|  |  |
| --- | --- |
| C:\Documents and Settings\admin\Рабочий стол\МИРЭА_ЭМБЛЕМА_приказ.JPG | |
| МИНОБРАНАУКИ РОССИИ | |
| Федеральное государственное бюджетное образовательное учреждение  высшего образования  «**МОСКОВСКИЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»**  **МИРЭА** | |
| Институт информационных технологий  Кафедра математического обеспечения и стандартизации информационных технологий | |
| **Отчет по лабораторной работе №4**  **Тема работы «СТРУКТУРЫ»** | |
| Выполнил: |  |
| Студент группы ИКБО-07-17  Отчет принял:  Преподаватель кафедры МОСИТ | Акжигитов Р. Р.  Миронов Антон Николаевич |
| Москва  2018 | |

Оглавление

[Задание 1 3](#_Toc510978310)

[Условия задания 3](#_Toc510978311)

[Декомпозиция 3](#_Toc510978312)

[Определение функций 3](#_Toc510978313)

[Реализация функций 3](#_Toc510978314)

[Функции ввода\вывода массива\значений 4](#_Toc510978315)

[Кодирование алгоритма программы 4](#_Toc510978316)

[Многомодульный код 7](#_Toc510978317)

[Вывод 11](#_Toc510978318)

# Задание 1

# Условия задания

Сведения о студенте:

* Номер зачетной книжки
* Фамилия
* Имя
* Дата рождения
* Номер телефона
* Дата поступления
* Признак удаления записи

Сортировка: метод прямого выбора. Отсортировать массив по дате рождения (перевести ее сначала в число, исчисляя начало 1900 годом).

Найти и удалить всех окончивших институт (т.е. более срок обучения более 6 лет).

# Декомпозиция

Создать 2 базовые структуры: Студент и Дата.

В сортировку передавать численное представление дня, полученное из поля год.

В структуре студент есть поле для определения, окончил ли он институт или нет, которое высчитывается сразу после инициализации датами поступления. В основной программе по этому полю можно проверять нужно ли его удалять.

# Определение функций

// сортировка выбором с функцией-селектором

void selectionSort(Student\* st, int size, int(\*selector)(Student st))

// получение массива "просроченных студентов"

Student\* getOverdued(Student\* st, int& size, Student\*& nonOverdued, int& sizeNonOverdued)

# Реализация функций

// сортировка выбором с функцией-селектором

void selectionSort(Student\* st, int size, int(\*selector)(Student st))

{

int min; // для поиска минимального элемента и для обмена

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

min = i; // запоминаем индекс текущего элемента

// ищем минимальный элемент, чтобы поместить на место i-ого

for (int j = i + 1; j < size; j++) { // для остальных элементов после i-ого

if (selector(st[j]) < selector(st[min])) // если элемент меньше минимального,

min = j; // запоминаем его индекс в min

}

swap(st[i], st[min]); // меняем местами i-ый и минимальный элементы

}

}

// получение массива "просроченных студентов"

Student\* getOverdued(Student\* st, int& size, Student\*& nonOverdued, int& sizeNonOverdued) {

Student\* overdued = new Student[size];

nonOverdued = new Student[size];

int cx = 0, cx1 = 0;

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

if (st[i].deleteFlag > 0) {

overdued[cx++] = st[i];

}

else {

nonOverdued[cx1++] = st[i];

}

}

size = cx;

sizeNonOverdued = cx1;

return overdued;

}

# Функции ввода\вывода массива\значений

// ввод даты с проверкой на базовые интервалы

void inputDate(Date& st)

// ввод числа с проверкой на диапазон

int input(int min, int max)

// ввод всех студентов

void getStruct(Student\*& st, int size)

// вывод всей структуры

void printStruct(Student\* st, int size)

# Кодирование алгоритма программы

// laba4.cpp : Defines the entry point for the console application.

