**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по курсовой работе**

**по дисциплине «Программирование» Тема:** Типы данных, определяемые пользователем. Структуры. Линейные структуры данных. Динамические массивы и двусвязные списки.

|  |  |  |
| --- | --- | --- |
| Студентка гр. 0324 |  | Сотина Е.А. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

## Цель работы.

Применить полученные знания о динамических массивах и двусвязных списках, организации структур;

**Основные теоретические положения.**

Структуры представляют собой группы связанных между собой, как правило, разнотипных переменных, объединенных в единый объект, в отличие от массива, все элементы которого однотипны. В языке C++ структура является видом класса и обладает всеми его свойствами. Чаще всего ограничиваются тем, как структуры представлены в языке С:

struct [имя\_типа] {

тип\_1 элемент\_1;

тип \_2 элемент\_2;

…

тип\_k элемент\_k;

} [ список\_описателей ];

Описание структуры начинается ключевым словом struct. Каждая входящая в структуру переменная называется членом (полем, элементом) структуры и описывается типом данных и именем. Поля структуры могут быть любого типа данных. Их количество не лимитировано.

Вся эта конструкция является инструкцией языка программирования, поэтому после нее всегда должен ставиться символ ‘;’.

При описании структуры память для размещения данных не выделяется. Работать с описанной структурой можно только после того, как будет определена переменная (переменные) этого типа данных, только при этом компилятор выделит необходимую память.

Для инициализации структуры значения ее элементов перечисляют в фигурных скобках в порядке их описания:

struct complex{

float real, im;

} data [2][2] = {

{{1,1}, {2,2}},

{{3,3}, {4,4}}

};

Все поля структурных переменных располагаются в непрерывной области памяти одно за другим. Общий объем памяти, занимаемый структурой, равен сумме размеров всех полей структуры. Для определения размера структуры следует использовать инструкцию sizeof().

Для того чтобы записать данные в структурную переменную, необходимо каждому полю структуры присвоить определенное значение. Для этого необходимо использовать оператор ‘’ («точка»):

struct Stack { // Cтек

float arr[100];

short topIndex;

};

…

Stack stack; // Объявляем переменную типа Stack

Stack.arr[0] = 1;

…

При доступе к определенному полю его следует рассматривать как обычную переменную, тип данных которой соответствует типу этого поля. Поля структур могут участвовать в качестве операндов любых выражений, допускающих использование операндов соответствующего типа данных.

Копирование данных из одной структурной переменной в другую осуществляется простой операцией присваивания, независимо от количества полей и размера структуры (это можно делать только в том случае, когда обе переменные одного и того же типа).

В программировании очень часто используются такие конструкции, как массивы структур. Например, сведения о студентах некоторой учебной группы можно хранить в массиве студентов:

t\_Student Gruppa\_N [30];

Был определен 30-элементный массив, каждый элемент которого предназначен для хранения данных одного студента. Получение доступа к данным некоторого студента из группы *N* осуществляется обычной индексацией переменной массива. Поскольку поля структуры могут быть любого типа данных, то они в свою очередь могут быть другой структурой или массивом других структур:

struct Stud

{

char FN[100];

short listNumber;

};

struct Group

{

int groupNumber;

short students;

Stud stud[30];

};

Но в структуре поля нельзя использовать элемент, тип которого совпадает с типом самой структуры, так как рекурсивное использование структур запрещено.

Любая структурная переменная занимает в памяти определенное положение, характеризующееся конкретным адресом. Для работы с адресами структурных переменных (как и для простых переменных) можно использовать указатели. Указатели на структурные переменные определяются точно так же, как и для обычных переменных. Разыменование указателя (обращение к данным по адресу, хранящемуся в указателе) осуществляется также обычным образом.

Через указатели можно работать с отдельными полями структур. Для доступа к полю структуры через указатель используется оператор ‘’ («стрелка»), а не «точка».

Структуры можно использовать в качестве параметров функций, как и обычные переменные. Для структур поддерживаются все три механизма передачи данных: по значению, через указатели и по ссылке.

Передачу структур в функции по значению необходимо использовать аккуратно:

void WriteStudent ( t\_Student S )

{

cout << "Фамилия: " << S.Fam << endl;

cout << "Имя: " << S.Name << endl;

cout << "Год рождения: " << S.Year << endl;

if ( S.Sex )

cout << "Пол: " << "М\n";

else

cout << "Пол: " << "Ж\n";

cout << "Средний балл: " << S.Grade << endl;

}

Вызов такой функции сопровождается дополнительным расходом памяти для создания локальной переменной *S*и дополнительными затратами времени на физическое копирование данных из аргумента в параметр *S*. Учитывая то, что объем структур может быть очень большим, эти дополнительные затраты вычислительных ресурсов могут быть чрезмерными.

Предпочтительно использование передачи структуры по указателю или ссылке:

void WriteStudent ( t\_Student \*S )

{

cout << "Фамилия: " << S -> Fam << endl;

cout << "Имя: " << S -> Name << endl;

cout << "Год рождения: " << S -> Year << endl;

if ( S -> Sex )

cout << "Пол: " << "М\n";

else

cout << "Пол: " << "Ж\n";

cout << "Средний балл: " << S -> Grade << endl;

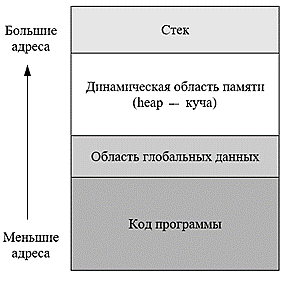
}

Фактической передачи данных в функцию не осуществляется. Дополнительные затраты памяти для создания локальной переменной небольшие – это адрес памяти (4 байта, независимо от размера самой структуры). Вызов такой функции будет происходить быстрее, а расход памяти будет существенно меньше, чем при передаче данных по значению.

Передача по ссылке по эффективности эквивалентна передаче данных через указатель. Однако, поскольку при передаче данных по ссылке все адресные преобразования берет на себя компилятор, существенно упрощается программирование действий со структурами. При использовании ссылочных параметров структурных типов доступ к членам структуры осуществляется обычным способом – с помощью оператора «точка».

Недостатком этих способов является то, что случайные изменения значений полей структуры внутри функции отразятся на значении аргумента после окончания работы функции. Если необходимо предотвратить изменения переданных по адресу аргументов, можно при определении соответствующего параметра объявить его константой (использовать спецификатор const).

Схема распределения памяти под программу показана на рис. 1.



*Рис. 1.* Схема распределения памяти под программу

Область кода программы предназначена для хранения инструкций функций программы, обеспечивающих обработку данных.

Данные в программе представляются переменными и константами.

Для хранения глобальных данных предназначена область глобальных данных.

Стек программы используется при вызове функций для передачи параметров и хранения локальных данных.

Распределение памяти для хранения всех обычных переменных осуществляется компилятором, адреса и объемы соответствующих участков памяти (в области глобальных данных) жестко закреплены за этими переменными на все время работы программы и изменены быть не могут.

Однако во многих задачах невозможно заранее предсказать, сколько места (количество переменных, объемы массивов и т. д.) потребуется для решения задачи – это так называемые задачи с неопределенной размерностью.

Решить эту проблему можно лишь в том случае, если иметь механизм, позволяющий создавать новые объекты по мере возникновения необходимости в этих объектах или изменять объемы памяти, выделенные под эти объекты (например, объемы массивов).

Между областью глобальных данных и стеком располагается так называемая динамическая область памяти, которую и можно использовать в процессе работы программы для реализации механизма динамического управления памятью.

Для того чтобы создать в динамической области некоторый объект, необходима одна обычная переменная-указатель (не динамическая переменная). Сколько таких объектов понадобится для одновременной обработки, столько необходимо иметь обычных переменных-указателей. Таким образом, проблема задач неопределенной размерности созданием одиночных динамических объектов решена быть не может.

Решить эту проблему поможет возможность создавать в динамической области памяти массивы объектов с таким количеством элементов, которое необходимо в данный момент работы программы, т. е. создание динамических массивов. Действительно, для представления массива требуется всего одна переменная-указатель, а в самом массиве, на который ссылается этот указатель, может быть столько элементов, сколько требуется в данный момент времени.

Для создания одномерного динамического массива, элементами которого являются, например, действительные числа, используется следующий синтаксис инструкции new (стиль С++):

double \*Arr = new double [100];

Освободить динамическую область от этого массива можно с помощью инструкции delete:

delete [] Arr;

После этого занятый участок памяти будет возвращен в список свободной памяти и может быть повторно использован для размещения других динамических объектов.

Язык C++ поддерживает и «старый», заимствованный от языка C, стиль работы с динамической областью. Довольно часто бывает полезно использовать именно этот механизм управления динамической памятью.

В языке C отсутствуют инструкции new и delete. Вместо них для управления динамической памятью используются библиотечные функции:

// Блок прототипов функций

{

void \*malloc (size);

void \*calloc(num, size);

void free( void \*memblock);

void \*realloc( void \*memblock, size);

}

Функция malloc выделяет в динамической области size байт памяти и возвращает адрес этого участка в виде указателя (void \*).

Поскольку возвращаемый указатель не привязан ни к какому типу данных, при работе с ним потребуется явное приведение типов данных (см. пример далее).

Функция calloc выделяет в динамической области size \* num байт памяти и возвращает адрес этого участка в виде указателя (void \*).

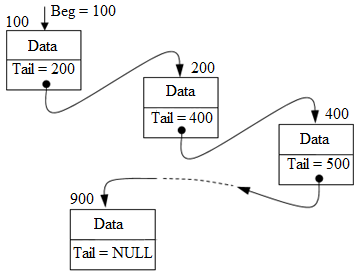
Функция free освобождает участок динамической памяти по адресу memblock и возвращает его в список свободной памяти для повторного использования.

Функция realloc позволяет изменить размер (уменьшить или увеличить) ранее выделенной по адресу memblock памяти, установив новый размер выделенного участка равным size байт. При увеличении размера выделенного участка данные, которые хранились в старом участке, копируются в новый участок памяти. При уменьшении объема выделенного участка данные, которые хранились в нем, усекаются до нового размера. Функция возвращает указатель на область памяти нового размера.

Работа с одномерным динамическим массивом осуществляется так же, как и с обычным. При этом стиль использования динамических массивов С имеет весомое преимущество над С++, которое заключается в изменении размерности массива. Дело в том, что в C++ нет функций увеличения размерности. Увеличить размер массива можно, создав новый динамический массив нужной размерности, скопировав данные из старого массива в новый и освободив память от старого массива.

Одномерный однонаправленный список представляет собой совокупность отдельных элементов, каждый из которых содержит две части – информационную () и адресную ().

Информационная часть предназначена для хранения полезных данных и может иметь практически любой тип. Адресная часть каждого элемента содержит адрес следующего элемента списка. Схематическое изображение такого списка представлено на рис. 2.



*Рис. 2.* Схематическое изображение односвязного списка

Для работы со списком достаточно знать только адрес его первого элемента (). Зная адрес первого элемента списка, можно последовательно получить доступ к любому другому его элементу.

Поскольку каждый элемент списка должен иметь две части, логичнее всего представить его в виде следующей структуры:

struct list

{

int data;

list \*tail;

};

Типовыми операциями при работе со списками являются:

1. создание списка;
2. освобождение памяти от списка (удаление списка);
3. доступ к заданному элементу списка для манипуляций с его информационной частью;
4. добавление нового элемента к списку;
5. удаление элемента из списка;
6. перестановка элемента списка на новую позицию внутри списка.

Достоинством подобных структур является простота добавления, удаления и перестановки элементов списка, которые осуществляются путем манипуляций с адресными частями без перезаписи всего списка.

Одним из недостатков односвязных списков является то, что узел (элемент списка) имеет указатель только на следующий элемент. Вернуться из текущего элемента к предыдущему явным способом невозможно.

Каждый узел двусвязного (двунаправленного) линейного списка содержит два поля указателей – на следующий и на предыдущий узлы. Указатель на предыдущий узел корня списка содержит нулевое значение. Указатель последнего узла также содержит нулевое значение.

Поскольку каждый элемент списка должен иметь три части, логичнее всего представить его в виде следующей структуры:

struct list

{

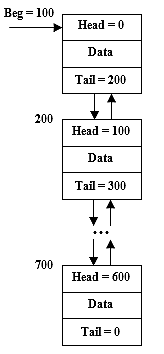
int data;

list \*head;

list \*tail;

};

На рис. 3 показано схематическое представление двусвязного списка. Поле содержит адрес предыдущего элемента, поле содержит адрес следующего элемента списка. Такая организация списка позволяет перемещаться по его элементам в двух направлениях.



