**Новосибирский государственный технический университет**

Лабораторная работа №4 по дисциплине **программирование**

Тема: Создание динамического списка объектов,

связанных наследованием. Полиморфизм

|  |  |
| --- | --- |
| Группа: АВТ-907  Студент: Смоленский И.В. | Преподаватель: Булатов А. Д. |

Новосибирск, 2020

**Задание**:

**Вариант 14**

Структура данных: бинарное дерево.

Способ хранения объектов в листьях дерева: ссылки на объекты.

## Теоретические сведения: Виртуальные функции

Виртуальная функция - это функция, вызов которой зависит от типа объекта. В ООП можно писать виртуальные функции так, чтобы объект определял, какую функцию необходимо вызвать, во время выполнения программы. Технику использования виртуальных функций называют полиморфизмом, намекая на то, что объекты, имеющие дело с виртуальными функциями, кажутся приспосабливающимися к контексту, в котором они используются.

Для лучшего понимания виртуальных функций ознакомьтесь с важным принципом классов С++, cвязанных отношением наследования. Cогласно правилам С++, указатель на базовый класс может ссылаться на объект этого класса или на объект любого производного от базового.

Пример: имеется 3 класса: класс В производный от класса А, а класс С производный от класса В.

A aObject; //Объявление объектов классов

B bОbject;

C cОbject;

По определению указатель на класс А может ссылаться на любой из этих объектов, так как они связаны наследованием. Эта взаимосвязь работает только в одном направлении.

A \*p;

p = &cОbject;

Этот принцип становится особенно важным, если в связанных отношениями родства классах определяются виртуальные функции. Эти Функции имеют точно такой же вид и программируются точно так же, как обыкновенные, но добавляется ключевое слово virtual.

class A {

public:

virtual void vf();

};

class B: public A {

public:

virtual void vf();

};

Когда в классе определяется виртуальная функция, имеющая одинаковое имя с виртуальной функцией класса-предка, такая функция называется замещающей. Указатель \*р ссылается на объект типа С, следовательно

р->vf(); //указывает на виртуальную функцию класса С.

#include <iostream.h>

class TValue {

public:

virtual double func(double x) { return x\*x; }

double culc(double x) { return func(x)/2; }

};

class DValue: public TValue {

public:

virtual double func(double x) { return x\*x\*x; }

};

void main() {

TValue obj1;

cout << obj1.culc(3) << endl;

DValue obj2;

cout << obj2.culc(3) << endl;

}

Виртуальными могут быть только нестатические функции-члены.

Виртуальность наследуется. После того как функция определена как виртуальная, ее повторное определение в производном классе (с тем же самым прототипом) создает в этом классе новую виртуальную функцию, причем спецификатор virtual может не использоваться.

Конструкторы не могут быть виртуальными, в отличие от деструкторов. Практически каждый класс, имеющий виртуальную функцию, должен иметь виртуальный деструктор.

**Код:**

**Bi\_Tree.h**

#pragma once

class Bi\_tree

{

public:

Bi\_tree();

Bi\_tree(Bankomat\*);

void push(Bankomat\*);

void remove(Bankomat\*);

Bankomat\* find(char\*);

virtual char\* toString();

private:

Bi\_tree\* l, \* r;

Bankomat\* value;

};

**Bi\_Tree.cpp**

#include <iostream>

#include "Bankomat.h"

#include "Bank\_transition+Bank\_withHistory.h"

#include "Node.h"

Bi\_tree::Bi\_tree(Bankomat\* f)

{

value = f;

l, r = NULL;

}

Bi\_tree::Bi\_tree() {

value = NULL;

}

Bi\_tree\* tree = NULL;

/\*ФУНКЦИЯ ЗАПИСИ ОБЪЕКТА В БИНАРНОЕ ДЕРЕВО\*/

void Bi\_tree::push(Bankomat\* a)

{

if (value == NULL) {

value = a;

return;

}

if (l == NULL && r == NULL)

{

l = new Bi\_tree(a);

r = new Bi\_tree(value);

if (l->value > r->value)

{

Bankomat\* buff = r->value;

r->value = l->value;

l->value = buff;

}

value = r->value;

return;

}

if (l != NULL && l->value > a)

{

l->push(a);

}

else

{

if (r == NULL) {

r = new Bi\_tree(a);

}

else {

r->push(a);

}

}

if (r != NULL) {

value = r->value;

}

else {

value = l->value;

}

}

void Bi\_tree::remove(Bankomat\* a)

{

if (value == NULL) {

return;

}

if (l == NULL && r == NULL) {

if (value == a) {

value = NULL;

}

return;

}

if (l != NULL && l->value > a)

{

l->remove(a);

if (l->value == NULL) {

l = NULL;

if (r != NULL) {

value = r->value;

}

else {

value = NULL;

}

}

}

else if (r != NULL)

{

r->remove(a);

if (r->value == NULL) {

r = NULL;

if (l != NULL) {

value = l->value;

}

else {

value = NULL;

}

}

}

}

Bankomat\* Bi\_tree::find(char\* id) {

if (value == NULL) {

return NULL;

}

if (\*value == id) {

std::cout << value << std::endl;

return value;

}

if (l == NULL & r == NULL) {

return NULL;

}

if (l != NULL && \*l->value > id) {

std::cout << l->find(id) << std::endl;

return l->find(id);

}

else if (r != NULL) {

std::cout << r->find(id) << std::endl;

return r->find(id);

}

}

char\* Bi\_tree::toString()

{

if (value == NULL) return 0;

if (l == NULL && r == NULL)

{

std::cout << \*value << std::endl;

}

if (l != NULL) {

l->toString();

}

if (r != NULL) {

r->toString();

}

return 0;

}

**Вывод**:   
Мной было разработано бинарное дерево, которое хранит объекты в листьях дерева, также был показан принцип работы полиморфизма.