//

#include "stdafx.h"

#include "iostream"

#include "stdlib.h"

#include "stdio.h"

#include <string>

using namespace std;

// структура Дата для отображения дат

struct Date {

int day : 6;

int month : 5;

int year;

};

// структура Студент для отображения множества свойств ему принадлежащих

struct Student {

string surname, name, recordNumber, phoneNumber;

Date birthDate, enterDate, todayDate;

int overdue = 6;

//(bool)(\*deleteF)(Student st) = [](Student st) { return st.todayDate.year - st.enterDate.year > st.overdue; };

int deleteFlag = todayDate.year - enterDate.year - overdue;

};

// сортировка выбором с функцией-селектором

void selectionSort(Student\* st, int size, int(\*selector)(Student st))

{

int min; // для поиска минимального элемента и для обмена

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

min = i; // запоминаем индекс текущего элемента

// ищем минимальный элемент, чтобы поместить на место i-ого

for (int j = i + 1; j < size; j++) { // для остальных элементов после i-ого

if (selector(st[j]) < selector(st[min])) // если элемент меньше минимального,

min = j; // запоминаем его индекс в min

}

swap(st[i], st[min]); // меняем местами i-ый и минимальный элементы

}

}

// проверка строки на число

bool isNumber(string str) {

if (str == "") return 0;

for (int i = 0; i < str.length(); i++)

if (str[i] - '0' > 9 || str[i] - '0' < 0) return 0;

return 1;

}

// ввод числа с проверкой на диапазон

int input(int min, int max) {

string temp;

bool flag = 0;

while (!isNumber(temp) || stoi(temp) > max || stoi(temp) < min) {

if (flag) cout << "Please input integer [" << min << ".." << max << "]: ";

flag = 1;

getline(cin, temp);

}

return stoi(temp);

}

// преобразование строки в число

bool strToInt(char\* str, int& num) {

try { num = stoi(str); return true; }

catch (exception e) {return false;}

return false;

}

// ввод даты с проверкой на базовые интервалы

void inputDate(Date& st) {

char temp[256];

int cx;

bool flag = 0;

bool secFlag = 0;

while (true) {

cx = 0;

flag = 0;

if (secFlag) cout << "Please enter right date format (dd mm yyyy): ";

secFlag = 1;

cin.getline(temp, 256);

char\* ptr = strtok(temp, " ");

while (ptr != NULL) {

try {

int num = stoi(ptr);

if (cx == 0)

if (num >= 1 && num <= 31)

st.day = num;

else flag = 1;

if (cx == 1)

if (num >= 1 && num <= 12)

st.month = num;

else flag = 1;

if (cx == 2)

if (num >= 1900 && num <= 3000)

st.year = num;

else flag = 1;

cx++;

}

catch (exception e) {

flag = 1;

}

if (flag) break;

ptr = strtok(NULL, " ");

}

if (flag == 0) return;

}

}

// вывод всей структуры

void printStruct(Student\* st, int size) {

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

cout << "Student: " << st[i].name << " " << st[i].surname << endl;

cout << "Birth's date: " << st[i].birthDate.day << "." << st[i].birthDate.month << "." << st[i].birthDate.year << endl;

cout << "Enter's date: " << st[i].enterDate.day << "." << st[i].enterDate.month << "." << st[i].enterDate.year << endl;

cout << "Today's date: " << st[i].todayDate.day << "." << st[i].todayDate.month << "." << st[i].todayDate.year << endl;

cout << "Overdue time: " << st[i].overdue << " years" << endl << endl;

}

}

// ввод всех студентов

void getStruct(Student\*& st, int size) {

st = new Student[size];

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

cout << "Student's name: "; cin >> st[i].name;

cout << "Student's surname: "; cin >> st[i].surname;

cout << "Student's order number: "; cin >> st[i].recordNumber;

cout << "Student's phone number: "; cin >> st[i].phoneNumber;

getchar();

cout << "Student's birth date (dd mm yyyy): ";

/\*st[i].birthDate.day = input(1, 31);

st[i].birthDate.month = input(1, 12);

st[i].birthDate.year = input(1900, 3000);\*/

inputDate(st[i].birthDate);

cout << "Student's enter date (dd mm yyyy): ";

inputDate(st[i].enterDate);

cout << "Student's today date (dd mm yyyy): ";

inputDate(st[i].todayDate);

cout << "Student's overdue (years): ";

st[i].overdue = input(1, 20);

}

}

// получение массива "просроченных студентов"

Student\* getOverdued(Student\* st, int& size, Student\*& nonOverdued, int& sizeNonOverdued) {

Student\* overdued = new Student[size];

nonOverdued = new Student[size];

int cx = 0, cx1 = 0;

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

if (st[i].deleteFlag > 0) {

overdued[cx++] = st[i];