*Рис. 3.* Схематическое изображение двусвязного списка

Основные действия, производимые над узлами двусвязного линейного списка (ДЛС):

1. инициализация списка;
2. добавление узла в список;
3. удаление узла из списка;
4. удаление корня списка;
5. вывод элементов списка;
6. вывод элементов списка в обратном порядке;
7. взаимообмен двух узлов списка.

Порядок действия очень похож на односвязный линейный список, но необходимо учитывать, что в двусвязном списке имеется два указателя: на следующий и предыдущий элементы.

**Постановка задачи.**

Необходимо создать массив структур, содержащий информацию о студентах: ФИО, пол, номер группы, номер в списке группы, оценки за прошедшую сессию (всего 3 экзамена и 5 дифференцированных зачетов), форма обучения, отметка времени о внесении или изменении данных. Ввод и изменение данных обо всех студентах должен осуществляться в файл students.

Написать функции, реализующие операции со структурами (ввод данных с клавиатуры):

1.   Создание новой записи о студенте.

2.   Внесение изменений в уже имеющуюся запись.

3.   Вывод всех данных о студентах.

4.   Вывод информации обо всех студентах группы N. N – инициализируется пользователем.

5.   Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию.

6.   Вывод количества студентов мужского и женского пола.

7.   Определение количества студентов, которые будут получать стипендию (стипендия начисляется, если у студента нет троек и очная форма обучения).

8.   Вывод данных о студентах, которые не получают стипендию; учатся только на «хорошо» и «отлично»; учатся только на «отлично»;

9.   Вывод данных о студентах, имеющих номер в списке – k.

10.   Вывод всех записей, сделанных в день, который введет пользователь. Вывод всех записей, сделанных после полудня. Вывод всех записей, сделанных до полудня.

Необходимо реализовать программу, которая выполняет следующие действия.

1. Формирование целочисленного одномерного массива размерности N, где:

a) пользователь вводит количество элементов в массиве, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы массива, N определяется автоматически по количеству введенных элементов;

в) массив считывается с файла, N определяется как количество элементов массива в файле.

2. Определение скорости создания динамического массива п. 1.

3. Вставка, удаление и получение элемента массива. Удаление и получение элемента необходимо реализовать по индексу и по значению.

4. Определение скорости вставки, удаления и получения элемента массива п. 3.

5. Формирование двусвязного списка размерности N, где:

a) пользователь вводит количество элементов в списке, который будет автоматически заполняться случайными числами (0 до 99);

б) пользователь вводит в консоль элементы списка, N определяется автоматически по количеству введенных элементов;

в) список считывается с файла, N определяется как количество элементов списка в файле.

6. Определение скорости создания двусвязного списка п. 5.

7. Вставка, удаление и получение элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.

8. Определение скорости вставки, удаление и получения элемента двусвязного списка п. 7.

Должна быть возможность запуска каждого пункта многократно, если есть возможность (если в списке/массиве нет элементов, то нельзя ничего удалить и об этом нужно сообщить пользователю). Необходимо сравнить результаты. Для этого пункты 1–4 и 5–8 должны принимать одинаковые значения.

**Выполнение работы.**

Код программы представлен в приложении А.

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Меню | |
| При запуске программы перед пользователем появляется окно с главным меню, где он может перейти к интересующей его теме. | Главное меню:  Меню для работы со записями о студентах:    Проверка на ввод символов, которые не входят в диапазон выбора: |

Продолжение Таблицы

|  |  |  |
| --- | --- | --- |
| Создание новой записи | | |
| При вводе пользователем корректного значения пункта меню и выбора создания новой записи, пользователь может добавить данные о студенте, но только на русском языке. | | Ввод информации выглядит следующим образом: |
| Внесение изменений в имеющуюся запись | | |
| Если пользователь допустил ошибку в создании карточки студента или данные изменились, он может внести изменения по пунктам, которые были созданы ранее. | При необходимости внесения изменений в запись о студенте, перед пользователем появляется меню:    Выбрав нужного ученика, на выбор пользователю даётся список информации, которую он может поменять:    Как только пользователь вводит корректное значение, его ответ обрабатывается и предлагается изменить существующие данные на новые. После пользователь возвращается (при необходимости) в главное меню. | |

Продолжение Таблицы

|  |  |
| --- | --- |
| Вывод всей информации о студентах | |
| Данное действие позволяет вывести всех студентов, находящихся в списке. |  |

Продолжение Таблицы

|  |  |
| --- | --- |
| Вывод всей информации о студентах в зависимости от условий | |
| Данные о студентах можно сортировать по номеру группы, по оценкам (хорошисты и отличники), по номеру в списке своей группы. | Если студент с нужным номером не найден:    Вывод информации: |
| Рейтинг студентов | |
| Пользователь в главном меню может выбрать такую опцию как рейтинг студентов, чтобы наглядно увидеть список студентов с минимальными данными и рейтингу их оценок. | Вывод топа самых успешных студентов с наивысшим по рейтингу средним баллом за прошедшую сессию |

Окончание Таблицы

|  |  |
| --- | --- |
| Информация о студентах в количестве | |
| Если пользователь хочет узнать количество студентов каждого пола или получивших стипендию. | Подсчёт количества студентов разных полов:    Получающих стипендию: |
| Вывод информации о студентах в зависимости от времени сделанной записи | |
| Пользователь может отсортировать записи по времени их создания или последнего редактирования: он может ввести дату, чтобы получить записи, сделанные в этот день, получить записи, сделанные до полудня или после полудня | Выбирая возможность получить записи в конкретный день, пользователь может ввести дату в определённом формате. Если записи были найдены, их список будет выведен:    В противном случае, пользователь получит сообщение о том, что таких записей не существует: |

|  |  |
| --- | --- |
| Ввод пользователем и обработка данных | Работа алгоритма и вывод на экран |
| Меню | |
| При запуске программы перед пользователем появляется окно с меню, где он может выбрать тип будущей вводимой последовательности. | Меню: |
| Преобразование выражение из одной записи в другую | |
| Пользователь может выбрать, из какой записи в какую преобразовать введённое им выражение. При этом, каждый раз его выражение будет проверяться на корректность.  Все промежуточные действия преобразования выводятся пользователю на экран |  |

Окончание Таблицы

|  |  |  |
| --- | --- | --- |
| Проверка на корректность | | |
| Чтобы проверить выражение на корректность, не преобразовывая его, пользователь может воспользоваться для этого специальной функцией | В случае, если пользователь ввёл некорректное выражение, он узнает почему: | |
| Вычисление выражений | | |
| Пользователь также может вычислить выражение в любой форме. Промежуточные действия будут также выведены на экран | |  |

**Выводы.**

Были изучены и применены структуры на практике.

Разработана программа, способная выводить и записывать данные, введённые пользователем. Также программа способна осуществлять сортировку введённых данных по параметрам, определяемые пользователем.

Были получены практические навыки работы со стеками и очередями; изучены обратная и прямая польская нотации; проведён сравнительный анализ этих структур данных.

Приложение А

**КОД ПРОГРАММЫ**

Название файла: cw.cpp

#include <iostream>

#include <fstream>

#include <string>

#include <iomanip>

#include <ctime>

#include <time.h>

#pragma warning(disable : 4996)

using namespace std;

struct Profile //анкета студента

{

string fullName;

char sex;

unsigned short int group;

unsigned short int numberList;

int term[8];

char depart[9];

string date; // Дата внесения изменнеия в запись (post/update)

};

//удаление лишних пробелов (для дат)

string DelSpaces(string s)

{

for (int j = 0; j < s.length(); j++)

{

if (s[j] == ' ')

{

while (s[j + 1] == ' ') s.erase(j + 1, 1);

}

}

if (s[0] == ' ') s.erase(0, 1);

if (s[s.length() - 1] == ' ') s.erase(s.length() - 1, 1);

return s;

}

//считывание с экрана информации о студенте

void newStudent()

{

Profile Student;

cin.clear();

char trash;

cout << "ФИО: ";

cin >> trash;

getline(cin, Student.fullName);

Student.fullName = trash + Student.fullName;

cout << "Группа: ";

cin >> Student.group;

cout << "Номер в списке: ";

cin >> Student.numberList;

cout << "Пол (F/M): ";

cin >> Student.sex;

cout << "Форма обучения (day/evening/distance): ";

cin >> Student.depart;

cout << "Оценки: ";

bool temp = false; //имеются ли двойки?

for (int i = 0; i < 8; i++)

{

cin >> Student.term[i];

if (Student.term[i] == 2) { temp = true; } //да, двойка имеется

}

//запоминание даты считывания

struct tm\* loctime;

time\_t curtime;

time(&curtime);

loctime = localtime(&curtime);

Student.date = asctime(loctime);

Student.date = DelSpaces(Student.date);

if (temp) { cout << '\n' << "Этот студент будет исключен. Профиль не будет сохранен в базе данных."; }

else

{

/\*Попытка создать файл с введёнными данными\*/

ofstream fout("students.txt", ios\_base::app);

if (!fout.is\_open()) { cout << '\n' << "Ошибка сохранения!"; }

else

{

/\* == Вывод записи == \*/

fout << Student.fullName << "\n";

fout << Student.group << "\n";

fout << Student.numberList << "\n";

fout << Student.sex << "\n";

fout << Student.depart << "\n";

for (int i = 0; i < 8; i++)

{

fout << Student.term[i] << " ";

}

fout << "\n";

fout << Student.date;

fout.close();

}

}

}

// Функция посчёта количества студентов

int countStudents()

{

ifstream fin("students.txt");

if (fin.is\_open())

{

int temp = 0; //количество строк

string data;

while (!fin.eof()) //пока указатель потока не достигнет конца файла

{

getline(fin, data); //считытвается строка

temp++;

}

fin.close();

int n;

n = temp / 7; //количество строк поделить на кол-во строк одной анкеты студента = кол-во анкет студента

return n;

}

else return 0;

}

//вывод файла всей информации о студентах

void outputStudents()

{

ifstream fin("students.txt");

if (!fin.is\_open()) // если файл не открыт

cout << "Файл не был открыт\n"; // сообщить об этом

else

{

int temp;

temp = countStudents();

if (temp == 0)

cout << "Файл пуст\n";

else

{

string data; // буфер промежуточного хранения считываемого из файла текста

while (!fin.eof())

{

getline(fin, data); // Считываем очередную строчку

cout << data << '\n'; // Выводим строку на экран

}

fin.close();

}

}

}

//подсчёт количества студентов разных полов

void F\_and\_M()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

getline(fin, student[i].fullName, '\n');

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

for (int j = 0; j < 8; j++) {

fin >> student[i].term[j];

}

fin >> trash;

getline(fin, student[i].date, '\n');

}

fin.close();

//подсчёт и вывод

int f = 0,

m = 0;

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

{

if (student[i].sex == 'F') { f++; }

if (student[i].sex == 'M') { m++; }

}

cout << "\nСтудентов женского пола: " << f << "\n";

cout << "Студентов мужского пола: " << m << "\n";

//конец подсчёта и вывода

delete[] student;

}

}

}

//Количество студентов со стипендией

void stipend()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool three; //\*наличие троек

int s = 0; //\*количество студентов со стипендией

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

three = false; //\*по умолчанию троек нет

for (int j = 0; j < 8; j++) {

fin >> student[i].term[j];

if (student[i].term[j] == 3) { three = true; } //\*если встречена тройка, запоминаем это

}

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

if (!(three) && (student[i].depart[1] == 'a')) { s++; } //\*подсчёт студентов со стипендией

}

fin.close();

cout << "\nВсего студентов со стипендией: " << s << '\n'; //\*вывод

delete[] student;

}

}

}

//Вывод данных о студентах без стипендии

void notStipend()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Степендия не может быть получена

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = false; //\*по умолчанию false

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

if (!(student[i].depart[1] == 'a')) { check = true; } //\*если не дневное обучение - степендии нет

for (int j = 0; j < 8; j++) {

fin >> student[i].term[j];

if (student[i].term[j] == 3) { check = true; } //\*если встречена тройка, запоминаем это

}

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

}

//конец вывода

}

fin.close();

delete[] student;

}

}

}

//Вывод данных о студентах-отличниках

void excellentTerm()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Является отличником

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = true; //\*по умолчанию true

