**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра Информационных систем**

отчет

**по лабораторной работе №2**

**по дисциплине «Программирование»**

Тема: Динамические линейные структуры данных

|  |  |  |
| --- | --- | --- |
| Студент гр. 0323 |  | Кольцов К.Э |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2021

**Цель работы.**

изучение свойств и организация динамических массивов и двусвязных списков; получение практических навыков в работе с динамическими массивами и двусвязными списками; проведение сравнительной характеристики скорости вставки, получения и удаления элементов из них.

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.

**Основные теоретические положения.**

При работе программы, выделенная ей память делится на 4 части:

1. код программы – хранит инструкции по обработке данных;
2. Область глобальных данных – хранит глобальные переменные, которые существуют все время работы программы;
3. Динамическая область памяти – хранит данные с неопределенной размерностью, созданные во время работы программы;
4. Стек – хранит обычные переменные, живут лишь в пределах своего блока кода. Управление памятью аппаратное.

В языке C++ реализованы операторы new и delete, для работы с динамической памятью.

new выделяет место в динамической памяти и возвращает адрес

delete удаляет объект по ссылке.

В данной работе рассматриваются две реализации массивов с неопределенной размерностью. Первый вариант – динамический массив. Работает как привычный массив, но если понадобиться больше места, чем изначально было задумано, то можно создать новый динамический массив, перезаписать в него имеющийся и сделать новый массив первым, а старый массив удалить.

Вторым вариантом является двусвязный список. В нем каждый элемент имеет три поля: фактическое (любой необходимый тип данных), и ссылки на предыдущий и следующий элементы. Так первый элемент имеет нулевую ссылку на предыдущего, а второй элемент имеет нулевую ссылку на следующего. За счет этого уменьшаются затраты на добавление элемента середину списка, ведь достаточно переписать два элемента, а не все за вставленным.

Для сравнения будут использоваться идентичные задания для двух реализаций динамических линейных структур и подсчет времени выполнения.

**Экспериментальные результаты.**

Программа начинается с выбора режима работы. Всего режима три:

s: работа с односвязным списком

d: работа с двусвязным списком

w: режим сравнения, где одна команда меняет оба списка.

Команда e в любом месте программы завершает ее работу.

Команды, доступные после выбора режима:

h: список команд;

n: заново задать список

v: вставить новый элемент в список

r: удалить элемент из списка

g: получить элемент из списка

w: сменить режим работы.

e: завершить работу программы.

Команда h выводит список команд, команда e завершает программу, остальные команды вызывают соответствующие методы в коде программы.

Команда n позволяет задать список тремя способами (в режиме сравнения только из файла):

f: из файла;

r: создать случайно, размер определяет пользователь

c: ввести с клавиатуры.

Команда v вставляет элемент. Сначала запрашивается элемент, затем индекс, куда этот элемент разместить.

Команда r удаляет элемент. Работает в режиме по значению (удаляются все элементы с подходящим значением) или по индексу (удаляется один элемент).

Команда g выводит элемент на экран. Работает в режиме по значению (возвращается индекс первого подходящего элемента) или по индексу (возвращается значение элемента).

Команда w позволяет снова выбрать режим работы. При смене режима старые структуры данных очищаются.

Команда e заканчивает работу программы.

Работу каждой функции можно просмотреть отдельно (см. рис. 1, 2, 3)

