ТЕХНИЧЕСКИ УНИВЕРСИТЕТ – ВАРНА

Факултет по изчислителна техника и автоматизация

Катедра „КНТ“

**СЕМЕСТРИАЛНА ДОМАШНА РАБОТА**

по дисциплината „Базово програмиране”

на тема: „Информационна система Монитори”

Вариант 117

|  |  |
| --- | --- |
| Изготвил:  Иво Мирославов Димов | Проверил: |
| Специалност: КСТ |  |
| Група: IV |  |
| Факултетен номер: 23621507 |  |

2024

​​Съдържание

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​

​**​**

1. **Задание на проекта**

**Информационна система Монитори**

Да се напише компютърна програма, реализираща информационна система за продажба на монитори (сериен номер, марка, модел, цвят, цена, диагонал на екрана, разделителна способност, вграден TV тунер(да/не), цвят, тип: нов, втора употреба, статус: в продажба, капариран, продаден; в промоция (указва се % на отстъпка – като 0% се счита, че не е в промоция!)). Програмата да съхранява и да обработва данни за наличните монитори. Максималният брой монитори, които могат да се поддържат, е 100.

Базова задача

A. Меню за избор на функциите от програмата. Функции от програмата са:

B. Добавяне на монитори

a. Добавяне на монитори, чрез добавяне в масива на n монитора, като не трябва да се превишава максималният брой елементи. Пример: Добавяне на списък от монитори. Въвежда се цяло число n и след него n на брой монитора . n не може да надвишава свободните елементи в масива с монитори.

C. Извеждане на всички монитори на екрана

a. Извеждане на всички монитори, чрез подходящо форматиране в таблица

D. Търсене и извеждане на екрана:

c. Търсене и извеждане на монитора с максимален размен на екрана

d. Търсене и извеждане на мониторите от определена марка.

E. Подреждане на основния масив, без да се извежда на екрана:

a. Сортиране на мониторите по цена.

F. Работа с външен файл(двоичен или текстов):

Извеждане на масива във файл.

b. Въвеждане на масива от файл.

Допълнение първо (+ базова задача)

G. Да се създаде подменю към основното с нови функции за:

a. Извеждане на мониторите на екрана подредени по марка и цена

b. Извеждане на мониторите по избрам диагонал подредени по разделителна способност.

Допълнение второ (+ базова задача)

H. Корекция на данни за монитори.

a. Въвеждат се марка и модел и данните за корекция

b. Въвежда се сериен/ни номер/а и се прави корекция на тези монитори

c. Определяне на промоционални %ти за модел монитори и/или за определени бройки монитори (по сериен номер)

I. Продажба на монитори:

d. Въвеждат се модел и марка и се избира от наличните монитори по сериен номер.

e. Капариране на монитор. След капариране се намалява реалната цена с капарото.

Допълнение трето(+ базова задача)

J. Данните в програмата да се попълват автоматично от файл при стартиране и да се записват автоматично във файл при затваряне на програмата.

1. **Анализ на решението**
2. Структури за данните в програмата

*Таблица  1. Структури*

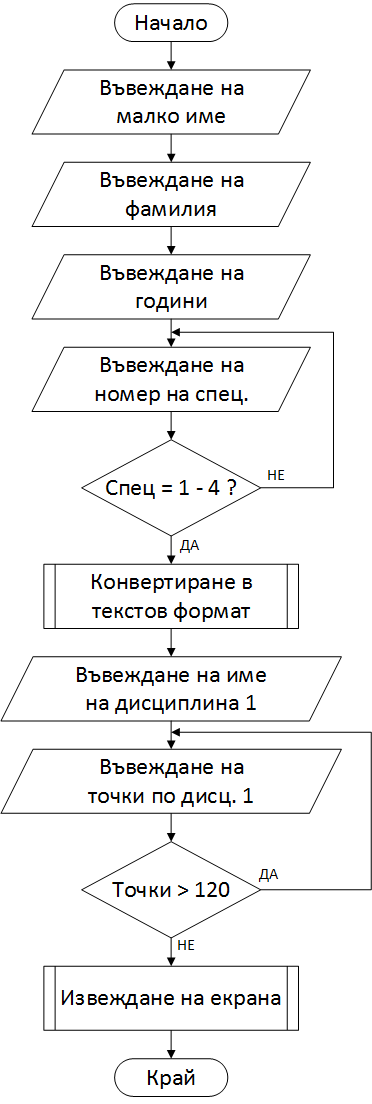
|  |  |  |
| --- | --- | --- |
| Структура | Обяснение | Примерни стойности |
| struct Monitors  {  int id = 0;  string manufacturer;  string model;  string color;  float price = 0;  float size = 0;  string resolution;  bool tvTuner = false;  string type;  string status = "Selling";  }; | id– сериен номер на монитора   manufacturer – име на производителя   model -модел на монитора  color -цвят на монитора  price- цена на монитора  size- диагонал на монитора  resolution- резолюция на монитора  tvTuner -дали монитора има TV тунер  type-вид на монитора  status- състояние на монитора | 23   AMD  ROG  green  45.76  14,6  1920x1080  No  New  Selling |

