror переменная errno?

*Ответ:*

Perror – выводит в стандартный поток ошибки сообщения, описывая ошибку, произошедшую при последнем системном вызове или вызове

Strerror – функция языков C/C++, транслирующая код ошибки, который обычно хранится в глобальной переменной errno, в сообщение об ошибке, понятном человеку.

**Листинг:**

. 1) controller.java

package by.bntu.linux.lab7;

import java.util.Comparator;

import java.util.List;

import java.util.Scanner;

import java.util.stream.Collectors;

public class Controller {

public static void main(String[] args) {

Input input = new Input();

Printer printer = new Printer();

List<Student> students = input.readFromFile();

Scanner scanner = new Scanner(System.in);

int choice;

do {

System.out.println("What do you want to do?\n" +

"1 - show students\n" +

"2 - add student\n" +

"3 - delete student by id\n" +

"4 - sort by surname\n" +

"5 - save student to file\n" +

"6 - exit");

choice = scanner.nextInt();

switch (choice) {

case 1:

students.forEach(System.out::println);

break;

case 2:

students.add(input.inputStudent());

break;

case 3:

System.out.println("Enter id: ");

int id = scanner.nextInt();

students = students

.stream()

.filter(s -> s.getId() != id)

.collect(Collectors.toList());

break;

case 4:

students.sort(Comparator.comparing(Student::getSurname));

break;

case 5:

printer.saveToFile(students);

break;

case 6:

break;

default:

System.out.println("Invalid number!");

}

} while (choice != 6);

}

}

2) student.java

package by.bntu.linux.lab7;

import java.util.Objects;

public class Student {

private int id;

private String surname;

private String name;

private String patronymic;

private String birthDate;

private String address;

private String phone;

private String faculty;

private int course;

private int group;

public Student() {}

public Student(int id, String surname, String name, String patronymic, String birthDate, String address,

String phone, String faculty, int course, int group) {

this.id = id;

this.surname = surname;

this.name = name;

this.patronymic = patronymic;

this.birthDate = birthDate;

this.address = address;

this.phone = phone;

this.faculty = faculty;

this.course = course;

this.group = group;

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

}

public String getSurname() {

return surname;

}

public void setSurname(String surname) {

this.surname = surname;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getPatronymic() {

return patronymic;

}

public void setPatronymic(String patronymic) {

this.patronymic = patronymic;

}

public String getBirthDate() {

return birthDate;

}

public void setBirthDate(String birthDate) {

this.birthDate = birthDate;

}

public String getAddress() {

return address;

}

public void setAddress(String address) {

this.address = address;

}

public String getPhone() {

return phone;

}

public void setPhone(String phone) {

this.phone = phone;

}

public String getFaculty() {

return faculty;

}

public void setFaculty(String faculty) {

this.faculty = faculty;

}

public int getCourse() {

return course;

}

public void setCourse(int course) {

this.course = course;

}

public int getGroup() {

return group;

}

public void setGroup(int group) {

this.group = group;

}

@Override

public boolean equals(Object o) {

if (this == o) return true;

if (o == null || getClass() != o.getClass()) return false;

Student student = (Student) o;

return id == student.id &&

course == student.course &&

group == student.group &&

Objects.equals(surname, student.surname) &&

Objects.equals(name, student.name) &&

Objects.equals(patronymic, student.patronymic) &&

Objects.equals(birthDate, student.birthDate) &&

Objects.equals(address, student.address) &&

Objects.equals(phone, student.phone) &&

Objects.equals(faculty, student.faculty);

}

@Override

public int hashCode() {

return Objects.hash(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

@Override

public String toString() {

return "Student{" +

"id=" + id +

", surname='" + surname + '\'' +

", name='" + name + '\'' +

", patronymic='" + patronymic + '\'' +

", birthDate='" + birthDate + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", faculty='" + faculty + '\'' +

", course=" + course +

", group=" + group +

'}';

}

}

3) printer.java

package by.bntu.linux.lab7;

import java.io.\*;

import java.util.List;

public class Printer {

public void saveToFile(List<Student> students) {

try(BufferedWriter writer = new BufferedWriter(new FileWriter("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

for(Student student : students) {

String line = student.getId() + " "

+ student.getSurname() + " " + student.getName() + " " + student.getPatronymic() + " "

+ student.getBirthDate() + " " + student.getAddress() + " " + student.getPhone() + " "

