# Лабораторная работа № 3 Тема. Классы для работы с динамическими структурами данных

**Теоретическое введение**. Объекты классов могут храниться в виде массивов или динамических связных списков. Если класс содержит конструктор, массив может быть инициализирован, причем конструктор вызывается столько раз, сколько задается элементов массива: **samp ob[4]={1,2,3,4};**

Список инициализации – это сокращение общей конструкции:

**samp ob[4]={samp(1,2),samp(3,4),samp(5,6),samp(7,8)};**

При создании динамических объектов используется оператор ***new***, который вызывает конструктор и производит инициализацию. Для разрушения динамического объекта используется оператор ***delete***, который может помещаться в деструкторе.

Кроме указателей на классы используются ссылки. Ссылка является скрытым указателем и работает как другое имя переменной. При передаче объекта через ссылку в функцию передается адрес объекта и не делается его копия. Это уменьшает вероятность ошибок, связанных с выделением динамической памяти и вызовом деструктора. Так, при передаче в функцию параметра-объекта может возникнуть ошибка из-за разрушения деструктором на выходе копии объекта, которая должна быть исправлена созданием конструктора копирования. В такой ситуации лучше передать в функцию ссылку на объект и возвратить ссылку на объект.

**Пример.** Создается класс **Tree** (бинарное дерево). Информационная часть узла дерева содержит целое число. Тестирование класса выполняется с помощью меню, которое позволяет сформировать дерево, вывести содержимое его узлов в порядке возрастания, найти узел по ключу, вывести содержимое листьев дерева (вершин, не имеющих потомков).

#include"vip\menu.cpp" //реализация работы с меню

#include <conio.h>

#include <string.h>

#include <iostream.h>

char bufRus[256];

char\*Rus(const char\*text){

CharToOem(text,bufRus);

return bufRus;}

struct node

{

int n; //информационное поле узла дерева

int count;

node\*left,\*right;

};

class Tree

{

public:

node\*root;

Tree(){root=0;}

Tree(int t); // Формирование дерева из t случайных чисел

void CopyTree(node\*&rootnew,node\*rootold);

/\* Копирует дерево с корнем rootold в дерево с корнем rootnew. В результате деревья находятся в различных динамических участках памяти.\*/

Tree(const Tree&ob); //конструктор копирования

// Рекурсивная функция, используемая в деструкторе (освобождение памяти)

void DelTree(node \*wer);

~Tree(){DelTree(root);}

void Push(node\*&wer,int data);// Вставка элемента в дерево

void Look(node\*wer); //- Вывод дерева на экран

node\*Find(node\*wer,int key); // Поиск по ключу

void PrintLeaves(node \*wer); // Вывод листьев дерева на экран

};

//\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Tree::Tree(int t) \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Tree::Tree(int t)

{

root=0;

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

Push(root,random(10)-5);

}

void Tree::CopyTree(node\*&rootnew,node\*rootold)

{

if(rootold->left!=0)

{Push(rootnew,(rootold->left)->n);CopyTree(rootnew,rootold->left);}

if(rootold->right!=0)

{Push(rootnew,(rootold->right)->n);CopyTree(rootnew,rootold->right);}

}

Tree::Tree(const Tree&ob)

{

if(ob.root==0)root=0;

else {

root=new node;

root->n=ob.root->n;

root->count=1;

root->left=0;

root->right=0;

CopyTree(root,ob.root);

}

}

void Tree::DelTree(node \*wer)

{

if(wer->left!=0)DelTree(wer->left);

if(wer->right!=0)DelTree(wer->right);

delete wer;

}

void Tree::Push(node\*&wer,int data)

{

if(wer==0)

{

wer=new node;

wer->n=data;

wer->left=0;wer->right=0;

wer->count=1;

}

else if(data<wer->n)Push(wer->left,data);

else if(data>wer->n)Push(wer->right,data);

else wer->count++;

}

void Tree::Look(node\*wer)

{

if(wer!=0)

{

Look(wer->left);

cout<<Rus("Число: ")<<wer->n<<" - "<<wer->count;

cout<<Rus(" штук")<<endl;

Look(wer->right);

}

}

node\* Tree::Find(node\*wer,int key)