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

for (int j = 0; j < 8; j++) {

fin >> student[i].term[j];

if ((student[i].term[j] == 3) || (student[i].term[j] == 4)) { check = false; } //\*если встречена 3 или 4, запоминаем это

}

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

}

//конец вывода

}

fin.close();

delete[] student;

}

}

}

//Вывод данных о хорошистах

void B\_GradeTerm()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Является отличником

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = true; //\*по умолчанию true

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

for (int j = 0; j < 8; j++) {

fin >> student[i].term[j];

if (student[i].term[j] == 3) { check = false; } //\*если встречена 3, запоминаем это

}

fin >> trash; //без него почему-то ломается

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

}

//конец вывода

}

fin.close();

delete[] student;

}

}

}

//Вывод данных о студентах группы N

void groupN(unsigned short int n)

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Является студентом группы N

int k = 0;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = false; //\*по умолчанию false

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

if (student[i].group == n) { check = true; }

fin >> student[i].depart;

for (int j = 0; j < 8; j++) { fin >> student[i].term[j]; }

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

k++;

}

//конец вывода

}

if (k == 0)

{

cout << "Таких студентов не было найдено\n";

}

fin.close();

delete[] student;

}

}

}

//Вывод данных о студентах номера k

void numberListK(unsigned short int k)

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Является студентом номера k

int n = 0;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = false; //\*по умолчанию false

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

if (student[i].numberList == k) { check = true; }

fin >> student[i].depart;

for (int j = 0; j < 8; j++) { fin >> student[i].term[j]; }

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

n++;

}

//конец вывода

}

if (n == 0)

{

cout << "Таких студентов не было найдено\n";

}

fin.close();

delete[] student;

}

}

}

//Вывод данных до полудня

void tillNoon()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Запись сделана до 12:00

int k = 0;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = false; //\*по умолчанию false

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

for (int j = 0; j < 8; j++) { fin >> student[i].term[j]; }

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//получаем время, когда была сделана запись

string date1 = student[i].date;

char date[9];

for (int i = 0; i < 8; i++)

{

date[i] = date1[i + date1.size() - 13];

}

date[8] = '\0'; //получается время в формате hh:mm:ss

int hh = (int)date[0] - (int)'0'; //преобразуем из string в int, вырезая ненужное и готовя для сравнения

hh \*= 10 + ((int)date[1] - (int)'0');

if (hh < 12) { check = true; } //если запись сделана раньше 12 часов, выводим

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

k++;

}

//конец вывода

}

if (k == 0)

{

cout << "Таких записей не было найдено.\n";

}

fin.close();

delete[] student;

}

}

}

//Вывод данных после полудня

void afterNoon()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Запись сделана после 12:00

int k = 0;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = false; //\*по умолчанию false

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

for (int j = 0; j < 8; j++) { fin >> student[i].term[j]; }

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//получаем время, когда была сделана запись

string date1 = student[i].date;

char date[9];

for (int i = 0; i < 8; i++)

{

date[i] = date1[i + date1.size() - 13];

}

date[8] = '\0'; //получается время в формате hh:mm:ss

//часы

int hh = (int)date[0] - (int)'0'; //преобразуем из string в int, вырезая ненужное и готовя для сравнения

hh \*= 10 + ((int)date[1] - (int)'0');

//минуты

int mm = (int)date[3] - (int)'0';

mm \*= 10 + ((int)date[4] - (int)'0');

//секунды

int ss = (int)date[6] - (int)'0';

ss \*= 10 + ((int)date[7] - (int)'0');

if (hh >= 12)

{

if (hh == 12) { if ((mm != 0) || (ss != 0)) { check = true; } }

else { check = true; }

} //если запись сделана после 12 часов, выводим

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

k++;

}

//конец вывода

}

if (k == 0)

{

cout << "Таких записей не было найдено.\n";

}

fin.close();

delete[] student;

}

}

}

void thatDate()

{

cout << "\nВведите дату в формате mmm dd yyyy (например, Mar 6 2020): ";

string Day;

char trash;

cin >> trash;

getline(cin, Day);

Day = trash + Day;

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

bool check; //\*true - надо вывести информацию. Запись сделана в данный день

int k = 0;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

check = false; //\*по умолчанию false

getline(fin, student[i].fullName);

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

for (int j = 0; j < 8; j++) { fin >> student[i].term[j]; }

fin >> trash;

getline(fin, student[i].date);

student[i].date = trash + student[i].date;

//сравниваем

//24 символа, если дата с %dd, 23 если %d в полной дате

//(Day).size(); 11 символов с %dd, 10 символов с %d

//при совпадении разница размеров должна давать 13. Отсеиваем при несовпадении

int daySize = (Day).size();

if (((student[i].date).size() - daySize) == 13)

{

string date1 = student[i].date;

bool same = true; //проверяем на совпадение месяцев и дней. По умолчанию true

for (int i = 0; i < 6; i++)

{

if (Day[i] != date1[i + 4]) //у date1 надо пропустить первые 4 символа

{

same = false;

break;

}

}

for (int i = daySize - 1; i > 8; i--)

{

if (Day[i] != date1[i + 13])

{

same = false;

break;

}

}

if (same) //если месяца совпали

{

check = true;

}

}

//вывод

if (check)

{

cout << student[i].fullName << "\n";

cout << student[i].group << "\n";

cout << student[i].numberList << "\n";

cout << student[i].sex << "\n";

cout << student[i].depart << "\n";

for (int j = 0; j < 8; j++) { cout << student[i].term[j] << " "; }

cout << "\n" << student[i].date << "\n";

k++;

}

//конец вывода

}

if (k == 0)

{

cout << "Таких записей не было найдено.\n";

}

fin.close();

delete[] student;

}

}

}

void topTerm()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

float\* term = new float[size]; //\*массив со средними оценками

char trash;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

getline(fin, student[i].fullName, '\n');

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

term[i] = 0;

for (int j = 0; j < 8; j++) {

fin >> student[i].term[j];

term[i] += (float)(student[i].term[j]);

}

term[i] /= 8;

fin >> trash;

getline(fin, student[i].date, '\n');

}

fin.close();

//сортировка и вывод

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

{

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

{

if (term[j] < term[j + 1])

{

float m = term[j];

term[j] = term[j + 1];

term[j + 1] = m;

string name = student[j].fullName;

student[j].fullName = student[j + 1].fullName;

student[j + 1].fullName = name;

}

}

}

//вывод

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

{

cout << i + 1 << ". " << student[i].fullName << " = " << term[i] << "\n";

}

//конец подсчёта и вывода

delete[] term;

delete[] student;

}

}

}

void changeFile()

{

ifstream fin("students.txt");

if (!fin.is\_open()) { cout << "Ошибка!\n"; }

else

{

int size;

size = countStudents();

if (size == 0) { cout << "База данных пуста.\n" << endl; }

else

{

Profile\* student = new Profile[size];

char trash;

for (int i = 0; i < size; i++) // Считываем данные всех студентов в массив структур

{

getline(fin, student[i].fullName, '\n');

fin >> student[i].group >> student[i].numberList >> student[i].sex;

fin >> student[i].depart;

for (int j = 0; j < 8; j++) {

fin >> student[i].term[j];

}

fin >> trash;

getline(fin, student[i].date, '\n');

student[i].date = trash + student[i].date;

}

fin.close();

//изменение информации

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

{

cout << i << ". " << student[i].fullName << "\n";

}

cout << "Введите номер, под которым находится студент, информацию о котором Вы хотите изменить: ";

int numbStud;

cin >> numbStud;

if (numbStud >= size) { cout << "Такой студент не был найден\n"; return; }

cout << "\n1. " << student[numbStud].fullName << "\n";

cout << "2. " << student[numbStud].group << "\n";

cout << "3. " << student[numbStud].numberList << "\n";

cout << "4. " << student[numbStud].sex << "\n";

cout << "5. " << student[numbStud].depart << "\n6. ";

for (int j = 0; j < 8; j++) { cout << student[numbStud].term[j] << " "; }

cout << "\nВведите номер, под которым находится информацию, которую Вы хотите изменить: ";

int sw;

cin >> sw;

bool check = false; //имеются ли изменения?

bool temp = false; //имеются ли двойки?

switch (sw)

{

case 1:

cout << "\nВведите новое ФИО студента: ";

cin >> trash;

getline(cin, student[numbStud].fullName);

student[numbStud].fullName = trash + student[numbStud].fullName;

check = true;

break;

case 2:

cout << "\nВведите новый номер группы студента: ";

cin >> student[numbStud].group;

check = true;

break;

case 3:

cout << "\nВведите новый номер студента в списке: ";

cin >> student[numbStud].numberList;

check = true;

break;

case 4:

cout << "\nВведите пол студента (F/M): ";

cin >> student[numbStud].sex;

check = true;

break;

case 5:

cout << "\nВведите новый формат обучения студента (day/evening/distance): ";

cin >> student[numbStud].depart;

check = true;

break;

case 6:

cout << "\nВведите оценки студента: ";

for (int i = 0; i < 8; i++)

{

cin >> student[numbStud].term[i];

if (student[numbStud].term[i] == 2) { temp = true; } //да, двойка имеется

}

if (temp) { cout << "У студента имеются неудовлетворительные оценки, поэтому он будет удалён.\n"; }

check = true;

break;

default:

cout << "\nВы ничего не выбрали. Производится выход из редактора.\n";

}

//изменения внесены

//вводим в файл

if (check)

{

/\*Попытка создать файл с введёнными данными\*/

ofstream fout("students.txt");

if (!fout.is\_open()) { cout << '\n' << "Ошибка сохранения!\n"; }

else

{

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

{

if (i == numbStud)

{

if (!temp) {

/\* == Вывод записи == \*/

fout << student[i].fullName << "\n";

fout << student[i].group << "\n";

fout << student[i].numberList << "\n";

fout << student[i].sex << "\n";

fout << student[i].depart << "\n";

for (int j = 0; j < 8; j++)

{

fout << student[i].term[j] << " ";

}

fout << "\n";

//запоминание даты считывания

struct tm\* loctime;

time\_t curtime;

string dateChange;

time(&curtime);

loctime = localtime(&curtime);

dateChange = asctime(loctime);

dateChange = DelSpaces(dateChange);

fout << dateChange;

}

cout << "Информация была успешно изменена.\n";

}

else

{

/\* == Вывод записи == \*/

fout << student[i].fullName << "\n";

fout << student[i].group << "\n";

fout << student[i].numberList << "\n";

fout << student[i].sex << "\n";

fout << student[i].depart << "\n";

for (int j = 0; j < 8; j++)

{

fout << student[i].term[j] << " ";

}

fout << "\n";

fout << student[i].date;

fout << "\n";

}

}

}

fout.close();

}

//конец вывода в файл

delete[] student;

}

}

}

int lb1()

{

bool check = true; //выход из меню

bool check1 = false; //выход из подменю

bool outp = false;

//false - заканчивает цикл, приводя непосредственно к выходу

do {

//system("cls");

char sw = ' '; //переключатель главного меню

char sw1 = ' '; //переключатель саб-меню

cout << "\nВыберите нужный раздел: \n";

cout << "\x1b[32m[1]\x1b[0m Создание новой записи о студенте\n";

cout << "\x1b[32m[2]\x1b[0m Изменить существующую запись \n";

cout << "\x1b[32m[3]\x1b[0m Отобразить все записи о студентах\n";

cout << "\x1b[32m[4]\x1b[0m Вывести данные о студентах, выполняющих поставленные условия\n";

cout << "\x1b[32m[5]\x1b[0m Вывести топ студентов по среднему баллу за сессию\n";

cout << "\x1b[32m[6]\x1b[0m Вывести число студентов, выполняющих поставленные условия\n";

cout << "\x1b[32m[7]\x1b[0m Вывести данные о студентах в зависимости от даты сделанной записи\n";

cout << "\x1b[32m[8]\x1b[0m Очистить экран консоли\n";

cout << "\x1b[32m[0]\x1b[0m Выйти в главное меню\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw;

while (cin.get() != '\n') { sw = ' '; }; //если строка содержит более одного символа, возвращается ошибка

switch (sw)

