# Лабораторная работа №8 "Динамические структуры данных"

**Вариант №15**

**Цель:** 1) Получить практические навыки работы с однонаправленными списками;

2) получить практические навыки работы с двунаправленными списками;

3) получить практические навыки работы с деревьями.

## **Задача:**

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.

|  |  |  |  |
| --- | --- | --- | --- |
| 15 | Тип информационного поля double.  Добавить в список после каждого элемента с отрицательным информационным полем элемент с информационным полем равным 0. | Тип информационного поля int.  Удалить из списка все элементы с четными номерами (2, 4, 6 и. т. д.). | Тип информационного поля char\*.  Найти количество элементов дерева, начинающихся с заданного символа |

**Решение:**

#include <iostream>

using namespace std;

// Заготовка однонаправленного списка

struct Data1

{

double a;

};

struct List

{

Data1 d;

List\* next;

};

// Заготовка двунаправленного списка

struct Data2

{

int a; // данные

};

struct List2

{

Data2 d;

List2\* prev; // указатель на предшествующий элемент

List2\* next; // указатель на последующий элемент

};

// Заготовка бинарного дерева

struct Data3

{

char a;

};

struct List3

{

Data3 d;

List3\* left;

List3\* right;

};

// Вывод однонаправленного списка на экран

void PrintSinglyLinkedList(List\* u)

{

setlocale(LC\_ALL, "Rus");

List\* p = u;

cout << "Наш однонаправленный список:\n\n";

while (p)

{

cout << p->d.a << " ";

p = p->next;

}

cout << "\n";

}

// Вывод двунаправленного списка на экран

void PrintDoublyLinkedList(List2\* Start)

{

setlocale(LC\_ALL, "Rus");

List2\* p = Start;

cout << "\nНаш двунаправленный список:\n\n";

while (p)

{

cout << p->d.a << " ";

p = p->next;

}

cout << "\n";

}

// Вывод бинарного дерева на экран

void PrintTreeList(List3\* StartTree)

{

setlocale(LC\_ALL, "Rus");

List3\* p = StartTree;

cout << "\nНаше бинарное дерево:\n\n";

while (p)

{

cout << p->d.a << " ";

p = p->left;

}

cout << "\n";

}

// Вставка в список одного элемента после отрицательного элемента

void Insert(List\* u)

{

List\* p = u;

List\* second = NULL;

while (p)

{

if (p->d.a < 0)

{

/\*p = u->next;

u->next = second;

second->d.a = number;

second->next = p;

return(second);\*/

}

p = p->next;

}

}

// Наполнение однонаправленного списка данными

void SinglyLinkedList()

{

List\* u = NULL;

u = new List; // Выделяем память под элемент списка

u->d.a = 3; // Заполняем поля с данными

u->next = NULL; // Указатель на следующий элемент пуст

List\* x; // Второй указатель

x = u; // Сейчас последний элемент списка совпадает с его началом

x->next = new List;

x = x->next;

x->d.a = -5.2;

x->next = NULL;

x->next = new List;

x = x->next;

x->d.a = 7;

x->next = NULL;

x->next = new List;

x = x->next;

x->d.a = 14.7;

x->next = NULL;

Insert(u);

PrintSinglyLinkedList(u);

}

// Наполнение двунаправленного списка данными

void DoublyLinkedList()

{

List2\* Start = NULL; // Начало списка

List2\* End = NULL; // Конец списка

// Создаём первый элемент

List2\* t = new List2;

t->d.a = 14;

t->prev = NULL;

t->next = NULL;

// Настроим на него оба указателя

Start = t;

End = t;

// Создаём второй элемент

t->next = new List2;

List2\* p = t;

t = t->next;

t->prev = p;

t->d.a = 7;

t->next = NULL;

End = t;

// Создаём третий элемент

t->next = new List2;

p = t;

t = t->next;

t->prev = p;

t->d.a = 2;

t->next = NULL;

End = t;

PrintDoublyLinkedList(Start);

}

// Наполнение бинарного дерева данными

void TreeList()

{

List3\* StartTree = new List3;

List3\* LeftTree = NULL;

List3\* RightTree = NULL;

// Создаем первый элемент

StartTree->d.a = '2';

StartTree->left = NULL;

StartTree->right = NULL;

LeftTree = StartTree;

RightTree = StartTree;

// Создаем элемент слева

StartTree->left = new List3;

List3\* p = LeftTree;

p->d.a = '1';

p->left = NULL;

p->right = NULL;

// Создаем элемент справа

StartTree->right = new List3;

List3\* q = RightTree;

q->d.a = '3';

q->left = NULL;

q->right = NULL;

PrintTreeList(StartTree);

}

int main()

{

SinglyLinkedList();

DoublyLinkedList();

TreeList();

return 0;

}

**Результат:**

Наш однонаправленный список:

3 -5.2 7 14.7

Наш двунаправленный список:

14 7 2

Наше бинарное дерево:

3