**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра информационных систем**

отчет

**по практической работе №2**

**по дисциплине «Программирование»**

Тема: Линейные структуры данных. Динамические массивы и двусвязные списки

|  |  |  |
| --- | --- | --- |
| Студент гр. 9894 |  | Леушкин С.М. |
| Преподаватель |  | Глущенко А.Г. |

Санкт-Петербург

2020

# **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.

# **Основные теоретические положения.**

Создание динамических массивов позволяет создавать в динамической области памяти массивы объектов с таким количеством элементов, которое необходимо в данный момент работы программы [1]. Для их представления требуется всего одна переменная-указатель, а в самом массиве, на который ссылается этот указатель, может быть столько элементов, сколько потребуется в данный момент времени. На рисунке 1 представлена инициализация динамического массива с действительными числами.



Рисунок 1 – Графическое представление инициализации массива

Одномерный двунаправленный список представляет собой совокупность отдельных элементов, каждый из которых содержит в себе две части – информационную и адресную. Информационная часть предназначена для хранения «полезных» данных и может иметь практически любой тип. Адресная часть каждого узла содержит адрес предыдущего и следующего узла списка. Указатель на предыдущий узел корня списка содержит нулевое значение. Указатель последнего узла также содержит нулевое значение. Схематическое изображение такого списка представлено на рисунке 2.

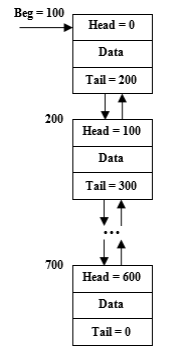


Рисунок 2 – Схематическое представление двусвязного списка

Основные действия, производимые над узлами двусвязного списка:

* инициализация списка;
* добавление узла в список;
* удаление узла из списка;
* удаление корня списка;
* вывод элементов списка;
* вывод элементов списка в обратном порядке;
* взаимообмен двух узлов списка.

# **Постановка задачи.**

Необходимо реализовать программу, которая выполняет следующие действия:

1. Формирование целочисленного одномерного массива размерности *N*, заполнение которого можно реализовать автоматически (пользователь вводит количество элементов в массиве, который будет заполнятся автоматически случайными числами в диапазоне от 0 до 99), в «ручном» режиме (пользователь вводит в консоль элементы массива, *N* определяется автоматически по количеству введенных элементов) и с помощью текстового файла (массив считывается с файла, *N* определяется как количество элементов массива в текстовом файле).
2. Определение скорости создания динамического массива п.1.
3. Реализация вставки, удаления и получения 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. Формирофание двусвязного списка размерности N*,* заполнение которого можно реализовать автоматически (пользователь вводит количество элементов в списке, который будет заполнятся автоматически случайными числами в диапазоне от 0 до 99), в «ручном» режиме (пользователь вводит в консоль элементы списка, *N* определяется автоматически по количеству введенных элементов) и с помощью текстового файла (список считывается с файла, *N* определяется как количество элементов списка в текстовом файле).
6. Определение скорости создания двусвязного списка п.5.
7. Реализация вставки, удаления и получения элемента двусвязного списка. Удаление и получение элемента необходимо реализовать по индексу и по значению.
8. Определение скорости вставки, удаления и получения элемента списка п.7.
9. Индивидуальное задание – увеличение каждого значения исходного двусвязного списка на случайное число в диапазоне от 0 до 5. Добавление в динамический массив всех положительных элементов динамического массива.

# **Выполнение работы.**

Листинг программы представлен в приложении А.

Доступ к функциям программы предоставляется с помощью меню, в котором необходимо выбрать один из пунктов. Меню реализовано в функции menu(). Выбор пунктов меню реализовано в функциях count\_val() и menu\_choice(). Дополнительные подменю также реализованы для двусвязного списка и динамического массива.

Заполнение элементов массива выполнено тремя способами:

1. автоматически – пользователю необходимо ввести размерность массива, а заполнение массива будет производится случайными значениями от 0 до 99. Способ реализован в функции fill\_array\_random().
2. «Ручным» способом – пользователь самостоятельно вводит элементы массива. Для того чтобы остановить ввод элементов, необходимо ввести любой нечисловой символ. Способ реализован в функции fill\_array\_self().
3. Заполнение с помощью текстового файла. Способ реализован в функции fill\_array\_from\_file().

Заполнение элементов двусвязного списка реализовано тремя способами:

1. автоматически – функция fill\_list\_random();
2. «ручным» способом – функция fill\_list\_self();
3. с помощью текстового файла – функция fill\_list\_from\_file().

Требования к текстовому файлу в двух случаях одинаковые: между числовыми значения ставится пробел, постронних знаков быть не должно.

Вставка элемента в массив реализована в функции insert\_array(). Удаление элемента массива реализовано в функции del\_array(). В зависимости от флага fl\_del\_array, будет производиться удаление по значению (флаг TRUE) или по индексу (флаг FALSE). В функции getNth\_array() реализован поиск элемента массива. В зависимости от флага fl\_array, будет производиться поиск по значению (флаг TRUE) или по индексу (флаг FALSE).

Вставка элемента в список реализована в функции insert\_list(). Удаление элемента списка реализовано в функции deleteNth\_list(). В зависимости от флага fl\_del\_list, будет производиться удаление по значению (флаг TRUE) или по индексу (флаг FALSE). В функции getNth\_list() реализован поиск элемента списка. В зависимости от флага fl\_list, будет производиться поиск по значению (флаг TRUE) или по индексу (флаг FALSE).

Индивидуальное задание реализовано в функции variant\_work(). Перед его выполнением необходимо создать массив и двусвязный список. В случае, если один из них (или оба) не будет создан, программа предложит его создать.

Вывод списка реализован в функции printDblLinkedList() и принимает на вход двусвязный список.

Вывод массива реализован в функции printArray() и принимает на вход динамический массив по указателю и размер данного массива.

Также в программе реализована проверка вводимых значений на их корректность в функции checkdigit(). На вход она ничего не принимает, а на выход она возвращает корректное целочисленное значение.

Также в программе реализована проверка ввода индекса на попадание в необходимый диапазон индексов в функции check\_index(). На вход она принимает индекс пользователя и размерность массива или списка. Возвращает фунция булевое значение 1, если индекс попадает в диапазон, или 0, если индекс не попадает в диапазон.

## Тестирование программы.

На рисунках 3 – 9 представлены скриншоты тестирования программы.

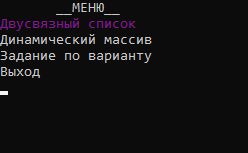


Рисунок 3 – Скриншот меню программы

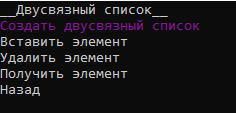


Рисунок 4 – Скриншот подменю двусвязного списка

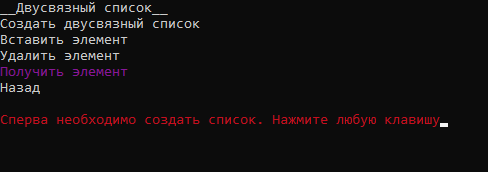


Рисунок 5 – Скриншот выбора пункта «Получить элемент» без созданного списка

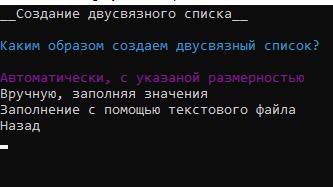


Рисунок 6 – Скриншот подменю заполнения двусвязного списка

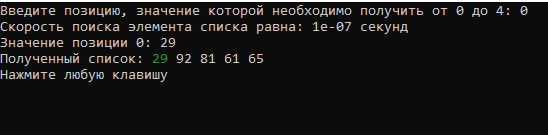


Рисунок 7 – Скриншот поиска элемента списка по индексу

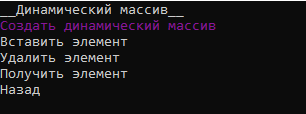


Рисунок 8 – Скриншот подменю динамического массива

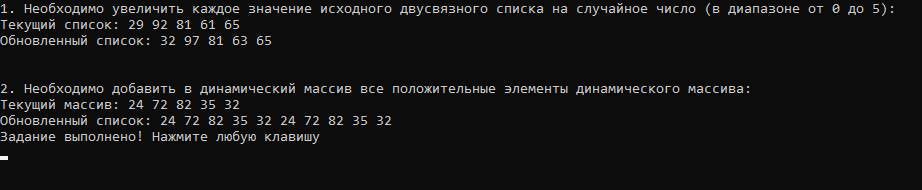


Рисунок 9 – Скриншот выполнения индвидиуального задания

Также было проведено сравнение скорости двусвязного списка и динамического массива при одинаковой размерности. Заполнение значений выполнялось автоматически. Сравнение происходило в создании структуры данных, поиска, вставки и удаления элемента (см. табл. 1).

