

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Московский государственный технологический университет «СТАНКИН»**

**(ФГБОУ ВО «МГТУ «СТАНКИН»)**

|  |  |
| --- | --- |
| **Институт**  **информационных**  **технологий** | **Кафедра**  **информационных технологий и вычислительных систем** |

**КУРСОВАЯ РАБОТА**  
ПО ДИСЦИПЛИНЕ

«СТРУКТУРЫ И АЛГОРИТМЫ ОБРАБОТКИ ДАННЫХ»

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| СТУДЕНТА | *2* | КУРСА | | *бакалавриата* | ГРУППЫ | *ИДБ-20-03* |
|  | | | *(уровень профессионального образования)* | |  | |

|  |
| --- |
| **ГОРДЕЙЧИКА ДАНИИЛА АЛЕКСЕЕВИЧА** |
| *(ФИО)* |

ТЕМА РАБОТЫ

|  |
| --- |
| Множество последовательностей типа float |

|  |  |
| --- | --- |
| Направление: | 09.03.01 Информатика и вычислительная техника |
| Профиль подготовки: | Программное обеспечение средств вычислительной техники и автоматизированных систем |

|  |  |  |  |
| --- | --- | --- | --- |
| Отчет сдан «\_\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г. | | | |
|  |  |  |  |
| Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ | | | |
|  | | | |
| Преподаватель | Лакунина О.Н., ст. преподаватель |  |  |
|  | *(Ф.И.О., должность, степень, звание.)* |  | *(подпись)* |

МОСКВА 2021

Оглавление

[Задание на курсовую работу 3](#_Toc59125181)

[Описание логических и физических структур данных 4](#_Toc59125182)

[Конечная схема реализуемой структуры](#_Toc59125183) 7

[Описание структур на языке С++](#_Toc59125185) 8

[Схема вызова функций](#_Toc59125186) 9

[Список функций и их назначение](#_Toc59125187) 10

[Исходный код программы с комментариями](#_Toc59125188) 12

# Задание на курсовую работу

Написать программу, реализующую логическую структуру данных – множество последовательностей типа float.

Программа должна работать в диалоговом режиме.

Каждая операция должна быть реализована в виде отдельной функции.

Множество и последовательность должны быть реализованы на базе структуры хранения – односвязный список.

Написать отчет по курсовой работе.

**Описание логических и физических структур данных**

**Множество** – логическая структура данных, представляющая из себя неупорядоченный набор уникальных элементов одного типа.

Элемент множества

Множество

Схема множества (на базе односвязного списка)

Н

null

С

Д

Д

С

Д

|  |  |
| --- | --- |
| Н | Указатель на начало односвязного списка |
| Д | Указатель на последовательность |
| С | Указатель на следующий элемент в односвязном списке |

**Список реализуемых функций**

1. Начать работу со множеством.
2. Закончить работу со множеством.
3. Сделать множество пустым.
4. Проверка: множество пусто / не пусто.
5. Удалить элемент из множества.
6. Взять элемент из множества
7. Добавить элемент во множество
8. Проверить принадлежит ли элемент множеству
9. Распечатать множество
10. Завершить работу программы

**Последовательность –** это линейно упорядоченный набор элементов, и элементы последовательности в каждый момент времени можно разделить на две части: прочитанную и непрочитанную. Для указания границы между прочитанной и непрочитанной частями служит указатель очередного элемента, который показывает на первый элемент из непрочитанной части.

