Лабораторная работа № 7. Виртуальные функции и полиморфизм

Часть 1 (до 3 баллов)

Вам даны классы BinaryOperation (бинарный оператор) и Number (число), которые наследуются от базового класса Expression (выражение). Ваша задача реализовать базовый класс Expression так, чтобы не было утечек памяти. Кроме этого подумайте, какие методы стоит сделать виртуальными.

```
#include <cassert> // assert
using namespace std;

struct Expression
{
	// здесь должен быть ваш код
};

struct Number : Expression
{
	Number(double value) : value_(value) {}
	double value() const { return value_; }
	double evaluate() const { return value_; }
private:
	double value_;
};

struct BinaryOperation : Expression
{
enum {
	PLUS = '+',
	MINUS = '-',
	DIV = '/',
	MUL = '*'
	};

BinaryOperation(Expression const *left, int op,
		Expression const *right):left_(left), op_(op), right_(right)
	{
	assert(left_ && right_);
	}

~BinaryOperation()
	{
	delete left_;
	delete right_;
	}

Expression const *left() const { return left_; }
Expression const *right() const { return right_; }
int operation() const { return op_; }
```

```
double evaluate() const
      {
      double left = left_->evaluate();
      double right = right_->evaluate();
      switch (op_)
            {
            case PLUS: return left + right;
            case MINUS: return left - right;
            case DIV: return left / right;
            case MUL: return left * right;
            }
      assert(0);
      return 0.0;
      }

private:
      Expression const *left_;
      Expression const *right_;
      int op_;
};
```

Проверьте полученную иерархию на следующем фрагменте кода:

```
//-------------------------------------------------------------
Expression * e1 = new Number(1.234);
Expression * e2 = new Number(-1.234);
Expression * e3 = new   BinaryOperation(e1,
                              BinaryOperation::DIV, e2);
cout<<e3->evaluate()<<endl;
//-------------------------------------------------------------
```

Часть 2 (до 5 баллов)

Добавьте к иерархии из предыдущего упражнения класс-наследник FunctionCall. FunctionCall должен представлять вызов одной из двух предопределенных математических функций: sqrt — извлечение квадратного корня и abs — вычисление модуля числа. Функция идентифицируется строкой, переданной в качестве параметра в конструктор. Не забудьте, что у функции должен быть аргумент (которым может быть любое выражение Expression)!

```
#include <string> // std::string
#include <cassert>
#include <cmath> // sqrt и fabs
// эти классы из предыдущего упражнения
struct Expression;
struct BinaryOperation;
struct Number;
struct FunctionCall : Expression
{
      /**
      * @name – это имя функции, возможные варианты
      * "sqrt" и "abs".
      *
      * Объекты, std::string можно
```

```
      * сравнивать с C-строками используя
      * обычный синтаксис ==.
      *
      * @arg – выражение-аргумент функции
      */

FunctionCall(/*аргументы*/)
      {
      //реализация
      }
// реализуйте оставшиеся методы из
// интерфейса структуры Expression и не забудьте
// удалить arg_, как это сделано в классе BinaryOperation
//также реализуйте предложенные ниже методы

std::string const & name() const
      {
      // реализация
      }

Expression const *arg() const
      {
      //реализация
      }

~FunctionCall(){/*реализация*/}

private:
      //а тут должны быть поля для FunctionCall
};
```

Проверьте полученную иерархию на следующем фрагменте кода:

```
//-------------------------------------------------------------
Expression* n32 = new Number(32.0);
Expression* n16 = new Number(16.0);
Expression* minus = new BinaryOperation(n32, BinaryOperation::MINUS, n16);
Expression* callSqrt = new FunctionCall("sqrt", minus);
Expression* n2= new Number(2.0);
Expression* mult = new BinaryOperation(n2, BinaryOperation::MUL, callSqrt);
Expression* callAbs = new FunctionCall("abs", mult);
cout<<callAbs->evaluate()<<endl;
//-------------------------------------------------------------
```