Таблица 1 – Сравнение скорости двусвязного списка и динамического массива

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Размерность | | 50 | 500 | 1 000 | 5 000 | 10 000 | 20 000 |
| Двусвязный список | Создание, сек | 4,00E-06 | 7,17E-05 | 9,25E-05 | 3,40E-04 | 7,40E-04 | 1,41E-03 |
| Поиск, сек | 1,50E-06 | 2,00E-07 | 1,80E-05 | 8,03E-05 | 1,30E-04 | 2,20E-04 |
| Вставка, сек | 1,80E-06 | 1,05E-05 | 1,19E-05 | 9,30E-06 | 2,20E-06 | 1,90E-06 |
| Удаление, сек | 8,00E-07 | 7,30E-06 | 1,93E-05 | 2,10E-04 | 2,00E-04 | 2,50E-04 |
| Динамический массив | Создание, сек | 1,55E-05 | 4,80E-04 | 1,78E-03 | 2,73E-02 | 0,08996 | 0,32672 |
| Поиск, сек | 1,00E-07 | 8,10E-06 | 2,00E-07 | 2,00E-07 | 3,00E-07 | 2,00E-07 |
| Вставка, сек | 3,10E-06 | 3,40E-06 | 2,80E-06 | 3,18E-05 | 4,59E-05 | 6,67E-05 |
| Удаление, сек | 3,10E-06 | 4,10E-06 | 4,00E-06 | 3,47E-05 | 3,63E-05 | 9,95E-05 |

Из представленных графиков (см. рис. 10 – 13) видно, что динамический массив, при размерности свыше 10000 элементов, уступает двусвязному списку по времени создания и вставки элемента. Можно предположить, что это связано с тем, что во время добавления элемента необходимо создавать полную копию массива, что затратно по времени, в то время как двусвязному списку необходимо лишь изменить адреса следующего и предыдущего узла. Однако, двусвязный список, при размерности свыше 10000 элементов, уступает массиву во время поиска элемента. Можно предположить, что в двусвязном списке это связано с последовательной проверкой условия на совпадение с искомым значением, что сильно сказывается на скорости поиска с большим количеством элементов.

Рисунок 10 – График зависимости скорости от времени создания объекта

Рисунок 11 – График зависимости скорости от времени поиска элемента

Рисунок 12 – График зависимости скорости от времени вставки элемента

Рисунок 13 – График зависимости скорости от времени удаления элемента

# **Выводы.**

Были освоены теоретические знания и закреплены практические навыки по работе со линейными структурами данных. Было проведено исследование по практическому применению линейных структур на примере создания массива и двусвязного списка с их дальнейшей обработкой. Приведённые графики показывают, что использование двусвязных списков могут позволить облегчить и ускорить работу с структурами данных.

# **Список использованных источников.**

1. Основы алгоритмизации и программирования на языке C++. Методические указания к практическим работам / сост.: А.Г. Глущенко. СПб.: Изд-во СПбГЭТУ «ЛЭТИ», 2020. 126с.

Приложение А

ЛИСТИНГ ПРОГРАММЫ

#include <iostream>

#include <conio.h>

#include <ctime>

#include <fstream>

#include <chrono>

#include <Windows.h>

#include <iomanip>

using namespace std;

int m\_count = 0; // счетчик главного меню

int subm\_l\_count = 0; // счетчик подменю списка

int subm\_a\_count = 0; // счетчик подменю массива

int subm\_l\_c\_count = 0; // счетчик подменю создания списка

int subm\_a\_c\_count = 0; // счетчик подменю создания массива

int subm\_l\_s\_count = 0; // счетчик подменю поиска элемента в списке

int subm\_a\_s\_count = 0; // счетчик подменю поиска элемента в массиве

int subm\_l\_d\_count = 0; // счетчик подменю удаления элемента в списке

int subm\_a\_d\_count = 0; // счетчик подменю удаления элемента в массиве

bool fl\_array\_create = FALSE; // флаг создания массива

bool fl\_list\_create = FALSE; // флаг создания списка

void SetColor(int text, int bg) //Функция смены цвета, взятая из Интернета

{

HANDLE hStdOut = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hStdOut, (WORD)((bg << 4) | text));

}

typedef struct \_Node {

int value;

struct \_Node\* next;

struct \_Node\* prev;

} Node;

typedef struct \_DblLinkedList {

size\_t size;

Node\* head;

Node\* tail;

} DblLinkedList;

///////////////////////// PROTOTYPES //////////////////////////

void menu(DblLinkedList\* list, int\* arr, int& arr\_size); // меню

void submenu\_list(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю списка

void submenu\_array(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю массива

void submenu\_list\_create(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю создания списка

void submenu\_array\_create(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю создания массива

void submenu\_list\_search(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю поиска элемента списка

void submenu\_array\_search(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю поиска элемента массива

void submenu\_list\_delete(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю удаления элемента списка

void submenu\_array\_delete(DblLinkedList\* list, int\* arr, int& arr\_size); // подменю удаления элемента массива

void menu\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление меню

void submenu\_list\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю списка

void submenu\_array\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю массива

void submenu\_list\_create\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю создания списка

void submenu\_array\_create\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю создания массива

void submenu\_list\_search\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю поиска элемента списка

void submenu\_array\_search\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю поиска элемента массива

void submenu\_list\_delete\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю удаления элемента списка

void submenu\_array\_delete\_choice(DblLinkedList\* list, int\* arr, int& arr\_size); // управление подменю удаления элемента массива

void printDblLinkedList(DblLinkedList\* list); // вывод списка

///////////////////////// PROTOTYPES //////////////////////////

int checkdigit() // проверка на корректный ввод

{

while (true)

{

int value;

// вводим число, которое хотим представить в двоичной форме

cin >> value; // число целое

if (cin.fail()) // ecли предыдущее извелечение оказлось неудачным,

{

cin.clear(); // то возвращаем cin в обычный режим работы

cin.ignore(32767, '\n'); // и удаляем из буфера значения предыдущего ввода

cout << "Недопустимое заданное число. Введите число правильно" << '\n';

}

else

{

cin.ignore(32767, '\n'); // удаляем из буфера значения предыдущего ввода

return value;

}

}

}

bool check\_index(int value, int size) // проверка попадания в диапазон

{

if (value >= 0 && value <= size)

return TRUE;

else

return FALSE;

}

///////////////////////////// DBLINKEDLIST///////////////

DblLinkedList\* createDblLinkedList() // создание списка

{

DblLinkedList\* tmp = (DblLinkedList\*)malloc(sizeof(DblLinkedList));

tmp->size = 0;

tmp->head = tmp->tail = NULL;

return tmp;

}

void pushFront(DblLinkedList\* list, int data) // добавление узла в начало списка

{

Node\* tmp = (Node\*)malloc(sizeof(Node));

tmp->value = data;

tmp->next = list->head;

tmp->prev = NULL;

if (list->head)

{

list->head->prev = tmp;

}

list->head = tmp;

if (list->tail == NULL)

{

list->tail = tmp;

}

list->size++;

}

void pushBack(DblLinkedList\* list, int value) // добавление узла в конец списка

{

Node\* tmp = (Node\*)malloc(sizeof(Node));

tmp->value = value;

tmp->next = NULL;

tmp->prev = list->tail;

if (list->tail)

{

list->tail->next = tmp;

}

list->tail = tmp;

if (list->head == NULL)

