МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ государственное БЮДЖЕТНОЕ образовательное учреждениевысшего образования

«НОВОСИБИРСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра защиты информации



**ОТЧЁТ по Лабораторной работе №1**

**по дисциплине «ПРОГРАММИРОВАНИЕ»**

**«Абстрактные структуры данных»**

|  |  |
| --- | --- |
| Факультет: АВТФ  Группа: АБс-222  Студент: Гатауллин Д.Р. | Преподаватель: Архипова А.Б. |

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

**Цели и задачи работы**: Изучение алгоритмов формирования и работы с абстрактными структурами данных.

**Задание к работе**:

1. Реализовать структуры данных с базовым набором операций:

a. Массив

b. Список (односвязный, двусвязный).

c. Очередь

d. Стек

e. Хеш таблицы

f. Деревья (2 Full Binary Tree)

2. Обосновать сложность выполнения каждой операции с позиции BigO нотации.

3. В отчете в контрольном примере представить визуальные формы,

подтверждающие соответствие результатам работы программы.

**Текст программы**

==========================================================

LB1.c

==========================================================

﻿#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <Windows.h>

#include <locale.h>

#include "table.h"

#include "stack.h"

#include "queue.h"

#include "massive.h"

#include "onelist.h"

#include "twolist.h"

#include "bin.h"

int main() {

setlocale(LC\_ALL, "Rus");

SetConsoleCP(1251); // устанавливаем кодировку Windows-1251 для ввода

SetConsoleOutputCP(1251);

char command[256];

char key[256];

char element[256];

HashTable\* ht = initHashTable();

Stack\* st = initStack();

Queue\* qu = initQueue();

StringArray\* ar = initStringArray();

NodeL1\* list1 = initList();

NodeL2\* list2 = initListT();

NodeTr\* tree = NULL;

printf("Список команд:\nhelp - Вывод Список Команд\n0 - Выход\n1 - Добавить элемент с ключом в Хеш-таблицу\n2 - Удалить элемент по ключу из Хеш-таблицы\n3 - Проверить наличия элемента по Хеш - Таблице\n4 - Вывести Хеш-таблицу\n5 - Очистить Хеш-таблицу\n6 - Добавить элемент в Стек\n7 - Удалить элемент из Стека\n8 - Вывести Стек\n9 - Очистить Стек\n10 - Добавить элемент в Очередь\n11 - Удалить элемент из Очереди\n12 - Вывести Очередь\n13 - Очистить Очередь\n14 - Добавление элемента в Массив\n15 - Удаление элемента из Массива\n16 - Просмотр элемента в Массиве\n17 - Сравнение элементов в Массиве\n18 - Вывод Массива\n19 - Очистка Массива\n20 - Добавить элемент в конец Односвязного списка\n21 - Удалить элемент из конца Односвязного списка\n22 - Проверка наличие элемента в Односвязном списке\n23 - Вывод Односвязного списка\n24 - Очистка Односвязного списка\n25 - Добавление в конец Двусвязного списка\n26 - Добавление в начало Двусвязного списка\n27 - Удаление в начале Двусвязного списка\n28 - Удаление в конце Двусвязного списка\n29 - Вывод Двусвязного списка\n30 - Очистка Двусвязного списка\n31 - Добавление элемента в Бинарное дерево по направлению\n32 - Удаление элемента из Бинарного дерева\n33 - Проверка наличия элемента в Бинарном дереве\n34 - Вывод Бинарного дерева\n35 - Очистка Бинарного дерева\n");

while (1) {

printf("Введите команду: ");

scanf("%255s", command);

if (strcmp(command, "0") == 0) {

break;

}

if (strcmp(command, "help") == 0) {

printf("Список команд:\nhelp - Вывод Список Команд\n0 - Выход\n1 - Добавить элемент с ключом в Хеш-таблицу\n2 - Удалить элемент по ключу из Хеш-таблицы\n3 - Проверить наличия элемента по Хеш - Таблице\n4 - Вывести Хеш-таблицу\n5 - Очистить Хеш-таблицу\n6 - Добавить элемент в Стек\n7 - Удалить элемент из Стека\n8 - Вывести Стек\n9 - Очистить Стек\n10 - Добавить элемент в Очередь\n11 - Удалить элемент из Очереди\n12 - Вывести Очередь\n13 - Очистить Очередь\n14 - Добавление элемента в Массив\n15 - Удаление элемента из Массива\n16 - Просмотр элемента в Массиве\n17 - Сравнение элементов в Массиве\n18 - Вывод Массива\n19 - Очистка Массива\n20 - Добавить элемент в конец Односвязного списка\n21 - Удалить элемент из конца Односвязного списка\n22 - Проверка наличие элемента в Односвязном списке\n23 - Вывод Односвязного списка\n24 - Очистка Односвязного списка\n25 - Добавление в конец Двусвязного списка\n26 - Добавление в начало Двусвязного списка\n27 - Удаление в начале Двусвязного списка\n28 - Удаление в конце Двусвязного списка\n29 - Вывод Двусвязного списка\n30 - Очистка Двусвязного списка\n31 - Добавление элемента в Бинарное дерево по направлению\n32 - Удаление элемента из Бинарного дерева\n33 - Проверка наличия элемента в Бинарном дереве\n34 - Вывод Бинарного дерева\n35 - Очистка Бинарного дерева\n");

