МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ УКРАИНЫ
НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ УКРАИНЫ
«КИЕВСКИЙ ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ»

ИСКЛЮЧИТЕЛЬНЫЕ СИТУАЦИИ С++

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторной работе № 5

по дисциплине «ООП»

Киев 2016

СОДЕРЖАНИЕ

1 ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Цель работы – изучить особенности генерации и обработки исключительных ситуаций в C++.

2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

Исключительная ситуация – это необычные или неожиданные обстоятельства, возникающие в работе программы: сбои аппаратуры, ошибки ввода-вывода, программные ошибки. Исключительные ситуации приводят к прерыванию программы, после чего следует необходимо обнаружить исключение и обработать его таким образом, чтобы исключить прерывание при выполнении программы.

Для работы с исключениями в C++ используются такие операторы:

– `try` – начало блока исключений;

– `catch` – начало блока, обрабатывающего исключение. Оператор `catch` может располагаться в любой части программы, не возвращает значения. Оператор используется для вывода на экран сообщений об исключении, в качестве аргумента может принимать ссылку и указатель, также может обрабатывать нетипизированные исключения. Формат оператора:

```
catch(тип_объекта) {программный код}
```

Если в качестве типа объекта используется …, то будут «пойманы» все исключения.

– `throw` – возбуждение (создание) исключения. Данный оператор может использоваться в нескольких форматах:

– `throw(type)` – исключение заданного типа;

– `throw(type1, type2, …)` – создание исключения, зависящего от нескольких типов

– `throw класс(аргументы)` – исключение заданного класса

– `throw()` – не генерируется исключение.

Для обработки исключений используются так называемые блоки `try-catch`:

```
try{
   //вызов функции, генерирующей исключение
}
catch(тип_объекта){
```


```
  //тело обработчика исключений
}
```

Например:

```
void func(){
  try{
     throw 1;
  }
  catch(int a){
     cout<<"Exception  "<<a<<endl;
     return;
  }
  cout<<"No exception detected!  "<<endl;
  return;
}
```

В случае, если кидаются нетипизированнный данные (экземпляры классов, структур и т.д.), лучше ловить их по ссылке. Если кидаются данные нескольких типов, то можно использовать несколько блоков catch, ловящих свой тип данных:

```
try{
     throw 1;
  //throw 'a';
  }
  catch(long b){
     cout<<"Пойман тип long  "<<b<<endl;
  }
  catch(char b){
     cout<<"Пойман тип char  "<<b<<endl;
  }
```

Когда возбуждается исключительная ситуация, программа просматривает стек функций до тех пор, пока не найдёт соответствующий catch. Если такой оператор не найден, то исключение обрабатывается в стандартном обработчике, показывая непонятные сообщения и аварийно завершая программу.

Блоки `try-catch` могут содержать вложенные блоки `try-catch`, и если не будет определено соответствующего оператора `catch` на текущем уровне вложения, исключение будет поймано на более высоком уровне. Операторы, следующие за оператором throw, никогда не выполняются:

```
try{
     throw;
  }
  catch(…){
     cout<<"Исключение  "<<b<<endl;
  }
```


Такой подход может применяться в случаях, когда не нужно передавать никаких данных в блок `catch`.

Методика использования исключений:

– определяется исключение в виде класса, конструктор класса при выбросе исключения будет генерировать исключение; в классе должны быть реализованы методы вывода сообщения об исключительной ситуации;

– определяются функции, которые могут генерировать исключения и определяются функции исключения данного типа;

– определяются арифметические, логические и др. действия, в результате которых генерируются исключения, инициализируются исключения при помощи конструктора класса;

– в основной программе определяется бесконечный цикл, обеспечивающий повторение заданной части программы определённое количество раз, пока не будут введены или ликвидированы последствия исключений;

– с помощью блока `try-catch` обрабатываются исключения, получают необходимые диагностические сообщения. `try-catch`.

Пример: возвести произвольное число в произвольную степень, предусмотреть генерацию исключительных ситуаций.

```
#include <iostream>
#include <math.h>
#include <conio.h>

class Error{
    double b;
    double e;
  public:
    Error(double base,
          double ex):b(base),e(ex){};
    void report();
};

void Error::report(){
  cout<<"Error!!"<<"Base "<<b;
  cout<<" To "<<e<<endl;
  getch();
}
```


```
double Power(double b,
             double e) throw (Error){
  if(b>0.0) return exp(e*log(b));
  if (b<0.0){
    double ipart;            //целая часть
    double fpart=modf(e,&ipart);
    if(fpart==0){
      if(fmod(ipart,2)!=0)
         return -exp(e*log(b));
      else return exp(e*log(b));
    }
    else throw Error(b,e);
  }
  else{
    if(e==0) return 1.0;
    if(e<1.0) throw Error(b,e);
    return 0.0;
  }
}

void main(){
  for(;;){
   try{
    try{
     double bb,ee,res;
     cout<<"Base? ";cin>>bb;
     cout<<"Exponent? ";cin>>ee;
     res=Power(bb,ee);
     cout<<"Result = "<<res<<endl;
    }
    catch(Error &e){
     e.report();
     throw e;
    }
    return;
   }
   catch(...){
     cout<<"Try again!"<<endl;
   }
  }
}
```

Использование иерархий классов, связанных наследованием, может оказаться весьма полезным при обработке исключений. Например, исключения для математической библиотеки можно организовать следующим образом:

```
class MathErr{};
class Overflow : public MathErr{};
class Underflow : public MathErr{};
class ZeroDivide : public MathErr{};
…
void f(){
  try{
    ///…
  }
  catch(ZeroDivide) {
    //обработка только объектов
    // данного класса и
    // всех производных от него
```


```
  }
  catch(MathErr){
  }
}
```