1. Реализация на условие A
2. Меню за избор на функциите в програмата

1. Блок схема на алгоритъма
2. Реализация на условие B
3. Алгоритъм Добавяне на студенти в групата.

Изисква от потребителят да въведе данните за един студент, описани в структурата Student в т.Х . Предвидено валидирано въвеждане на данни (например: за специалност се въвежда числото от 1 до 4, отговарящо на съответната специалност). Оценката се изчислява автоматично, чрез извикване на подалгоритъм за изчисление, реализиран с функцията int calcFinalMark(const unsigned short points);

1. Блок схема на алгоритъма



*Фиг. 3.. Блок схема на Алгоритъм въвеждане на данни за студент*

1. Функция, с която е реализиран алгоритъма

void inputStudentData(Student& st);

1. Входни данни на функцията

Student& st – променлива от тип структура Student, в която се съхраняват данните за един студент

1. Изходни данни на функцията или данни, които се извеждат

Няма. Функцията връща данните за студента, през параметъра Student& st.

1. Помощни функции

int calcFinalMark(const unsigned short points) – Изчислява оценката по предмет спрямо точките на студента. Функцията връща като резултат:

* от 2 до 6 - при точки в диапазона 0-120
* -1 при невалидни точки (>120).

 Параметри:

const unsigned short points – точки на студент по дисциплина

1. Реализация на условие C
2. Алгоритъм Извеждане на данни на студент

При добавени данни в масива със студенти, те се извеждат на екрана в табличен вид. Предвидена е проверка за въведени студенти, в противен случай се извежда подходящо съобщение на екрана за добавяне на данни. За отпечатването в табличен вид се използват подалгоритми за отпечатване на заглавен ред (хедър) с заглавия на колоните, завършващ ред (футър) и отпечатване на ред с данни и тяхното форматиране. Тези подалгоритми за реализирани с функциите

void printTableHeader(), void printTableFooter(); void printTableRow(const Student& st, const unsigned int recordNumber = 0);

1. Блок схема на алгоритъма

*Фиг. 4. Алгоритъм Извеждане на студенти*

1. Функция, с която е реализиран алгоритъма

void printStudentsInTable(const Student listStudents[], const unsigned short listSize);

1. Входни данни на функцията

const Student listStudents[]- списък със студентите

const unsigned short listSize – размер на масива (брой на студентите в масива)

1. Изходни данни на функцията или данни, които се извеждат

Няма изходни данни. Функцията отпечатва студентите в табличен вид.

1. Помощни функции

void printTableHeader() – отпечатва заглавният ред с направляващите линии и заглавията на колоните на таблицата

void printTableFooter() – отпечатва завършващият ред с направляващата линия в края на таблицата

void printTableRow(const Student& st, const unsigned int recordNumber = 0) – отпечатва данните за един студент на един ред с подходящо форматиране. За извеждане на оценката с име, се използва функция string finalMarkToString(const unsigned int finalMark). Параметри:

const Student& st – променлива за един студент

const unsigned int recordNumber = 0 – пореден номер на записа в таблицата

string finalMarkToString(const unsigned int finalMark) – Връща текстов формат на оценката с име и стойност (Например: Отличен (6)). Параметри:

const unsigned int finalMark – оценка по шестобалната система – от 2 до 6

1. Реализация на условие D
2. Алгоритъм Извеждане на студенти по специалност

Въвежда се номер на специалност в интервала от 1 до 4, съответстващ на специалностите КСТ, СИТ, ИИ, КС, като е реализирана валидация на въвежданият номер. Номерът се преобразува в текстов низ, използвайки подалгоритъм за конвертиране, реализиран чрез функцията char\* specualtyNumberToString(const unsigned short specialtyNum). В нов списък се отделят тези, студенти, които са от желаната специалност и се отпечатват на екрана в табличен вид. За отпечатването се използва функцията void printStudentsInTable(const Student listStudents[], const unsigned short listSize)