{

list->head = tmp;

}

list->size++;

}

Node\* getNth\_list\_index(DblLinkedList\* list, size\_t index) //получение узла от индекса

{

Node\* tmp = list->head;

size\_t i = 0;

while (tmp && i < index)

{

tmp = tmp->next;

i++;

}

return tmp;

}

bool check\_getNth\_list\_value(DblLinkedList\* list, int value) // проверка существования значения узла в списке

{

Node\* tmp = list->head;

size\_t index = 0;

while (tmp && tmp->value != value)

{

tmp = tmp->next;

index++;

}

if (tmp)

return TRUE;

else

return FALSE;

}

int getNth\_list\_value(DblLinkedList\* list, int value) // получение индекса от значения узла

{

Node\* tmp = list->head;

size\_t index = 0;

while (tmp && tmp->value != value)

{

tmp = tmp->next;

index++;

}

return index;

}

void printDblLinkedList\_index(DblLinkedList\* list, int index) // вывод списка с выделенным значением

{

bool fl\_stop = TRUE;

int count = 0;

Node\* tmp = list->head;

while (tmp)

{

if (count == index && fl\_stop)

{

SetColor(2, 0);

cout << tmp->value << " ";

SetColor(7, 0);

fl\_stop = FALSE;

}

else

cout << tmp->value << " ";

tmp = tmp->next;

count++;

}

cout << endl;

}

void getNth\_list(DblLinkedList\* list, int\* arr, int& arr\_size, bool fl\_list) // функция получения значения или индекса узла списка

{

system("cls"); // очищаем консоль

chrono::steady\_clock sc;

int value;

int index;

if (fl\_list) // TRUE - ищем индекс. FALSE - ищем значение

{

cout << "Введите значение, позицию которого необходимо получить: ";

value = checkdigit();

if (!check\_getNth\_list\_value(list, value))

{

cout << "Такого значения нет. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

submenu\_list(list, arr, arr\_size);

}

else

{

auto start = sc.now(); // устанавливаем начало отсчета времени события

index = getNth\_list\_value(list, value);

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость поиска элемента списка равна: " << time\_span.count() << " секунд" << endl;

cout << "Позиция значения " << value << ": " << index << endl;

cout << "Полученный список: ";

printDblLinkedList\_index(list, index);

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_list(list, arr, arr\_size); // возращаемся в подменю

}

}

else

{

cout << "Введите позицию, значение которой необходимо получить от 0 до " << list->size - 1 << ": ";

index = checkdigit();

if (!check\_index(index, list->size - 1))

{

cout << "Значение выходит за рамки диапазона. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

getNth\_list(list, arr, arr\_size, fl\_list);

}

auto start = sc.now(); // устанавливаем начало отсчета времени события

value = getNth\_list\_index(list, index)->value;

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость поиска элемента списка равна: " << time\_span.count() << " секунд" << endl;

cout << "Значение позиции " << index << ": " << value << endl;

cout << "Полученный список: ";

printDblLinkedList\_index(list, index);

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_list(list, arr, arr\_size); // возращаемся в подменю

}

}

void insert(DblLinkedList\* list, size\_t index, int value) // вставка узла в список

{

Node\* elm = NULL;

Node\* ins = NULL;

elm = getNth\_list\_index(list, index);

ins = (Node\*)malloc(sizeof(Node));

ins->value = value;

ins->prev = elm;

ins->next = elm->next;

if (elm->next)

{

elm->next->prev = ins;

}

elm->next = ins;

if (!elm->prev)

{

list->head = elm;

}

if (!elm->next)

{

list->tail = elm;

}

list->size++;

}

void insert\_list(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

chrono::steady\_clock sc;

int index;

int value;

printDblLinkedList(list);

cout << endl << "Введите позицию нового значения от 0 до " << list->size - 1 << ": ";

index = checkdigit();

if (!check\_index(index, list->size - 1))

{

cout << "Значение выходит за рамки диапазона. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

insert\_list(list, arr, arr\_size);

}

cout << "Введите новое значение: ";

value = checkdigit();

auto start = sc.now(); // устанавливаем начало отсчета времени события

if (index == 0)

pushFront(list, value);

else

if (index == list->size - 1)

{

pushBack(list, value);

index++;

}

else

insert(list, index - 1 , value);

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость вставки элемента списка равна: " << time\_span.count() << " секунд" << endl;

cout << "Новый список: ";

printDblLinkedList\_index(list, index);

cout << "Список успешно обновлен. Нажмите любую клавишу" << endl;

\_getch();

submenu\_list(list, arr, arr\_size);

}

void deleteNth(DblLinkedList\* list, size\_t index) // удаление узла из списка

{

Node\* elm = NULL;

int tmp = NULL;

elm = getNth\_list\_index(list, index);

if (elm->prev)

{

elm->prev->next = elm->next;

}

if (elm->next)

{

elm->next->prev = elm->prev;

}

tmp = elm->value;

if (!elm->prev)

{

list->head = elm->next;

}

if (!elm->next)

{

list->tail = elm->prev;

}

free(elm);

list->size--;

}

void deleteNth\_list(DblLinkedList\* list, int\* arr, int& arr\_size, bool fl\_del\_list) // функция удаления узла из списка

{

system("cls"); // очищаем консоль

chrono::steady\_clock sc;

int value;

int index;

cout << "Исходный список: ";

printDblLinkedList(list);

if (fl\_del\_list) // TRUE - удаляем по значению. FALSE - удаляем по индексу

{

cout << "Введите значение, которое будет удалено (удаление первого вхождения): ";

value = checkdigit();

if (!check\_getNth\_list\_value(list, value))

{

cout << "Такого значения нет. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

submenu\_list(list, arr, arr\_size);

}

else

{

auto start = sc.now(); // устанавливаем начало отсчета времени события

index = getNth\_list\_value(list, value);

deleteNth(list, index);

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость удаления элемента списка равна: " << time\_span.count() << " секунд" << endl;

if (!list->size)

{

fl\_list\_create = FALSE;

cout << "Все элементы списка были удалены! Необходимо заново создать список." << endl;

}

else

{

cout << "Полученный список: ";

printDblLinkedList(list);

}

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_list(list, arr, arr\_size); // возращаемся в подменю

}

}

else

{

cout << "Введите позицию, значение которой необходимо получить от 0 до " << list->size - 1 << ": ";

index = checkdigit();

if (!check\_index(index, list->size - 1))

{

cout << "Значение выходит за рамки диапазона. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

deleteNth\_list(list, arr, arr\_size, fl\_del\_list);

}

auto start = sc.now(); // устанавливаем начало отсчета времени события

deleteNth(list, index);

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость удаления элемента списка равна: " << time\_span.count() << " секунд" << endl;

if (!list->size)

{

fl\_list\_create = FALSE;

cout << "Все элементы списка были удалены! Необходимо заново создать список." << endl;

}

else

{

cout << "Полученный список: ";

printDblLinkedList(list);

}

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_list(list, arr, arr\_size); // возращаемся в подменю

}

}

void printDblLinkedList(DblLinkedList\* list) // вывод списка

{

Node\* tmp = list->head;

while (tmp)

{

cout << tmp->value << " ";

tmp = tmp->next;

}

cout << endl;

}

void trans(DblLinkedList\* list, int first, int second) // обмен элементами (не использовался, но придуман)

{

Node\* elm1 = NULL;

Node\* elm2 = NULL;

elm1 = getNth\_list\_index(list, first);

int first\_data = elm1->value;

elm2 = getNth\_list\_index(list, second);

int second\_data = elm2->value;

if (first == 0)

pushFront(list, second\_data);

else

insert(list, first - 1, second\_data);

deleteNth(list, first + 1);

if (second == 0)

pushFront(list, first\_data);

else

insert(list, second - 1, first\_data);

deleteNth(list, second + 1);

}

///////////////////////////// DBLINKEDLIST///////////////

//////////////////////////// ARRAY /////////////////////

void printArray(const int\* arr, const int size) // вывод массива

{

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

{

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

}

cout << endl;

}

void printArray\_index(const int\* arr, const int size, const int index) // вывод массива с выделенным значением