+ student.getFaculty() + " " + student.getCourse() + " " + student.getGroup();

writer.write(line);

}

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

}

4) input.java

package by.bntu.linux.lab7;

import java.io.BufferedReader;

import java.io.FileReader;

import java.io.IOException;

import java.util.ArrayList;

import java.util.List;

import java.util.Scanner;

public class Input {

public List<Student> readFromFile() {

try(BufferedReader reader = new BufferedReader(new FileReader("/home/sasha/Desktop/linux/lab7/individual/src/students.txt"))){

String line;

List<Student> students = new ArrayList<>();

while ((line = reader.readLine()) != null) {

String[] fields = line.split(" ");

Student student =

new Student(Integer.parseInt(fields[0]), fields[1], fields[2], fields[3], fields[4], fields[5],

fields[6], fields[7], Integer.parseInt(fields[8]), Integer.parseInt(fields[9]));

students.add(student);

}

return students;

} catch (IOException ex) {

throw new IllegalArgumentException(ex.getMessage(), ex);

}

}

public Student inputStudent() {

Scanner scanner = new Scanner(System.in);

System.out.print("Surname: ");

String surname = scanner.nextLine();

System.out.print("Name: ");

String name = scanner.nextLine();

System.out.print("Patronymic: ");

String patronymic = scanner.nextLine();

System.out.print("Birth date: ");

String birthDate = scanner.nextLine();

System.out.print("Address: ");

String address = scanner.nextLine();

System.out.print("Phone: ");

String phone = scanner.nextLine();

System.out.print("Faculty: ");

String faculty = scanner.nextLine();

System.out.print("Course: ");

int course = scanner.nextInt();

System.out.print("Group: ");

System.out.print("Id: ");

int id = scanner.nextInt();

int group = scanner.nextInt();

return new Student(id, surname, name, patronymic, birthDate, address, phone, faculty, course, group);

}

}

5) view.java

#pragma once

void printMatrixResult(int);

void printMatrix(int\*\*, int, int);

6) controller.java

import java.io.File;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.Scanner;

public class Controller {

public static void main(String[] args) throws IOException {

Scanner sc = new Scanner(new File("in.txt"));

PrintWriter pw = new PrintWriter(new File("out.txt"));

String[] student;

while (sc.hasNextLine()) {

student = sc.nextLine().split("\\s+");

double sum = 0;

for (int i = 1; i < student.length; i ++) {

sum += Integer.parseInt(student[i]);

}

if (sum / (student.length - 1) > 7) {

pw.println(student[0].toUpperCase());

}

}

sc.close();

pw.close();

}}

Министерство образования Республики Беларусь

Белорусский Национальный Технический Университет

Факультет информационных технологий и робототехники

Кафедра «Программное обеспечение информационных систем и технологий»

**Отчет**

По лабораторной работе №10

По дисциплине **«Операционные системы»**

Тема: **«Linux Processes»**

**Выполнили:**

Студенты группы 10701118

Дубоделов А.В.

Лебедь А.С.

**Преподаватель:**

Станкевич С.Н.

Минск 2020

**Цель работы:**

Изучить фундаментальные концепции многозадачности в ОС, а также научиться порождать и управлять многозадачными приложениями в ОС Linux. **Требования:**

1) На базе полученных знаний разработать модульное консольное приложение на С/С++ согласно варианту задании A и B с использованием архитектурного шаблона проектирования MVC.

2) АПРЕЩАЕТСЯ в программе использовать под любым предлогом ГЛОБАЛЬНЫЕ переменные!

3) Для повышения производительности программы и закрепления навыков работы с памятью везде, где это возможно, необходимо использовать ДИНАМИЧЕСКОЕ выделение и освобождение памяти, а также осуществлять работу через УКАЗАТЕЛИ.

4) Все функции должны быть сгруппированы по соответствующим отдельным файлам и вынесены в отдельную библиотеку.

5) Все функции должны быть самодостаточные, т.е. при их разработке необходимо придерживаться принципа Single Responsibility Principle.

6) Для компиляции, компоновки и выполнения программы использовать средства автосборки (к примеру, Makefiles и утилиту make).

