Пермский Национальный Исследовательский  
Политехнический Университет

**Лабораторная работа №8**

Теория алгоритмов

за 2 семестр

Вариант № 1

Выполнил:

Студент группы РИС 20-1-бз

Курганов Н.В.

20-ЭТФ-631

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

2021

# Динамические структуры данных

1. Сформировать однонаправленный список, тип информационного поля указан в варианте.
2. Распечатать полученный список.
3. Выполнить обработку списка в соответствии с заданием.
4. Распечатать полученный список.
5. Удалить список из памяти.
6. Сформировать двунаправленный список, тип информационного поля указан в варианте.
7. Распечатать полученный список.
8. Выполнить обработку списка в соответствии с заданием.
9. Распечатать полученный список.
10. Удалить список из памяти.
11. Сформировать идеально сбалансированное бинарное дерево, тип информационного поля указан в варианте.
12. Распечатать полученное дерево.
13. Выполнить обработку дерева в соответствии с заданием, вывести полученный результат.
14. Преобразовать идеально сбалансированное дерево в дерево поиска.
15. Распечатать полученное дерево.

# 2.

|  |  |  |  |
| --- | --- | --- | --- |
| №  варианта | Однонаправленный | Двунаправленный | Бинарное дерево |
| 1 | Тип информационного поля int.  Удалить из списка все элементы с четными информационными полями. | Тип информационного поля char\*.  Добавить в список элемент с заданным номером. | Тип информационного поля char.  Найти количество элементов с заданным ключом. |

# 3.

#include <locale.h>

#include <stdio.h>

#include <iostream>

#include <locale>

#include <conio.h>

#include "point.h"

#include "doublePoints.h"

#include "BinaryTree.h"

using namespace std;

//меню первого уровня

void MenuFirst()

{

cout << ("1 - Работа с однонаправленным списоком\n");

cout << ("2 - Работа с двунаправленным списоком\n");

cout << ("3 - Работа с бинарным деревом\n");

cout << ("Esc выход\n\n");

}

//меню второго уровня для односторонних списков

void MenuSecond1()

{

cout << ("1 - Сформировать однонаправленный список\n");

cout << ("2 - Распечатать полученный список\n");

cout << ("3 - Удалить из списка все элементы с четными информационными полями\n");

cout << ("Esc выход\n\n");

}

//меню второго уровня для двусторонних списков

void MenuSecond2()

{

cout << ("1 - Сформировать двунаправленный список\n");

cout << ("2 - Распечатать список\n");

cout << ("3 - Добавить в список элемент с заданным номером\n");

cout << ("Esc выход\n\n");

}

//меню второго уровня для бинарного дерева

void MenuSecond3()

{

cout << ("1 - Сформировать идеально сбалансированное бинарное дерево\n");

cout << ("2 - Распечатать дерево\n");

cout << ("3 - Найти количество элементов с заданным ключом\n");

cout << ("4 - Преобразовать идеально сбалансированное дерево в дерево поиска\n");

cout << ("Esc выход\n\n");

}

int main() {

setlocale(LC\_ALL, "Rus");//меняем язык консоли на русский

point\* oneList = 0;//односторонний список

doublePoints\* doubleList = 0;//двусторнний список

binnaryTree\* tree = 0;//идеальное бинарное дерево

binnaryTree\* Btree = 0;//дерево поиска

while (1) {

MenuFirst();//выводим меню

char c = \_getch();//получаем нажатую кнопку

bool stop = false;

int s = 0;

char ch;

int num;

switch (c) {

case '1':

while (!stop)

{

MenuSecond1();//выводи меню

c = \_getch();//получаем конпку

switch (c)

{

case '1':

//удаляем все списки

oneList = 0;

doubleList = 0;

tree = 0;

Btree = 0;

//создаём однонаправленный список

cout << ("Введите длину списка: ");

cin >> s;

cout << ("\n");

oneList = makeOne\_list(s);

break;

case '2':

if (oneList)

{

//если список существует то печатаем его

print\_list(oneList);

}

else

{

cout << ("Список не создан\n");

}

break;

case '3':

if (oneList) {

//если список существует то удаляем все чётные элементы

oneList = del\_even\_data(oneList);

}

else {

cout << ("Список не создан\n");

}

break;

case 27:

//если нажали esc то выходим из цикла

stop = true;

break;

}

}

break;

case '2':

while (!stop) {

MenuSecond2();//выводим меню

c = \_getch();//получаем нажатую кнопку

switch (c) {

case '1':

//удаляем все списки

oneList = 0;

doubleList = 0;

tree = 0;

Btree = 0;

//создаём двунаправленный список

cout << ("Введите длину списка: ");

cin >> s;

cout << ("\n");

doubleList = makeTwo\_list(s);

break;

case '2':

if (doubleList) {

//если список существует, то печатаем его

print\_list(doubleList);

cout << ("\n");

}

else {

cout << ("Список не создан\n");

}

break;

case '3':

if (doubleList) {

//если список существует, то добавляем элемент после индекса s

cout << ("Введите индекс после которого вставить новый элемент: ");

cin >> s;

cout << ("\n");

doubleList = add\_ToList(doubleList, s);

}

else {

cout << ("Список не создан\n");

}

break;

case 27:

//если нажали esc то выходим из цикла

stop = true;

break;

}

}

break;

case '3':

while (!stop) {

MenuSecond3();

c = \_getch();

switch (c) {

case '1':

//удаляем все списки

oneList = 0;

doubleList = 0;

tree = 0;

Btree = 0;

//создаём идиальное бинарное дерево

tree = new binnaryTree;

cout << ("Введите длину дерева: ");

cin >> s;

cout << ("\n");

tree = Tree(s, tree);

break;

case '2':

if (tree) {

//если дерево создано, то печатаем идеальное бинарное дерево

print\_Tree(tree);

cout << ("\n");

}

else {

cout << ("Список не создан\n");

}

break;

case '3':

if (tree) {

//если дерево создано, то подсчитываем элементы

cout << ("Введите первый элемент: ");

cin >> ch;

cout << ("\n");

num = find\_in\_Tree(tree, ch);

cout << "Колличество символов " << ch << " в дереве = " << num << ("\n");

}

else {

cout << ("Список не создан\n");

}

break;

case '4':

if (tree) {

//если бинарное дерево создано, то создаем на его основе дерево поиска

Btree = new binnaryTree;

Btree = createBT(Btree, tree);

//выводим дерево поиска

print\_Tree(Btree);

cout << ("\n");

}

else {

cout << ("Список не создан\n");

}

break;

case 27:

//если нажали esc то выходим из цикла

stop = true;

break;

}

}

break;

case 27:

//если нажали esc то выходим из программы

return 0;

}

}

return 0;

}

# 4.

# 