{

bool fl\_stop = TRUE;

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

{

if (i == index && fl\_stop)

{

SetColor(2, 0);

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

SetColor(7, 0);

fl\_stop = FALSE;

}

else

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

}

cout << endl;

}

void getNth\_array(DblLinkedList\* list, int\* arr, int& arr\_size, bool fl\_array) // функция получения индекса или значения массива

{

bool fl\_stop = TRUE;

system("cls"); // очищаем консоль

chrono::steady\_clock sc;

int value = 0;

int index = 0;

if (fl\_array) // TRUE - ищем индекс. FALSE - ищем значение

{

cout << "Введите значение, позицию которого необходимо получить: ";

value = checkdigit();

auto start = sc.now(); // устанавливаем начало отсчета времени события

while (fl\_stop && index <= arr\_size - 1)

{

if (value == arr[index])

fl\_stop = FALSE;

else

index++;

}

if (index > arr\_size - 1)

{

cout << "Такого значения нет. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

submenu\_array(list, arr, arr\_size);

}

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость поиска элемента массива равна: " << time\_span.count() << " секунд" << endl;

cout << "Позиция значения " << value << ": " << index << endl;

cout << "Полученный массив: ";

printArray\_index(arr, arr\_size, index);

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_array(list, arr, arr\_size);

}

else

{

cout << "Введите позицию, значение которой необходимо получить (от 0 до " << arr\_size -1 << "): ";

index = checkdigit();

if (!check\_index(index, arr\_size - 1))

{

cout << "Значение выходит за рамки диапазона. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

getNth\_array(list, arr, arr\_size, fl\_array);

}

auto start = sc.now(); // устанавливаем начало отсчета времени события

value = arr[index];

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость поиска элемента массива равна: " << time\_span.count() << " секунд" << endl;

cout << "Значение позиции " << index << ": " << value << endl;

cout << "Полученный массив: ";

printArray\_index(arr, arr\_size, index);

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_array(list, arr, arr\_size);

}

}

int\* addtoArray(int \*arr, int temp, int &size) // добавление элемента в конец массива

{

int\* newArray = new int [size + 1];

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

{

newArray[i] = arr[i];

}

newArray[size] = temp;

arr = newArray;

return arr;

}

int\* insert\_value\_ar(int\* arr, int temp, int& size, int index) // вставка элемента в массив

{

int\* newArray = new int[size + 1];

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

{

newArray[i] = arr[i];

}

newArray[index] = temp;

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

{

newArray[i] = arr[i - 1];

}

arr = newArray;

size++;

return arr;

}

void insert\_array(DblLinkedList\* list, int\* arr, int& arr\_size) // функция вставки элемента в массив

{

system("cls"); // очищаем консоль

chrono::steady\_clock sc;

int index;

int value;

printArray(arr, arr\_size);

cout << endl <<"Введите позицию нового значения от 0 до " << arr\_size - 1 << ": ";

index = checkdigit();

if (!check\_index(index, arr\_size - 1))

{

cout << "Значение выходит за рамки диапазона. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

insert\_array(list, arr, arr\_size);

}

cout << "Введите новое значение: ";

value = checkdigit();

auto start = sc.now(); // устанавливаем начало отсчета времени события

if (index == arr\_size - 1)

index++;

arr = insert\_value\_ar(arr, value, arr\_size, index);

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость вставки элемента массива равна: " << time\_span.count() << " секунд" << endl;

cout << "Новый массив: ";

printArray\_index(arr, arr\_size, index);

cout << "Массив успешно обновлен. Нажмите любую клавишу" << endl;

\_getch();

submenu\_array(list, arr, arr\_size);

}

int\* delete\_value\_ar(int\* arr, int& size, int index) // удаление элемента в массиве

{

int\* newArray = new int[size - 1];

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

{

newArray[i] = arr[i];

}

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

{

newArray[i] = arr[i + 1];

}

arr = newArray;

size--;

return arr;

}

void del\_array(DblLinkedList\* list, int\* arr, int& arr\_size, bool fl\_del\_array) // функция удаления элемента в массиве

{

bool fl\_stop = TRUE;

system("cls"); // очищаем консоль

chrono::steady\_clock sc;

int value = 0;

int index = 0;

cout << "Исходный массив: ";

printArray(arr, arr\_size);

if (fl\_del\_array) // TRUE - удаляем по значению. FALSE - удаляем по индексу

{

cout << "Введите значение, которое будет удалено (удаление первого вхождения): ";

value = checkdigit();

auto start = sc.now(); // устанавливаем начало отсчета времени события

while (fl\_stop && index <= arr\_size - 1)

{

if (value == arr[index])

fl\_stop = FALSE;

else

index++;

}

if (index > arr\_size - 1)

{

cout << "Такого значения нет. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

submenu\_array(list, arr, arr\_size);

}

arr = delete\_value\_ar(arr, arr\_size, index);

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость удаления элемента массива равна: " << time\_span.count() << " секунд" << endl;

if (!arr\_size)

{

fl\_array\_create = FALSE;

cout << "Все элементы массива были удалены! Необходимо заново создать массив." << endl;

}

else

{

cout << "Полученный массив: ";

printArray(arr, arr\_size);

}

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_array(list, arr, arr\_size);

}

else

{

cout << "Введите позицию, значение которой необходимо удалить (от 0 до " << arr\_size - 1 << "): ";

index = checkdigit();

if (!check\_index(index, arr\_size - 1))

{

cout << "Значение выходит за рамки диапазона. Нажмите любую клавишу, чтобы попробовать снова." << endl;

\_getch();

del\_array(list, arr, arr\_size, fl\_del\_array);

}

auto start = sc.now(); // устанавливаем начало отсчета времени события

arr = delete\_value\_ar(arr, arr\_size, index);

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость удаления элемента массива равна: " << time\_span.count() << " секунд" << endl;

if (!arr\_size)

{

fl\_array\_create = FALSE;

cout << "Все элементы массива были удалены! Необходимо заново создать массив." << endl;

}

else

{

cout << "Полученный массив: ";

printArray(arr, arr\_size);

}

cout << "Нажмите любую клавишу" << endl;

\_getch();

submenu\_array(list, arr, arr\_size);

}

}

//////////////////////////// ARRAY /////////////////////

//////////////////////////// FILL /////////////////////

void fill\_array\_random(DblLinkedList\* list, int\* arr, int& arr\_size) // автоматическое заполнения массива

{

system("cls"); // очищаем консоль

int index = 0;

chrono::steady\_clock sc;

cout << "Введите размерность массива: ";

arr\_size = checkdigit();

if (arr\_size <= 0)

{

cout << "Размерность не может быть отрицательная или нулевая. Нажмите любую клавишу, чтобы продолжить.";

\_getch();

submenu\_array(list, arr, arr\_size);

}

auto start = sc.now(); // устанавливаем начало отсчета времени события

while (index <= arr\_size)

{

int temp = rand() % 99;

arr = addtoArray(arr, temp, index);

index++;

}

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость создания массива равна: " << time\_span.count() << " секунд" << endl;

cout << "Созднанный массив: ";

printArray(arr, arr\_size);

cout << "Массив успешно создан. Нажмите любую клавишу" << endl;

\_getch();

fl\_array\_create = TRUE;

submenu\_array(list, arr, arr\_size);

}

void fill\_array\_self(DblLinkedList\* list, int\* arr, int& arr\_size) // ручное заполнение массива

{

system("cls"); // очищаем консоль

int temp;

int index = 0;

chrono::steady\_clock sc;

bool fl\_stop = TRUE;

cout << "Вводите числовые значения. Если хотите закончить, то введите любой нечисловой символ" << endl;

auto start = sc.now(); // устанавливаем начало отсчета времени события

while (fl\_stop)

{

cout << index << ": ";

cin >> temp; // число целое

if (cin.fail()) // ecли предыдущее извелечение оказлось неудачным,

{

cin.clear(); // то возвращаем cin в обычный режим работы

cin.ignore(32767, '\n'); // и удаляем из буфера значения предыдущего ввода

fl\_stop = FALSE;