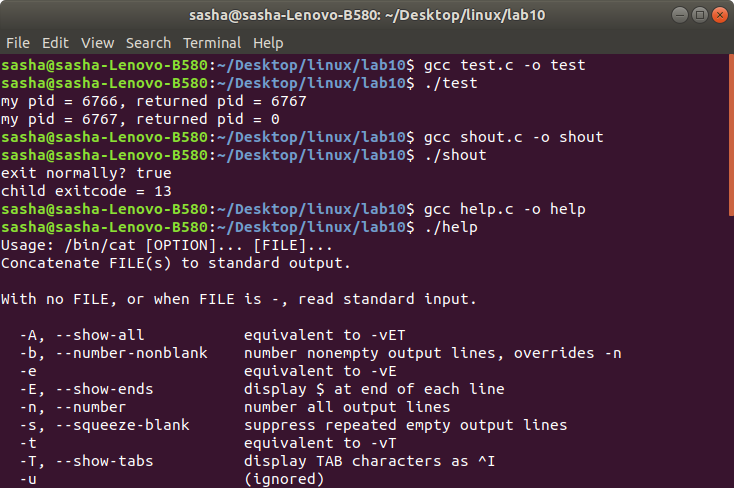
7) При разработке программ придерживайтесь соглашений по написанию кода на C/C++ (Code-Convention).

8) Контрольные вопросы по лабораторной работе и ответы на них должны быть записаны в конспект.

**Индивидуальное задание:**

1) Написать программу, которая будет превращать в демона сама себя.

*Результаты выполнения*



**Контрольные вопросы:**

**1.** Основы многозадачности (multitasking) и мультипрограммирования в системах пакетной обработки, в системах разделения времени и в системах реального времени? Мультипроцессорная обработка?

*Ответ:*

Разделяют мультипрограммирование в пакетных системах, системах реального времени и в системах разделения времени. Пакетная обработка используется для достижения максимальной эффективности использования ресурсов вычислительной машины при выполнении вычислительных задач путём сбалансированной загрузки её компонентов, как например, АЛУ и УВВ.

**2**. Понятие процесса как второй фундаментальной абстракции в Linux-системах, основные концепции, поддержка процессов ядром ОС Linux? Древовидная иерархия процессов в Linux, процесс init, Linux-команды для мониторинга процессов и просмотра их иерархии?

*Ответ:*

Процесс - это объект ОС Linux, который состоит из адресного пространства памяти и набора структур данных. По сути, процесс это запущенная программа или служба. Каждый запущенный процесс в ОС Linux может породить дополнительные процессы. Процесс, запустивший новый процесс называется родительским процессом. Новый процесс по отношению к создавшему его процессу называется дочерним. Процессы - это не то же самое, что задачи: процессы являются частью операционной системы.

**3**. Планирование процессов и их приоритет? Уступчивость процесса и использование системного вызова nice?

*Ответ:*

Знание алгоритмов **планирования** **процессов** **и** системы **приоритетов** ОС Unix во многих случаях позволяют **системному** администратору успешно справиться с проблемами повышения производительности. В случае если компьютер, работающий под управлением ОС Unix, решает задачи в реальном времени, значение **планирования** **процессов** еще больше возрастает. В предлагаемой статье дан обзор вопросов, связанных с **приоритетами** **процессов** **и** **их** **планированием** в ОС Unix. Однако ряд моментов **планирования**, специфических для мультипроцессорных систем, в статье не рассматривается.

**4.** Структура данных Linux-процесса, идентификатор и атрибуты процесса? Способы получения информации о процессе? Использование соответствующих системных вызовов? Как получить имя пользователя из его числового имени, т.е. из UID?

*Ответ:*

Атрибуты процесса. Процесс в ядре представляется просто как структура с множеством полей (определение структуры можно прочитать здесь). Но так как статья посвящена системному программированию, а не разработке ядра, то несколько абстрагируемся и просто акцентируем внимание на важных для нас полях процесса: Идентификатор процесса (pid). Открытые файловые дескрипторы (fd). Обработчики сигналов (signal handler).

**5**. Какие бывают типы процессов (порождающий (родитель), порождённый (потомок), демон, зомби, приведение)?

*Ответ:*

Демоны и потоки. Среди всех процессов можно выделить несколько особых типов процессов. Системные процессы являются частью ядра и всегда находятся в оперативной памяти. ... Демоны отличаются от обычных процессов только тем, что они работают в неинтерактивном режиме. Если с обычным процессом всегда ассоциирован какой-то терминал или псевдотерминал, через который осуществляется взаимодействие процесса с пользователем, то демон такого терминала не имеет. ... Но бывают случаи, когда родительский процесс завершается раньше дочернего. Процессы, не имеющие родителя, называются "сиротами".