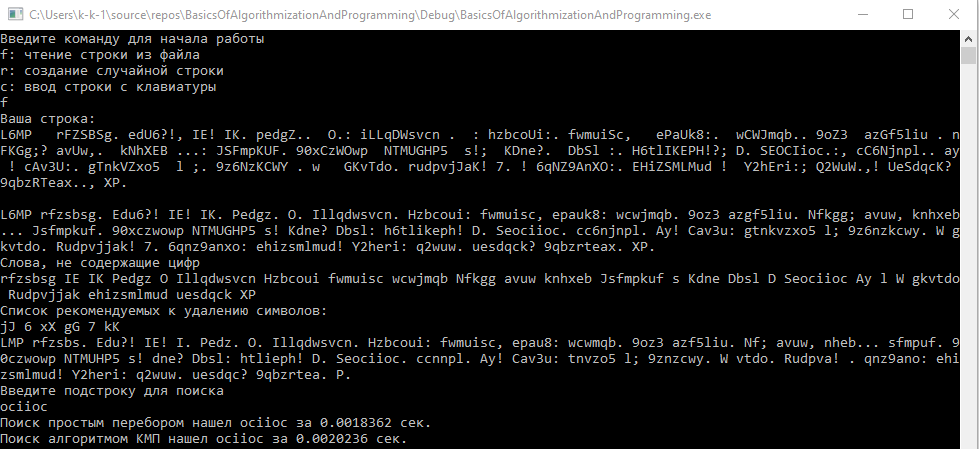


Рисунок 1 – Начало работы программы

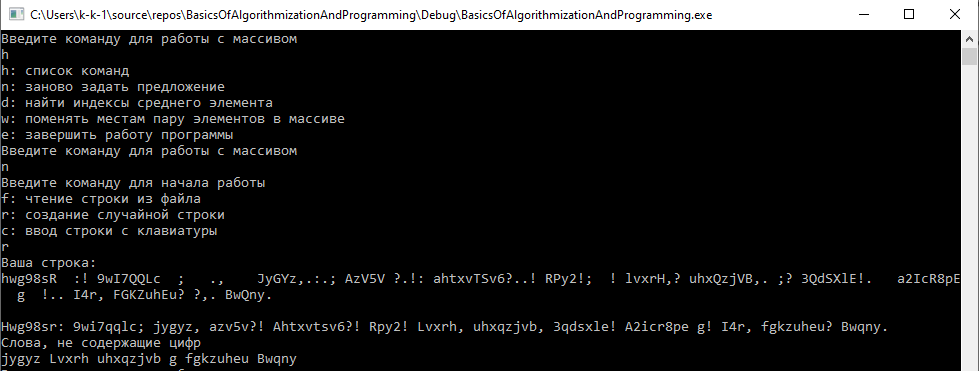


Рисунок 2 – Команды h и n

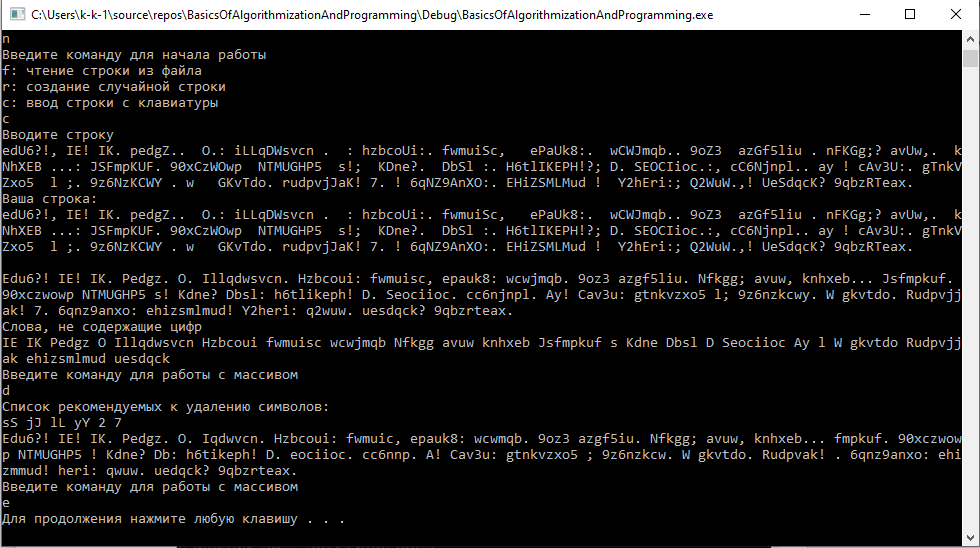


Рисунок 3 – Варианты команды n и выход из программы

**Обработка результатов эксперимента.**

Программа выводит результат корректной обработки строки и правильно находит подстроку в предложении.

**Выводы.**

В ходе данной лабораторной работы я изучил различные способы работы со строчным типом данных в c++.

