Часть 1 (до 5 баллов). Реализуйте класс «Линейный однонаправленный список», как показано в заготовке, поясните принцип организации полей и методов в структуре «Узел», классе «Линейный список». В каком доступе находятся поля и методы структуры TNode? Как называется класс/структура, содержащие в качестве поля ссылку или указатель на самого себя? Какие методы класса TList могут быть сделаны константными? Дополните код класса TList деструктором, адекватно удаляющим список. Продемонстрируйте его работу. Вынесите реализацию всех методов класса за пределы класса. Убедитесь в работоспособности класса на примере приведенного кода в теле main().

```
#include <iostream>
using namespace std;

struct TNode //структура для описания узла в списке
{
	int info;
	TNode *next;
	TNode(int x){
		info=x;
		next=0;}
};

class TList // сам класс «Линейный список»
{
private:
	TNode * head; //голова списка
	TNode * tail; //хвост списка
public:
	TList()
```

```
	{
		//ваш код
	}

	bool addToTail(int x){//добавляем элемент в конец списка
		//ваш код
	}

	bool isEmpty(){ // пуст ли список
		//ваш код
	}

	void print() // вывод списка на печать
	{
		//ваш код
	}

	bool delFromEnd()
	{
		//ваш код
	}

	bool addAfterNode(int valueToSearch, int valueToAdd)
		//добавление после заданного узла
	{
		//ваш код
	}

	bool delNode(int valueToDelete) //удаление заданного узла
	{
		//ваш код
	}
};

int main()
{
TList myList;

if (myList.isEmpty())
	for (int i=0; i<5; i++)
		myList.addToTail(i);

if (!myList.isEmpty()) myList.print();
myList.addAfterNode(0,-1000);

if (!myList.isEmpty()) myList.print();
myList.addAfterNode(2,-222);

if (!myList.isEmpty()) myList.print();
myList.addAfterNode(4,-444);

if (!myList.isEmpty()) myList.print();
myList.delNode(-222);
```

```
if (!myList.isEmpty()) myList.print();

while (!myList.isEmpty())
      {
            myList.delFromEnd();
            myList.print();
      }
return 0;
}
```

Часть 2 (до 5 баллов). В данной лабораторной работе все описания классов должны быть вынесены в отдельный заголовочный файл. Реализация функций-элементов класса должна быть написана в отдельном модуле, а основную программу, иллюстрирующую применение всех методов вашего класса, следует реализовать еще одним модулем. Во всех заданиях предусмотрите конструкторы с аргументами по умолчанию, а также дружественную перегруженную операцию вывода в поток и чтения из потока. Помните, что в каждом классе должны быть предусмотрены константные функции get.

Варианты

1. Создайте класс с именем Complex для выполнения арифметических действий с комплексными числами. Напишите программу для проверки вашего класса. Комплексные числа должны быть представлены в форме RePart+ImPart*i, где i*i = –1. Используйте переменные с плавающей точкой для представления закрытых данных класса.
   Создайте конструктор, деструктор и открытые функции-элементы для следующих действий:
   a) сложение 2 комплексных чисел;
   b) вычитание 2 комплексных чисел;
   c) умножение 2 комплексных чисел.
   Перегрузите операцию вывода в поток для печати комплексного числа в виде |A|exp{i*z}.

2. Создайте класс с именем Rational для выполнения арифметических действий с дробями. Напишите программу для проверки вашего класса. Используйте целые переменные для представления закрытых данных класса – числителя и знаменателя. Создайте функцию-конструктор класса, которая позволяет объекту этого класса принимать начальные значения при его объявлении. Конструктор должен содержать значения по умолчанию на случай отсутствия заданных начальных и должен хранить дроби в сокращенном виде. Создайте открытые функции-элементы для случаев:
   a) сложение 2 дробей (здесь и далее результат должен храниться в сокращенном виде);
   b) вычитание 2 дробей;
   c) перемножение 2 дробей;
   d) деление 2 дробей;
   e) печать дроби в десятичном виде.
   Перегрузите операцию вывода в поток для печати дроби в виде a/b.

3. То же, что и в варианте 2, но для дроби, числитель и знаменатель которой – комплексные числа. Перегруженная операция печати должна выглядеть следующим образом: на экран должна выводиться дробь в виде (ReA+i*ImA)/(ReB+i*ImB).