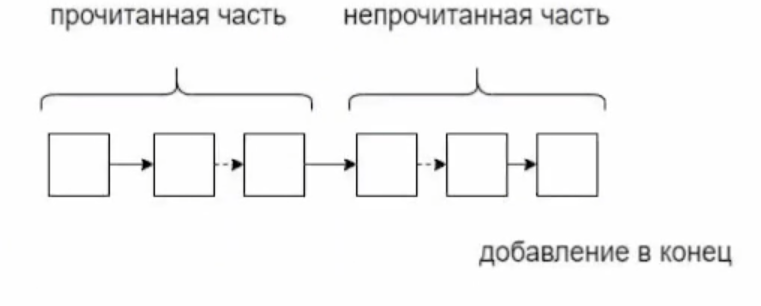
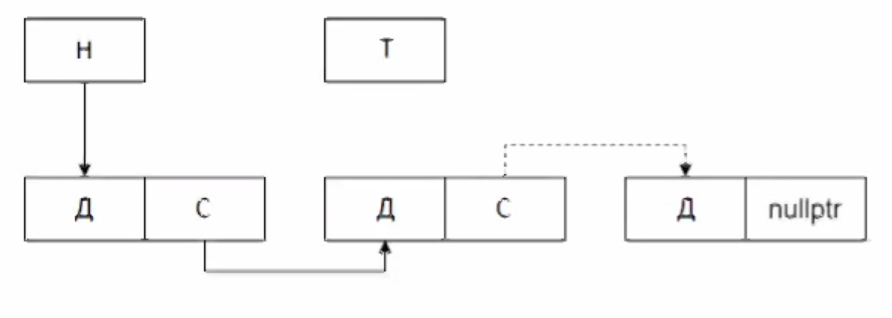


Схема последовательности (на базе односвязного списка)



|  |  |
| --- | --- |
| Н | Указатель на начало последовательности |
| Т | Указатель на текущий элемент |
| Д | Указатель на данные float |
| С | Указатель на следующий элемент |

**Список реализуемых функций:**

1. Начать работу с последовательностью.
2. Закончить работу с последовательностью.
3. Сделать последовательность пустой.
4. Проверка: последовательность пуста / не пуста.
5. Показать значение очередного элемента.
6. Пропустить очередной элемент
7. Прочесть очередной элемент
8. Изменить значение очередного элемента
9. Добавить элемент в конец
10. Установить указатель очередного элемента в начало
11. Проверка наличия непрочитанных элементов
12. Распечатать последовательность

**Конечная схема реализуемой структуры данных**

Последовательность (на базе односвязного списка)

Множество (на базе односвязного списка)

\*head

\*data

\*next

\*data

\*next

\*data

null

float data

\*next

float data

\*next

float data

null

\*head

\*pointer

**Описание структур на языке С++**

struct SetNode { //Узел множества

SequenceNode\* data; //Элемент данных множества

SetNode\* next;//Указатель на следующий элемент множества

};

struct SequenceNode { //Узел последовательности

float data; //Элемент данных последовательности

SequenceNode\* next;//Указатель на следующий элемент

//последовательности

};

**Схема вызова функций**

main()

SetMenu()

PrintSetMenu()

GetCommandofSet ()

CheckStart()

CheckCmdifSetisClear()

isSetClearandshowmsg()

GetStartedSet()

FinishWorkSet()

ClearSet()

isSetClear()

DeleteElement()

TakeElement()

AddtoSet()

isElementinSet()

PrintSet()

ClearSet()

SequenceMenu()

PrintSequence()

PrintSet()

PrintSequenceMenu()

GetCommandofSequence()

CheckcmdifitisClear()

isItClearandshowmsg()

GetStartedSequence()

FinishWorkSequence()

ClearMemory()

isItClear()

PrintElement()

SkipElement()

ReadElement()

ChangeElement()

AddElement()

SetPointerToHead()

isThereUnread()

InitializeSetElement()

isAllocated()

areSeqstheSame()

InitializeSequenceElement()

**Список функций и их назначение**

**Множество**

1. void SetMenu()

Функция-обработчик меню

1. void GetCommandofSet(SetNode\*& head, SetNode\*& object, bool& isStarted)

Обработка команды, которую ввел пользователь

1. void GetStartedSet(bool& isStarted)

Начать работу со множеством

1. void FinishWorkSet(SetNode\*& head, bool& isStarted)