1. Блок схема на алгоритъма

*Фиг. 5. Блок схема на Алгоритъм Извеждане на студенти по специалност*

1. Функция с която е реализиран алгоритъма

void reportStudentsBySpecialty(const Student listStudents[], const unsigned short listSize, const char\* specialty);

1. Входни данни на функцията

const Student listStudents[] - списък със студенти

const unsigned short listSize – размер на масива (брой на студентите)

const char \*specialty – име на специалността, по която ще се търсят студенти

1. Изходни данни на функцията или данни, които се извеждат

Няма изходни данни. Функцията извежда студентите, които са от желаната специалност.

1. Реализация на условие E
2. Алгоритъм Сортиране на студенти по малко име по азбучен ред

Студентите се сортират по малко име във възходящ ред (азбучен ред), използвайки алгоритъма на пряката селекция (SelectonSort). За сортирането се използва подалгоритъм за размяна на данните на двама студенти, реализиран чрез функцията void swapStudents(Student& st1, Student& st2);

1. Блок схема на алгоритъма

*Фиг. 6. Алгоритъм Сортиране на студенти по азбучен ред*

1. Функция с която е реализиран алгоритъма

void sortListStudentsByFirstName(Student listStudents[], const unsigned short listSize)

1. Входни данни на функцията

Student listStudents[] – списък със студентите. След сортирането, списъкът автоматично се променя.

const unsigned short listSize – размер на масива (брой на студентите)

1. Изходни данни на функцията или данни, които се извеждат

Няма изходни данни. Функцията сортира масива със студенти по азбучен ред.

1. Помощни функции.

void swapStudents(Student& st1, Student& st2) - разменя данните на двама студенти в масива със студенти

1. Реализация на условие F
2. Алгоритъм Извеждане на масива със студенти във файл

Данните със студентите се записват в текстов файл чрез избор от главното меню. Данните за студентите се съхраняват всяка на отделен ред. Записването се реализира чрез функцията int saveStudentsToFile(const char fileName[], const Student listStudents[], const unsigned short listSize).

1. Блок схема на алгоритъма

Тук се поставя блок схема на описания алгоритъм.

1. Функция с която е реализиран алгоритъма

 int saveStudentsToFile(const char fileName[], const Student listStudents[], const unsigned short listSize)

1. Входни данни на функцията

const char fileName[] – име на текстовият файл за съхраняване на данните за студентите

const Student listStudents[] – масив (списък) със студентите

const unsigned short listSize – размер на масива

1. Изходни данни на функцията или данни, които се извеждат

int - код на грешка:

* -1 – файлът не може да се отвори за запис на данните
* 0 – данните са записани успешно във файла

1. Алгоритъм Въвеждане на масива със студенти от файл

Данните със студентите се прочитат от текстов файл чрез избор от главното меню. Данните за студентите се четат ред по ред. Четенето се реализира чрез функцията int loadStudentsFromFile(const char fileName[], Student listStudents[], unsigned short& listSize)

1. Функция с която е реализиран алгоритъма

int loadStudentsFromFile(const char fileName[], Student listStudents[], unsigned short& listSize)

1. Входни данни на функцията

const char fileName[] – име на текстовият файл за съхраняване на данните за студентите

const Student listStudents[] – масив (списък) със студентите

1. Изходни данни на функцията или данни, които се извеждат

int - код на грешка:

* -1 – файлът не може да се отвори за запис на данните
* 0 – данните са прочетени успешно от файла

При успешно прочитане на данни, функцията записва в параметъра unsigned short& listSize,  размера на масива (броят на студентите, за които са прочетени данни)

1. Реализация на условие … - допълнение първо
2. Анализ на алгоритъма, който трябва да се реализира

Моля, опишете.

1. Блок схема на алгоритъма

Тук се поставя блок схема на описания алгоритъм.