**6**. Состояние процесса, время выполнения процесса, понятие очереди обработки процессов? Подходы, используемые для обработки очередей: квантование, приоритет, смешанный?

*Ответ:*

Например, приоритет очереди системных процессов устанавливается выше, чем приоритет очередей пользовательских процессов. А приоритет очереди процессов, запущенных студентами, ниже, чем для очереди процессов, запущенных преподавателями.

**7**. Как осуществить порождение дочернего процесса? Концепция развилки, создание (ветвление) и исполнение новых процессов (передача управления)? Использование системных вызовов fork (vfork) и семейства вызовов exec (execl, execlp, execle, execv, execvp, execve)?

*Ответ:*

Так как любой процесс содержит хотя бы одну нить, создание (т.е. порождение) и завершение процессов подразумевает создание и завершение нитей. ... Для создания нового потока управления в рамках уже существующей программы. AIX создает новый процесс. Для создания нового процесса, выполняющего другую программу. В этом случае вскоре после вызова функции fork вызывается одна или несколько функций exec. В программе с несколькими нитями новые потоки управления создаются не функцией fork, а функцией pthread\_create. Функцию fork следует применять только для запуска программ.

**8**. Как осуществить ожидание выполнения дочернего процесса? Использование системных вызовов wait и waitpid?

*Ответ:*

wait, waitpid - ожидает завершения процесса. СИНТАКСИС. #include <sys/types.h> #include <sys/wait.h>. ... Функция wait приостанавливает выполнение текущего процесса до тех пор, пока дочерний процесс не завершится, или до появления сигнала, который либо завершает текущий процесс, либо требует вызвать функцию-обработчик. Если дочерний процесс к моменту вызова функции уже завершился (так называемый "зомби" ("zombie")), то функция немедленно возвращается. Системные ресурсы, связанные с дочерним процессом, освобождаются.

**9**. Как завершить выполнение процесса? Использование системного вызова \_exit и библиотечной С-функции exit? Что произойдет с работающим потомком при завершении родительского процесса?

*Ответ:*

Системный вызов \_exit завершает процесс, он вызывает освобождение сегмента данных процесса, сегмента стека, распределения областей и закрытия всех открытых дескрипторов файловю. Но запись в таблице процессов, в которой был зарегистрирован этот процесс, не удаляется, чтобы там был зарегистрирован статус завершения процесса и информация о его выполнении. Такой остаток процесса называется процессом - “зомби”. ... Системные функции типа exec. Существует группа системных функций, которые предназначаются для перезапуска программы. #include<unistd.h>. ... exit (0); } } Системный вызов exec() заставляет вызывающий процесс изменить свой контекст и выполнить другую программу.

**10**. В каких целях используют библиотечную функцию system? Как она работает?

*Ответ:*

Электронно-библиотечная система (ЭБС) — это предусмотренный федеральными государственными образовательными стандартами высшего профессионального образования (ФГОС ВПО) России обязательный элемент библиотечно-информационного обеспечения учащихся вузов, представляющий собой базу данных, содержащую издания учебной, учебно-методической и иной литературы, используемой в образовательном процессе.

**Листинг:**

* 1. **help.cpp**

#define \_GNU\_SOURCE

#include <unistd.h>

int main() {

char\* args[] = { "/bin/cat", "--help", NULL };

execve("/bin/cat", args, environ);

return 1;

}

* 1. **shout.cpp**

#include <stdio.h>

#include <unistd.h>

#include <errno.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() {

int pid = fork();

switch(pid) {

case -1:

perror("fork");

return -1;

case 0:

// Child

return 13;

default: {

// Parent

int status;

waitpid(pid, &status, 0);

printf("exit normally? %s\n", (WIFEXITED(status) ? "true" : "false"));

printf("child exitcode = %i\n", WEXITSTATUS(status));

break;

}

}

return 0;

}

* 1. **test.cpp**

#include <stdio.h>

#include <unistd.h>

#include <errno.h>

#include <sys/wait.h>

#include <sys/types.h>

int main() {

int pid = fork();

switch(pid) {

case -1:

perror("fork");

return -1;

case 0:

// Child

printf("my pid = %i, returned pid = %i\n", getpid(), pid);

break;

default:

// Parent

printf("my pid = %i, returned pid = %i\n", getpid(), pid);

break;

}

return 0;

}