Закончить работу со множеством

1. void ClearSet(SetNode\*& head)

Удалить множество

1. void isSetClear(SetNode\* head)

Проверка: множество пустое / не пустое

1. void DeleteElement(SetNode\*& head)

Удалить элемент множества

1. void TakeElement(SetNode\*& head, SetNode\*& object)

Взять элемент множества

1. void AddtoSet(SetNode\*& head)

Добавить элемент во множество

1. void isElementinSet(SetNode\* head)

Проверить наличие элемента во множестве

1. void PrintSet(SetNode\* head, int printMode)

Распечатать множество

1. bool CheckCommandifitisClear(int command)

Проверить команду на допустимость, если множество пустое

1. SequenceNode\* isItClearandShowMsg(SequenceNode\* head)

Проверить пусто ли множество и вывести сообщение

**Последовательность**

1. void SequenceMenu(SequenceNode\*& head)

Функция-обработчик меню

1. int GetCommandofSequence(SequenceNode\*& head, SequenceNode\*& pointer, SequenceNode\*& object, bool& isStarted)

Обработка команды, которую ввёл пользователь

1. void GetStartedSequence(bool& isStarted)

Начало работы с последовательностью

1. void FinishWorkSequence(SequenceNode\*& head)

Закончить работу с последовательностью

1. void ClearMemory(SequenceNode\*& head, SequenceNode\*& pointer)

Очистить последовательность

1. void isItClear(SequenceNode\* head, SequenceNode\* pointer)

Проверка: последовательность пуста / не пуста

1. void PrintElement(SequenceNode\* pointer)

Вывести на экран очередной элемент

1. void SkipElement(SequenceNode\*& pointer)

Пропустить очередной элемент

1. void ReadElement(SequenceNode\*& pointer, SequenceNode\*& object)

Прочитать очередной элемент

1. void ChangeElement(SequenceNode\*& pointer)

Изменить элемент

1. void AddElement(SequenceNode\*& head, SequenceNode\*& pointer)

Добавить элемент

1. void SetPointerToHead(SequenceNode\* head, SequenceNode\*& pointer)

Установить указатель в начало

1. void isThereUnread(SequenceNode\* pointer)

Проверить есть ли непрочитанные элементы

1. void PrintSequence(SequenceNode\* head, SequenceNode\* pointer)\

Распечатать последовательность

**Исходный код программы с комментариями**

**Файл main.cpp**

/\*Подключение необходимых библиотек\*/

#include <stdio.h>

#include <iostream>

#include <Windows.h>

#include <stdlib.h>

#include <iomanip>

#include "Sequence.h"

struct SetNode { //Узел множества

SequenceNode\* data; //Указатель на последовательность

SetNode\* next; //Указатель на следующий элемент

};

void ClearSet(SetNode\*& head) { //Удаление множества

while (head) {

SetNode\* nodetoDelete = head;

head = head->next;

//Удаление последовательности

ClearMemory(nodetoDelete->data);

//Очистка множества

free(nodetoDelete);

}

head = NULL;

}

void GetStartedSet(bool& isStarted) {//Начать работу со множеством

if (isStarted) { //Если работа уже начата

system("cls");

printf("Вы уже работаете с множеством!\nЧтобы начать работу с новым множеством, закончите с текущим [2]!\n");

system("pause");

}

else

isStarted = true;

}

void FinishWorkSet(SetNode\*& head, bool& isStarted) {//Закончить работу со множеством

isStarted = false;

system("cls");

if (head) //Если имеются данные, очистить их

ClearSet(head);

printf("Вы закончили работу со структурой\n");

system("pause");

}

void isAllocated(SetNode\* node) {//Проверить выделена ли память под новый элемент

if (!node) { //Если нет, то завершить работу программы

printf("Память не выделена, программа закроется!");

Sleep(1000);

exit(0);