{

case '1': //[1] новая запись о студенте

newStudent();

break;

case '2': //[2] изменение записей

changeFile();

break;

case '3': //[3] вывод всех данных

outputStudents();

break;

case '4': //[4] данные о студентах, выполняющих поставленные условия

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Вывод информации о студентах группы N\n";

cout << "\x1b[32m[2]\x1b[0m Вывод информации о студентах, которые не получат стипендию\n";

cout << "\x1b[32m[3]\x1b[0m Вывод информации о студентах, которые имеют только оценки \"4\" и \"5\"\n";

cout << "\x1b[32m[4]\x1b[0m Вывод информации о студентах, которые имеют только оценки \"5\"\n";

cout << "\x1b[32m[5]\x1b[0m Вывод информации о студентах с номером k в списке\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] вывод данных студентов группы N

unsigned short int n;

cout << "\nВведите номер группы: ";

cin >> n;

groupN(n);

break;

case '2': //[2] вывод данных студентов без стипендии

notStipend();

break;

case '3': //[3] вывод данных студентов-хорошистов

B\_GradeTerm();

break;

case '4': //[4] вывод данных студентов-отличников

excellentTerm();

break;

case '5': //[5] вывод данных студентов номера k

unsigned short int k;

cout << "\nВведите номер студента в списке: ";

cin >> k;

numberListK(k); //

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '5': //[5] вывод топа студентов по средней оценке

topTerm();

break;

case '6': //[6] число студентов, выполняющих поставленные условия

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Количество студентов мужского и женского пола\n";

cout << "\x1b[32m[2]\x1b[0m Количество студентов, получающих стипендию\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] количество F и M

F\_and\_M();

break;

case '2': //[2] количество студентов со стипендией

stipend();

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '7': //[7] данные о студентах в зависимости от даты

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Вывод записей, сделанных в указанный день\n";

cout << "\x1b[32m[2]\x1b[0m Вывод записей, сделанных после полудня\n";

cout << "\x1b[32m[3]\x1b[0m Вывод записей, сделанных до полудня\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] вывод всех записей, сделанных в этот день

thatDate();

break;

case '2': //[2] записи после полудня

afterNoon();

break;

case '3': //[2] записи до полудня

tillNoon();

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '8': //[8] Очистить экран

system("cls");

break;

case '0': //[0] Закрыть программу

cout << "Выход из программы...\n";

check = false; //выход из цикла

break;

default: //в случае, если введено что-то иное

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

break;

}

} while (check);

system("Pause");

return 0;

}

// ДИНАМИЧЕСКИЙ ОДНОМЕРНЫЙ ЦЕЛОЧИСЛЕННЫЙ МАССИВ

// == Прочие действия ==

// Вывод массива

void outputArr(int\*& arr, int size)

{

if (size <= 0) { cout << "Ошибка!\n"; return; }

for (int i = 0; i < size; i++) { cout << arr[i] << ' '; }

cout << '\n';

}

// Создание массива размера N. Данные - случайные числа от 0 до 99

int\* createArr(int N)

{

if (N <= 0) { cout << "Ошибка!\n"; return NULL; }

int\* arr = new int[N]();

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

{

arr[i] = rand() % 100; //заполнение случайным числом от 0 до 99

}

return arr;

}

// == ФУНКЦИИ ПО ПУНКТАМ ==

//пункт 1.a по массивам: N случайных чисел

int taskArr1a(int\*& arr, bool time, int repeat)

{

clock\_t start, end; //таймер

int n;

cout << "Введите количество элементов нового массива: ";

while (!(cin >> n) || (n <= 0)) //проверка на корректность ввода

{

cout << "Ошибка! Новый массив не был создан!\n";

cin.clear();

cin.sync();

while (cin.get() != '\n');

cout << "Введите количество элементов нового массива: ";

}

if (time) { start = clock(); } //начало таймера

for (int r = 0; r < repeat; r++) { if (arr) { delete[] arr; } arr = createArr(n); }

if (time) { end = clock(); } //конец таймера

cout << "Создан новый массив: ";

outputArr(arr, n);

if (time) { cout << "Время выполнения (только создание): " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

return n;

}

//пункт 1.b по массивам: произвольный размер

int taskArr1b(int\*& arr, bool time)

{

clock\_t start, end;

int x, size = 1;

cout << "Введите элементы нового массива: ";

cin >> x;

if (!cin) { cout << "Ошибка! Новый массив не был создан\n"; }

else

{

if (time) { start = clock(); } //начало таймера

if (arr) { delete[] arr; }

arr = createArr(1);

arr[0] = x;

while (cin && cin.get() != '\n') //считываются все значения, пока не встречен мусор

{

int\* tmp;

cin >> x;

if (cin)

{

size++;

tmp = (int\*)realloc(arr, size \* sizeof(int));

if (tmp != NULL) { arr = tmp; arr[size - 1] = x; }

}

}

if (time) { end = clock(); } //конец таймера

cout << "Создан новый массив: ";

outputArr(arr, size);

cout << "Длина массива — " << size << " элементов.\n";

if (time) { cout << "Время выполнения (только создание): " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

return size;

}

return 0;

}

//пункт 1.с по массивам: произвольный размер с файла

int taskArr1с(int\*& arr, bool time, int repeat)

{

clock\_t start, end;

int x, size = 1;

ifstream fin("numbers.txt");

if (!fin.is\_open()) { cout << "Файл не был найден.\n"; } // если файл не открыт, сообщить об этом

else

{

fin >> x;

if (!fin) { cout << "Ошибка! Новый массив не был создан\n"; }

else

{

if (time) { start = clock(); } //начало таймера

if (arr) { delete[] arr; }

arr = createArr(1);

arr[0] = x;

while (fin && fin.get() != '\n') //считываются все значения, пока не встречен мусор

{

int\* tmp;

fin >> x;

if (fin)

{

size++;

tmp = (int\*)realloc(arr, size \* sizeof(int));

if (tmp != NULL) { arr = tmp; arr[size - 1] = x; }

}

}

if (time) { end = clock(); } //конец таймера

cout << "Создан новый массив: ";

outputArr(arr, size);

cout << "Длина массива — " << size << " элементов.\n";

if (time) { cout << "Время выполнения (только создание): " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

return size;

}

}

return 0;

}

//3.a - вставка нового элемента

void taskArr3a(int\*& arr, bool time, int& size, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Массив не был изменён!\n"; return; }

if (size < 1) { cout << "Ошибка! Массив не был найден!\n"; return; }

int x;

cout << "Введите значение нового элемента: ";

cin >> x;

if (!cin) { cout << "Ошибка! Массив не был изменён!\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); } //начало таймера

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

{

int\* tmp;

size++;

tmp = (int\*)realloc(arr, size \* sizeof(int));

if (tmp != NULL) { arr = tmp; arr[size - 1] = x; }

}

if (time) { end = clock(); } //конец таймера

cout << "Создан новый массив: ";

outputArr(arr, size);

cout << "Длина массива — " << size << " элементов.\n";

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

return;

}

return;

}

//3ItemInd - получить элемент по индексу

void taskArr3ItemInd(int\*& arr, bool time, int& size, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (size < 1) { cout << "Ошибка! Массив не был найден!\n"; return; }

int x;

cout << "Введите индекс элемента, который хотите получить (нумерация элементов с нуля): ";

cin >> x;

if (!cin || x >= size || x < 0) { cout << "Ошибка! Элемент не был найден.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); } //начало таймера

int\* item = nullptr;

for (int i = 0; i < repeat; i++) { item = &arr[x]; }

if (time) { end = clock(); } //конец таймера

if (item != nullptr) { cout << "Получен элемент " << \*item << " с адресом " << item << '\n'; }

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

return;

}

return;

}

//3Item - получить элемент по значению

void taskArr3Item(int\*& arr, bool time, int& size, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (size < 1) { cout << "Ошибка! Массив не был найден!\n"; return; }

int x;

cout << "Введите значение элемента, который хотите получить: ";

cin >> x;

if (!cin) { cout << "Ошибка! Некорректный ввод.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); } //начало таймера

int\* item = nullptr;

for (int r = 0; r < repeat; r++)

{

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

{

if (arr[i] == x) { item = &arr[i]; break; }

}

if (item == nullptr) { cout << "Не найдено\n"; break; }

}

if (time) { end = clock(); } //конец таймера

if (item != nullptr) { cout << "Получен элемент " << \*item << " с адресом " << item << '\n'; }

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

return;

}

//3DelInd - удаление элемента по индексу

void taskArr3DelInd(int\*& arr, bool time, int& size, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (size < 1) { cout << "Ошибка! Массив не был найден!\n"; return; }

int x;

cout << "Введите индекс, элемент которого хотите удалить (нумерация элементов с нуля): ";

cin >> x;

if (!cin || x >= size || x < 0) { cout << "Ошибка! Элемент не был найден.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); } //начало таймера

for (int r = 0; r < repeat; r++)

{

if (!cin || x >= size || x < 0) { cout << "Ошибка! Повторение не удалось, так как такой элемент не был найден.\n"; r = repeat; break; }

for (int j = x; j < size; j++) { arr[j] = arr[j + 1]; } //сдвиг влево на место удаляемого элемента

int\* tmp;

size--;

tmp = (int\*)realloc(arr, size \* sizeof(int));

if (tmp != NULL) { arr = tmp; }

}

if (time) { end = clock(); } //конец таймера

cout << "Новый массив: ";

if (size) { outputArr(arr, size); }

else { cout << "Массив пуст.\n"; }

cout << "Длина массива — " << size << " элементов.\n";

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

return;

}

return;

}

//3Del - удаление элемента по значению

//Удаляется только 1 элемент, так как доступна функция повтора

void taskArr3Del(int\*& arr, bool time, int& size, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (size < 1) { cout << "Ошибка! Массив не был найден!\n"; return; }

int x;

cout << "Введите значение элемента, который хотите удалить: ";

cin >> x;

if (!cin) { cout << "Ошибка! Некорректный ввод.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

bool check; //найден ли элемент?

if (time) { start = clock(); } //начало таймера

for (int r = 0; r < repeat; r++)

{

check = false;

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

{

if (arr[i] == x)

{

for (int j = i; j < size; j++) { arr[j] = arr[j + 1]; } //сдвиг влево на место удаляемого элемента

int\* tmp;

size--;

tmp = (int\*)realloc(arr, size \* sizeof(int));

if (tmp != NULL) { arr = tmp; }

check = true;

break;

}

}

if (!check) { cout << "Элемент не найден\n"; r = repeat; break; }

}

if (time) { end = clock(); } //конец таймера

if (check)

{

cout << "Элемент удалён\n";

cout << "Новый массив: ";

if (size) { outputArr(arr, size); }

else { cout << "Массив пуст.\n"; }

cout << "Длина массива — " << size << " элементов.\n\n";

}

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

return;

}

return;

}

// ДВУСВЯЗНЫЙ СПИСОК

struct list

{

int data;

list\* tail;

list\* head;

};

// == Прочие действия ==

// Вывод списка, начиная с введённого узла

void outputList(list\* beg)

{

if (!beg) { cout << "Список не найден!"; return; }

list\* curr = beg;

while (curr)

{

cout << curr->data << ' ';

curr = curr->tail;

}

cout << '\n';

}

// Очищение списка, начиная с введённого узла

void deleteList(list\*& beg)

{

list\* Next;

while (beg)

{

Next = beg->tail;

delete beg;

beg = Next;

}

}

// Длина списка

int lenList(list\* roster)

{

size\_t len = 0;

while (roster)

{

len++;

roster = roster->tail;

}

return len;

}

// == Создание списка размера N. Данные - случайные числа от 0 до 99 ==

list\* createList(int N)

{

if (N <= 0) { cout << "Ошибка!\n"; return NULL; }

list\* Curr = 0, //текущий элемент

\* Next = 0; //следующий

for (int i = 0; i < N; i++) //заполнение списка с конца

{

Curr = new list; //новый элемент

Curr->data = rand() % 100; //заполнение случайным числом от 0 до 99

Curr->tail = Next; //в адресной части - следующий элемент

if (Next) //если существует следующий элемент

{

Next->head = Curr;

} //закрепляем прошлый узел с текущим

Next = Curr; //переходим к следующему элементу

}

Curr->head = 0; //его предыдущий адрес должен отсылаться на NULL

return Curr; //адрес последнего элемента возвращается как адрес первого

}

// == Заполнение списка данными. N определяется автоматически ==

//Добавление нового элемента к списку после указанного узла. Не добавляет элемент в начало списка

list\* addItem(list\* roster, int a)

{

list\* temp;

temp = new list; //создание добавляемого узла

temp->data = a; //заполняем значение

temp->tail = roster->tail;

temp->head = roster;

if (roster->tail) { (roster->tail)->head = temp; }

roster->tail = temp;

return temp; //адрес добавленного узла

}

//доступ к заданному элементу списка (по индексу) для манипуляций с его информационной частью

list\* itemList(list\* beg, int index)