4. Модифицируйте класс myTime (о котором говорилось на лекциях) так, чтобы включить функцию-элемент tick, которая дает приращение времени, хранящегося в объекте myTime, равное одной секунде. Объект Time должен всегда находиться в непротиворечивом состоянии. Напишите программу для проверки функции-элемента tick в цикле, которая печатала бы время в каком-либо

стандартном формате на каждой итерации цикла и иллюстрировала правильную работу функции-элемента tick. Удостоверьтесь в правильности работы в следующих случаях:

a) приращение с переходом в следующую минуту;
b) приращение с переходом в другой час;
c) приращение с переходом в другой день.

Предусмотрите перегруженную операцию вывода в поток, а также перегруженную операцию инкремента в префиксной и постфиксной формах.

5. Модифицируйте класс myDate (о котором говорилось в лекциях) так, чтобы в нем присутствовала функция-элемент nextDay, которая будет увеличивать дату на 1 день. Объект myDate должен всегда находиться в непротиворечивом состоянии. Напишите программу, проверяющую функцию nextDay в цикле и печатающую время в стандартном формате на каждой итерации цикла. Проверьте правильность работы функции в следующих случаях:

   a) приращение с переходом в следующий месяц;
   b) приращение с переходом в следующий год.

   Предусмотрите перегруженную операцию вывода в поток, а также перегруженную операцию инкремента в префиксной и постфиксной формах.

6. Создайте класс параллелограмм, который хранит только декартовы координаты его четырех углов. Конструктор вызывает набор функций, которые принимают 4 группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 50,0. Это множество функций также должно проверять, что переданные координаты определяют параллелограмм. Должны быть предусмотрены функции-элементы, вычисляющие длины сторон параллелограмма, периметр и площадь. Включите функцию, которая определяла бы, не является ли параллелограмм прямоугольником.
   Перегрузите операцию вывода в поток для печати всей информации об объекте.

7. Создайте класс Rectangle (прямоугольник). Класс имеет атрибуты length (длина) и width (ширина), каждый из которых по умолчанию равен 1. Он имеет функции-элементы, которые вычисляют периметр и площадь прямоугольника. Он имеет функции записи и чтения для длины и ширины. Функции записи должны проверять, что длина и ширина – числа с плавающей запятой, находящиеся в пределах от 0,0 до 20,0.
   Перегрузите операцию вывод в поток для печати всей информации о прямоугольнике.

8. Создайте класс прямоугольник, который хранит только декартовы координаты четырех углов прямоугольника. Конструктор вызывает набор функций, которые принимают 4 группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 20,0. Это множество функций также должно проверять, что переданные координаты определяют прямоугольник. Должны быть предусмотрены функции-элементы, вычисляющие длину и ширину прямоугольника, периметр и площадь. Включите функцию, которая определяла бы, не является ли прямоугольник квадратом.
   Перегрузите операцию вывод в поток для печати всей информации о прямоугольнике.

9. Модифицируйте класс прямоугольник из варианта №8 так, чтобы включить в него функцию draw, которая изображает прямоугольник внутри окна 25 на 25. Включите функцию setFillCharacter, чтобы задавать символ, которым будет заполняться прямоугольник внутри. Включите функцию setPerimeterCharacter, чтобы задавать символ, которым будут печататься границы прямоугольника. Включите функцию поворота прямоугольника на 90 градусов вокруг одной из его вершин против и по часовой стрелке.

Перегрузите операцию вывод в поток для печати всей информации о прямоугольнике и рисования прямоугольника.

10. Создайте класс треугольник, хранящий только декартовы координаты вершин. Конструктор вызывает набор функций, которые принимают 3 группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 50,0. Функции также должны проверять, чтобы треугольник не «схлопывался» в прямую линию. Должны быть предусмотрены функции-элементы, вычисляющие длину сторон, периметр и площадь треугольника. Включите функцию, которая определяла бы, не является ли треугольник равнобедренным, равносторонним или прямоугольным.
    Перегрузите операцию вывода в поток для печати всей информации о треугольнике.

11. Создайте класс треугольник, хранящий длины двух сторон и значение угла между ними. Должны быть предусмотрены функции-элементы, вычисляющие длину третьей стороны, значения 2 оставшихся углов, периметр и площадь треугольника. Включите функцию, которая определяла бы, не является ли треугольник равнобедренным, равносторонним или прямоугольным.
    Перегрузите операцию вывода в поток для печати всей информации о треугольнике.