}

}

SetNode\* InitializeSetElement() {//Инициализировать новый элемент

SetNode\* newNode = (SetNode\*)malloc(sizeof(SetNode));//Выделить память под новый элемент

isAllocated(newNode);//Проверить выделена ли память

newNode->data = NULL;

newNode->next = NULL;

SequenceMenu(newNode->data);//Вызвать меню последовательности для данных нового элемента множества

return newNode;

}

void AddtoSet(SetNode\*& head) {//Добавить элемент во множество

if (!head) {//Инициализировать корень, если он отсутствует

head = InitializeSetElement();

return;

}

SetNode\* newNode = InitializeSetElement(); //Инициализировать новый элемент

SetNode\* t = head;

SetNode\* lastSeqNode = t;

while (t) { //Получить последний элемент и узнать, есть ли уже такой

if (areSeqstheSame(newNode->data, t->data)) {//Проверить имеется ли такой же элемент во множестве

system("cls");

printf("Такой элемент уже есть в множестве!\n");

system("pause");

ClearSet(newNode);//Удалить выделенный элемент, если такой уже имеется

return;

}

if (!t->next) //Проверяет является ли элемент последним

lastSeqNode = t;

t = t->next;

}

lastSeqNode->next = newNode; //Присвоить новый элемент указателю на следующий элемент у последнего элемента

}

void PrintSet(SetNode\* head, int printMode) {//Распечатать множество

SetNode\* t = head;

int i = 1;

while (t) {

SetColor(2); //Установка необходимого цвета

printf("Элемент %d{", i);

SetColor(0);

SequenceNode\* seqt = t->data;

if (seqt) //Если последовательность не пустая, вывести её данные

while (seqt) {

std::cout << std::fixed << std::setprecision(3) << seqt->data;

seqt = seqt->next; //Перейти к следующему элементу в последовательности

if (seqt)

printf(" ");

}

else

printf("Пустая");

SetColor(2);

printf("} ");

if (printMode)

printf("\n");

SetColor(0);

t = t->next; //Перейти к следующему элементу

i++;

}

printf("\n\n");

}

void isSetClear(SetNode\* head) {//Проверить пусто ли множество

system("cls");

if (head)//Данные имеются, значит множество не пусто

printf("Структура не пустая!\n");

else

printf("Структура пустая!\n");

system("pause");

}

bool CheckCommandifSetisClear(int command) {//Проверить допустимость выполнения команды, если множество пустое

int allowedCommands[7] = { 1, 2, 3, 4, 7, 9, 10 };//Разрешенные команды

for (int i = 0; i < 7; i++)

{

if (allowedCommands[i] == command)//Выход, если команда разрешена

return false;

}

return true;

}

SetNode\* isSetClearandShowMsg(SetNode\* head) {//Вывести сообщение, если множество пустое

if (!head) {

system("cls");

printf("Структура пуста, вы не можете использовать эту операцию!\n");

system("pause");

}

return head;

}

void DeleteRoot(SetNode\*& head) {//Удалить корень множества

SetNode\* nodetoDelete = head;

head = head->next; //Получить новый корень

nodetoDelete->next = NULL;

ClearSet(nodetoDelete);//Удалить элемент множества

}

void DeleteElement(SetNode\*& head) {//Удалить элемент множества

SetNode\* newNode = InitializeSetElement();//Получить элемент, который хотим удалить во множестве

SequenceNode\* newSeqNode = newNode->data;

if (areSeqstheSame(head->data, newSeqNode)) { //Если введённые данные совпадают с корнем, то удалить корень

ClearSet(newNode); //Удалить введенный введенные даные

DeleteRoot(head); //Удалить корень

system("cls");

printf("Элемент удалён!\n");

system("pause");

return;

}

SetNode\* t = head;

while (t->next) { //удаление элемента отличного от корня

SequenceNode\* seqt = t->next->data;

if (areSeqstheSame(seqt, newSeqNode)) {//Поиск элемента, который нужно удалить

SetNode\* nodetoDelete = t->next;

t->next = nodetoDelete->next;