}

else

{

cin.ignore(32767, '\n'); // удаляем из буфера значения предыдущего ввода

arr = addtoArray(arr, temp, arr\_size);

arr\_size++;

}

index++;

}

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость создания массива равна: " << time\_span.count() << " секунд" << endl;

cout << "Созднанный массив: ";

printArray(arr, arr\_size);

cout << "Массив успешно создан. Нажмите любую клавишу" << endl;

\_getch();

fl\_array\_create = TRUE;

submenu\_array(list, arr, arr\_size);

}

void fill\_array\_from\_file(DblLinkedList\* list, int\* arr, int& arr\_size) // заполнение массива через файл

{

system("cls"); // очищаем консоль

ifstream fin;

chrono::steady\_clock sc;

fin.open("file\_array.txt");

auto start = sc.now(); // устанавливаем начало отсчета времени события

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

{

int temp = 0;

while (!fin.eof()) // пока не дошли до конца файла

{

fin >> temp;

if (fin.fail()) // ecли предыдущее извелечение оказлось неудачным,

{

fin.clear(); // то возвращаем cin в обычный режим работы

fin.ignore(32767, '\n'); // и удаляем из буфера значения предыдущего ввода

cout << "Файл содержит некорректные числовые данные. Необходимо закрыть программу и ввести в текстовый файл данные корректно (между числовыми значениями пробел, посторонних знаков быть не должно)." << endl;

cout << "Пример: 50 -3 45 14 0 0 0 1" << endl;

cout << "Нажмите, чтобы закрыть программу" << endl;

\_getch();

exit(3);

}

else

{

arr = addtoArray(arr, temp, arr\_size);

arr\_size++;

}

}

}

else

{

cout << "Файл открыть не удалось! Закройте программу и создайте в корневой папке программы файл file\_array.txt для заполнения массива через файл." << endl;

cout << "Нажмите, чтобы закрыть программу" << endl;

\_getch();

exit(3);

}

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

fin.close();

cout << "Скорость создания массива равна: " << time\_span.count() << " секунд" << endl;

cout << "Созднанный массив: ";

printArray(arr, arr\_size);

cout << "Массив успешно создан. Нажмите любую клавишу" << endl;

\_getch();

fl\_array\_create = TRUE;

submenu\_array(list, arr, arr\_size);

}

void fill\_list\_random(DblLinkedList\* list, int\* arr, int& arr\_size) // автоматическое заполнение списка

{

system("cls"); // очищаем консоль

int size = 0;

int index = 0;

chrono::steady\_clock sc;

cout << "Введите размерность списка: ";

size = checkdigit();

if (size <= 0)

{

cout << "Размерность не может быть отрицательная или нулевая. Нажмите любую клавишу, чтобы продолжить.";

\_getch();

submenu\_list(list, arr, arr\_size);

}

auto start = sc.now(); // устанавливаем начало отсчета времени события

while (index <= size - 1)

{

int temp = rand() % 99;

pushBack(list, temp);

index++;

}

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость создания списка равна: " << time\_span.count() << " секунд" << endl;

cout << "Созднанный список: ";

printDblLinkedList(list);

cout << "Список успешно создан. Нажмите любую клавишу" << endl;

\_getch();

fl\_list\_create = TRUE;

submenu\_list(list, arr, arr\_size);

}

void fill\_list\_self(DblLinkedList\* list, int\* arr, int& arr\_size) // ручное заполнение списка

{

system("cls"); // очищаем консоль

int temp;

int index = 0;

chrono::steady\_clock sc;

bool fl\_stop = TRUE;

cout << "Вводите числовые значения. Если хотите закончить, то введите любой нечисловой символ" << endl;

auto start = sc.now(); // устанавливаем начало отсчета времени события

while (fl\_stop)

{

cout << index << ": ";

cin >> temp; // число целое

if (cin.fail()) // ecли предыдущее извелечение оказлось неудачным,

{

cin.clear(); // то возвращаем cin в обычный режим работы

cin.ignore(32767, '\n'); // и удаляем из буфера значения предыдущего ввода

fl\_stop = FALSE;

}

else

{

cin.ignore(32767, '\n'); // удаляем из буфера значения предыдущего ввода

pushBack(list, temp);

}

index++;

}

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

cout << "Скорость создания списка равна: " << time\_span.count() << " секунд" << endl;

cout << "Созднанный список: ";

printDblLinkedList(list);

cout << "Список успешно создан. Нажмите любую клавишу" << endl;

\_getch();

fl\_list\_create = TRUE;

submenu\_list(list, arr, arr\_size);

}

void fill\_list\_from\_file(DblLinkedList\* list, int\* arr, int& arr\_size) // заполнение массива через файл

{

system("cls"); // очищаем консоль

ifstream fin;

chrono::steady\_clock sc;

fin.open("file\_list.txt");

auto start = sc.now(); // устанавливаем начало отсчета времени события

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

{

int temp = 0;

while (!fin.eof()) // пока не дошли до конца файла

{

fin >> temp;

if (fin.fail()) // ecли предыдущее извелечение оказлось неудачным,

{

fin.clear(); // то возвращаем cin в обычный режим работы

fin.ignore(32767, '\n'); // и удаляем из буфера значения предыдущего ввода

cout << "Файл содержит некорректные числовые данные. Необходимо закрыть программу и ввести в текстовый файл данные корректно (между числовыми значениями пробел, посторонних знаков быть не должно)." << endl;

cout << "Пример: 50 -3 45 14 0 0 0 1" << endl;

cout << "Нажмите, чтобы закрыть программу" << endl;

\_getch();

exit(3);

}

else

pushBack(list, temp);

}

}

else

{

cout << "Файл открыть не удалось! Закройте программу и создайте в корневой папке программы файл file\_list.txt для заполнения списка через файл." << endl;

cout << "Нажмите, чтобы закрыть программу" << endl;

\_getch();

exit(3);

}

auto end = sc.now(); // устанавливаем конец отсчета времени события

auto time\_span = static\_cast<chrono::duration<double>>(end - start); // высчитываем время, затраченное на событие

fin.close();

cout << "Скорость создания списка равна: " << time\_span.count() << " секунд" << endl;

cout << "Созднанный список: ";

printDblLinkedList(list);

cout << "Список успешно создан. Нажмите любую клавишу" << endl;

\_getch();

fl\_list\_create = TRUE;

submenu\_list(list, arr, arr\_size);

}

//////////////////////////// FILL /////////////////////

///////////////////////// VARIAT /////////////////////

void variant\_work(DblLinkedList\* list, int\* arr, int& arr\_size) // индивидуальное задание

{

system("cls"); // очищаем консоль

int temp;

cout << "1. Необходимо увеличить каждое значение исходного двусвязного списка на случайное число (в диапазоне от 0 до 5):" << endl;

cout << "Текущий список: ";

printDblLinkedList(list);

Node\* tmp = list->head;

while (tmp)

{

temp = rand() % 6;

tmp->value = tmp->value + temp;

tmp = tmp->next;

}

free(tmp);

cout << "Обновленный список: ";

printDblLinkedList(list);

cout << endl << endl;

cout << "2. Необходимо добавить в динамический массив все положительные элементы динамического массива:" << endl;

cout << "Текущий массив: ";

printArray(arr, arr\_size);

int current\_arr\_size = arr\_size;

for (int index = 0; index <= current\_arr\_size - 1; index++)

{

if (arr[index] >= 0)

{

arr = addtoArray(arr, arr[index], arr\_size);

arr\_size++;

}

}

cout << "Обновленный список: ";

printArray(arr, arr\_size);

cout << "Задание выполнено! Нажмите любую клавишу" << endl;

\_getch();

menu(list, arr, arr\_size);

}

///////////////////////// VARIAT /////////////////////

///////////////////////// MENU ///////////////////

void subconf\_array\_delete\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_a\_d\_count)

{

case 0:

del\_array(list, arr, arr\_size, TRUE);

break;

case 1:

del\_array(list, arr, arr\_size, FALSE);

break;

case 2:

submenu\_array(list, arr, arr\_size);

break;

default:

break;

}

}

void subconf\_list\_delete\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_l\_d\_count)