{

//index--; //если индексация не с нуля

while (beg && (index--))

{

beg = beg->tail;

if (!beg) { return 0; } //элемент не существует

}

return beg;

}

//удаление узла из списка;

list\* deleteItem(list\* delItem, list\* beg)

{

list\* prev, \* next;

prev = delItem->head; //элемент, предшедствующий удаляемому узлу

next = delItem->tail; //следующий элемент после удаляемого

//если НЕ первый

if (prev) { prev->tail = delItem->tail; }

else //если в начале

{

delete delItem;

if (next) { next->head = prev; return next; } //если следующий элемент существует - вернуть его как начало

else { return NULL; } //если нет - список будет удалён

}

//если НЕ последний

if (next) { next->head = delItem->head; }

delete delItem;

return beg;

}

// == ФУНКЦИИ ПО ПУНКТАМ ==

//пункт 1.a по спискам: N случайных чисел

void taskList1a(list\*& beg, bool time, int repeat)

{

clock\_t start, end;

int n;

cout << "Введите длину нового списка: ";

while (!(cin >> n) || (n <= 0))

{

cout << "Ошибка! Новый список не был создан!\n";

cin.clear();

cin.sync();

while (cin.get() != '\n');

cout << "Введите длину нового списка: ";

}

if (beg) { deleteList(beg); }

if (time) { start = clock(); }

for (int r = 0; r < repeat; r++) { beg = createList(n); }

if (time) { end = clock(); }

cout << "Создан новый список: ";

outputList(beg);

if (time) { cout << "Время выполнения (только создание): " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

//пункт 1.b по спискам: произвольный размер

void taskList1b(list\*& beg, bool time)

{

clock\_t start, end;

int x;

cout << "Введите элементы нового списка: ";

cin >> x;

if (!cin) { cout << "Ошибка! Некорректный ввод.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); }

if (beg) { deleteList(beg); }

beg = createList(1);

beg->data = x;

while (cin && cin.get() != '\n') //считываются все значения, пока не встречен мусор

{

cin >> x;

if (cin)

{

addItem(beg, x);

beg = beg->tail;

}

}

if (time) { end = clock(); }

while (beg->head) { beg = beg->head; }

cout << "Создан новый список: ";

outputList(beg);

cout << "Длина списка — " << lenList(beg) << " элементов.\n";

if (time) { cout << "Время выполнения (только создание): " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

}

//пункт 1.c по спискам: произвольный размер, считывание с файла

void taskList1c(list\*& beg, bool time)

{

clock\_t start, end;

int x;

ifstream fin("numbers.txt");

if (!fin.is\_open()) { cout << "Файл не был найден.\n"; }// если файл не открыт, сообщить об этом

else

{

fin >> x;

if (!fin) { cout << "Ошибка! Некорректный ввод.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); }

if (beg) { deleteList(beg); }

beg = createList(1);

beg->data = x;

while (fin && fin.get() != '\n') //считываются все значения, пока не встречен мусор

{

fin >> x;

if (fin)

{

addItem(beg, x);

beg = beg->tail;

}

}

if (time) { end = clock(); }

while (beg->head) { beg = beg->head; }

cout << "Создан новый список: ";

outputList(beg);

cout << "Длина списка — " << lenList(beg) << " элементов.\n";

if (time) { cout << "Время выполнения (только создание): " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

fin.close();

}

}

}

//3.a - вставка нового элемента !!после первого элемента

void taskList3a(list\*& beg, bool time, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Список не был изменён!\n"; return; }

if (!beg) { cout << "Ошибка! Список не был найден!\n"; return; }

int x;

cout << "Введите значение нового элемента: ";

cin >> x;

if (!cin) { cout << "Ошибка! Некорректный ввод.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); } //начало таймера

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

{

addItem(beg, x);

}

if (time) { end = clock(); } //конец таймера

cout << "Создан новый список: ";

outputList(beg);

cout << "Длина списка — " << lenList(beg) << " элементов.\n";

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

return;

}

//3ItemInd - получить элемент по индексу

void taskList3ItemInd(list\*& beg, bool time, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (!beg) { cout << "Ошибка! Список не был найден!\n"; return; }

int x;

cout << "Введите индекс элемента, который хотите получить (нумерация элементов с нуля): ";

cin >> x;

if (!cin || x >= lenList(beg) || x < 0) { cout << "Ошибка! Элемент не был найден.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); } //начало таймера

list\* item = nullptr;

for (int i = 0; i < repeat; i++) { item = itemList(beg, x); }

if (time) { end = clock(); } //конец таймера

if (item != nullptr) { cout << "Получен элемент " << item->data << " с адресом " << item << '\n'; }

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

return;

}

//3Item - получить элемент по значению

void taskList3Item(list\*& beg, bool time, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (!beg) { cout << "Ошибка! Список не был найден!\n"; return; }

int x;

cout << "Введите значение элемента, который хотите получить: ";

cin >> x;

if (!cin || x >= lenList(beg) || x < 0) { cout << "Ошибка! Элемент не был найден.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

bool check = false; //найден ли элемент?

if (time) { start = clock(); } //начало таймера

list\* tmp;

list\* item = nullptr;

for (int r = 0; r < repeat; r++)

{

tmp = beg;

size\_t i = 0;

do

{

if (tmp->data == x) { item = tmp; break; }

tmp = tmp->tail;

i++;

} while (tmp);

if (item == nullptr) { cout << "Не найдено\n"; break; }

}

if (time) { end = clock(); } //конец таймера

if (item != nullptr) { cout << "Получен элемент " << item->data << " с адресом " << item << '\n'; }

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

return;

}

//3DelInd - удаление элемента по индексу

void taskList3DelInd(list\*& beg, bool time, int repeat)

{

clock\_t start, end;

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (!beg) { cout << "Ошибка! Список не был найден!\n"; return; }

int x;

cout << "Введите индекс элемента, который хотите удалить (нумерация элементов с нуля): ";

cin >> x;

if (!cin || x >= lenList(beg) || x < 0) { cout << "Ошибка! Элемент не был найден.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

if (time) { start = clock(); } //начало таймера

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

{

if (itemList(beg, x)) { deleteItem(itemList(beg, x), beg); }

else { cout << "Ошибка! Такой элемент не был найден.\n"; break; }

}

if (time) { end = clock(); } //конец таймера

if (beg)

{

cout << "Создан новый список: ";

outputList(beg);

cout << "Длина списка — " << lenList(beg) << " элементов.\n";

}

else { cout << "Список был удалён\n"; }

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

return;

}

//3Del - удаление элемента по значению

void taskList3Del(list\*& beg, bool time, int repeat)

{

clock\_t start, end;

bool check = false; //удалён ли элемент

if (repeat < 1) { cout << "Ошибка! Количество повторений не может быть меньше 1!\n"; return; }

if (!beg) { cout << "Ошибка! Список не был найден!\n"; return; }

int x;

cout << "Введите значение элемента, который хотите удалить: ";

cin >> x;

if (!cin) { cout << "Ошибка! Некорректный ввод.\n"; cin.clear(); while (cin.get() != '\n'); }

else

{

list\* delItem;

if (time) { start = clock(); } //начало таймера

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

{

delItem = beg;

do {

if (delItem->data == x)

{

beg = deleteItem(delItem, beg);

check = true;

break;

}

delItem = delItem->tail;

} while (delItem);

if (!check) { break; }

}

if (time) { end = clock(); } //конец таймера

if (beg)

{

if (check)

{

cout << "Создан новый список: ";

outputList(beg);

cout << "Длина списка — " << lenList(beg) << " элементов.\n";

}

else { cout << "Элемент не был найден\n"; }

}

else { cout << "Список был удалён\n"; }

if (time) { cout << "Время выполнения: " << ((double)end - start) / ((double)CLOCKS\_PER\_SEC) << '\n'; }

}

return;

}

int lb2()

{

bool time = true;

int repeat = 1; //количество повторений действия

// ДИНАМИЧЕСКИЙ МАССИВ

int size = 0;

int\* arr = new int[size]();

// ДВУСВЯЗНЫЙ СПИСОК

list\* beg = NULL;

bool check = true; //выход из меню

bool check1 = false; //выход из подменю

bool outp = false;

//false - заканчивает цикл, приводя непосредственно к выходу

do {

//system("cls");

char sw = ' '; //переключатель главного меню

char sw1 = ' '; //переключатель саб-меню

cout << "\nВыберите нужный раздел: \n";

cout << "\x1b[32m[1]\x1b[0m Создание целочисленного одномерного массива\n";

cout << "\x1b[32m[2]\x1b[0m Работа с элементами массива \n";

cout << "\x1b[32m[3]\x1b[0m Создание двусвязного списка\n";

cout << "\x1b[32m[4]\x1b[0m Работа с узлами двусвязного списка\n";

cout << "\x1b[32m[5]\x1b[0m Включить/выключить счётчик времени при выполнении какого-либо действия. Состояние: ";

if (time) { cout << "\x1b[32mВключено\x1b[0m\n"; }

else { cout << "\x1b[33mВыключено\x1b[0m\n"; }

cout << "\x1b[32m[6]\x1b[0m Установить другое число повторов действия. Количество повторений сейчас: " << repeat << '\n';

cout << "\x1b[32m[7]\x1b[0m Очистить экран консоли\n";

cout << "\x1b[32m[8]\x1b[0m Включить/выключить отображение списка и массива. Состояние: ";

if (outp) { cout << "\x1b[32mВключено\x1b[0m\n"; }

else { cout << "\x1b[33mВыключено\x1b[0m\n"; }

cout << "\x1b[32m[0]\x1b[0m Выйти в главное меню\n";

if (outp)

{

cout << "Массив: ";

if (size) { outputArr(arr, size); }

else { cout << "Не создан.\n"; }

cout << "Список: ";

if (beg) { outputList(beg); }

else { cout << "Не создан.\n"; }

}

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw;

while (cin.get() != '\n') { sw = ' '; }; //если строка содержит более одного символа, возвращается ошибка

switch (sw)

{

case '1': //[1] Создание целочисленного одномерного массива

do {

check1 = false;

sw1 = ' ';

cout << "\n\x1b[32m[1]\x1b[0m Ввести количество элементов в массиве, чтобы получить массив со случайными числами (от 0 до 99)\n";

cout << "\x1b[32m[2]\x1b[0m Ввести значение элементов массива. Его размер определяется автоматически\n";

cout << "\x1b[32m[3]\x1b[0m Считать массив с файла\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] Случайные числа

size = taskArr1a(arr, time, repeat);

break;

case '2': //[2] Заполнение массива

size = taskArr1b(arr, time);

break;

case '3': //[3] Заполнение массива с файла

size = taskArr1с(arr, time, repeat);

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '2': //[2] Работа с элементами массива

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Вставить новый элемент в массив\n";

cout << "\x1b[32m[2]\x1b[0m Удалить элемент массива (по индексу)\n";

cout << "\x1b[32m[3]\x1b[0m Удалить элемент массива (по значению)\n";

cout << "\x1b[32m[4]\x1b[0m Получить элемент массива по индексу\n";

cout << "\x1b[32m[5]\x1b[0m Получить элемента массива по значению\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] Вставить новый элемент в массив

taskArr3a(arr, time, size, repeat);

break;

case '2': //[2] Удалить элемент массива (по индексу)

taskArr3DelInd(arr, time, size, repeat);

break;

case '3': //[3] Удалить элемент массива (по значению)

taskArr3Del(arr, time, size, repeat);

break;

case '4': //[4] Получить элемент массива по индексу

taskArr3ItemInd(arr, time, size, repeat);

break;

case '5': //[5] Получить элемента массива по значению

taskArr3Item(arr, true, size, repeat);

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '3': //[3] Создание двусвязного списка

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Ввести количество элементов в списке, чтобы получить список со случайными числами (от 0 до 99)\n";

cout << "\x1b[32m[2]\x1b[0m Ввести значение узлов списка. Его размер определяется автоматически\n";

cout << "\x1b[32m[3]\x1b[0m Считать список с файла\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] Случайные числа

taskList1a(beg, time, repeat);

break;

case '2': //[2] Заполнение списка

taskList1b(beg, time);

break;

case '3': //[3] Заполнение списка с файла

taskList1c(beg, time);

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '4': //[4] Работа с узлами двусвязного списка