//Удаления множества

ClearSet(newNode);

nodetoDelete->next = NULL;

ClearSet(nodetoDelete);

system("cls");

printf("Элемент удалён!\n");

system("pause");

return;

}

t = t->next;

}

//Удалить элемент, который ввели, чтобы найти элемент для удаления

system("cls");

printf("Такой элемент отсутствует!\n");

system("pause");

ClearSet(newNode);

}

void TakeElement(SetNode\*& head, SetNode\*& object) {//Взять элемент

SetNode\* t = head;

system("cls");

printf("Взятый элемент: ");

if (!t->next) {//Если элемент является корнем

ClearSet(object);

object = head;

head = NULL;

if (object->data)

PrintSequence(object->data, NULL);

else

printf("пустая последовательность!\n");

system("pause");

return;

}

while (t->next) {

if (!t->next->next) {//Поиск последнего элемента

ClearSet(object); //Удалит то что лежало до этого

object = t->next; //Присвоить в объект взятый элемент

t->next = NULL;

if (object->data)

PrintSequence(object->data, NULL); //Вывести взятый объект

else

printf("пустая последовательность!\n");

system("pause");

return;

}

t = t->next;

}

}

void isElementinSet(SetNode\* head) {//Проверить принадлежит ли элемент множеству

SetNode\* newNode = InitializeSetElement();//Ввести элемент на проверку

system("cls");

SetNode\* t = head;

while (t) {//Проход по всем элементам множества

if (areSeqstheSame(t->data, newNode->data)) {//Если последовательности одинаковые

printf("Элемент принадлежит множеству!\n");

system("pause");

return;

}

t = t->next;

}

printf("Элемент не принадлежит множеству!\n");

system("pause");

}

void GetCommandofSet(SetNode\*& head, SetNode\*& object, bool& isStarted) {//Обработка команды, полученной от пользователя

printf("Команда ~ ");

int command = 0;

std::cin >> command;//Ввод команды пользователя

if (command != 1 && command != 10 && !CheckStart(isStarted)) //Проверить начата ли работа, и допустимые при этом команды

return;

if (CheckCommandifSetisClear(command) && !isSetClearandShowMsg(head)) //Проверить допустима ли команда, если множество пустое

return;

switch (command)//Вызов нужной для пользователя функции

{

case 1:

GetStartedSet(isStarted);//Начать работу

break;

case 2:

FinishWorkSet(head, isStarted);//Закончить работу

break;

case 3:

system("cls");

if (head) {

ClearSet(head);//Очистить структуру

printf("Вы сделали структуру пустой!\n");

}

else

printf("Структура уже пустая!\n");

system("pause");

break;

case 4:

isSetClear(head);//Проверить пуста ли структура

break;

case 5:

DeleteElement(head);//Удалить элемент

break;

case 6:

TakeElement(head, object);//Взять элемент

break;

case 7:

AddtoSet(head);//Добавить элемент

break;

case 8:

isElementinSet(head);//Проверить принадлежит ли элемент множеству

break;

case 9:

system("cls");

if (head)

PrintSet(head, 1);//Распечатать множество

else

printf("Структура пустая!\n\n");

system("pause");

break;

case 10:

if (head)

ClearSet(head);//Очистить множество

if (object)

ClearSet(object);//Очистить объект

exit(0);

break;

default:

system("cls");

printf("Вы ввели неизвестную команду!\n");

system("pause");

break;

}

}

void PrintSetMenu() {//Печатает меню

printf("[1] - Начать работу с множеством\n");

printf("[2] - Закончить работу с множеством\n");

printf("[3] - Сделать пустой\n");

printf("[4] - Структура пустая?\n");

printf("[5] - Удалить элемент из множества\n");

printf("[6] - Взять элемент из множества\n");

printf("[7] - Добавить элемент в множество\n");

printf("[8] - Проверить принадлежит ли элемент множеству\n");

printf("[9] - Распечатать структуру\n");

printf("[10] - Завершить работу программы\n");

printf("\n");