{

case 0:

deleteNth\_list(list, arr, arr\_size, TRUE);

break;

case 1:

deleteNth\_list(list, arr, arr\_size, FALSE);

break;

case 2:

submenu\_list(list, arr, arr\_size);

break;

default:

break;

}

}

void subconf\_array\_search\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_a\_s\_count)

{

case 0:

getNth\_array(list, arr, arr\_size, TRUE);

break;

case 1:

getNth\_array(list, arr, arr\_size, FALSE);

break;

case 2:

submenu\_array(list, arr, arr\_size);

break;

default:

break;

}

}

void subconf\_list\_search\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_l\_s\_count)

{

case 0:

getNth\_list(list, arr, arr\_size, TRUE);

break;

case 1:

getNth\_list(list, arr, arr\_size, FALSE);

break;

case 2:

submenu\_list(list, arr, arr\_size);

break;

default:

break;

}

}

void subconf\_array\_create\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_a\_c\_count)

{

case 0:

fill\_array\_random(list, arr, arr\_size);

break;

case 1:

fill\_array\_self(list, arr, arr\_size);

break;

case 2:

fill\_array\_from\_file(list, arr, arr\_size);

break;

case 3:

submenu\_array(list, arr, arr\_size);

default:

break;

}

}

void subconf\_list\_create\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_l\_c\_count)

{

case 0:

fill\_list\_random(list, arr, arr\_size);

break;

case 1:

fill\_list\_self(list, arr, arr\_size);

break;

case 2:

fill\_list\_from\_file(list, arr, arr\_size);

break;

case 3:

submenu\_list(list, arr, arr\_size);

default:

break;

}

}

void subconf\_list\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_l\_count)

{

case 0:

submenu\_list\_create(list, arr, arr\_size);

break;

case 1:

if (fl\_list\_create)

insert\_list(list, arr, arr\_size);

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать список. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_list\_create(list, arr, arr\_size);

}

break;

case 2:

if (fl\_list\_create)

submenu\_list\_delete(list, arr, arr\_size);

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать список. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_list\_create(list, arr, arr\_size);

}

break;

case 3:

if (fl\_list\_create)

submenu\_list\_search(list, arr, arr\_size);

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать список. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_list\_create(list, arr, arr\_size);

}

break;

case 4:

menu(list, arr, arr\_size);

default:

break;

}

}

void subconf\_array\_val(DblLinkedList\* list, int\* arr, int& arr\_size)

{

switch (subm\_a\_count)

{

case 0:

submenu\_array\_create(list, arr, arr\_size);

break;

case 1:

if (fl\_array\_create)

insert\_array(list, arr, arr\_size);

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать массив. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_array\_create(list, arr, arr\_size);

}

break;

case 2:

if (fl\_array\_create)

submenu\_array\_delete(list, arr, arr\_size);

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать массив. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_array\_create(list, arr, arr\_size);

}

break;

case 3:

if (fl\_array\_create)

submenu\_array\_search(list, arr, arr\_size);

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать массив. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_array\_create(list, arr, arr\_size);

}

break;

case 4:

menu(list, arr, arr\_size);

default:

break;

}

}

void conf\_val(DblLinkedList\* list, int\* arr, int& arr\_size) // выбор после подтверждения в зависимости от значения текущего

{

switch (m\_count)

{

case 0:

submenu\_list(list, arr, arr\_size);

break;

case 1:

submenu\_array(list, arr, arr\_size);

break;

case 2:

if (fl\_array\_create)

{

if (fl\_list\_create)

variant\_work(list, arr, arr\_size);

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать список. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_list\_create(list, arr, arr\_size);

}

}

else

{

SetColor(4, 0);

cout << endl << "Сперва необходимо создать массив. Нажмите любую клавишу";

SetColor(7, 0);

\_getch();

submenu\_array\_create(list, arr, arr\_size);

}

break;

case 3:

SetColor(0, 0);

exit(0);

break;

default:

break;

}

}

void submenu\_array\_delete\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_d\_count--;

if (subm\_a\_d\_count < 0) subm\_a\_d\_count = 0;

submenu\_array\_delete(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_d\_count++;

if (subm\_a\_d\_count > 2) subm\_a\_d\_count = 2;

submenu\_array\_delete(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_delete\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_delete\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_d\_count--;

if (subm\_a\_d\_count < 0) subm\_a\_d\_count = 0;

submenu\_array\_delete(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_d\_count++;

if (subm\_a\_d\_count > 2) subm\_a\_d\_count = 2;

submenu\_array\_delete(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_delete\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_delete\_choice(list, arr, arr\_size);

}

}

void submenu\_list\_delete\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_d\_count--;

if (subm\_l\_d\_count < 0) subm\_l\_d\_count = 0;

submenu\_list\_delete(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_d\_count++;

if (subm\_l\_d\_count > 2) subm\_l\_d\_count = 2;

submenu\_list\_delete(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_delete\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_delete\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_d\_count--;

if (subm\_l\_d\_count < 0) subm\_l\_d\_count = 0;

submenu\_list\_delete(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_d\_count++;

if (subm\_l\_d\_count > 2) subm\_l\_d\_count = 2;

submenu\_list\_delete(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_delete\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_delete\_choice(list, arr, arr\_size);

}

}

void submenu\_array\_search\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_s\_count--;

if (subm\_a\_s\_count < 0) subm\_a\_s\_count = 0;

submenu\_array\_search(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_s\_count++;

if (subm\_a\_s\_count > 2) subm\_a\_s\_count = 2;

submenu\_array\_search(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_search\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_search\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_s\_count--;

if (subm\_a\_s\_count < 0) subm\_a\_s\_count = 0;

submenu\_array\_search(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_s\_count++;

if (subm\_a\_s\_count > 2) subm\_a\_s\_count = 2;

submenu\_array\_search(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_search\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_search\_choice(list, arr, arr\_size);

}

}

void submenu\_list\_search\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_s\_count--;

if (subm\_l\_s\_count < 0) subm\_l\_s\_count = 0;

submenu\_list\_search(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_s\_count++;

if (subm\_l\_s\_count > 2) subm\_l\_s\_count = 2;

submenu\_list\_search(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_search\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_search\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_s\_count--;

if (subm\_l\_s\_count < 0) subm\_l\_s\_count = 0;

submenu\_list\_search(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_s\_count++;

if (subm\_l\_s\_count > 2) subm\_l\_s\_count = 2;

submenu\_list\_search(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_search\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_search\_choice(list, arr, arr\_size);

}

}

void submenu\_array\_create\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_c\_count--;

if (subm\_a\_c\_count < 0) subm\_a\_c\_count = 0;

submenu\_array\_create(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_c\_count++;

if (subm\_a\_c\_count > 3) subm\_a\_c\_count = 3;

submenu\_array\_create(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_create\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_create\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_c\_count--;

if (subm\_a\_c\_count < 0) subm\_a\_c\_count = 0;

submenu\_array\_create(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_c\_count++;

if (subm\_a\_c\_count > 3) subm\_a\_c\_count = 3;

submenu\_array\_create(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_create\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_create\_choice(list, arr, arr\_size);

}

}

void submenu\_list\_create\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_c\_count--;

if (subm\_l\_c\_count < 0) subm\_l\_c\_count = 0;

submenu\_list\_create(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_c\_count++;

if (subm\_l\_c\_count > 3) subm\_l\_c\_count = 3;

submenu\_list\_create(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_create\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_create\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_c\_count--;

if (subm\_l\_c\_count < 0) subm\_l\_c\_count = 0;

submenu\_list\_create(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_c\_count++;

if (subm\_l\_c\_count > 3) subm\_l\_c\_count = 3;

submenu\_list\_create(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_create\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_create\_choice(list, arr, arr\_size);

}

}

void submenu\_list\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_count--;

if (subm\_l\_count < 0) subm\_l\_count = 0;

submenu\_list(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_count++;

if (subm\_l\_count > 4) subm\_l\_count = 4;

submenu\_list(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_l\_count--;

if (subm\_l\_count < 0) subm\_l\_count = 0;

submenu\_list(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_l\_count++;

if (subm\_l\_count > 4) subm\_l\_count = 4;

submenu\_list(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_list\_val(list, arr, arr\_size);

break;

default:

submenu\_list\_choice(list, arr, arr\_size);