}

else {

nonOverdued[cx1++] = st[i];

}

}

size = cx;

sizeNonOverdued = cx1;

return overdued;

}

int main()

{

//Student\* arr = new Student[3];

//Student vasya = { "Pupkin", "Vasya", "22813378", "78005553535", { 11, 12, 1999 }, { 28, 8, 2017 }, { 31, 3, 2020 }, 2 };

//cout << vasya.deleteFlag << endl;

//Student petya = { "Popov", "Petya", "22813378", "78005553535", { 20, 8, 2000 }, { 28, 8, 2017 }, { 31, 3, 2028 }};

//Student masha = { "Ivanova", "Maria", "22813378", "78005553535", { 3, 2, 1999 }, { 28, 8, 2017 }, { 31, 3, 2020 }};

//Student arr[3] = { vasya, petya, masha };

Student\* arr = 0;

cout << "Input number of students: ";

int size = input(1, 100);

getStruct(arr, size);

//cout << vasya.surname << " " << vasya.deleteFlag << " " << (vasya.overdue = 6) << " " << vasya.deleteFlag << endl;

printStruct(arr, size);

selectionSort(arr, size, [](Student st) { return st.birthDate.year \* 365 + st.birthDate.month \* 30 + st.birthDate.day; });

printStruct(arr, size);

int size1;

Student\* lastStudents = 0;

//getOverdued(arr, size, lastStudents, size1);

cout << "Overdued..." << endl;

Student\* temp = getOverdued(arr, size, lastStudents, size1);

printStruct(temp, size);

cout << "Non-overdued..." << endl;

printStruct(lastStudents, size1);

system("pause");

return 0;

}

# Многомодульный код

unitDef.h

#pragma once

#define \_CRT\_SECURE\_NO\_WARNINGS

#include "iostream"

#include "stdlib.h"

#include "stdio.h"

#include <string>

using namespace std;

// структура Дата для отображения дат

struct Date {

int day : 6;

int month : 5;

int year;

};

// структура Студент для отображения множества свойств ему принадлежащих

struct Student {

string surname, name, recordNumber, phoneNumber;

Date birthDate, enterDate, todayDate;

int overdue = 6;

//(bool)(\*deleteF)(Student st) = [](Student st) { return st.todayDate.year - st.enterDate.year > st.overdue; };

int deleteFlag = todayDate.year - enterDate.year - overdue;

};

unitIO.h

#pragma once

#ifndef Student

#include "unitDef.h"

#endif

// проверка строки на число

bool isNumber(string str) {

if (str == "") return 0;

for (int i = 0; i < str.length(); i++)

if (str[i] - '0' > 9 || str[i] - '0' < 0) return 0;

return 1;

}

// ввод числа с проверкой на диапазон

int input(int min, int max) {

string temp;

bool flag = 0;

while (!isNumber(temp) || stoi(temp) > max || stoi(temp) < min) {

if (flag) cout << "Please input integer [" << min << ".." << max << "]: ";

flag = 1;

getline(cin, temp);

}

return stoi(temp);

}

// преобразование строки в число

bool strToInt(char\* str, int& num) {

try { num = stoi(str); return true; }

catch (exception e) { return false; }

return false;

}

// ввод даты с проверкой на базовые интервалы

void inputDate(Date& st) {

char temp[256];

int cx;

bool flag = 0;

bool secFlag = 0;

while (true) {

cx = 0;

flag = 0;

if (secFlag) cout << "Please enter right date format (dd mm yyyy): ";

secFlag = 1;

cin.getline(temp, 256);

char\* ptr = strtok(temp, " ");

while (ptr != NULL) {

try {

int num = stoi(ptr);

if (cx == 0)

if (num >= 1 && num <= 31)

st.day = num;

else flag = 1;

if (cx == 1)

if (num >= 1 && num <= 12)

st.month = num;

else flag = 1;

if (cx == 2)

if (num >= 1900 && num <= 3000)

st.year = num;

else flag = 1;

cx++;

}

catch (exception e) {

flag = 1;

}

if (flag) break;

ptr = strtok(NULL, " ");

}

if (flag == 0) return;