1. Функция с която е реализиран алгоритъма

Тук се представя функцията, с която се реализира алгоритъма с нейния прототип. Описва се как трябва да се използва и какво се случва след нейната употреба.

1. Входни данни на функцията

Тук се представят входните данни на функцията: какви са, защо са избрани точно те, за какво ще се използват.

1. Изходни данни на функцията или данни, които се извеждат

Тук се представят изходните данни на функцията или какво ще се изведе на екрана от нея. Ако не връща резултат и не извежда данни, се описва какво е действието ѝ. Изяснява се защо е избран точно този тип и начин на извеждане на данните.

1. Реализация на условие … - допълнение второ
2. Анализ на алгоритъма, който трябва да се реализира

Моля, опишете.

1. Блок схема на алгоритъма

Тук се поставя блок схема на описания алгоритъм.

1. Функция с която е реализиран алгоритъма

Тук се представя функцията, с която се реализира алгоритъма с нейния прототип. Описва се как трябва да се използва и какво се случва след нейната употреба.

1. Входни данни на функцията

Тук се представят входните данни на функцията: какви са, защо са избрани точно те, за какво ще се използват.

1. Изходни данни на функцията или данни, които се извеждат

Тук се представят изходните данни на функцията или какво ще се изведе на екрана от нея. Ако не връща резултат и не извежда данни, се описва какво е действието ѝ. Изяснява се защо е избран точно този тип и начин на извеждане на данните.

1. Реализация на допълнение трето
2. Анализ на алгоритъма, който трябва да се реализира

Моля, опишете.

1. Блок схема на алгоритъма

Тук се поставя блок схема на описания алгоритъм.

1. Функция с която е реализиран алгоритъма

Тук се представя функцията, с която се реализира алгоритъма с нейния прототип. Описва се как трябва да се използва и какво се случва след нейната употреба.

1. Входни данни на функцията

Тук се представят входните данни на функцията: какви са, защо са избрани точно те, за какво ще се използват.

1. Изходни данни на функцията или данни, които се извеждат

Тук се представят изходните данни на функцията или какво ще се изведе на екрана от нея. Ако не връща резултат и не извежда данни, се описва какво е действието ѝ. Изяснява се защо е избран точно този тип и начин на извеждане на данните.

1. **Примерно действие на програмата**
2. Условие A
3. Снимка на изгледа с примерни входни данни
4. Входни данни

Въвеждане на опция от главното меню. При въвеждане на опция са възможни следните ситуации:

А) Въвеждане на опция < 1 – в този случай т. к. променливата съхраняваща опцията е беззнакова, ще се преизчисли стойността от дясната положителната граница на диапазона и ще се получи положително число

Б) Въвеждане на опция > последният номер на опция – в този случай, програмата ще поиска от потребителят да въведе отново опция, т. к. има предвидена валидация на въвеждането.

* В) Въвеждане на  опция в интервала [1 – предпоследната опция] – в този случай програмата изпълнява действията, заложени за изпълнение при дадената опция
* Г) Въвеждане на последната опция – в този случай това е опция „Изход“, което води до прекратяване на изпълнението на цялата програма.

1. Снимка на изгледа с примерни изходни данни

Моля, поставете снимката тук.

*Фиг. 7. Въвеждане на невалидна опция - Случай А) и Б)*

*Фиг. 8. Въвеждане на валидна опция на меню В)*

*Фиг. 9 Въвеждане на номера на последната опция – Изход – Случай Г)*

1. Условие B
2. Примерни входни данни

Въвежда се броят на студентите и след това се Въвеждат се данни за студент последователно. При въвеждането се спазват изискванията за въвеждане на данни спрямо типовете им! Предвидена е валидация на въвеждането на номер на специалност (от 1 до 4) и точки по дисциплина (от 0 до 120). При въвеждане на стойности извън допустимите се очаква ново въвеждане с коректна стойност.

1. Снимка на изгледа с примерни данни

1. Условие C
2. Примерни входни данни

Извеждането на студенти на екрана (отпечатване) не изисква въвеждане на входни данни. Предвидена е проверка за празен масив (списък) със студенти и извеждане на подходящо съобщение. При наличие на стойности се отпечатват студентите в таблица.

1. Снимка на изгледа с примерни данни

Моля, поставете снимката тук.