}

if (strcmp(command, "1") == 0) {

printf("Введите ключ: ");

scanf("%255s", key);

printf("Введите элемент: ");

scanf("%255s", element);

HSET(ht, key, element);

printf("Вы добавили элемент %s с ключом %s\n", element, key);

}

if (strcmp(command, "2") == 0) {

printf("Введите ключ: ");

scanf("%255s", key);

HDEL(ht, key);

printf("Вы удалили элемент с ключом %s\n", key);

}

if (strcmp(command, "3") == 0) {

printf("Введите ключ: ");

scanf("%255s", key);

if (HGET(ht, key) != NULL) printf("True\n");

else printf("False\n");

}

if (strcmp(command, "4") == 0) {

printHashTable(ht);

}

if (strcmp(command, "5") == 0) {

freeHashTable(ht);

ht = initHashTable();

printf("Таблица очищена.\n");

}

if (strcmp(command, "6") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

SPUSH(st, element);

printf("Вы добавили элемент %s\n", element);

}

if (strcmp(command, "7") == 0) {

printf("Вы удалили элемент %s\n", SPOP(st));

}

if (strcmp(command, "8") == 0) {

SPrint(st);

}

if (strcmp(command, "9") == 0) {

SClear(st);

Stack\* st = initStack();

printf("Стек очищен.\n");

}

if (strcmp(command, "10") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

QPUSH(qu, element);

printf("Вы добавили элемент %s\n", element);

}

if (strcmp(command, "11") == 0) {

printf("Вы удалили элемент %s\n", QPOP(qu));

}

if (strcmp(command, "12") == 0) {

printQueue(qu);

}

if (strcmp(command, "13") == 0) {

clearQueue(qu);

Queue\* qu = initQueue();

printf("Очередь очищена.\n");

}

if (strcmp(command, "14") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

printf("Введите индекс: ");

scanf("%255s", key);

int num = atoi(key);

addString(ar, num, element);

}

if (strcmp(command, "15") == 0) {

printf("Введите индекс: ");

scanf("%255s", key);

int num = atoi(key);

removeString(ar, num);

}

if (strcmp(command, "16") == 0) {

printf("Введите индекс: ");

scanf("%255s", key);

int num = atoi(key);

printString(ar, num);

}

if (strcmp(command, "17") == 0) {

printf("Введите индекс 1: ");

scanf("%255s", key);

int num1 = atoi(key);

printf("Введите индекс 2: ");

scanf("%255s", key);

int num2 = atoi(key);

compareStrings(ar, num1, num2);

}

if (strcmp(command, "18") == 0) {

printStringArray(ar);

}

if (strcmp(command, "19") == 0) {

freeStringArray(ar);

StringArray\* ar = initStringArray();

printf("Массив очищен.\n");

}

if (strcmp(command, "20") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

append(&list1, element);

}

if (strcmp(command, "21") == 0) {

removeLast(&list1);

}

if (strcmp(command, "22") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

printf(contains(list1, element) ? "True\n" : "False\n");

}

if (strcmp(command, "23") == 0) {

printList(list1);

}

if (strcmp(command, "24") == 0) {

clearList(&list1);

NodeL1\* list1 = initList();

printf("Односвязный список очищен.\n");

}

if (strcmp(command, "25") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

appendT(&list2, element);

}

if (strcmp(command, "26") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

prependT(&list2, element);

}

if (strcmp(command, "27") == 0) {

removeFirstT(&list2);

}

if (strcmp(command, "28") == 0) {

removeLastT(&list2);

}

if (strcmp(command, "29") == 0) {

printListT(list2);

}

if (strcmp(command, "30") == 0) {

clearListT(&list2);

NodeL2\* list2 = initList();

printf("Двусвязный список очищен.\n");

}

if (strcmp(command, "31") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

printf("Введите положение (LCR): ");

scanf("%255s", key);

tree = insertNode(tree, element, key);

printf("Вы успешно добавили элемент.\n");

}

if (strcmp(command, "32") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

tree = deleteNode(tree, element);

printf("Вы успешно удалили под элементом его ветви.n");

}

if (strcmp(command, "33") == 0) {

printf("Введите элемент: ");

scanf("%255s", element);

if (findNode(tree, element) != NULL) printf("True\n");

else printf("False\n");

}

if (strcmp(command, "34") == 0) {

printTree(tree);

}

if (strcmp(command, "35") == 0) {

clearTree(&tree);

NodeTr\* tree = NULL;

printf("Бинарное дерево очищено.\n");

}

}

freeHashTable(ht);

SClear(st);

clearQueue(st);

freeStringArray(ar);

clearList(&list1);

clearListT(&list2);

clearTree(&tree);

return 0;