ПРИЛОЖЕНИЕ А

Листинг программы

#include <iostream>

#include <string>

#include <chrono>

#include <ctime>

#include <fstream>

unsigned singleN = 0; // Глобальная переменная, для постоянного доступа к размеру односвязного списка

unsigned doubleN = 0; // Глобальная переменная, для постоянного доступа к размеру двусвязного списка

struct singleList // элемент односвязанного списка, ссылка первого элемента является самим списком

{

int data;

singleList \*tail;

};

struct doubleList // элемент двусвязанного списка, ссылка первого элемента является самим списком

{

int data;

doubleList \*head;

doubleList \*tail;

};

/\*

Три метода, создающие список, первый элемент которого сохраняется в переданной ссылке

Перед созданием проводится освобождение памяти от прежде записанного списка

все методы по созданию списка двойную ссылку на элемент, и перегружены по одно- и двусвязному

\*/

void userCreate(singleList\*\*);

void userCreate(doubleList\*\*);

void randomCreate(singleList\*\*);

void randomCreate(doubleList\*\*);

void fileCreate(singleList\*\*);

void fileCreate(doubleList\*\*);

void remove(singleList\*\*); // Метод для удаления созданного списка

void remove(doubleList\*\*);

/\*

Метод для вставки элемента элемента по индексу.

Принимает ссылку на список, значение и опционально индекс, без индекса вставка в конец списка.

Если индекс за пределами списка, добавляются промежуточные нулевые элементы и сам элемент

\*/

void add(singleList\*\*, int, unsigned = singleN);

void add(doubleList\*\*, int, unsigned = doubleN);

// Метод для удаления элемента по значению, удаляет все элементы с полученным значением

void removeEquels(singleList\*\*, int);

void removeEquels(doubleList\*\*, int);

void removeIndex(singleList\*\*, unsigned);// Метод для удаления по индексу

void removeIndex(doubleList\*\*, unsigned);

// Возвращает индекс первого элемент списка, равный параметру. Если нет такого - возвращает -1

int getEquels(singleList\*, int);

int getEquels(doubleList\*, int);

int getIndex(singleList\*, unsigned); // Возвращает значение элемента по индексу

int getIndex(doubleList\*, unsigned);

// Метод, выводящий список на экран. В отличии от остольных, получает \*list, то есть копию ссылки на список

void show(singleList\*);

void show(doubleList\*);

int main()

{

setlocale(LC\_ALL, "Russian"); //для корректного вывода текста

std::cout.setf(std::ios::fixed); //для кореектного вывода дробных чисел

std::cout.precision(7); //только первые 7 знаков после запятой значащие для вывода временного интервала

std::srand(time(0)); //для новых значений каждый раз

singleList \*sList = 0;

doubleList \*dList = 0;

char answer;

bool flag = true, flag1 = true, flag2 = true;

while (flag) {//цикл для приема команд от пользователя

std::cout << "Выберите тип работы:\n" \

"s: работа с односвязным списком\n" \

"d: работа с двусвязным\n" \

"w: режим сравнения, где одна команда меняет оба списка\n" \

"e: завершить работу программы\n";

std::cin >> answer;

std::cin.ignore(INT\_MAX, '\n');

switch (answer)

{

case 's':

flag1 = true;

while (flag1) {//цикл для приема команд от пользователя

std::cout << "Введите команду для работы с односвязным списком\n" \

"Подсказка: команда h выводит все доступные команды\n";

std::cin >> answer;

std::cin.ignore(INT\_MAX, '\n');

switch (answer)

{

case 'h':

std::cout << "h: список команд\n" \

"n: заново задать список\n" \

"v: вставить новый элемент в список\n" \

"r: удалить элемент из списка\n" \

"g: получить элемент из списка\n" \

"w: сменить режим работы. Внимание, при смене режима работы списки обнуляются\n" \

"e: завершить работу программы\n";

break;

case 'n':

flag2 = true;

std::cout << "Выберете метод формирования списка\n" \

"f: чтение из файла\n" \

"r: создание случайной строки\n" \

"c: ввод строки с клавиатуры\n";

while (flag2) {//цикл для создания нового предложения

std::cin >> answer;

std::cin.ignore(INT\_MAX, '\n');

switch (answer)