*Фиг. 10. Извеждане на празен списък със студенти.*

*Фиг. 11. Извеждане на списък със студенти в таблица.*

1. Условие D
2. Примерни входни данни

За изпълнението на справката за студенти по специалност, се въвежда предварително номер на специалността, с което се симулира избор на специалност, вместо въвеждане на текстов низ. Предвидена е валидация на избора от 1до 4 (за 4-те специалности). Ако няма въведени студени в масива (списъка) или няма студенти от дадена специалност се извеждат подходящи съобщения на екрана

1. Снимка на изгледа с примерни данни

*Фиг. 12. Избор на опция 3 но с липсващи данни*

*Фиг. 13. Избор на невалиден номер на специалност*

*Фиг. 14. Извеждане на данни за студенти от КСТ*

*Фиг. 15. Избор на специалност без студенти*

1. Условие E
2. Примерни входни данни.

При сортирането се подават като параметри масива със студенти и текущият брой въведени студенти (размера на масива). Студентите се сортират по азбучен ред на малките имена. Резултатът от сортирането е подреден оригиналния масив. Ако няма въведени студенти в масива, се извежда подходящо съобщение на екрана.

1. Снимка на изгледа с примерни входни данни

Моля, поставете снимката тук.

1. Снимка на изгледа с примерни изходни данни

Моля, поставете снимката тук.

1. Условие F
2. Снимка на изгледа с примерни входни данни

Моля, поставете снимката тук.

1. Снимка на изгледа с примерни изходни данни

Моля, поставете снимката тук.

1. Допълнение първо
2. Снимка на изгледа с примерни входни данни

Моля, поставете снимката тук.

1. Снимка на изгледа с примерни изходни данни

Моля, поставете снимката тук.

1. Допълнение второ
2. Снимка на изгледа с примерни входни данни

Моля, поставете снимката тук.

1. Снимка на изгледа с примерни изходни данни

Моля, поставете снимката тук.

1. Допълнение трето
2. Снимка на изгледа с примерни входни данни

Моля, поставете снимката тук.

1. Снимка на изгледа с примерни изходни данни

Моля, поставете снимката тук.

1. **ПРИЛОЖЕНИЕ 1**
2. Изходен код на програмата

#include <iostream>

#include <iomanip>

#include <string>

#include <fstream>

using namespace std;

//Example using constants using keyword "constant"

const unsigned short MAX\_STUDENTS = 100;

const unsigned short MAX\_SUBJECT\_POINTS = 120;

//constants for char limit length

const unsigned int MAX\_NAME\_SIZE = 15;

const unsigned int MAX\_SUBJECT\_NAME\_SIZE = 4;

const unsigned int MAX\_FAC\_NUMBER\_SIZE = 9;

const unsigned int MAX\_MAIN\_MENU\_OPTIONS = 7;

const unsigned int MAX\_SPECIALTIES = 4;

//Example using constants using prepcocessor directive "#define"

//

//specialty names

#define KST\_NAME "КСТ"

#define SIT\_NAME "СИТ"

#define II\_NAME "ИИ"

#define KS\_NAME "КС"

#define N\_A\_NAME "N.A." //define not available subject name

//table columns width

#define COLUMN\_1\_WIDTH  2

#define COLUMN\_2\_WIDTH  8

#define COLUMN\_3\_WIDTH 15

#define COLUMN\_4\_WIDTH 15

#define COLUMN\_5\_WIDTH 11

#define COLUMN\_6\_WIDTH  7

#define COLUMN\_7\_WIDTH  5

#define COLUMN\_8\_WIDTH 13

#define COLUMN\_SEPARATORS 9

//Auto calculation total table width

#define TABLE\_WIDTH  (COLUMN\_1\_WIDTH + COLUMN\_2\_WIDTH + COLUMN\_3\_WIDTH + COLUMN\_4\_WIDTH + COLUMN\_5\_WIDTH + COLUMN\_6\_WIDTH + COLUMN\_7\_WIDTH + COLUMN\_8\_WIDTH + COLUMN\_SEPARATORS)

//Enumeration menu options

enum MENU\_OPTIONS

{

MENU\_ADD = 1, //Menu ADD number 1

MENU\_PRINT, //Menu PRINT number 2

MENU\_REPORTS, //Menu REPORT number 3

MENU\_SORT, //Menu SORT number 4

MENU\_SAVE,

MENU\_LOAD

};