Часть 3 (до 7 баллов)

Создайте иерархию классов, используя наследование от абстрактного базового класса. В каждой программе необходимо соблюсти принцип разделения интерфейса и реализации класса. В каждом варианте необходимо написать программу, иллюстрирующую применение всех методов ваших классов. Прежде чем приступить к написанию программ, продумайте (или
посоветуйтесь с преподавателем), какие необходимы функции в каждом из классов. Также продумайте, что следует поместить в закрытые или защищенные переменные. Особенно обратите внимание на то, какие функции следует сделать чистыми виртуальными. Обязательно предусмотрите виртуальный деструктор! В основной программе обязательно используйте
динамическое связывание (без него работа не принимается).

Варианты

1. Координаты точки на плоскости – это способ описания ее положения в двумерном пространстве. Способов описания может быть много, но мы ограничимся декартовыми, полярными и естественными координатами. Рассмотрите класс «Координата» как базовый, а перечисленные способы описания положения на плоскости реализуйте конкретными классами. Можете также поэкспериментировать с функциями – друзьями классов, которые могут переводить один способ описания в другой.
2. Усложним задачу №1 – перейдем в пространство. Здесь остановимся на декартовых, сферических и цилиндрических координатах. Далее сделайте то же, что и в первой задаче.
3. Пока с утра вы едете в университет, вы являетесь пассажиром. Реализуйте абстрактный базовый класс «Пассажир» и конкретные классы – пассажир с билетом, льготной сезонкой, транспортной картой (помните, что их бывает несколько видов), «заяц». Попробуйте подсчитать, сколько пассажиров каждого «вида» едет в час пик и какова выручка кондуктора с них.
4. Реализуйте иерархию Форма -> Точка -> Круг -> Сфера. Можете попробовать написать для этих классов функции рисования, а не только расчета площади, периметра и т.п.
5. То же, что и в задаче №5, но иерархия Форма -> Точка -> Многоугольник -> Многогранник.
6. Все мы любим отдыхать. Но абстрактное понятие «нерабочий день» может на самом деле оказаться конкретным выходным, праздником или отпуском. Реализуйте такую иерархию. Разумеется, самый большой приоритет имеют выходные – на них могут попасть и праздники, и отпуск.
7. Вспомним основы механики. Наиболее общее понятие «движение» конкретизировалось сначала понятиями прямолинейное и криволинейное движение. Прямолинейное движение в свою очередь еще более сильно конкретизировали: равномерное и неравномерное движение. А уж из неравномерного движения выделяли еще равноускоренное движение. С криволинейным все немного проще – достаточно рассмотреть движение по окружности как частный случай более общего криволинейного движения. Реализуйте иерархию классов, используя абстрактный базовый класс «Движение». Не забудьте про основные характеристики движения!
8. Напишите абстрактный класс «Студент», которому наследуют конкретные классы отличник, хорошист, троечник, должник. Не забудьте успевающим студентам начислить стипендию, а неуспевающим – дату пересдачи долгов. Естественно, количество пересдач ограничено – не более 3. Предусмотрите возможность отчисления неуспевающих студентов или ухода их в академический отпуск.
9. При слове «Авиабилет» у вас наверняка возникают самые разные ассоциации – ведь это может быть билет на простой междугородний рейс или на международный. Это может быть билет первого класса, бизнес-класса, второго класса… Реализуйте иерархию «Авиабилет» -> «Междугородний» -> «Международный». Помните, что требования на приобретение международного авиабилета более жесткие, значит данных в этом классе (или функций) будет больше.
10. Представьте, что вы только что выиграли чемпионат по программированию. Вы стали абстрактным победителем, чтобы о вас знали как о конкретном человеке, победившем в конкретном чемпионате, надо определить несколько вещей. Во-первых, свои персональные данные (хотя бы ФИО). Во-вторых, вид и тур соревнования: университетский тур, городской, областной, федеральный или международный. Начиная с абстрактного класса «Победитель чемпионата», реализуйте приведенную выше иерархию классов.