12. Создайте класс прямая призма, хранящий только декартовы координаты вершин основания и высоту призмы. Конструктор вызывает набор функций, которые принимают группы координат и проверяют, чтобы каждая из координат x и y находилась в первом квадранте в диапазоне от 0,0 до 250,0. Должны быть предусмотрены функции-элементы, вычисляющие длину ребер, периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем призмы. Перегрузите операцию вывода в поток так, чтобы она печатала, какая фигура лежит в основании, и ее основные характеристики.

13. Создайте класс пирамида, хранящий только декартовы координаты вершин основания и вершины пирамиды. Конструктор вызывает набор функций, которые принимают группы координаты. Должны быть предусмотрены функции-элементы, вычисляющие длину ребер, периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем пирамиды.
    Перегрузите операцию вывода в поток так, чтобы она печатала, какая фигура лежит в основании, и ее основные характеристики.

14. Создайте класс конус, хранящий только декартовы координаты центра основания, радиус основания и высоту конуса. Должны быть предусмотрены функции-элементы, рассчитывающие периметр и площадь основания, а также площадь боковой поверхности, площадь поверхности и объем конуса.
    Перегрузите операцию вывода в поток для печати всей информации об объекте.

15. Создайте класс усеченный конус, хранящий только декартовы координаты центра основания, радиусы оснований и высоту конуса. Должны быть предусмотрены функции-элементы, рассчитывающие периметр и площадь оснований, а также площадь боковой поверхности, площадь поверхности и объем конуса.
    Перегрузите операцию вывода в поток для печати всей информации об объекте.

16. Создайте класс усеченная пирамида, хранящий только декартовы координаты вершин оснований. Конструктор вызывает набор функций, которые принимают группы координат одного основания. Высота пирамиды задается случайным образом, координаты второго основания вычисляются в соответствии с высотой. Должны быть предусмотрены функции-элементы, вычисляющие периметр и

площадь основания, а также площадь боковой поверхности, площадь поверхности и объем пирамиды.

Перегрузите операцию вывода в поток для печати всей информации об объекте (какая фигура лежит в основании, ее основные характеристики).

17. Создайте класс "Спортсмен", в котором будут храниться данные о ФИО атлета, представляемой стране, виде спорта, установленных рекордах, завоеванных местах на первенствах страны, континента, мира и олимпиадные достижения. Предусмотрите методы "Установить рекорд", "Получить медаль" (помните, что медали бывают разные...). Предусмотрите также все необходимые методы установки и чтения данных-элементов.

    Перегрузите операцию вывода в поток для вывода информации о спортсмене.

18. Создайте класс «Запись в адресной книге». В нем хранятся фамилия и имя человека, номера телефонов (нескольких, в т.ч. домашних, рабочих и сотовых), район и адрес проживания, e-mail (несколько). Конструктор должен вызывать функцию, считывающую эти данные из текстового файла. Напишите функции-элементы для установки и чтения данных. Предусмотрите метод поиска номеров сотовых телефонов, метод формирования текстового файла с заголовком и подписью (как заготовки письма на электронный адрес).

    Перегрузите операцию вывода в поток для печати информации об объекте вида «Запись в адресной книге».

19. Создайте класс матрица размерностью $n \times n$, который хранит только размерность матрицы и максимальное по модулю значение элемента матрицы (и указатель на целое). Конструктор должен вызывать функцию заполнения матрицы случайными числами в заданном диапазоне. Напишите функции-элементы для:
    a) транспонирования матрицы;
    b) умножения матрицы на число;
    c) сложения матриц;
    d) умножения двух матриц.

    Перегрузите операцию вывода в поток для печати всей информации об объекте. Перегрузите также оператор вычитания матриц.

20. Создайте класс правильный многоугольник, который хранит число вершин и их координаты. Конструктор вызывает набор функций, которые проверяют, чтобы число вершин было не менее 3, чтобы многоугольник был правильным, и в случае ошибки устанавливают значения всех вершин в 0. Должны быть предусмотрены функции-элементы, вычисляющие периметр, площадь многоугольника. Должна быть предусмотрена функция-элемент вывода на печать информации о числе сторон многоугольника. Напишите программу-драйвер, иллюстрирующую применение вашего класса.