}

==========================================================

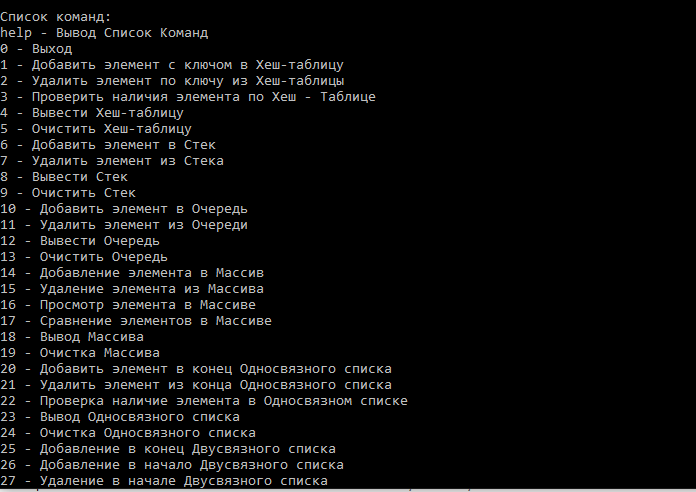
Остальной код в моём репозитории (PS для комментарий нужно скачать код, поскольку мои файлы сохранены не в формате UTF-8) https://github.com/SpongebobGatik/GataullinLB1/tree/main

==========================================================

1. Хеш-таблица: все операции выполняются за O(1), так как мы работаем фактически с индексом (хеширование)
2. Стек: все операции выполняются за O(1), так как мы просто добавляем элемент в следующий указатель, также и удаляем с возвратом предыдущего указателя
3. Очередь: то же самое, что и в стеке
4. Массив: добавление элемента по индексу выполняется за O(N), так как мы сдвигаем все элементы, также и с удалением, получение значение по индексу и сравнение двух значений выполняется за O(1), поскольку выполнятся эти функции за 1 операцию
5. Односвязный список: все операции в моём случае выполняются за O(N), так как я не вводил переменную, которая сохраняла указатель конца списка
6. Двусвязный список: добавление и удаление в начало выполняются за O(1), так как я просто записываю элемент в указатель перед указателем всего списка, а удаление и добавление в конец у меня происходит за O(N), так как, как и в случае односвязного списка, я просто не вводил переменную, отвечающая за указатель конца списка
7. Абсолютное бинарное дерево: все операции выполняются за O(N), так как во всех случаях мы рекурсивно обходим все узлы дерева

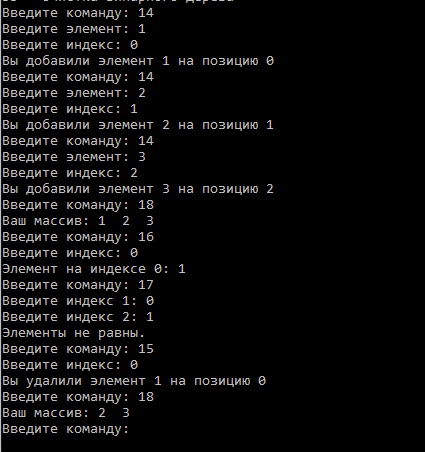
==========================================================

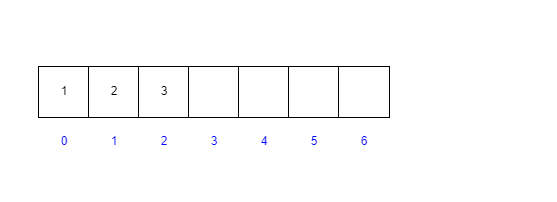
**Результат работы программы**



Изображение выглядит как текст, снимок экрана, Шрифт

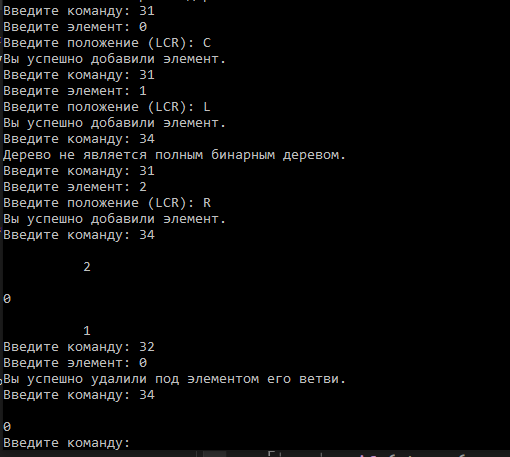
Автоматически созданное описание





Изображение выглядит как линия, диаграмма, Прямоугольник, График

Автоматически созданное описание



**Вывод**: Был создан интерфейс, где пользователь, по его желанию, вводить различные команды, связанные со структурами данных различного рода. Все временные сложности операций расписаны. В контрольном примере показаны работоспособность массива и Full Binary Tree, в случае массива предоставлена визуализация (они совпадают с моим выводом). Остальные операции также работают корректно (при желании можно их всех проверить).