}

}

// вывод всей структуры

void printStruct(Student\* st, int size) {

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

cout << "Student: " << st[i].name << " " << st[i].surname << endl;

cout << "Birth's date: " << st[i].birthDate.day << "." << st[i].birthDate.month << "." << st[i].birthDate.year << endl;

cout << "Enter's date: " << st[i].enterDate.day << "." << st[i].enterDate.month << "." << st[i].enterDate.year << endl;

cout << "Today's date: " << st[i].todayDate.day << "." << st[i].todayDate.month << "." << st[i].todayDate.year << endl;

cout << "Overdue time: " << st[i].overdue << " years" << endl << endl;

}

}

// ввод всех студентов

void getStruct(Student\*& st, int size) {

st = new Student[size];

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

cout << "Student's name: "; cin >> st[i].name;

cout << "Student's surname: "; cin >> st[i].surname;

cout << "Student's order number: "; cin >> st[i].recordNumber;

cout << "Student's phone number: "; cin >> st[i].phoneNumber;

getchar();

cout << "Student's birth date (dd mm yyyy): ";

/\*st[i].birthDate.day = input(1, 31);

st[i].birthDate.month = input(1, 12);

st[i].birthDate.year = input(1900, 3000);\*/

inputDate(st[i].birthDate);

cout << "Student's enter date (dd mm yyyy): ";

inputDate(st[i].enterDate);

cout << "Student's today date (dd mm yyyy): ";

inputDate(st[i].todayDate);

cout << "Student's overdue (years): ";

st[i].overdue = input(1, 20);

}

}

unitOperations.h

#pragma once

#include "unitDef.h"

// сортировка выбором с функцией-селектором

void selectionSort(Student\* st, int size, int(\*selector)(Student st))

{

int min; // для поиска минимального элемента и для обмена

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

min = i; // запоминаем индекс текущего элемента

// ищем минимальный элемент, чтобы поместить на место i-ого

for (int j = i + 1; j < size; j++) { // для остальных элементов после i-ого

if (selector(st[j]) < selector(st[min])) // если элемент меньше минимального,

min = j; // запоминаем его индекс в min

}

swap(st[i], st[min]); // меняем местами i-ый и минимальный элементы

}

}

// получение массива "просроченных студентов"

Student\* getOverdued(Student\* st, int& size, Student\*& nonOverdued, int& sizeNonOverdued) {

Student\* overdued = new Student[size];

nonOverdued = new Student[size];

int cx = 0, cx1 = 0;

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

if (st[i].deleteFlag > 0) {

overdued[cx++] = st[i];

}

else {

nonOverdued[cx1++] = st[i];

}

}

size = cx;

sizeNonOverdued = cx1;

return overdued;

}

main.cpp

#include "unitDef.h"

#include "unitOperations.h"

#include "unitIO.h"

int main()

{

//Student\* arr = new Student[3];

//Student vasya = { "Pupkin", "Vasya", "22813378", "78005553535", { 11, 12, 1999 }, { 28, 8, 2017 }, { 31, 3, 2020 }, 2 };

//cout << vasya.deleteFlag << endl;

//Student petya = { "Popov", "Petya", "22813378", "78005553535", { 20, 8, 2000 }, { 28, 8, 2017 }, { 31, 3, 2028 }};

//Student masha = { "Ivanova", "Maria", "22813378", "78005553535", { 3, 2, 1999 }, { 28, 8, 2017 }, { 31, 3, 2020 }};

//Student arr[3] = { vasya, petya, masha };

Student\* arr = 0;

cout << "Input number of students: ";

int size = input(1, 100);

getStruct(arr, size);

//cout << vasya.surname << " " << vasya.deleteFlag << " " << (vasya.overdue = 6) << " " << vasya.deleteFlag << endl;

printStruct(arr, size);

selectionSort(arr, size, [](Student st) { return st.birthDate.year \* 365 + st.birthDate.month \* 30 + st.birthDate.day; });

printStruct(arr, size);

int size1;

Student\* lastStudents = 0;

//getOverdued(arr, size, lastStudents, size1);

cout << "Overdued..." << endl;

Student\* temp = getOverdued(arr, size, lastStudents, size1);

printStruct(temp, size);

cout << "Non-overdued..." << endl;

printStruct(lastStudents, size1);

system("pause");

return 0;

}

# Вывод

В этой работе пришлось столкнуться со структурами, выравниванием, заполнением полей, узнать о способах обращения к полям (через точку и через ->, явное/неявное приведение типов). Как хранится структура в памяти, и сколько выделяется под нее.