{

case 'f':

fileCreate(&sList);

flag2 = false;

break;

case 'r':

randomCreate(&sList);

flag2 = false;

break;

case 'c':

userCreate(&sList);

flag2 = false;

break;

default:

std::cout << "Вы выбрали несуществующую команду\n";

break;

}

}

answer = 'n';

break;

case 'v':

std::cout << "Введите число\n";

int num, index;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index <= singleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

add(&sList, num, index);

break;

case 'r':

std::cout << "Удалять по индексу? y/n\n";

std::cin >> answer;

if (answer == 'y') {

int index;

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index < singleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

removeIndex(&sList, index);

}

else {

std::cout << "Введите число\n";

int num;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

removeEquels(&sList, num);

}

break;

case 'g':

std::cout << "Искать по индексу? y/n\n";

std::cin >> answer;

if (answer == 'y') {

int index;

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index < singleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Значение элемента по индексу " << index << " = " << getIndex(sList, index) << '\n';

}

else {

std::cout << "Введите число\n";

int num;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Первый элемент со значением " << num << " находится по индексу " << getEquels(sList, num) << '\n';

}

break;

case 'w':

remove(&sList);

flag1 = false;;

break;

case 'e':

flag1 = false;

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду";

break;

}

}

break;

case 'd':

flag1 = true;

while (flag1) {//цикл для приема команд от пользователя

std::cout << "Введите команду для работы с двусвязным списком\n" \

"Подсказка: команда h выводит все доступные команды\n";

std::cin >> answer;

std::cin.ignore(INT\_MAX, '\n');

switch (answer)

{

case 'h':

std::cout << "h: список команд\n" \

"n: заново задать список\n" \

"v: вставить новый элемент в список\n" \

"r: удалить элемент из списка\n" \

"g: получить элемент из списка\n" \

"w: сменить режим работы. Внимание, при смене режима работы списки обнуляются\n" \

"e: завершить работу программы\n";

break;

case 'n':

flag2 = true;

std::cout << "Выберете метод формирования списка\n" \

"f: чтение из файла\n" \

"r: создание случайной строки\n" \

"c: ввод строки с клавиатуры\n";

while (flag2) {//цикл для создания нового предложения

std::cin >> answer;

std::cin.ignore(INT\_MAX, '\n');

switch (answer)

{

case 'f':

fileCreate(&dList);

flag2 = false;

break;

case 'r':

randomCreate(&dList);

flag2 = false;

break;

case 'c':

userCreate(&dList);

flag2 = false;

break;

default:

std::cout << "Вы выбрали несуществующую команду\n";

break;

}

}

answer = 'n';

break;

case 'v':

std::cout << "Введите число\n";

int num, index;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index <= doubleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

add(&dList, num, index);

break;

case 'r':

std::cout << "Удалять по индексу? y/n\n";

std::cin >> answer;

if (answer == 'y') {

int index;

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index < doubleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

removeIndex(&dList, index);

}

else {

std::cout << "Введите число\n";

int num;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

removeEquels(&dList, num);

}

break;

case 'g':

std::cout << "Искать по индексу? y/n\n";

std::cin >> answer;

if (answer == 'y') {

int index;

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index < doubleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Значение элемента по индексу " << index << " = " << getIndex(dList, index) << '\n';

}

else {

std::cout << "Введите число\n";

int num;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Первый элемент со значением " << num << " находится по индексу " << getEquels(dList, num) << '\n';

}

break;

case 'w':

remove(&dList);

flag1 = false;;

break;

case 'e':

flag1 = false;

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду";

break;

}

}

break;

case 'w':

flag1 = true;

while (flag1) {//цикл для приема команд от пользователя

std::cout << "Введите команду для работы с одно- и двусвязными списками и сравнить время выполнения\n" \

"Подсказка: команда h выводит все доступные команды\n";

std::cin >> answer;

std::cin.ignore(INT\_MAX, '\n');

switch (answer)

{

case 'h':

std::cout << "h: список команд\n" \

"n: заново задать список\n" \

"v: вставить новый элемент в список\n" \

"r: удалить элемент из списка\n" \

"g: получить элемент из списка\n" \

"w: сменить режим работы. Внимание, при смене режима работы списки обнуляются\n" \