}

void SetMenu() {//Обработчик меню

SetNode\* head = NULL;//Корень списка

SetNode\* object = NULL;//Объект

bool isStarted = false;//Начата ли работа

while (true) {

if (head)

PrintSet(head, 0);//Распечатать множество

else

if (isStarted)

printf("Структура пустая\n\n");

PrintSetMenu();//Распечатать меню

GetCommandofSet(head, object, isStarted);//Обработать команду пользователя

system("cls");

}

}

int main() {

setlocale(LC\_ALL, "Russian");

SetMenu();//Вызвать меню множества

}

**Файл Sequence.h**

//Подключение необходимых библиотек

#include <stdio.h>

#include <iostream>

#include <Windows.h>

#include <stdlib.h>

#include <iomanip>

struct SequenceNode { //Узел последовательности

float data;

SequenceNode\* next;//Указатель на следующий элемент

};

void isItClear(SequenceNode\* head, SequenceNode\* pointer) {//проверить пустая ли последовательнсоть

system("cls");

if (head)//Данные есть => не пустая

printf("Структура НЕ пустая!\n");

else

printf("Структура пустая!\n");

system("pause");

}

bool CheckStart(bool isStarted) {//Проверить начата ли работа

if (!isStarted) {

system("cls");

printf("Выполнена недопустимая операция, сначала выполните [1]!\n");

system("pause");

}

return isStarted;

}

void GetStartedSequence(bool& isStarted) {//Начать работу

if (isStarted) {//Если уже начали работу, то вывести это

system("cls");

printf("Вы уже работаете с последовательностью!\nЧтобы начать работу с новой, закончите с текущей [2]!\n");

system("pause");

}

else//Иначе начать работу

isStarted = true;

}

void FinishWorkSequence(SequenceNode\*& head) {//Закончить работу с последовательностью

system("cls");

printf("Вы закончили работу с последовательностью\n");

system("pause");

}

void CheckisAllocated(SequenceNode\* node) {//Проверить выделилась ли память

if (!node) {

printf("Память не выделена, программа закроется!");

Sleep(1000);

exit(0);//Закрыть программу

}

}

SequenceNode\* InitializeSequenceElement() {//Инициализировать элемент последовательности

SequenceNode\* newNode = (SequenceNode\*)malloc(sizeof(SequenceNode));//Выделить память

CheckisAllocated(newNode);//Проверить выделилась ли память

printf("Введите число: ");

std::cin >> newNode->data;//Ввести данные

newNode->next = NULL;

return newNode;

}

void AddElement(SequenceNode\*& head, SequenceNode\*& pointer) {//Добавить элемент

system("cls");

if (!head) {//Если нет корня, то инициализировать корень

head = InitializeSequenceElement();

pointer = head;//Установить рабочий указатель на корень

return;

}

SequenceNode\* t = head;

while (t->next) { //Получить последний элемент

t = t->next;

}

t->next = InitializeSequenceElement();//Добавить новый элемент в конец

if (!pointer)//Если указатель уходил за пределы, присвоить ему новый элемент

pointer = t->next;

}

void SetColor(int colorMode) {//Установить цвет символов в консоли

switch (colorMode)//В зависимости от режима

{

case 0:

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), (WORD)((0 << 4) | 7)); //стандартный

break;

case 1:

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), FOREGROUND\_GREEN);

break;

case 2:

SetConsoleTextAttribute(GetStdHandle(STD\_OUTPUT\_HANDLE), (WORD)((0 << 4) | 6)); //желтый

break;

default:

break;

}

}

void PrintSequence(SequenceNode\* head, SequenceNode\* pointer) {//Распечатать последовательность