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Вставить новый элемент в список\n";

cout << "\x1b[32m[2]\x1b[0m Удалить узел списка (по индексу)\n";

cout << "\x1b[32m[3]\x1b[0m Удалить узел списка (по значению)\n";

cout << "\x1b[32m[4]\x1b[0m Получить элемент списка по индексу\n";

cout << "\x1b[32m[5]\x1b[0m Получить элемент списка по значению\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] Вставить новый элемент в массив

taskList3a(beg, time, repeat);

break;

case '2': //[2] Удалить элемент массива (по индексу)

taskList3DelInd(beg, time, repeat);

break;

case '3': //[3] Удалить элемент массива (по значению)

taskList3Del(beg, time, repeat);

break;

case '4': //[4] Получить элемент списка по индексу

taskList3ItemInd(beg, time, repeat);

break;

case '5': //[5] Получить элемент списка по значению

taskList3Item(beg, time, repeat);

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '5': //[5] Включить/отключить счётчик времени при выполнении какого-либо действия

time = !time;

if (time) { cout << "Счётчик времени теперь включён.\n"; }

else { cout << "Счётчик времени теперь отключён.\n"; }

break;

case '6': //[6] Установить другое число повторов действия. По умолчанию равен 1

do

{

int x = 0;

check1 = false;

cout << "Введите число повторений (выше нуля): ";

cin >> x;

while (cin.get() != '\n');

if (!cin || x <= 0) {

cout << "Ошибка! Количество повторений не может быть меньше 1. Значение не было изменено.\n";

check1 = true;

}

else { repeat = x; }

} while (check1);

break;

case '7': //[7] Очистить экран

system("cls");

break;

case '8': //[5] Включить/отключить счётчик времени при выполнении какого-либо действия

outp = !outp;

if (outp) { cout << "Отображение списка и массива включено.\n"; }

else { cout << "Отображение списка и массива отключено.\n"; }

break;

case '0': //[0] Закрыть программу

cout << "Выход из программы...\n";

check = false; //выход из цикла

break;

default: //в случае, если введено что-то иное

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

break;

}

} while (check);

if (size) { delete[] arr; }

deleteList(beg);

system("Pause");

return 0;

}

#include <iostream>

#include <string>

using namespace std;

//удаляет лишние пробелы, введённые пользователем

string DelSpaces(string s)

{

for (size\_t j = 0; j < s.length(); j++)

{

if (s[j] == ' ')

{

while (s[j + 1] == ' ') s.erase(j + 1, 1);

}

}

if (s[0] == ' ') s.erase(0, 1);

if (s[s.length() - 1] == ' ') s.erase(s.length() - 1, 1);

return s;

}

//проверка, является ли символ цифрой

bool isNumber(char s)

{

if ((s <= '9') && (s >= '0')) { return true; }

return false;

}

//проверка, является ли текущий символ операцией. если да - возвращает его приоритет

int priorOperation(char symb)

{

switch (symb) {

case '+':

case '-':

return 1;

case '\*':

case '/':

return 2;

case '(':

return -1;

case ')':

return -2;

default:

return 0; //не является операцией

}

}

//структура стека для символов

struct stack

{

string data;

stack\* prev = 0;

};

//создание (инициализирование) стека и добавление в него первого элемента

stack\* init(string data) {

stack\* tmp = new stack; //создаём начало стека

tmp->data = data;

tmp->prev = 0;

return tmp;

}

//добавление элемента в существующий стек

stack\* pushBack(stack\* stk, string data) {

stack\* newHead = new stack;

newHead->data = data;

newHead->prev = stk;

return newHead;

}

//добавление элементов в стек

void push(stack\*& stk, string data) {

if (!stk) { stk = init(data); } //если стек пуст

else { stk = pushBack(stk, data); }

}

//вывод элементов из стека в строку. oper - разный вывод в зависимости от операции

string outputStk(stack\*& head, int oper)

{

//oper = 0 вывести весь стек

//oper = -2 вывести стек до "(" уничтожая "("

//oper = -1 вывести стек до ")" уничтожая ")"

//oper = 1 вывести весь стек до "(" не уничтожая его

//oper = 3 вывести верхний элемент

string output = "\0";

if (oper == 3)

{

output += head->data;

head = head->prev;

}

else

{

while (head)

{

if (priorOperation(head->data[0]) > 0) //игнорируются скобки

{

output += head->data;

output += " ";

}

head = head->prev;

if (oper && head && ((head->data == "(") || (head->data == ")"))) //если не весь стек, то до "(" или ")"

{

if (oper == -2 || oper == -1) { head = head->prev; } //уничтожается "("

break; //выход из вывода

}

}

}

return output; //в конце строки всегда должен быть пробел

}

//обратная польская нотация

string polishNotation(string& expr)

{

stack\* operation = 0; //стек операций

string push\_symb = "\0";

string result = "\0"; //обработанное выражение

int k = 0; //индекс строки

string strval = "\0";

int prior = 0;

do {

if (!expr[k]) { break; }

cout << "Рассматриваем символ " << expr[k] << "\n";

prior = priorOperation(expr[k]);

if (prior) //если является операцией/скобкой

{

cout << "Символ " << expr[k] << " является операцией\n";

//Если подытожить:

//Игнор если: "++-"

//НЕ ИГНОР: ")", "\*+"

//Если ")" - текущий символ уничтожается

//Если не игнор "+" или "-", текущий символ переходит в пустой стек

switch (prior)

{

case -1: //все открывающиеся скобки - сразу в стек операций

cout << "Символ " << expr[k] << " является открывающейся скобкой, поэтому он идёт в стек\n";

push\_symb = expr[k];

push(operation, push\_symb);

break;

case 2: //"\*" и "/" - вывод стека, если прошлый символ "\*" или "/"

cout << "Символ " << expr[k] << " имеет приоритет 2\n";

if (operation && (priorOperation(operation->data[0]) == 2))

{

cout << "Приоритет прошлого символа также равен 2, поэтому происходит выход символов из стека\n";

result += outputStk(operation, 1);

}

cout << "Текущий символ попадает в стек\n";

push\_symb = expr[k];

push(operation, push\_symb);

break;

case -2: //закрывается скобка - выход операций до открывающейся скобки

cout << "Символ " << expr[k] << " является закрывающейся скобкой, поэтому происходит выход всех операций до открывающейся скобки\n";

result += outputStk(operation, -2);

break;

case 1:

//перед помещением в стек операций + и -, смотрится последняя операция в стеке

//если у текущего символа приоритет меньше или равно - освобождение стека. Текущий символ помещается в стек

cout << "Символ " << expr[k] << " имеет приоритет 1\n";

if (operation && (priorOperation(operation->data[0]) >= 1))

{

result += outputStk(operation, 1);

cout << "Приоритет прошлого символа равен 1 или меньше, поэтому происходит выход символов из стека\n";

}

push\_symb = expr[k];

push(operation, push\_symb); //если приоритет у текущего символа больше - в стек

cout << "Текущий символ попадает в стек\n";

break;

}

k++;

cout << "Переходим к следующему символу...\n";

}

else

{

cout << "Символ " << expr[k] << " является числом\n";

strval = "\0"; //очищение строки

do { //пока текущий элемент НЕ является операцией

strval += expr[k]; //сюда попадают ТОЛЬКО числа

k++;

} while (expr[k] && !priorOperation(expr[k]));

cout << "Считали число " << strval << "\n";

result += strval; //все числа попадают сразу же в строку выхода

cout << "Оно сразу попадает в строку вывода\n";

result += " "; //пробел после числа

}

cout << "Текущая строка вывода: " << result << "\n";

} while (expr[k]);

if (operation) { result += outputStk(operation, 0); } //освобождение всего стека, если там чёт есть

cout << "Если в стеке что-то осталось, выводим все операции. \n";

return result;

}

//Обратную польскую в обычное

string fromPolishNotation(string& expr)

{

string result = "\0"; //обработанное выражение

stack\* numbs = 0; //стек для чисел/выражений

size\_t k = 0; //индекс строки

string strval = "\0"; //строка для чисел/выражений

string tmp1 = "\0"

, tmp2 = "\0";

//все числа - в стек

//как встречена операция - забираем числа, используем на них операцию

do {

cout << "Рассматриваем символ " << expr[k] << "\n";

if (priorOperation(expr[k])) //если является операцией

{

cout << "Символ " << expr[k] << " является операцией\n";

strval = "\0"; //очищение строку

tmp2 = outputStk(numbs, 3); //вытаскиваем второе выражение

cout << "Вытащили второе выражение из стека " << tmp2 << " \n";

tmp1 = outputStk(numbs, 3); //вытаскиваем первое выражение

cout << "Вытащили первое выражение из стека " << tmp1 << " \n";

if (expr[k + 1] && (expr[k] == '+' || expr[k] == '-')) { strval += "("; } //если НЕ последнее действие, ставит скобки

strval += tmp1 + expr[k] + tmp2;

if (expr[k + 1] && (expr[k] == '+' || expr[k] == '-')) { strval += ")"; }

cout << "Получившееся выражение: " << strval << ". Добавляем в стек\n";

push(numbs, strval); //Добавляем в стек получившееся выражение

k++; //переходим на следующий символ

}

else //если число или пробел

{

if (isNumber(expr[k]))

{

cout << "Символ " << expr[k] << " является числом\n";

strval = "\0"; //очищение строку

while (!(expr[k] == ' ')) //пока текущий элемент НЕ является пробелом

{

strval += expr[k]; //считываем числа

k++;

}

cout << "Считали число " << strval << ". Оно идёт в стек\n";

push(numbs, strval); //все числа попадают в стек

}

k++;

}

} while (expr[k]);

result = outputStk(numbs, 3);

return result;

}

//в прямую польскую нотацию

string directPolishNotation(string& expr)

{

stack\* operation = 0 //стек операций

, \* output = 0; //стек выхода

string push\_symb = "\0";

string result = "\0"; //обработанное выражение

int k = expr.length() - 1; //индекс строки

string strval = "\0";

int prior = 0;

do {

if (!expr[k]) { break; }

cout << "Рассматриваем символ " << expr[k] << "\n";

prior = priorOperation(expr[k]);

if (prior) //если является операцией/скобкой

{

cout << "Символ " << expr[k] << " является операцией\n";

push\_symb = "\0";

//Если подытожить:

//Игнор если: "++-"

//НЕ ИГНОР: "(", "\*+"

//Если "(" - текущий символ уничтожается

//Если не игнор "+" или "-", текущий символ переходит в пустой стек

switch (prior)

{

case -2: //все закрывающиеся скобки - сразу в стек операций

cout << "Символ " << expr[k] << " является закрывающейся скобкой. Помещаем в стек\n";

push\_symb = expr[k];

push(operation, push\_symb);

break;

case 2: //"\*" и "/" - сразу в стек операций

cout << "Символ " << expr[k] << " имеет приоритет 2. Помещаем в стек\n";

push\_symb = expr[k];

push(operation, push\_symb);

break;

case -1: //открывается скобка - выход операций до закрывающейся скобки

cout << "Символ " << expr[k] << " является открывающейся скобкой. Вывод операций до закрывающейся\n";

result += outputStk(operation, -1);

break;

case 1:

//перед помещением в стек операций + и -, смотрится последняя операция в стеке

//если у текущего символа приоритет меньше или равно - освобождение стека. Текущий символ помещается в стек

cout << "Символ " << expr[k] << " имеет приоритет 1\n";

if (operation && (priorOperation(operation->data[0]) >= 1))

{

cout << "Прошлая операция в стеке имеет приоритет равный 1 или больше. Вывод операций\n";

result += outputStk(operation, 1);

}

cout << "Символ " << expr[k] << " помещаем в стек\n";

push\_symb = expr[k];

push(operation, push\_symb); //если приоритет у текущего символа больше - в стек

break;

}

k--;

}

else

{

cout << "Символ " << expr[k] << " является числом\n";

strval = "\0"; //очищение строки

do { //пока текущий элемент НЕ является операцией

strval += expr[k]; //сюда попадают ТОЛЬКО числа (в обратной записи)

k--;

} while ((k > 0) && !priorOperation(expr[k]));

push\_symb = "\0";

for (size\_t i = strval.length(); i > 0; i--) { push\_symb += strval[i - 1]; } //отражаем полученное число

cout << "Получаем число " << push\_symb << " и помещаем в строку вывода\n";

result += strval + " "; //число сразу в стек выхода

}

cout << "Текущая строка вывода: " << result << "\n";