"e: завершить работу программы\n";

break;

case 'n':

flag2 = true;

std::cout << "Для режима сравнения доступна лишь опция создания из файла, его можно корректировать во время работы программы\n";

fileCreate(&sList);

fileCreate(&dList);

break;

case 'v':

std::cout << "Введите число\n";

int num, index;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index <= doubleN) && (index <= singleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

add(&sList, num, index);

add(&dList, num, index);

break;

case 'r':

std::cout << "Удалять по индексу? y/n\n";

std::cin >> answer;

if (answer == 'y') {

int index;

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index < doubleN) && (index < singleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

removeIndex(&sList, index);

removeIndex(&dList, index);

}

else {

std::cout << "Введите число\n";

int num;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

removeEquels(&sList, num);

removeEquels(&dList, num);

}

break;

case 'g':

std::cout << "Искать по индексу? y/n\n";

std::cin >> answer;

if (answer == 'y') {

int index;

std::cout << "Введите индекс\n";

while (true) {

std::cin >> index;

if ((std::cin) && (index >= 0) && (index < doubleN) && (index < singleN))

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Значение элемента по индексу " << index << " = " << getIndex(sList, index) << '\n';

std::cout << "Значение элемента по индексу " << index << " = " << getIndex(dList, index) << '\n';

}

else {

std::cout << "Введите число\n";

int num;

while (true) {

std::cin >> num;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

std::cout << "Первый элемент со значением " << num << " находится по индексу " << getEquels(sList, num) << '\n';

std::cout << "Первый элемент со значением " << num << " находится по индексу " << getEquels(dList, num) << '\n';

}

break;

case 'w':

remove(&sList);

remove(&dList);

flag1 = false;;

break;

case 'e':

flag1 = false;

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду";

break;

}

}

break;

case 'e':

flag = false;

break;

default:

std::cout << "Вы указали несуществующую команду";

break;

}

}

system("pause");

return 0;

}

void userCreate(singleList \*\*list) {

singleList \*listPoint = 0;

std::cout << "Введите массив чисел одной строчкой\n";

int num = 0;

std::string arr;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

bool flagAns;

do {

remove(list);

flagAns = false;

std::getline(std::cin, arr);

char \*point = &arr[0];

start = std::chrono::system\_clock::now();

while (\*point) {

bool flagNum = false;

bool flagMinus = false;

while ((\*point < '0') || (\*point > '9')) {

if (!\*point)

break;

if ((\*point != ' ') && ((\*point != '-') || (\*(point + 1) < '0') || (\*(point + 1) > '9'))) {

std::cout << "В вашем вводе некоректный символ, используйте пробелы для разделения чисел\n";

flagAns = true;

break;

}

if (\*point == '-')

flagMinus = true;

point++;

}

if (flagAns)

break;

while ((\*point >= '0') && (\*point <= '9')) {

flagNum = true;

if (flagMinus)

num = num \* 10 - \*point + 48;

else

num = num \* 10 + \*point - 48;

point++;

}

if (flagNum) {

if (!\*list) {

\*list = new singleList{ num, 0 };

listPoint = \*list;

}

else {

listPoint->tail = new singleList{ num, 0 };

listPoint = listPoint->tail;

}

num = 0;

singleN++;

}

}

} while (flagAns);

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Создание пользовательского односвязного списка заняло " << listTime.count() << " сек.\n\n";

}

void userCreate(doubleList \*\*list) {

std::cout << "Введите массив чисел одной строчкой\n";

int num = 0;

std::string arr;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

bool flagAns;

do {

remove(list);

flagAns = false;

std::getline(std::cin, arr);

char \*point = &arr[0];

start = std::chrono::system\_clock::now();

while (\*point) {

bool flagNum = false;

bool flagMinus = false;

while ((\*point < '0') || (\*point > '9')) {

if (!\*point)

break;

if ((\*point != ' ') && ((\*point != '-') || (\*(point + 1) < '0') || (\*(point + 1) > '9'))) {

std::cout << "В вашем вводе некоректный символ, используйте пробелы для разделения чисел\n";

flagAns = true;

break;