11. Попробуйте немного себя в роли бухгалтера, начисляющего зарплату. Само по себе понятие «Зарплата» не особенно конкретное: оно включает и почасовую, и ставочную зарплату, и комиссионные,
    и процент с продаж (если работа связана с продажей). Реализуйте данные классы.

12. Вычислительное средство само по себе является понятием абстрактным. Оно может представлять из себя бухгалтерские счёты, арифмометр, калькулятор или современную ПЭВМ. Реализуйте такую иерархию.

13. Поговорим о транспортных средствах. На дороге можно встретить грузовой, легковой и пассажирский транспорт. Каждый из этих видов транспорта обладает общими, а также специфическими характеристиками (подумайте, какими). Напишите иерархию классов «Автомобиль» (абстрактный базовый) и далее 3 производных конкретных класса. Подумайте, какие характеристики нужны для вывода на печать, попробуйте также рассчитать средний тормозной путь, грузоподъемность, вместимость соответствующих транспортных средств.

14. Вся компьютерная графика делится на векторную и растровую. Так как сильно отличаются принципы построения рисунков этих видов, можно выделить программы, поддерживающие тот или иной вид графики и сохраняющие рисунки в том или ином формате. Попробуйте написать программу с использованием полиморфизма и абстрактного класса, реализующую данную схему.

15. Не за горами сессия, и скоро такое абстрактное понятие, как «Контроль успеваемости», превратится во вполне реальное понятие зачет или экзамен (в устной или письменной форме, или, быть может,
    в форме теста). Подумайте, какие элементы и функции входят в абстрактный базовый класс «Контроль успеваемости» и как их реализовать для конкретных производных классов. Напишите программу.

16. Попробуйте выстроить логическую цепочку: Студент университета N -> Студент факультета N -> Студент специальности N -> Студент курса N. Вместо N каждый раз будет нужно подставить название вуза, факультета, специальности, номер курса. Реализуйте цепочку через абстрактные и конкретные классы. Возможно, для конкретной специальности придется ввести новые предметы, а для конкретного курса – количество часов на изучение определенных дисциплин.

17. Вы уже не первый год изучаете языки программирования. Наверняка вы знаете, что их можно подразделить на языки структурного программирования и языки объектно-ориентированного программирования. К первым можно отнести, например, C и Pascal, ко вторым – C++, Java... Реализуйте эту структуру через абстрактные и конкретные классы.

18. Понятие "здание" относится к абстрактным. Чтобы его конкретизировать, необходимо знать, является ли оно жилым домом, или торговым центром, или административным зданием, либо спортивным (крытый каток, бассейн) или культурным (театр) объектом... Разумеется, у всех них будут какие-то общие характеристики (например, адрес), и у каждого из них – свои особенные характеристики (скажем, у жилого дома – этажность, количество подъездов, наличие газовых коммуникаций, ГВС, наличие и количество лифтов, наличие мусоропровода и т.д., у бассейна – количество акваторий и их размер (например, две 50-метровые акватории по 9 дорожек), количество мест в раздевалке, душ...). Реализуйте иерархию классов.

19. Все основные неорганические вещества на нашей планете можно подразделить на соли, кислоты, основания и оксиды. Естественно, все категории веществ состоят из химических элементов. Сделав класс «Химический элемент» абстрактным базовым, опишите приведенную здесь иерархию.

Вспомните, что все эти вещества могут взаимодействовать друг с другом, превращаясь друг в друга по определенным законам.

20. Напишите абстрактный класс «Учащийся». Его конкретными реализациями будут ученик, студент, аспирант. Подумайте, как выстроить подобную иерархию и какими общими и специфическими характеристиками будет обладать каждый класс. Напишите реализацию данной иерархии.