SequenceNode\* t = head;

int colorMode = 1;

while (t) {//пока существует t

if (t == pointer) {//Изменить цвет, если это рабочий указатель

SetColor(2);

std::cout << std::fixed << std::setprecision(3) << t->data << " "; //Вывод числа с 3 знаками после запятой

colorMode = 0;

}

else {

SetColor(colorMode);

std::cout << std::fixed << std::setprecision(3) << t->data << " ";

}

t = t->next;

}

SetColor(0);

printf("\n\n");

}

void ClearMemory(SequenceNode\*& head, SequenceNode\*& pointer) {//Очистить память от последовательности

SequenceNode\* t = head;

while (t) {//Пока существует t

SequenceNode\* nodetoDelete = t;

t = t->next;

free(nodetoDelete);//Очистка памяти

}

pointer = head = NULL;

}

void ClearMemory(SequenceNode\*& head) {//Очистка последовательности извне (например при очистке множества)

SequenceNode\* t = head;

while (t) {

SequenceNode\* nodetoDelete = t;

t = t->next;

free(nodetoDelete);

}

head = NULL;

}

void PrintElement(SequenceNode\* pointer) {//Распечатать элемент последовательности

system("cls");

if (pointer)//Если элемент существует распечатать его

std::cout << std::fixed << std::setprecision(3) << "Очередной элемент: " << pointer->data << std::endl;

else

printf("Вы не можете посмотреть значение очередного элемента, т.к. непрочитанные элементы отстутствуют!\n");

system("pause");

}

void SkipElement(SequenceNode\*& pointer) {//Пропустить очередной элемент

if (pointer)

pointer = pointer->next;//Сдвиг указателя

else {

system("cls");

printf("Вы не можете пропустить очередной элемент, т.к. непрочитанные элементы отсутствуют\n");

system("pause");

}

}

void ReadElement(SequenceNode\*& pointer, SequenceNode\*& object) {//Прочитать очередной элемент

system("cls");

if (pointer) {//Если элемент существует

object = pointer; //Присвоить объекту очередной элемент

std::cout << std::fixed << std::setprecision(3) << "Прочитанный элемент: " << object->data << std::endl;

pointer = pointer->next;//Сдвиг рабочего указателя

}

else

printf("Вы не можете прочитать очередной элемент, т.к. непрочитанные элементы отсутствуют\n");

system("pause");

}

void ChangeElement(SequenceNode\*& pointer) {//Изменить очередной элемент

system("cls");

if (pointer) {//Если очередной элемент существует

printf("Введите новое значение: ");

std::cin >> pointer->data;//Ввести новой значение

}

else {

printf("Вы не можете изменить очередной элемент, т.к. все элементы прочитаны!\n");

system("pause");

}

}

bool CheckCommandifitisClear(int command) {//Проверить доступна ли команда, если последовательность пустая

int allowedCommands[5] = { 1, 2, 3, 4, 9 };//Разрешенные команды

for (int i = 0; i < 5; i++)

{

if (allowedCommands[i] == command)//Выйти если команда разрешена

return false;

}

return true;

}

SequenceNode\* isItClearandShowMsg(SequenceNode\* head) {//Если последовательность пуста, вывести сообщение

if (!head) {

system("cls");

printf("Структура пуста, вы не можете использовать эту операцию!\n");

system("pause");

}

return head;

}

void SetPointerToHead(SequenceNode\* head, SequenceNode\*& pointer) {//Установить указатель на начало

if (pointer == head) {//Если уже установлен

system("cls");

printf("Указатель уже установлен в начале!\n");

system("pause");

}

else

pointer = head;

}

void isThereUnread(SequenceNode\* pointer) {//Есть ли непрочитанные элементы

system("cls");

if (pointer)//Если рабочий указатель не за пределами

printf("Непрочитанные элементы есть!\n");

else

printf("Непрочитанных элементов нет!\n");

system("pause");