}

}

void submenu\_array\_choice(DblLinkedList\* list, int\* arr, int& arr\_size)

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_count--;

if (subm\_a\_count < 0) subm\_a\_count = 0;

submenu\_array(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_count++;

if (subm\_a\_count > 4) subm\_a\_count = 4;

submenu\_array(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

subm\_a\_count--;

if (subm\_a\_count < 0) subm\_a\_count = 0;

submenu\_array(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

subm\_a\_count++;

if (subm\_a\_count > 4) subm\_a\_count = 4;

submenu\_array(list, arr, arr\_size);

break;

case 0xD: // подтвердить

subconf\_array\_val(list, arr, arr\_size);

break;

default:

submenu\_array\_choice(list, arr, arr\_size);

}

}

void menu\_choice(DblLinkedList\* list, int\* arr, int& arr\_size) // в зависимости от стрелок изменяем меню

{

int k1;

k1 = \_getch(); // получаем символ стрелки без вывода знака

if (k1 == 0xE0) // если стрелки

{

switch (k1)

{

case 0x48: // стрелка вверх

m\_count--;

if (m\_count < 0) m\_count = 0;

menu(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

m\_count++;

if (m\_count > 3) m\_count = 3;

menu(list, arr, arr\_size);

break;

case 0xD: // подтвердить

conf\_val(list, arr, arr\_size);

break;

default:

menu\_choice(list, arr, arr\_size);

}

}

switch (k1)

{

case 0x48: // стрелка вверх

m\_count--;

if (m\_count < 0) m\_count = 0;

menu(list, arr, arr\_size);

break;

case 0x50: // стрелка вниз

m\_count++;

if (m\_count > 3) m\_count = 3;

menu(list, arr, arr\_size);

break;

case 0xD: // подтвердить

conf\_val(list, arr, arr\_size);

break;

default:

menu\_choice(list, arr, arr\_size);

}

}

void submenu\_array\_delete(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_a\_d\_count == 0)

{

cout.width(15);

cout << right << "\_\_Удаление элемента в динамическом массиве\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом удаляем элемент в массиве?" << endl << endl;

SetColor(7, 0);

SetColor(5, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(7, 0);

cout << "По индексу" << endl;

cout << "Назад" << endl;

submenu\_array\_delete\_choice(list, arr, arr\_size);

}

if (subm\_a\_d\_count == 1)

{

cout.width(15);

cout << right << "\_\_Удаление элемента в динамическом массиве\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом удаляем элемент в массиве?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(5, 0);

cout << "По индексу" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_array\_delete\_choice(list, arr, arr\_size);

}

if (subm\_a\_d\_count == 2)

{

cout.width(15);

cout << right << "\_\_Удаление элемента в динамическом массиве\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом удаляем элемент в массиве?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

cout << "По индексу" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_array\_delete\_choice(list, arr, arr\_size);

}

}

void submenu\_list\_delete(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_l\_d\_count == 0)

{

cout.width(15);

cout << right << "\_\_Удаление элемента в двусвязном списке\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом удаляем элемент в списке?" << endl << endl;

SetColor(7, 0);

SetColor(5, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(7, 0);

cout << "По индексу" << endl;

cout << "Назад" << endl;

submenu\_list\_delete\_choice(list, arr, arr\_size);

}

if (subm\_l\_d\_count == 1)

{

cout.width(15);

cout << right << "\_\_Удаление элемента в двусвязном списке\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом удаляем элемент в списке?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(5, 0);

cout << "По индексу" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_list\_delete\_choice(list, arr, arr\_size);

}

if (subm\_l\_d\_count == 2)

{

cout.width(15);

cout << right << "\_\_Удаление элемента в двусвязном списке\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом удаляем элемент в списке?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

cout << "По индексу" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_list\_delete\_choice(list, arr, arr\_size);

}

}

void submenu\_array\_search(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_a\_s\_count == 0)

{

cout.width(15);

cout << right << "\_\_Поиск элемента в динамическом массиве\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом ищем элемент в массиве?" << endl << endl;

SetColor(7, 0);

SetColor(5, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(7, 0);

cout << "По индексу" << endl;

cout << "Назад" << endl;

submenu\_array\_search\_choice(list, arr, arr\_size);

}

if (subm\_a\_s\_count == 1)

{

cout.width(15);

cout << right << "\_\_Поиск элемента в динамическом массиве\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом ищем элемент в массиве?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(5, 0);

cout << "По индексу" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_array\_search\_choice(list, arr, arr\_size);

}

if (subm\_a\_s\_count == 2)

{

cout.width(15);

cout << right << "\_\_Поиск элемента в динамическом массиве\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом ищем элемент в массиве?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

cout << "По индексу" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_array\_search\_choice(list, arr, arr\_size);

}

}

void submenu\_list\_search(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_l\_s\_count == 0)

{

cout.width(15);

cout << right << "\_\_Поиск элемента в двусвязном списке\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом ищем элемент в списке?" << endl << endl;

SetColor(7, 0);

SetColor(5, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(7, 0);

cout << "По индексу" << endl;

cout << "Назад" << endl;

submenu\_list\_search\_choice(list, arr, arr\_size);

}

if (subm\_l\_s\_count == 1)

{

cout.width(15);

cout << right << "\_\_Поиск элемента в двусвязном списке\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом ищем элемент в списке?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

SetColor(5, 0);

cout << "По индексу" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_list\_search\_choice(list, arr, arr\_size);

}

if (subm\_l\_s\_count == 2)

{

cout.width(15);

cout << right << "\_\_Поиск элемента в двусвязном списке\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом ищем элемент в списке?" << endl << endl;

SetColor(7, 0);

cout << "По значению (первое вхождение)" << endl;

cout << "По индексу" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_list\_search\_choice(list, arr, arr\_size);

}

}

void submenu\_array\_create(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_a\_c\_count == 0)

{

cout.width(15);

cout << right << "\_\_Создание динамического массива\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем динамический массив?" << endl << endl;

SetColor(7, 0);

SetColor(5, 0);

cout << "Автоматически, с указаной размерностью" << endl;

SetColor(7, 0);

cout << "Вручную, заполняя значения" << endl;

cout << "Заполнение с помощью текстового файла" << endl;

cout << "Назад" << endl;

submenu\_array\_create\_choice(list, arr, arr\_size);

}

if (subm\_a\_c\_count == 1)

{

cout.width(15);

cout << right << "\_\_Создание динамического массива\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем динамический массив?" << endl << endl;

SetColor(7, 0);

cout << "Автоматически, с указаной размерностью" << endl;

SetColor(5, 0);

cout << "Вручную, заполняя значения" << endl;

SetColor(7, 0);

cout << "Заполнение с помощью текстового файла" << endl;

cout << "Назад" << endl;

submenu\_array\_create\_choice(list, arr, arr\_size);

}

if (subm\_a\_c\_count == 2)

{

cout.width(15);

cout << right << "\_\_Создание динамического массива\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем динамический массив?" << endl << endl;

SetColor(7, 0);

cout << "Автоматически, с указаной размерностью" << endl;

cout << "Вручную, заполняя значения" << endl;

SetColor(5, 0);

cout << "Заполнение с помощью текстового файла" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_array\_create\_choice(list, arr, arr\_size);

}

if (subm\_a\_c\_count == 3)

{

cout.width(15);

cout << right << "\_\_Создание динамического массива\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем динамический массив?" << endl << endl;

SetColor(7, 0);

cout << "Автоматически, с указаной размерностью" << endl;

cout << "Вручную, заполняя значения" << endl;

cout << "Заполнение с помощью текстового файла" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_array\_create\_choice(list, arr, arr\_size);

}

}

void submenu\_list\_create(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_l\_c\_count == 0)

{

cout.width(15);

cout << right << "\_\_Создание двусвязного списка\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем двусвязный список?" << endl << endl;

SetColor(7, 0);

SetColor(5, 0);

cout << "Автоматически, с указаной размерностью" << endl;