//Enumeration specialty numbers

enum SPECIALTIES

{

KST = 1, //KST number 1

SIT, //SIT number 2

II, //II number 3

KS //KS number 4

};

struct Subject

{

char name[MAX\_SUBJECT\_NAME\_SIZE];

unsigned short totalPoints;

unsigned short finalMark;

};

struct Student

{

char firstName[MAX\_NAME\_SIZE];

char lastName[MAX\_NAME\_SIZE];

char facultyNumber[MAX\_FAC\_NUMBER\_SIZE];

char specialty[MAX\_SUBJECT\_NAME\_SIZE];

unsigned short age;

Subject sbj1, sbj2, sbj3;

};

//function prototypes

unsigned int mainMenu();

void inputStudentData(Student& st);

//functions for print data in table

void printStudentsInTable(const Student listStudents[], const unsigned short listSize);

void printTableHeader();

void printTableFooter();

void printTableRow(const Student& st, const unsigned int recordNumber = 0);

void reportStudentsBySpecialty(const Student listStudents[], const unsigned short listSize, const char\* specialty);

void sortListStudentsByFirstName(Student listStudents[], const unsigned short listSize);

//help functions

void clearInputBuffer();

char\* specualtyNumberToString(const unsigned short specialtyNum);

int calcFinalMark(const unsigned short points);

string finalMarkToString(const unsigned int finalMark);

void swapStudents(Student& st1, Student& st2);

//functions to save/load data to/from text file

int saveStudentsToFile(const char filename[], const Student listStudents[], const unsigned short listSize);

int loadStudentsFromFile(const char filename[], Student listStudents[], unsigned short &listSize);

int main()

{

system("CHCP 1251");

unsigned int mainMenuOption;

unsigned short numStudents = 0;

unsigned short specialty;

Student listStudents[MAX\_STUDENTS]; //list of total students

do

{

mainMenuOption = mainMenu();

switch (mainMenuOption)

{

case MENU\_ADD:

do {

cout << "\nВъведете брой студенти: ";

cin >> numStudents;

} while (numStudents < 2 || numStudents > MAX\_STUDENTS);

for (int i = 0; i < numStudents; i++) {

cout << "\nВъведете данни за студент " << i+1 << ": " << endl;

inputStudentData(listStudents[i]);

}

break;

case MENU\_PRINT:

if (numStudents == 0) {

cout << "Няма въвeдени данни за студенти! Моля въведете данни!";

}

else {

cout << "Списък на студентите: " << endl;

printStudentsInTable(listStudents, numStudents);

}

break;

case MENU\_REPORTS:

if (numStudents == 0) {

cout << "Няма въвeдени данни за студенти! Моля въведете данни!";

}

else {

do {

cout << "Специалност: [1- КСТ, 2- СИТ, 3- ИИ, 4- КС]: ";

cin >> specialty;

} while (specialty > MAX\_SPECIALTIES);

reportStudentsBySpecialty(listStudents, numStudents, specualtyNumberToString(specialty));

}

break;

case MENU\_SORT:

if (numStudents == 0) {

cout << "Няма въвeдени данни за студенти! Моля въведете данни!";

}

else{

cout << "\nСортиране на студентите по име..." << endl;

                       sortListStudentsByFirstName(listStudents, numStudents);

cout << "Край. Данните след сортиране са: " << endl;

printStudentsInTable(listStudents, numStudents);

}

break;

case MENU\_SAVE:

if (numStudents == 0) {

cout << "Няма въвeдени данни за студенти! Моля въведете данни!";

}

else {

saveStudentsToFile("students.txt", listStudents, numStudents);

}

break;

case MENU\_LOAD:

            loadStudentsFromFile("students.txt", listStudents, numStudents);

break;

default: cout << "Изход от програмата!" << endl;

}

cout << endl; system("PAUSE"); system("CLS");

} while (mainMenuOption != MAX\_MAIN\_MENU\_OPTIONS);

return 0;

}

unsigned int mainMenu()