}

if (\*point == '-')

flagMinus = true;

point++;

}

if (flagAns)

break;

while ((\*point >= '0') && (\*point <= '9')) {

flagNum = true;

if (flagMinus)

num = num \* 10 - \*point + 48;

else

num = num \* 10 + \*point - 48;

point++;

}

if (flagNum) {

if (!\*list) {

\*list = new doubleList{ num, 0, 0 };

}

else {

while ((\*list)->tail)

\*list = (\*list)->tail;

(\*list)->tail = new doubleList{ num, \*list, 0 };

while ((\*list)->head)

\*list = (\*list)->head;

}

num = 0;

doubleN++;

}

}

} while (flagAns);

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Создание пользовательского двусвязного списка заняло " << listTime.count() << " сек.\n\n";

}

void randomCreate(singleList \*\*list) {

remove(list);

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

std::cout << "Введите количество элементов\n";

while (true) {

std::cin >> singleN;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

start = std::chrono::system\_clock::now();

\*list = new singleList{ std::rand() % 100, 0 };

singleList \*listPoint = \*list;

for (int i = 1; i < singleN; i++) {

listPoint->tail = new singleList{ std::rand() % 100, 0 };;

listPoint = listPoint->tail;

}

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Создание рандомного односвязного списка заняло " << listTime.count() << " сек.\n\n";

}

void randomCreate(doubleList \*\*list) {

remove(list);

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

std::cout << "Введите количество элементов\n";

while (true) {

std::cin >> doubleN;

if (std::cin)

break;

std::cout << "Вы неправильно ввели число. Повторите ввод\n";

std::cin.clear();

std::cin.ignore(INT\_MAX, '\n');

}

start = std::chrono::system\_clock::now();

\*list = new doubleList{ std::rand() % 100, 0, 0 };

for (int i = 1; i < doubleN; i++) {

(\*list)->tail = new doubleList{ std::rand() % 100, \*list, 0 };

\*list = (\*list)->tail;

}

while ((\*list)->head)

\*list = (\*list)->head;

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Создание рандомного двусвязного списка заняло " << listTime.count() << " сек.\n\n";

}

void fileCreate(singleList \*\*list) {

std::string arr;

std::ifstream in;

remove(list);

in.open("SimpleArr.txt");

if (in.is\_open()) {

std::getline(in, arr);

int num = 0;

singleList \*listPoint = 0;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

char \*point = &arr[0];

bool flagAns = false;;

start = std::chrono::system\_clock::now();

while (\*point) {

bool flagNum = false;

bool flagMinus = false;

while ((\*point < '0') || (\*point > '9')) {

if (!\*point)

break;

if ((\*point != ' ') && ((\*point != '-') || (\*(point + 1) < '0') || (\*(point + 1) > '9'))) {

std::cout << "В файле некоректный символ, используйте пробелы для разделения чисел\n" \

"Изменить файл можно незакрывая программу\n";

flagAns = true;

break;

}

if (\*point == '-')

flagMinus = true;

point++;

}

if (flagAns) {

remove(list);

break;

}

while ((\*point >= '0') && (\*point <= '9')) {

flagNum = true;

if (flagMinus)

num = num \* 10 - \*point + 48;

else

num = num \* 10 + \*point - 48;

point++;

}

if (flagNum) {

if (!\*list) {

\*list = new singleList{ num, 0 };

listPoint = \*list;

}

else {

listPoint->tail = new singleList{ num, 0 };

listPoint = listPoint->tail;

}

num = 0;

singleN++;

}

}

end = std::chrono::system\_clock::now();

listTime = end - start;

if (!flagAns) {

show(\*list);

std::cout << "Создание односвязного списка из файла заняло " << listTime.count() << " сек.\n\n";

}

}

else

std::cout << "Файл не доступен\n";

in.close();

}

void fileCreate(doubleList \*\*list) {

std::string arr;

std::ifstream in;

remove(list);

in.open("SimpleArr.txt");

if (in.is\_open()) {

std::getline(in, arr);

int num = 0;