} while (k >= 0);

if (operation) { result += outputStk(operation, 0); } //освобождение всего стека, если там чёт есть

cout << "Освобождаем стек, если там что-то осталось. Текущая строка вывода: " << result << "\n";

strval = result;

result = "\0";

cout << "Отражаем строку\n";

for (size\_t i = strval.length(); i > 0; i--) { result += strval[i - 1]; } //отражаем полученное выражение

return result;

}

//Прямую польскую в обычное

string fromDirectPolishNotation(string& expr)

{

string result = "\0"; //обработанное выражение

stack\* numbs = 0; //стек для чисел/выражений

int k = expr.length() - 1; //индекс строки

string strval = "\0" //строка для чисел/выражений

, tmp1 = "\0"

, tmp2 = "\0"

, push\_symb = "\0";

//все числа - в стек

//как встречена операция - забираем числа, используем на них операцию

do {

cout << "Рассматриваем символ " << expr[k] << "\n";

if (priorOperation(expr[k])) //если является операцией

{

cout << "Символ " << expr[k] << " является операцией\n";

strval = "\0"; //очищение строку

tmp2 = outputStk(numbs, 3); //вытаскиваем второе выражение

cout << "Вытащили второе выражение из стека " << tmp2 << " \n";

tmp1 = outputStk(numbs, 3); //вытаскиваем первое выражение

cout << "Вытащили первое выражение из стека " << tmp1 << " \n";

if ((k != 0) && (expr[k] == '+' || expr[k] == '-')) { strval += "("; } //если НЕ последнее действие, ставит скобки

strval += tmp2 + expr[k] + tmp1;

if ((k != 0) && (expr[k] == '+' || expr[k] == '-')) { strval += ")"; }

cout << "Получившееся выражение: " << strval << ". Добавляем в стек\n";

push(numbs, strval); //Добавляем в стек получившееся выражение

k--; //переходим на следующий символ

}

else //если число или пробел

{

if (isNumber(expr[k]))

{

cout << "Символ " << expr[k] << " является числом\n";

strval = "\0"; //очищение строки

while (!(expr[k] == ' ')) //пока текущий элемент НЕ является пробелом

{

strval += expr[k]; //считываем числа

k--;

}

push\_symb = "\0";

for (size\_t i = strval.length(); i > 0; i--) { push\_symb += strval[i - 1]; } //отражаем полученное число

cout << "Считали число " << push\_symb << ". Оно идёт в стек\n";

push(numbs, push\_symb); //все числа попадают в стек

}

k--;

}

} while (k >= 0);

result = outputStk(numbs, 3);

return result;

}

//инициализация переменных. возвращает 0, если всё ок

bool initVar(string& expr)

{

char sw = '\0';

for (size\_t i = 0; i < expr.length() - 1; i++)

{

if (!priorOperation(expr[i]) && !isNumber(expr[i])) //если не является операцией и не является числом

{

cout << "Найден неизвестный символ: " << expr[i] << ". Инициализировать его как переменную? (Y/N)\n";

cin >> sw;

while (cin.get() != '\n') { sw = ' '; }; //если строка содержит более одного символа, возвращается ошибка

switch (sw)

{

case 'N':

return 1;

case 'Y':

string toReplace = "\0"

, replaceWith = "\0";

toReplace = expr[i];

cout << "Введите значение переменной " << expr[i] << ": ";

getline(cin, replaceWith);

for (size\_t j = 0; j < replaceWith.length() - 1; j++)

{

if (!isNumber(replaceWith[j])) //если встречено не число

{

cout << "Значение переменной не может содержать иные символы кроме цифр.\n";

return 1;

}

}

size\_t pos = 0;

while ((pos = expr.find(toReplace, pos)) != string::npos)

{

expr.replace(pos, toReplace.size(), replaceWith);

pos += replaceWith.size();

}

i = 0;

break;

}

}

}

return 0;

}

//проверка на корректность ввода и обработка выражения. возвращает 1, если найдена ошибка в вводе, 0 если всё ок

bool invalidInput(string& expr)

{

if (priorOperation(expr[0]) > 0) { cout << "Выражение не может начинаться операцией\n"; return 1; } //выражение не может иметь операцию в начале

if (priorOperation(expr[expr.length() - 1]) > 0) { cout << "Выражение не может заканчиваться операцией\n"; return 1; } //выражение не может иметь операцию в конце

//удаление всех пробелов

for (size\_t j = 1; j < expr.length() - 1; j++) //пробел не может стоять между числами

{

if (expr[j] == ' ')

{

if (!(priorOperation(expr[j - 1]) || priorOperation(expr[j + 1]))) //если пробелом разделены два числа

{

cout << "Выражение содержит лишние символы\n"; return 1; //ввод некорректный

}

expr.erase(j, 1); //если пробел разделяет число и операцию - стираем пробел

}

}

int opBrace = 0,

cBrace = 0;

bool op = false;

//добавляет "\*" там, где его не хватает. Проверяет корректность потенциальных переменных

for (size\_t i = 0; i < expr.length(); i++)

{

if (!expr[i]) { break; }

if (!(isNumber(expr[i]) || priorOperation(expr[i]))) //не является числом или операцией (т.е. потенциальная переменная)

{

//работаем со следующим символом

if (expr[i + 1] != '\0') //если он существует

{

if (expr[i] == expr[i + 1]) { cout << "Выражение содержит лишние символы\n"; return 1; } //если две одинаковой переменной подряд - ошибка

if (isNumber(expr[i + 1])) { cout << "Выражение содержит лишние символы\n"; return 1; } //если следующий символ число - ошибка

if (!(priorOperation(expr[i + 1]) && expr[i + 1] != '(')) //если является операцией и не ")" - ничего не делаем

{

expr.insert(i + 1, "\*"); //если "(" или другая переменная - вставляем "\*"

}

}

//работа с предыдущим символом

if (i > 0) //если он существует

{

if (!(priorOperation(expr[i - 1]) && expr[i - 1] != ')')) //если является операцией и не ")" - ничего не делаем

{

expr.insert(i, "\*"); //если число или ")" - вставляем "\*"

}

}

//таким образом:

// x5 xx - ошибка

// x\* x+ \*x +x x) (x - игнор

// xb )x x( 5x - превращается в x\*b )\*x x\*( 5\*x

}

if (isNumber(expr[i])) //является числом

{

//работа со следующим символом || 2( превращается в 2\*(

if (expr[i + 1] != '\0') //если он существует

{

if (expr[i + 1] == '(') { expr.insert(i + 1, "\*"); } //если "(" - вставляем "\*"

}

//работа с предыдущим символом || )2 превращается в )\*2

if (i > 0) //если он существует

{

if (expr[i - 1] == ')') { expr.insert(i, "\*"); } //если ")" - вставляем "\*"

}

}

if (priorOperation(expr[i]) > 0) //если является операцией, не скобки

{

op = true; //запоминаем, что в выражении есть операция

//работа со следующим символом

if (expr[i + 1] != '\0') //если он существует

{

if (priorOperation(expr[i + 1]) > 0) { cout << "Две операции не могут находиться рядом\n"; return 1; } //если две операции рядом - ошибка

if (priorOperation(expr[i + 1]) == ')') { cout << "Неправильный порядок скобок\n"; return 1; } //если после операции ")" - ошибка. пример: \*)

}

//работа со следующим символом

if (i > 0) //если он существует

{

if (priorOperation(expr[i - 1]) == '(') { cout << "Неправильный порядок скобок\n"; return 1; } //если после операции "(" - ошибка. пример: (\*

}

}

if (expr[i] == '(')

{

opBrace++; //подсчёт "("

//внутри скобок должны быть как минимум одна операция с двумя переменными/числами

size\_t j = i + 1;

bool check = false;

while (expr[j] != '\0') //поиск символа-операции

{

switch (priorOperation(expr[j]))

{

case -1:

case 0: //если не операция или "(", скип

j++;

break;

case 1:

case 2: //если встречена операция + - \* /, запоминаем

check = true;

break;

case -2: //если ")" встречена раньше других операций - ошибка

cout << "Неправильный порядок скобок\n"; return 1;

}

if (check) { break; }

}

}

if (expr[i] == ')')

{

cBrace++; //подсчёт ")"

if (cBrace > opBrace) { cout << "Неправильный порядок скобок\n"; return 1; } //если закрывающаяся скобка встречена раньше открывающейся - ошибка

}

}

if (!op) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если в выражении нет операций, это ошибка

if (opBrace != cBrace) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если количество скобок не совпадает

if (initVar(expr)) { return 1; }

return 0;

}

//Проверка на корректность написания обратной польской

bool invalidInputPolish(string& expr)

{

//Первые два объекта - числа или переменные

size\_t j = 0;

while (expr[j] != ' ')

{

if (priorOperation(expr[j])) { cout << "Выражение должно иметь в начале два числа или переменные\n"; return 1; }

j++;

if (j == expr.length()) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если не найдено пробела или операции

}

j++;

while (expr[j] != ' ')

{

if (priorOperation(expr[j])) { cout << "Выражение должно иметь в начале два числа или переменные\n"; return 1; }

j++;

if (j == expr.length()) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если не найдено пробела или операции

}

//Всегда заканчивается на операцию

if (!priorOperation(expr[expr.length() - 1])) { cout << "Выражение должно заканчиваться операцией\n"; return 1; }

//Количество операций = количество чисел - 1

size\_t oper = 0

, numbs = 0;

for (size\_t i = 0; i < expr.length(); i++)

{

if (priorOperation(expr[i])) { oper++; }

else //не должно быть переменной+число, т.е. 5x x5 sg5d 3436f3

{

if (expr[i] != ' ') //пробел пропускаем

{

if (!isNumber(expr[i])) //если встречена потенциальная переменная

{

if (expr[i + 1] != ' ') { cout << "Выражение имеет лишние символы\n"; return 1; } //после неё должен быть пробел

numbs++; //если всё ок, считаем её за число

}

else //если встречено число

{

j = i + 1;

while (expr[j] != ' ') //проверяем, нет ли внутри числа лишнего до пробела

{

if (!isNumber(expr[i])) { cout << "Выражение имеет лишние символы\n"; return 1; }

j++;

}

i = j; //пропускаем все символы до пробела

numbs++;

}

}

}

}

if (oper != (numbs - 1)) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; }

if (initVar(expr)) { return 1; }

return 0;

}

//Проверка прямой польской

bool invalidInputDirectPolish(string& expr)

{

//Последние два объекта - числа или переменные

size\_t j = expr.length() - 1;

while (expr[j] != ' ')

{

if (priorOperation(expr[expr.length() - 1])) { cout << "Выражение должно иметь в конце два числа или переменные\n"; return 1; }

j--;

if (j == 0) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если не найдено пробела или операции

}

j--;

while (expr[j] != ' ')

{

if (priorOperation(expr[expr.length() - 1])) { cout << "Выражение должно иметь в конце два числа или переменные\n"; return 1; }

j--;

if (j == 0) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; } //если не найдено пробела или операции

}

//Всегда начинается на операцию

if (!priorOperation(expr[0])) { cout << "Выражение должно начинаться с операции\n"; return 1; }

//Количество операций = количество чисел - 1

size\_t oper = 0

, numbs = 0;

for (size\_t i = 0; i < expr.length(); i++)

{

if (priorOperation(expr[i])) { oper++; }

else //не должно быть переменной+число, т.е. 5x x5 sg5d 3436f3

{

if (expr[i] != ' ') //пробел пропускаем

{

if (!isNumber(expr[i])) //если встречена потенциальная переменная

{

if (expr[i + 1] != ' ') { cout << "Выражение содержит лишние символы\n"; return 1; } //после неё должен быть пробел

numbs++; //если всё ок, считаем её за число

}

else //если встречено число

{

j = i + 1;

while (j < expr.length() && expr[j] != ' ') //проверяем, нет ли внутри числа лишних символов до пробела

{

if (!isNumber(expr[i])) { cout << "Выражение содержит лишние символы\n"; return 1; }

j++;

}

i = j; //пропускаем все символы до пробела

numbs++;

}

}

}

}

if (oper != (numbs - 1)) { cout << "Количество операций не соответствует количеству чисел\n"; return 1; }

if (initVar(expr)) { return 1; }

return 0;

}

//Вычисление в обратной

float calculate(string expr)

{

float result = 0; //обработанное выражение

stack\* numbs = 0; //стек для чисел

size\_t k = 0; //индекс строки

string strval = "\0"; //строка для чисел/выражений

float tmp1 = 0

, tmp2 = 0;