{

unsigned int menuCounter = 0;

unsigned int choise;

cout << "        МЕНЮ        " << endl;

cout << ++menuCounter << ". Въвеждане на студенти." << endl;

cout << ++menuCounter << ". Извеждане на студенти." << endl;

cout << ++menuCounter << ". Справка (Студенти от специалност)." << endl;

cout << ++menuCounter << ". Сортиране на студенти." << endl;

cout << ++menuCounter << ". Запазване във файл." << endl;

cout << ++menuCounter << ". Зареждане от файл." << endl;

cout << ++menuCounter << ". Изход." << endl;

do

{

cout << "\nВъведете Вашият избор: ";

cin >> choise;

} while (choise < 1 || choise > menuCounter);

return choise;

}

void inputStudentData(Student& st)

{

unsigned short specialtyNumber;

clearInputBuffer();

cout << "Първо име: \t";cin.getline(st.firstName, sizeof(st.firstName));

cout << "Фамилия: \t"; cin.getline(st.lastName , sizeof(st.lastName));

cout << "Години: \t"; cin >> st.age;

clearInputBuffer();

cout << "Фак. номер: \t"; cin.getline(st.facultyNumber, sizeof(st.facultyNumber));

do {

cout << "Специалност: [1- КСТ, 2- СИТ, 3- ИИ 4- КС]: ";

cin >> specialtyNumber;

} while (specialtyNumber > MAX\_SPECIALTIES);

//get the specialty name using convert function!

strcpy\_s(st.specialty, specualtyNumberToString(specialtyNumber));

clearInputBuffer();

cout << "Въвeдете информация за Дисциплина 1: " << endl;

cout << "Име на Дисциплина: "; cin.getline(st.sbj1.name, sizeof(st.sbj1.name));

do {

cout << "Точки по Дисциплина [0-" << MAX\_SUBJECT\_POINTS << "]: ";

cin >> st.sbj1.totalPoints;

} while (st.sbj1.totalPoints > MAX\_SUBJECT\_POINTS);

//auto calculation final mark using function

st.sbj1.finalMark = calcFinalMark(st.sbj1.totalPoints);

//ToDo: Add data for Subject 2 and Subject 3

//

//Not must be implement at this time!!! For self work!!!

}

void printStudentsInTable(const Student listStudents[], const unsigned short listSize)

{

printTableHeader();

for (int i = 0; i < listSize; i++) {

printTableRow(listStudents[i], i + 1);

}

printTableFooter();

}

void printTableHeader()

{

cout << endl << setw(TABLE\_WIDTH) << setfill('-') << "" << setfill(' ');

cout << endl << "|"

<< setw(COLUMN\_1\_WIDTH) << "No" << "|"

<< setw(COLUMN\_2\_WIDTH) << left << "ФАК. NO" << "|"

<< setw(COLUMN\_3\_WIDTH) << left << "ИМЕ" << "|"

<< setw(COLUMN\_4\_WIDTH) << left << "ФАМИЛИЯ" << "|"

<< setw(COLUMN\_5\_WIDTH) << right << "СПЕЦИАЛНОСТ" << "|"

<< setw(COLUMN\_6\_WIDTH) << left << "ПРЕДМЕТ" << "|"

<< setw(COLUMN\_7\_WIDTH) << right << "ТОЧКИ" << "|"

<< setw(COLUMN\_8\_WIDTH) << right << "ОЦЕНКА" << "|";

cout << endl << setw(TABLE\_WIDTH) << setfill('-') << "" << setfill(' ');

}

void printTableFooter()

{

cout << endl << setw(TABLE\_WIDTH) << setfill('-') << "" << setfill(' ');

}

void printTableRow(const Student& st, const unsigned int recordNumber)

{

cout << endl << "|"

<< setw(COLUMN\_1\_WIDTH) << setfill('0') << recordNumber << "|" << setfill(' ')

<< setw(COLUMN\_2\_WIDTH) << left << st.facultyNumber << "|"

<< setw(COLUMN\_3\_WIDTH) << left << st.firstName << "|"

<< setw(COLUMN\_4\_WIDTH) << left << st.lastName << "|"

<< setw(COLUMN\_5\_WIDTH) << right << st.specialty << "|"

<< setw(COLUMN\_6\_WIDTH) << left << st.sbj1.name << "|"

<< setw(COLUMN\_7\_WIDTH) << right << st.sbj1.totalPoints << "|"