SetColor(7, 0);

cout << "Вручную, заполняя значения" << endl;

cout << "Заполнение с помощью текстового файла" << endl;

cout << "Назад" << endl;

submenu\_list\_create\_choice(list, arr, arr\_size);

}

if (subm\_l\_c\_count == 1)

{

cout.width(15);

cout << right << "\_\_Создание двусвязного списка\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем двусвязный список?" << endl << endl;

SetColor(7, 0);

cout << "Автоматически, с указаной размерностью" << endl;

SetColor(5, 0);

cout << "Вручную, заполняя значения" << endl;

SetColor(7, 0);

cout << "Заполнение с помощью текстового файла" << endl;

cout << "Назад" << endl;

submenu\_list\_create\_choice(list, arr, arr\_size);

}

if (subm\_l\_c\_count == 2)

{

cout.width(15);

cout << right << "\_\_Создание двусвязного списка\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем двусвязный список?" << endl << endl;

SetColor(7, 0);

cout << "Автоматически, с указаной размерностью" << endl;

cout << "Вручную, заполняя значения" << endl;

SetColor(5, 0);

cout << "Заполнение с помощью текстового файла" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_list\_create\_choice(list, arr, arr\_size);

}

if (subm\_l\_c\_count == 3)

{

cout.width(15);

cout << right << "\_\_Создание двусвязного списка\_\_" << endl << endl;

SetColor(3, 0);

cout << "Каким образом создаем двусвязный список?" << endl << endl;

SetColor(7, 0);

cout << "Автоматически, с указаной размерностью" << endl;

cout << "Вручную, заполняя значения" << endl;

cout << "Заполнение с помощью текстового файла" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_list\_create\_choice(list, arr, arr\_size);

}

}

void submenu\_list(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_l\_count == 0)

{

cout.width(15);

cout << right << "\_\_Двусвязный список\_\_" << endl;

SetColor(5, 0);

cout << "Создать двусвязный список" << endl;

SetColor(7, 0);

cout << "Вставить элемент" << endl;

cout << "Удалить элемент" << endl;

cout << "Получить элемент" << endl;

cout << "Назад" << endl;

submenu\_list\_choice(list, arr, arr\_size);

}

if (subm\_l\_count == 1)

{

cout.width(15);

cout << right << "\_\_Двусвязный список\_\_" << endl;

cout << "Создать двусвязный список" << endl;

SetColor(5, 0);

cout << "Вставить элемент" << endl;

SetColor(7, 0);

cout << "Удалить элемент" << endl;

cout << "Получить элемент" << endl;

cout << "Назад" << endl;

submenu\_list\_choice(list, arr, arr\_size);

}

if (subm\_l\_count == 2)

{

cout.width(15);

cout << right << "\_\_Двусвязный список\_\_" << endl;

cout << "Создать двусвязный список" << endl;

cout << "Вставить элемент" << endl;

SetColor(5, 0);

cout << "Удалить элемент" << endl;

SetColor(7, 0);

cout << "Получить элемент" << endl;

cout << "Назад" << endl;

submenu\_list\_choice(list, arr, arr\_size);

}

if (subm\_l\_count == 3)

{

cout.width(15);

cout << right << "\_\_Двусвязный список\_\_" << endl;

cout << "Создать двусвязный список" << endl;

cout << "Вставить элемент" << endl;

cout << "Удалить элемент" << endl;

SetColor(5, 0);

cout << "Получить элемент" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_list\_choice(list, arr, arr\_size);

}

if (subm\_l\_count == 4)

{

cout.width(15);

cout << right << "\_\_Двусвязный список\_\_" << endl;

cout << "Создать двусвязный список" << endl;

cout << "Вставить элемент" << endl;

cout << "Удалить элемент" << endl;

cout << "Получить элемент" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_list\_choice(list, arr, arr\_size);

}

}

void submenu\_array(DblLinkedList\* list, int\* arr, int& arr\_size)

{

system("cls"); // очищаем консоль

if (subm\_a\_count == 0)

{

cout.width(15);

cout << right << "\_\_Динамический массив\_\_" << endl;

SetColor(5, 0);

cout << "Создать динамический массив" << endl;

SetColor(7, 0);

cout << "Вставить элемент" << endl;

cout << "Удалить элемент" << endl;

cout << "Получить элемент" << endl;

cout << "Назад" << endl;

submenu\_array\_choice(list, arr, arr\_size);

}

if (subm\_a\_count == 1)

{

cout.width(15);

cout << right << "\_\_Динамический массив\_\_" << endl;

cout << "Создать динамический массив" << endl;

SetColor(5, 0);

cout << "Вставить элемент" << endl;

SetColor(7, 0);

cout << "Удалить элемент" << endl;

cout << "Получить элемент" << endl;

cout << "Назад" << endl;

submenu\_array\_choice(list, arr, arr\_size);

}

if (subm\_a\_count == 2)

{

cout.width(15);

cout << right << "\_\_Динамический массив\_\_" << endl;

cout << "Создать динамический массив" << endl;

cout << "Вставить элемент" << endl;

SetColor(5, 0);

cout << "Удалить элемент" << endl;

SetColor(7, 0);

cout << "Получить элемент" << endl;

cout << "Назад" << endl;

submenu\_array\_choice(list, arr, arr\_size);

}

if (subm\_a\_count == 3)

{

cout.width(15);

cout << right << "\_\_Динамический массив\_\_" << endl;

cout << "Создать динамический массив" << endl;

cout << "Вставить элемент" << endl;

cout << "Удалить элемент" << endl;

SetColor(5, 0);

cout << "Получить элемент" << endl;

SetColor(7, 0);

cout << "Назад" << endl;

submenu\_array\_choice(list, arr, arr\_size);

}

if (subm\_a\_count == 4)

{

cout.width(15);

cout << right << "\_\_Динамический массив\_\_" << endl;

cout << "Создать динамический массив" << endl;

cout << "Вставить элемент" << endl;

cout << "Удалить элемент" << endl;

cout << "Получить элемент" << endl;

SetColor(5, 0);

cout << "Назад" << endl;

SetColor(7, 0);

submenu\_array\_choice(list, arr, arr\_size);

}

}

void menu(DblLinkedList\* list, int\* arr, int &arr\_size) // меню

{

system("cls"); // очищаем консоль

if (m\_count == 0)

{

cout.width(15);

cout << right << "\_\_МЕНЮ\_\_" << endl;

SetColor(5, 0);

cout << "Двусвязный список" << endl;

SetColor(7, 0);

cout << "Динамический массив" << endl;

cout << "Задание по варианту" << endl;

cout << "Выход" << endl;

menu\_choice(list, arr, arr\_size);

}

if (m\_count == 1)

{

cout.width(15);

cout << right << "\_\_МЕНЮ\_\_" << endl;

cout << "Двусвязный список" << endl;

SetColor(5, 0);

cout << "Динамический массив" << endl;

SetColor(7, 0);

cout << "Задание по варианту" << endl;

cout << "Выход" << endl;

menu\_choice(list, arr, arr\_size);

}

if (m\_count == 2)

{

cout.width(15);

cout << right << "\_\_МЕНЮ\_\_" << endl;

cout << "Двусвязный список" << endl;

cout << "Динамический массив" << endl;

SetColor(5, 0);

cout << "Задание по варианту" << endl;

SetColor(7, 0);

cout << "Выход" << endl;

menu\_choice(list, arr, arr\_size);

}

if (m\_count == 3)

{

cout.width(15);

cout << right << "\_\_МЕНЮ\_\_" << endl;

cout << "Двусвязный список" << endl;

cout << "Динамический массив" << endl;

cout << "Задание по варианту" << endl;

SetColor(5, 0);

cout << "Выход" << endl;

SetColor(7, 0);

menu\_choice(list, arr, arr\_size);

}

}

///////////////////////// MENU ///////////////////

int main()

{

setlocale(0, ""); // локализация

DblLinkedList\* list = createDblLinkedList(); // создаем пустой список

int size = 0;

int\* arr = (int\*)malloc(size); // создаем пустой массив

arr[0] = 0;

srand(time(NULL));

menu(list, arr, size);

return 0;

}