    Перегрузите операцию вывода в поток для печати всей информации об объекте.

Лабораторная работа № 2. Наследование

Создайте иерархию классов, используя наследование. В каждой программе необходимо соблюсти принцип разделения интерфейса и реализации класса (иными словами, не забывайте выделять заголовочные файлы). В каждом варианте необходимо написать программу, иллюстрирующую применение всех методов ваших классов. Прежде чем приступить к написанию программ, продумайте (или уточните у преподавателя), какие необходимы функции в каждом из классов (может, где-то необходимо считать координаты, где-то площади и объемы, а где-то хранить фамилии и года поступления): как в базовом, так и в классах-наследниках. Также продумайте, что следует поместить в закрытые (а, возможно, защищенные) переменные. Предусмотрите возможность переопределения методов базового класса в производном. Приветствуется демонстрация наследования перегруженных операций. Возможно, в некоторых вариантах лучше воспользоваться композицией, чем наследованием.

Варианты

1. Точка -> Квадрат -> Куб.
2. Четырехугольник -> Трапеция -> Параллелограмм.
3. Точка -> Четырехугольник -> Параллелепипед.
4. Точка -> Треугольник -> Треугольная призма.
5. Точка -> Круг -> Сфера.
6. Факультет:
    a) администрация;
    b) преподаватель;
    c) студент.
7. Учащийся в университете:
    a) студент;
    b) аспирант;
    c) слушатель.
8. Студент:
    a) первокурсник;
    b) студент 2-4 курса;
    c) дипломник.
9. Круг -> Конус -> Усеченный конус.
10. Треугольник -> Треугольная пирамида -> Усеченная треугольная пирамида.
11. Прямолинейное движение делится на равномерное и равноускоренное. Равноускоренное в свою очередь может реализовываться свободным падением по вертикали.
12. Криволинейное движение можно разбить на движение по окружности и падение тела, брошенного под углом к горизонту. Из движения тела, брошенного под углом к горизонту, можно выделить свободное падение тела, брошенного горизонтально.
13. Спортсмен -> Легкоатлет -> Спринтер.
14. Точка -> Треугольник -> Равнобедренный треугольник -> Равносторонний треугольник.
15. Транспортное средство:
    a) трамвай;
    b) троллейбус;
    c) автобус.
16. Чемпионат по программированию -> Университетский тур -> Городской тур -> Областной тур.
17. Студент -> Математик -> Математик-программист.
18. Среди накопителей информации можно выделить такие классы, как жесткий диск и флеш-карта. Среди жестких дисков, в свою очередь, можно выделить класс съемных дисков.

19. Точка -> Треугольник -> Прямоугольный треугольник -> Равнобедренный прямоугольный треугольник.
20. Точка –> Четырехугольник -> Ромб -> Квадрат.

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.

Лабораторная работа № 3. Обработка исключений

В данной лабораторной работе вам нужно написать класс исключения и программу, способную генерировать и обрабатывать определенный вид исключения (программа должна содержать блоки try, catch, точку throw). Позаботьтесь о том, чтобы исключение действительно могло возникнуть, продемонстрируйте работу перехватчика и обработчика исключений.

Варианты

Обработать следующие исключения:

1. Взятие квадратного корня из отрицательного числа.
2. Нехватка памяти при динамическом ее выделении.
3. Попытка записи в файл, открытый только для чтения.
4. Ввод пользователем вещественного числа вместо целого.
5. Ввод пользователем строки вместо числа.
6. Ввод пользователем специального символа вместо целого числа.
7. Взятие натурального логарифма от нуля.
8. Взятие натурального логарифма от отрицательного числа.
9. Ввод пользователем несуществующей даты, например, «31 февраля».
10. Ввод несуществующего времени, например, 56 час. 335 мин. 98 сек.
11. Ввод пользователем отрицательного возраста.
12. Ввод пользователем отрицательной зарплаты.
13. Попытка чтения из несуществующего файла.
14. Попытка записи в несуществующий файл.
15. Неопределенность вида «0/0».
16. Неопределенность вида «0*∞».
17. Ввод числа, спецсимволов или русских букв вместо символов латинского алфавита.
18. Ввод числа, спецсимволов или латинских букв вместо символов русского алфавита.
19. Превышение при вводе максимально возможной длины строки.
20. Выход индекса за пределы массива.