<< setw(COLUMN\_8\_WIDTH) << right << finalMarkToString(st.sbj1.finalMark) << "|";

}

void reportStudentsBySpecialty(const Student listStudents[], const unsigned short listSize, const char\* specialty)

{

unsigned int numStudents = 0;

//temporary list of students

Student listStudentsBySpecialty[MAX\_STUDENTS];

for (int i = 0; i < listSize; i++){

if (strcmp(listStudents[i].specialty, specialty) == 0) {

listStudentsBySpecialty[numStudents++] = listStudents[i];

}

}

//check for founded students

if (numStudents == 0) {

cout << "Няма студенти от специалност " << specialty;

}

else {

cout << "Брой студенти от специалност " << specialty << ": " << numStudents << endl;

cout << "Данни за студентите:" << endl;

printStudentsInTable(listStudentsBySpecialty, numStudents);

}

}

void sortListStudentsByFirstName(Student listStudents[], const unsigned short listSize)

{

int max;

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

max = i;

for (int j = listSize - 1; j > i; j--) {

if (strcmp(listStudents[max].firstName, listStudents[j].firstName) > 0) {

max = j;

}

}

if (max != i) {

swapStudents(listStudents[max], listStudents[i]);

}

}

}

int saveStudentsToFile(const char fileName[], const Student listStudents[], const unsigned short listSize)

{

ofstream  fp;

fp.open(fileName);

if (!fp) {

cout << endl << "Файлът не може да бъде отворен!";

return -1;

}

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

{

fp << listStudents[i].facultyNumber << "\n";

fp << listStudents[i].firstName << "\n";

fp << listStudents[i].lastName << "\n";

fp << listStudents[i].specialty << "\n";

fp << listStudents[i].sbj1.name << "\n";

fp << listStudents[i].sbj1.totalPoints << "\n";

fp << listStudents[i].sbj1.finalMark << "\n";

}

fp.close();    cout << "Готово";

return 0;

}

int loadStudentsFromFile(const char fileName[], Student listStudents[], unsigned short& listSize)

{

ifstream  fp;

Student st;

unsigned int i = 0;

//Read file

fp.open(fileName);

if (!fp) {

cout << endl << "Файлът не може да бъде отворен!";

return -1;

}

while (!fp.eof())

{

fp.getline(st.facultyNumber, sizeof(st.facultyNumber), '\n');

fp.getline(st.firstName, sizeof(st.firstName), '\n');

fp.getline(st.lastName, sizeof(st.lastName), '\n');

fp.getline(st.specialty, sizeof(st.specialty), '\n');

fp.getline(st.sbj1.name, sizeof(st.sbj1.name), '\n');

fp >> st.sbj1.totalPoints;

fp >> st.sbj1.finalMark; fp.ignore();

listStudents[i] = st;

i++;

}

listSize = i;

fp.close(); cout << "Готово";

return 0;

}

void clearInputBuffer()

{

cin.ignore(1i64, '\n');  cin.clear();

}

char\* specualtyNumberToString(const unsigned short specialtyNum)

{

switch (specialtyNum)

{

case 1: return (char\*)KST\_NAME; break;

case 2: return (char\*)SIT\_NAME; break;

case 3: return (char\*)II\_NAME; break;

case 4: return (char\*)KS\_NAME; break;

default: return (char \*)N\_A\_NAME;

}

}

int calcFinalMark(const unsigned short points)

{

if (points > 120) {

return -1; //error code for invalid points

}

//return mark value indipending points range

if (points < 50) {

return 2;

}

else if (points < 62) {

return 3;

}

else if (points < 74) {

return 4;

}

else if (points < 89) {

return 5;

}

else {

return 6;

}

}

string finalMarkToString(const unsigned int finalMark)

{

string finalMarkStr;

switch (finalMark)

{

case 2: finalMarkStr = "Слаб (2)"; break;

case 3: finalMarkStr = "Среден (3)"; break;

case 4: finalMarkStr = "Добър (4)"; break;

case 5: finalMarkStr = "Мн. добър (5)"; break;

case 6: finalMarkStr = "Отличен (6)"; break;

default: finalMarkStr = "Нев. о-ка"; //example: return invalid information

}

return finalMarkStr;

}

void swapStudents(Student& st1, Student& st2)

{

Student st; st = st1; st1 = st2; st2 = st;

}