{

if(wer==0) return 0;

else if(key<wer->n) return Find(wer->left,key);

else if(key>wer->n) return Find(wer->right,key);

else return wer;

}

void Tree::PrintLeaves(node \*wer)

{

if(wer==0)return;

else if( (wer->left==0)&&(wer->right==0) ) {

cout<<Rus(“Число: “)<<wer->n<<”-“<<wer->count;

cout<<Rus(“штук”)<<endl;

}

else

{

PrintLeaves(wer->left);

PrintLeaves(wer->right);

}

}

//-------------------------------- MAIN ----------------------------------------

int main(int argc, char\* argv[])

{

Tree tr;

node \*u;

int k=0,max,kol;

char menu[][100]={ {" PushElement "}, {" ShowTree "}, {" FindElement "},

{" PrintLeaves "}, {" EXIT "}, };

kol=5;//КОЛИЧЕСТВО СТРОК МЕНЮ. Используется в выравнивании строк

// меню по центру.

//------------------ВЫРАВНИВАНИЕ СТРОК МЕНЮ ПО ЦЕНТРУ----------------

max=viravnivaniestrok(menu,kol);

//------------------------ МЕНЮ НА ЭКРАНЕ---------------------------------------

textmode(C80);

while(1){

switch(mmm(kol,menu,max,k))

{ case 0: {

int data;

cout<<Rus("Введите число:");

cin>>data;

tr.Push(tr.root,data);

k=0;break;

}

case 1: {

if(tr.root==0)cout<<Rus("Дерево пустое");

else

{

cout<<Rus("Наше дерево:")<<endl;

tr.Look(tr.root);

}

while(!kbhit());

k=1;break;

}

case 2: {

if(tr.root==0)cout<<Rus("Дерево пустое");

else

{

int key;

cout<<Rus("Введите искомое число:");

cin>>key;

if((u=tr.Find(tr.root,key))!=0){

cout<<Rus("Элементов: ");

cout<<key;

cout<<Rus(" найдено ");

cout<<u->count<<Rus(" штук");

}

else cout<<Rus("Таких элементов нет!");

}

while(!kbhit());

k=2;break;

}

case 3: {

if(tr.root==0)cout<<Rus("Дерево пустое");

else{

cout<<Rus("Листья:")<<endl;

tr.PrintLeaves(tr.root);

}

while(!kbhit());

k=3;break;

}

case 4:{

exit(0);

}

} }

return 0;

}

Задания для самостоятельного решения

При решении задач необходимо описать класс, который используется для представления элементов динамической структуры данных. Затем разрабатывается класс для работы с используемой динамической структурой данных, которая при тестировании класса может быть построена путем ввода данных: a) с клавиатуры; б) из файла. Возможны два варианта решения:

а) динамическая структура данных постоянно хранится в памяти;

б) динамическая структура данных хранится в файле.

1. Создать класс для работы со стеком. Элемент стека – действительное число. Применить класс для вывода возрастающих серий последовательности действительных чисел: a) в обратном порядке; б) в том же порядке (серия – упорядоченная последовательность максимальной длины).

2. Построить класс для работы со стеком. Элемент стека – целое число. Ввести две неубывающие последовательности чисел в два стека. Использовать третий стек для слияния двух последовательностей в одну неубывающую.

3. Создать класс для работы со стеком. Элемент стека – символ. Сформировать два стека, содержащие последовательности символов. Подсчитать общее число элементов в стеках, предусмотреть восстановление их исходного расположения.

4. Создать класс для работы со стеком. Элемент стека – символ. Использовать стек для проверки правильности расстановки скобок трех типов (круглых, квадратных и фигурных) в выражении.

5. Построить класс для работы с односвязным списком. Элемент списка – действительное число. Сформировать список, содержащий неубывающую последовательность чисел, и преобразовать его так, чтобы последовательность была невозрастающей. Для этого необходимо совершить переворот списка, т. е. такую переустановку указателей в списке, при которой элементы его следуют друг за другом в обратном порядке.

6. Построить класс для работы с односвязным списком. Элементы списка – целые числа. Сформировать список, упорядочить элементы списка по возрастанию, используя сортировку: a) методом выбора; б) методом пузырька; в) методом вставки.