}

int GetNumberofElements(SequenceNode\* seqNode) {//Получить количество элементов (используется для сравнения последовательностей)

int i = 0;

while (seqNode) {

i++;

seqNode = seqNode->next;//Перейти к следующему элементу

}

return i;

}

bool areSeqstheSame(SequenceNode\* firstSeq, SequenceNode\* secondSeq) {//Сравнение последовательностей

if (GetNumberofElements(firstSeq) != GetNumberofElements(secondSeq))//Если не равно количество элементов, то выйти

return false;

while (firstSeq && secondSeq) { //проверка по элементу

if (firstSeq->data != secondSeq->data)//Если не равны, то выйти

return false;

firstSeq = firstSeq->next;//Переход к следующему элементу

secondSeq = secondSeq->next;

}

return true;

}

int GetCommandofSequence(SequenceNode\*& head, SequenceNode\*& pointer, SequenceNode\*& object, bool& isStarted) {//Обработать команду

printf("Команда ~ ");

int command = 0;

std::cin >> command;//Получить команду

if (command != 1 && !CheckStart(isStarted))//Проверить команду на допустимость, если работа не начата

return 1;

if (CheckCommandifitisClear(command) && !isItClearandShowMsg(head))//Проверить команду на допустимость, если

return 1;

switch (command)

{

case 1:

GetStartedSequence(isStarted);//Начать работу с последовательностью

break;

case 2:

FinishWorkSequence(head);//Закончить работу с последовательностью

return 0;

case 3:

system("cls");

if (head) {

ClearMemory(head, pointer);//Очистить память от последовательности

printf("Вы сделали структуру пустой\n");

}

else

printf("Структура уже пустая!\n");

system("pause");

break;

case 4:

isItClear(head, pointer);//Проверить пустая ли последовательность

break;

case 5:

PrintElement(pointer);//Распечатать элемент

break;

case 6:

SkipElement(pointer);//Пропустить элемент

break;

case 7:

ReadElement(pointer, object);//Прочитать элемент

break;

case 8:

ChangeElement(pointer);//Изменить элемент

break;

case 9:

AddElement(head, pointer);//Добавить элемент

break;

case 10:

SetPointerToHead(head, pointer);//Установить указатель в начало

break;

case 11:

isThereUnread(pointer);//Проверить есть ли непрочитанные

break;

case 12:

system("cls");

PrintSequence(head, pointer);//Распечатать последовательность

system("pause");

break;

default:

system("cls");

printf("Вы ввели неизвестную команду!\n");

system("pause");

break;

}

return 1;

}

void PrintSequenceMenu() {//Распечатать последовательность

printf("[1] - Начать работу с последовательностью\n");

printf("[2] - Закончить работу с последовательностью (вернуться к множеству)\n");

printf("[3] - Сделать пустой\n");

printf("[4] - Структура пустая?\n");

printf("[5] - Показать значение очередного элемента\n");

printf("[6] - Пропустить очередной элемент\n");

printf("[7] - Прочесть очередной элемент\n");

printf("[8] - Изменить значение очередного элемента\n");

printf("[9] - Добавить элемент в конец\n");

printf("[10] - Установить указатель очередного элемента в начало\n");

printf("[11] - Есть ли непрочитанные элементы?\n");

printf("[12] - Распечатать последовательность\n");

printf("\n");

}

void SequenceMenu(SequenceNode\*& head) {//Меню обработчик

SequenceNode\* pointer = head;//Указатель на начало последовательности

SequenceNode\* object = NULL;//Указатель на объект

bool isStarted = false;

while (true) {

system("cls");

if (head)//Если корень существует, распечатать последовательность

PrintSequence(head, pointer);

else//Иначе структура пустая

if (isStarted)

printf("Структура пустая!\n\n");

PrintSequenceMenu();//Распечатать меню последовательности

if (!GetCommandofSequence(head, pointer, object, isStarted))//Обработать команду пользователя

return; //Выход из меню последовательности

}

}