//все числа - в стек

//как встречена операция - забираем числа, используем на них операцию

do {

cout << "Рассматриваем символ " << expr[k] << "\n";

if (priorOperation(expr[k])) //если является операцией

{

cout << "Символ " << expr[k] << " является операцией\n";

strval = "\0"; //очищение строку

tmp2 = stof(outputStk(numbs, 3)); //вытаскиваем второе выражение

cout << "Вытащили второе выражение из стека " << tmp2 << " \n";

tmp1 = stof(outputStk(numbs, 3)); //вытаскиваем первое выражение

cout << "Вытащили первое выражение из стека " << tmp1 << " \n";

switch (expr[k])

{

case '+':

result = tmp1 + tmp2;

cout << "Сложили и получили число " << result << " \n";

break;

case '-':

result = tmp1 - tmp2;

cout << "Вычли и получили число " << result << " \n";

break;

case '\*':

result = tmp1 \* tmp2;

cout << "Умножили и получили число " << result << " \n";

break;

case '/':

result = tmp1 / tmp2;

cout << "Разделили и получили число " << result << " \n";

break;

}

strval = to\_string(result);

cout << "Добавляем его в стек \n";

push(numbs, strval); //Добавляем в стек получившееся выражение

k++; //переходим на следующий символ

}

else //если число или пробел

{

if (isNumber(expr[k]))

{

cout << "Символ " << expr[k] << " является числом\n";

strval = "\0"; //очищение строку

while (!(expr[k] == ' ')) //пока текущий элемент НЕ является пробелом

{

strval += expr[k]; //считываем числа

k++;

}

cout << "Получаем число " << strval << ". Оно попадает в стек\n";

push(numbs, strval); //все числа попадают в стек

}

k++;

}

} while (expr[k]);

result = stof(outputStk(numbs, 3)); //вытаскиваем получившееся выражение

return result;

}

// == ФУНКЦИИ ПО ПУНКТАМ ==

//Обычное выражение в обратную польскую нотацию

string toPolishNotation()

{

string expr; //выражение пользователя

string result; //результат преобразования

//Ввод выражения двумя способами: с клавиатуры и \* с файла

cout << "Введите выражение: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInput(expr))

{

result = polishNotation(expr);

cout << "Преобразованное выражение: " << result << '\n';

}

else { cout << "Ошибка! Некорректное выражение\n"; return "\0"; }

//После обработанного выражения:

return result;

}

//Обратную польскую в обычное

string fromPolishNotationToInfix()

{

string expr; //выражение пользователя

string result; //результат преобразования

//Ввод выражения двумя способами: с клавиатуры и \* с файла

cout << "Введите выражение: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputPolish(expr))

{

result = fromPolishNotation(expr);

cout << "Преобразованное выражение: " << result << '\n';

}

else { cout << "Ошибка! Некорректное выражение\n"; return "\0"; }

//После обработанного выражения:

return result;

}

//Обычное выражение в прямую польскую нотацию

string toDirectPolishNotation()

{

string expr; //выражение пользователя

string result; //результат преобразования

//Ввод выражения двумя способами: с клавиатуры и \* с файла

cout << "Введите выражение: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInput(expr))

{

result = directPolishNotation(expr);

cout << "Преобразованное выражение: " << result << "\n";

}

else { cout << "Ошибка! Некорректное выражение\n"; return "\0"; }

//После обработанного выражения:

return result;

}

//Прямую польскую в обычное

string fromDirectPolishNotationToInfix()

{

string expr; //выражение пользователя

string result; //результат преобразования

//Ввод выражения двумя способами: с клавиатуры и \* с файла

cout << "Введите выражение: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputDirectPolish(expr))

{

result = fromDirectPolishNotation(expr);

cout << "Преобразованное выражение: " << result << "\n";

}

else { cout << "Ошибка! Некорректное выражение\n"; return "\0"; }

//После обработанного выражения:

return result;

}

//Обратную польскую в прямую

string fromPolishNotationToDirect()

{

string expr; //выражение пользователя

string result; //результат преобразования

//Ввод выражения двумя способами: с клавиатуры и \* с файла

cout << "Введите выражение: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputPolish(expr))

{

result = fromPolishNotation(expr);

result = directPolishNotation(result);

cout << "Преобразованное выражение: " << result << '\n';

}

else { cout << "Ошибка! Некорректное выражение\n"; return "\0"; }

//После обработанного выражения:

return result;

}

//Прямую польскую в обратную

string fromDirectToPolishNotation()

{

string expr; //выражение пользователя

string result; //результат преобразования

//Ввод выражения двумя способами: с клавиатуры и \* с файла

cout << "Введите выражение: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputDirectPolish(expr))

{

result = fromDirectPolishNotation(expr);

result = polishNotation(result);

cout << "Преобразованное выражение: " << result << '\n';

}

else { cout << "Ошибка! Некорректное выражение\n"; return "\0"; }

//После обработанного выражения:

return result;

}

//Проверка на корректность обычного выражения

void taskInvalidInput()

{

string expr; //выражение пользователя

cout << "Введите выражение, которое нужно проверить на корректность: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInput(expr)) { cout << "Следующее выражение является правильным: " << expr << "\n"; }

else { cout << "Выражение содержит ошибку\n"; }

}

//Проверка на корректность обратной польской записи

void taskInvalidInputPolish()

{

string expr; //выражение пользователя

cout << "Введите выражение, которое нужно проверить на корректность: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputPolish(expr)) { cout << "Следующее выражение является правильным: " << expr << "\n"; }

else { cout << "Выражение содержит ошибку\n"; }

}

//Проверка на корректность прямой польской записи

void taskInvalidInputDirectPolish()

{

string expr; //выражение пользователя

cout << "Введите выражение, которое нужно проверить на корректность: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputDirectPolish(expr)) { cout << "Следующее выражение является правильным: " << expr << "\n"; }

else { cout << "Выражение содержит ошибку\n"; }

}

//Вычисление в обычной форме

void taskCalculateInfix()

{

string expr; //выражение пользователя

cout << "Введите выражение, которое нужно вычислить: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInput(expr))

{

expr = polishNotation(expr);

cout << "Ответ: " << calculate(expr) << "\n";

}

else { cout << "Выражение некорректно\n"; }

}

//Вычисление обратной польской

void taskCalculate()

{

string expr; //выражение пользователя

cout << "Введите выражение, которое нужно вычислить: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputPolish(expr))

{

cout << "Ответ: " << calculate(expr) << "\n";

}

else { cout << "Выражение некорректно\n"; }

}

//Вычисление прямой польской

void taskCalculateDirect()

{

string expr; //выражение пользователя

cout << "Введите выражение, которое нужно вычислить: ";

getline(cin, expr);

expr = DelSpaces(expr); //удаляет пробелы в начале и в конце, пробелы, идущие подряд

if (!invalidInputDirectPolish(expr))

{

expr = fromDirectPolishNotation(expr);

expr = polishNotation(expr);

cout << "Ответ: " << calculate(expr) << "\n";

}

else { cout << "Выражение некорректно\n"; }

}

int lb3()

{

setlocale(0, "");

string result = "\0"; //строка выхода

float res = 0;

bool check = true; //выход из меню

bool check1 = false; //выход из подменю

//false - заканчивает цикл, приводя непосредственно к выходу

do {

char sw = ' '; //переключатель главного меню

char sw1 = ' '; //переключатель саб-меню

cout << "\nВыберите нужный раздел: \n";

cout << "\x1b[32m[1]\x1b[0m Преобразование введённого выражения\n";

cout << "\x1b[32m[2]\x1b[0m Проверить выражение на корректность\n";

cout << "\x1b[32m[3]\x1b[0m Вычислить выражение\n";

cout << "\x1b[32m[4]\x1b[0m Очистить экран консоли\n";

cout << "\x1b[32m[0]\x1b[0m Выйти в главное меню\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw;

while (cin.get() != '\n') { sw = ' '; }; //если строка содержит более одного символа, возвращается ошибка

switch (sw)

{

case '1': //[1] Преобразование введённого выражения

do {

check1 = false;

sw1 = ' ';

cout << "\n\x1b[32m[1]\x1b[0m Ввести обычное выражение и преобразовать в обратную польскую запись\n";

cout << "\x1b[32m[2]\x1b[0m Ввести обычное выражение и преобразовать в прямую польскую запись\n";

cout << "\x1b[32m[3]\x1b[0m Ввести обратную польскую запись и преобразовать в обычное выражение\n";

cout << "\x1b[32m[4]\x1b[0m Ввести прямую польскую запись и преобразовать в обычное выражение\n";

cout << "\x1b[32m[3]\x1b[0m Ввести обратную польскую запись и преобразовать в прямую польскую запись\n";

cout << "\x1b[32m[4]\x1b[0m Ввести прямую польскую запись и преобразовать в обратную польскую запись\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] обычное выражение => обратная польская

toPolishNotation();

break;

case '2': //[2] обычное выражение => прямая польская

toDirectPolishNotation();

break;

case '3': //[3] обратная польская => обычное выражение

fromPolishNotationToInfix();

break;

case '4': //[4] прямая польская => обычное выражение

fromDirectPolishNotationToInfix();

break;

case '5': //[5] обратная польская => прямая польская

fromPolishNotationToDirect();

break;

case '6': //[6] прямая польская => обратная польская

fromDirectToPolishNotation();

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '2': //[2] Проверить выражение на корректность

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Проверить на корректность простого выражения\n";

cout << "\x1b[32m[2]\x1b[0m Проверить на корректность выражения в обратной польской записи\n";

cout << "\x1b[32m[3]\x1b[0m Проверить на корректность выражения в прямой польской записи\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] корректность простого выражения

taskInvalidInput();

break;

case '2': //[2] в обратной польской записи

taskInvalidInputPolish();

break;

case '3': //[3] в прямой польской записи

taskInvalidInputDirectPolish();

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '3': //[3] Вычислить выражение

do {

check1 = false;

sw1 = ' ';

cout << "\nВыберите, что хотите сделать: \n";

cout << "\x1b[32m[1]\x1b[0m Вычислить обычное выражение\n";

cout << "\x1b[32m[2]\x1b[0m Вычислить выражение в обратной польской записи\n";

cout << "\x1b[32m[3]\x1b[0m Вычислить выражение в прямой польской записи\n";

cout << "\x1b[32m[0]\x1b[0m Вернуться назад\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw1;

while (cin.get() != '\n') { sw1 = ' '; };

switch (sw1)

{

case '1': //[1] Вычислить обычное выражение

taskCalculateInfix();

break;

case '2': //[2] Вычислить выражение в обратной польской записи

taskCalculate();

break;

case '3': //[3] Вычислить выражение в прямой польской записи

taskCalculateDirect();

break;

case '0': //[0] Назад

break;

default:

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

check1 = true; //цикл пойдёт заново

break;

}

} while (check1);

break;

case '4': //[4] Очистка экрана

system("cls");

break;

case '0': //[0] Закрыть программу

cout << "Выход из программы...\n";

check = false; //выход из цикла

break;

default: //в случае, если введено что-то иное

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

break;

}

} while (check);

system("Pause");

return 0;

}

int main()

{

setlocale(0, "");

bool check = true; //выход из меню

//false - заканчивает цикл, приводя непосредственно к выходу

do {

//system("cls");

char sw = ' '; //переключатель главного меню

cout << "\nВыберите нужный раздел: \n";

cout << "\x1b[32m[1]\x1b[0m Работа со структурами; Записи о студентах\n";

cout << "\x1b[32m[2]\x1b[0m Работа с динамическими массивами и двусвязными списками \n";

cout << "\x1b[32m[3]\x1b[0m Работа со стеками; Польская и обратная нотация\n";

cout << "\x1b[32m[0]\x1b[0m Закрыть программу\n";

cout << "Пожалуйста, введите число, чтобы выполнить нужное действие: ";

cin >> sw;

while (cin.get() != '\n') { sw = ' '; }; //если строка содержит более одного символа, возвращается ошибка

switch (sw)

{

case '1': //[1] Создание целочисленного одномерного массива

lb1();

break;

case '2': //[2] Работа с элементами массива

lb2();

break;

case '3': //[3] Польская нотация, стеки

lb3();

break;

case '0': //[0] Закрыть программу

cout << "Выход из программы...\n";

check = false; //выход из цикла

break;

default: //в случае, если введено что-то иное

cout << "Ошибка! Пожалуйста, попробуйте снова\n";

break;

}

} while (check);

return 0;

}