7. Построить класс для работы с односвязным списком. Элементы списка – действительные числа. Создать два упорядоченных по невозрастанию списка, слить их в один (также упорядоченный по невозрастанию), построив новый список.

8. Построить класс для работы с односвязным списком. Элементы списка – слова. Создать список, содержащий некоторую последова­тельность слов. Заменить в списке каждое вхождение заданного слова другим (также заданным).

9. Построить класс для работы с односвязным списком. Создать два списка: List1 и List2. Проверить, содержатся ли элементы списка List1 в списке List2 в указанном списком List1 порядке.

10. Построить класс для работы с односвязным списком. Элементы списка – целые числа. Создать список List1. Построить список List2, содержащий порядковые номера максимальных элементов списка List1.

11. Построить класс для работы с двусвязным списком. Элементы списка – действительные числа. Создать список List1, содержащий последовательность . Построить список List2, содержащий последовательность .

12. Создать класс для работы с бинарным деревом, узлы которого содержат целые числа. Построить дерево, затем копию дерева. Подсчитать число листьев в нем (листьями называются узлы, не содержащие поддеревьев).

13. Построить класс для работы с бинарным деревом, узлы которого содержат действительные числа. Создать дерево. Определить высоту дерева (максимальное число узлов, принадлежащих пути от корня дерева до любого из его листьев). Подсчитать число элементов, равных максимальному.

14. Построить класс для работы с бинарным деревом, узлы которого содержат действительные числа. Создать дерево для заданной последовательности чисел. Используя его, упорядочить последовательность по возрастанию, убыванию.

15. Построить класс для работы со списком. Элемент списка содержит информацию о заявке на авиабилет: пункт назначения, номер рейса, фамилию и инициалы пассажира, желаемую дату вылета.

Программа должна обеспечивать: хранение всех заявок в виде списка, добавление заявок в список, удаление заявок, вывод заявок по заданному номеру рейса и дате вылета, вывод всех заявок.

16. Построить класс для работы с бинарным деревом, узел которого содержит информацию о заявках на авиабилеты (в последовательности, используемой для упорядочения заявок): желаемую дату вылета, номер рейса, фамилию и инициалы пассажира, пункт на­значения.

Программа должна обеспечивать: хранение всех заявок в виде бинарного дерева, добавление заявок, удаление заявок, вывод заявок по заданному номеру рейса и дате вылета, вывод всех заявок.

17. Построить класс для работы со списком, который содержит динамическую информацию о наличии автобусов в парке: номер автобуса, фамилию и инициалы водителя, номер маршрута, признак местонахождения автобуса – на маршруте или в парке.

Программа должна обеспечивать: начальное формирование списка, введение номера автобуса при выезде и установление программой значения признака «автобус на маршруте». Аналогичным образом изменяется информация об автобусе при его возвращении в парк. По запросу выдаются сведения об автобусах, находящихся в парке, или об автобусах, находящихся на маршруте.

18. Решить предыдущую задачу, используя не список, а бинарное дерево.

19. Построить класс для работы с бинарным деревом, содержащим англо-русский словарь.

20. Построить класс для работы со списком, содержащим информацию о поездах дальнего следования. Элемент списка содержит следующую информацию о поезде: номер поезда, станция назначения, время отправления.

Составить программу, которая обеспечивает первоначальное формирование списка, производит вывод списка, вводит номер поезда и выводит информацию о нем, вводит название станции назначения и выводит данные о всех поездах, следующих до этой станции.

Тесты

1. На каком этапе (компиляция, выполнение) происходит выделение памяти под динамические структуры данных (ДСД)?
2. Какие основные операции выполняются над следующими ДСД:
3. стек;
4. односвязный список;
5. односвязная очередь;
6. двусвязная очередь;
7. бинарное дерево?
8. Есть ли ошибки в деструкторе дерева:

~Tree(){del\_tree(root);}

void del\_tree(Node \*root){

if(!root){

delete root;

del\_tree(root->left);

del\_tree(root->right);}} ?

1. Есть ли ошибки в деструкторе списка:

~List(){

Node \*V;

if(begin){

V=begin;

while(V){

delete V;

V=V->next;}}} ?

Здесь begin – указатель на начало списка.