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

char \*point = &arr[0];

bool flagAns = false;;

start = std::chrono::system\_clock::now();

while (\*point) {

bool flagNum = false;

bool flagMinus = false;

while ((\*point < '0') || (\*point > '9')) {

if (!\*point)

break;

if ((\*point != ' ') && ((\*point != '-') || (\*(point + 1) < '0') || (\*(point + 1) > '9'))) {

std::cout << "В файле некоректный символ, используйте пробелы для разделения чисел\n" \

"Изменить файл можно незакрывая программу\n";

flagAns = true;

break;

}

if (\*point == '-')

flagMinus = true;

point++;

}

if (flagAns) {

remove(list);

break;

}

while ((\*point >= '0') && (\*point <= '9')) {

flagNum = true;

if (flagMinus)

num = num \* 10 - \*point + 48;

else

num = num \* 10 + \*point - 48;

point++;

}

if (flagNum) {

if (!\*list) {

\*list = new doubleList{ num, 0, 0 };

}

else {

while ((\*list)->tail)

\*list = (\*list)->tail;

(\*list)->tail = new doubleList{ num, \*list, 0 };

while ((\*list)->head)

\*list = (\*list)->head;

}

num = 0;

doubleN++;

}

}

end = std::chrono::system\_clock::now();

listTime = end - start;

if (!flagAns) {

show(\*list);

std::cout << "Создание двусвязного списка из файла заняло " << listTime.count() << " сек.\n\n";

}

}

else

std::cout << "Файл не доступен\n";

in.close();

}

void remove(singleList \*\*list) {

singleList \*buff;

while (\*list) {

buff = \*list;

\*list = (\*list)->tail;

delete buff;

}

singleN = 0;

}

void remove(doubleList \*\*list) {

if (doubleN) {

while ((\*list)->tail) {

\*list = (\*list)->tail;

delete (\*list)->head;

}

delete \*list;

\*list = 0;

doubleN = 0;

}

}

void add(singleList \*\*list, int num, unsigned index) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

singleList \*pointList;

if (!index) {

pointList = new singleList{ num,\*list };

\*list = pointList;

}

else {

pointList = \*list;

for (int i = 1; i < index; i++) {

if (!pointList->tail) {

pointList->tail = new singleList{ 0,0 };

singleN++;

}

pointList = pointList->tail;

}

pointList->tail = new singleList{ num,pointList->tail };

singleN++;

}

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Добавление элемента в односвязный спискок заняло " << listTime.count() << " сек.\n\n";

}

void add(doubleList \*\*list, int num, unsigned index) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

if (!index) {

if (\*list) {

(\*list)->head = new doubleList{ num, 0, \*list };

\*list = (\*list)->head;

}

else

\*list = new doubleList{ num, 0, 0 };

}

else {

for (int i = 1; i < index; i++) {

if (!(\*list)->tail) {

(\*list)->tail = new doubleList{ 0, \*list, 0 };

doubleN++;

}

\*list = (\*list)->tail;

}

(\*list)->tail = new doubleList{ num, \*list, (\*list)->tail };

if ((\*list)->tail->tail)

(\*list)->tail->tail->head = (\*list)->tail;

while ((\*list)->head)

\*list = (\*list)->head;

doubleN++;

}

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Добавление элемента в двусвязный спискок заняло " << listTime.count() << " сек.\n\n";

}

void removeEquels(singleList \*\*list, int num) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

singleList \*pointList = \*list, \*yet = \*list;

while ((\*list)->data == num) {

\*list = (\*list)->tail;

delete pointList;

singleN--;

pointList = \*list;

if (!\*list)

break;

}

while (pointList) {

if (pointList->data == num) {

yet->tail = pointList->tail;

delete pointList;

singleN--;

pointList = yet->tail;

}

else {

yet = pointList;

pointList = pointList->tail;

}

}

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Удаление элемента из односвязного списка заняло " << listTime.count() << " сек.\n\n";

}

void removeEquels(doubleList \*\*list, int num) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

while ((\*list)->data == num) {

doubleN--;

if (!(\*list)->tail) {

delete \*list;

\*list = 0;

break;

}

else {

\*list = (\*list)->tail;

delete (\*list)->head;

(\*list)->head = 0;

}

}

while ((\*list)->tail) {

if ((\*list)->data == num) {

doubleN--;

(\*list)->head->tail = (\*list)->tail;

\*list = (\*list)->head;

delete (\*list)->tail->head;

(\*list)->tail->head = \*list;

}

else

\*list = (\*list)->tail;

}

if ((\*list)->data == num) {

doubleN--;

\*list = (\*list)->head;

delete (\*list)->tail;

(\*list)->tail = 0;

}

while ((\*list)->head)

\*list = (\*list)->head;

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Удаление элемента из двусвязного списка заняло " << listTime.count() << " сек.\n\n";

}

void removeIndex(singleList \*\*list, unsigned index) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

singleList \*pointList;

if (!index) {

pointList = \*list;

\*list = (\*list)->tail;

delete pointList;

}

else {

singleList \*yet = \*list;

for (int i = 1; i < index; i++) {

yet = yet->tail;

}

pointList = yet->tail;

yet->tail = pointList->tail;

delete pointList;

}

singleN--;

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Удаление элемента из односвязного списка заняло " << listTime.count() << " сек.\n\n";

}

void removeIndex(doubleList \*\*list, unsigned index) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

if (!index) {

doubleN--;

if (!(\*list)->tail) {

delete \*list;

\*list = 0;

}

else {

\*list = (\*list)->tail;

delete (\*list)->head;

(\*list)->head = 0;

}

}

else {

for (int i = 1; i <= index; i++) {

\*list = (\*list)->tail;

}

if ((\*list)->tail) {

(\*list)->head->tail = (\*list)->tail;

\*list = (\*list)->head;

delete (\*list)->tail->head;

(\*list)->tail->head = \*list;

}

else {

\*list = (\*list)->head;

delete (\*list)->tail;

(\*list)->tail = 0;

}

}

singleN--;

while ((\*list)->head)

\*list = (\*list)->head;

end = std::chrono::system\_clock::now();

listTime = end - start;

show(\*list);

std::cout << "Удаление элемента из двусвязного списка заняло " << listTime.count() << " сек.\n\n";

}

int getEquels(singleList \*point, int num) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

int i = 0;

while (point) {

if (point->data == num) {

end = std::chrono::system\_clock::now();

listTime = end - start;

std::cout << "Поиск элемента в односвязном списке заняло " << listTime.count() << " сек.\n\n";

return i;

}

point = point->tail;

i++;

}

end = std::chrono::system\_clock::now();

listTime = end - start;

std::cout << "Безуспешный поиск элемента в односвязном списке заняло " << listTime.count() << " сек.\n\n";

return -1;

}

int getEquels(doubleList \*point, int num) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

int i = 0;

while (point) {

if (point->data == num) {

end = std::chrono::system\_clock::now();

listTime = end - start;

std::cout << "Поиск элемента в двусвязном списке заняло " << listTime.count() << " сек.\n\n";

return i;

}

point = point->tail;

i++;

}

end = std::chrono::system\_clock::now();

listTime = end - start;

std::cout << "Безуспешный поиск элемента в двусвязном списке заняло " << listTime.count() << " сек.\n\n";

return -1;

}

int getIndex(singleList \*point, unsigned index) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

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

point = point->tail;

}

end = std::chrono::system\_clock::now();

listTime = end - start;

std::cout << "Поиск элемента в односвязном списке заняло " << listTime.count() << " сек.\n\n";

return point->data;

}

int getIndex(doubleList \*point, unsigned index) {

std::chrono::system\_clock::time\_point start, end;

std::chrono::duration<double> listTime;

start = std::chrono::system\_clock::now();

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

point = point->tail;

}

end = std::chrono::system\_clock::now();

listTime = end - start;

std::cout << "Поиск элемента в двусвязном списке заняло " << listTime.count() << " сек.\n\n";

return point->data;

}

void show(singleList \*point) {

std::cout << "Ваш список:\n";

while (point) {

std::cout << point->data << ' ';

point = point->tail;

}

std::cout << '\n';

}

void show(doubleList \*point) {

std::cout << "Ваш список:\n";

while (point) {

std::cout << point->data << ' ';

point = point->tail;

}

std::cout << '\n';

}