Организация исключений в виде иерархий может иметь большое значение для надёжности кода. Исключение обычно перехватывается обработчиком базового класса, а не обработчиком собственного класса, например:

```
#include <iostream>
#include <conio.h>

class MathErr{
  //…
  public:
  virtual void error_print(){
    cout<<"Math Error!"<<endl;
  }
};

class Int_overflow : public MathErr{
  const char *op;
  int a1,a2;
 public:
  Int_overflow(const char *p,
               int a,
               int b):op(p),a1(a),a2(b){};
  virtual void error_print(){
   MathErr::error_print();
   cout<<"Oveflow - "<<op;
   cout<<" ( "<<a1<<" , "<<a2<<" ) \n";
   _getch();
   }
};

int add(int x, int y){
 if((x>0&&y>0&&x>INT_MAX-y)||
    (x<0&&y<0&&x<INT_MIN-y))
      throw Int_overflow("+",x,y);
return x+y;
}

void main(){
  try{
    int i1= add(1,2);
    cout<<i1<<endl;
    int i2 = add(INT_MAX,-2);
    cout<<i2<<endl;
    int i3 = add(INT_MAX,2);
    cout<<i3<<endl;
  }
  catch(MathErr &m){
   m.error_print();
  }
}
```


2.1. Необработанные исключения

Любое исключение, возникшее в процессе работы программы, обязательно обрабатывается, если для него имеется оператор `catch()`; если же исключение не находит соответствующий `catch()`, то необработанное исключение передаётся по цепочке вызова функции до тех пор, пока не останется нерассмотренных исключений. Если `catch()` не будет найден, то исключение обрабатывается стандартными функциями `unexpected()`, `terminate()`, `abort()`.

Основное назначение `unexpected()` – подготовить программу к прерыванию, далее вызывается функция `terminate()`, вызывающая `abort()` – ненормальное завершение программы. Если необходимо обработать неспецифицированные исключения без использования стандартной `unexpected()`, не допускающей вмешательства, то выполняется переопределение функций `unexpected()` и `terminate()` при помощи функций `set_unexpected()` и `set_terminate()`.

3 ЗАДАНИЯ

Разработать пошаговый алгоритм решения расчётной части задачи и подзадач (при необходимости). Разработать UML диаграмму классов. Выполнить программную реализацию задания согласно варианту. Прототипы классов должны содержаться в отдельном *.h-файле. В программе обязательно предусмотреть вывод информации об исполнителе работы (ФИО), номере варианта, выбранном уровне сложности и задании. Предусмотреть возможность повторного запуска программы (запрос о желании выйти из программы, или продолжить работу). Ключевые моменты программы обязательно должны содержать комментарии.

Уровень А (1 балл)

1. Вычислить сумму двух матриц. При выходе значения суммы за заданный пользователем диапазон и задании неверного размера матриц генерировать исключительную ситуацию.

2. Осуществить поиск заданного элемента в массиве. При отсутствии необходимого значения генерировать исключительную ситуацию.

3. Осуществить поиск заданного файла и вывод на экран его содержимого. В случаях отсутствия файла или невозможности его прочитать генерировать исключительную ситуацию.

4. Вычислить значение выражения, генерировать исключение при невозможности вычисления и при выходе вычисленного значения за диапазон, заданный пользователем:

$$\frac{\ln(x)}{\ln(x-2)}$$

5. Для заданной матрицы определить обратную (с вычислением детерминанта) с генерацией исключительных ситуаций при задании некорректной размерности матрицы и отсутствии обратной матрицы.

6. Осуществить поиск заданного числа в файле. При отсутствии заданного числа или файла генерировать исключительную ситуацию.

7. Определить класс short int, работающий как стандартный, но генерирующий исключительную ситуацию при превышении разрешённого диапазона значений.

8. Вычислить значение выражения, генерировать исключение при невозможности вычисления и при выходе вычисленного значения за диапазон, заданный пользователем:

$$\frac{x}{\sqrt{x^2+3x+2}}$$

9. Разработать базовый класс «Исключение» и производные классы для конкретных исключительных ситуаций: ошибки преобразования строки в число, недопустимое преобразование типов, переполнение.

10.Определить класс bool, работающий как стандартный, но генерирующий исключительную ситуацию при превышении разрешённого диапазона значений.

11.Осуществить замену символа в файле. При отсутствии файла и невозможности произвести замену (файл имеет атрибут «readonly») генерировать исключительную ситуацию.

12.Определить класс «Целое число» с генерацией исключительной ситуации «выход за рамки допустимого диапазона».

13.Вычислить значение выражения, генерировать исключение при невозможности вычисления:

$$arctg^2(x+1)-\frac{n!}{n-m}$$

14.Осуществить поиск заданного файла и запись в него введённого текста. В случаях отсутствия файла или невозможности записи в него генерировать исключительную ситуацию.

15.Разработать программу, которая формирует исключительную ситуацию при превышении отведенного для неё времени работы. Пользователь может продолжить работу, задав необходимые данные с клавиатуры.

16.Определить класс unsigned int, работающий как стандартный, но генерирующий исключительную ситуацию при превышении разрешённого диапазона значений.

17.Определить класс «Обратные тригонометрические функции». При выходе аргумента за область допустимых значений генерировать исключение.

18.Вычислить произведение двух матриц. При задании некорректных размерностей матрицы генерировать исключение.

19.Определить класс date, который хранит дату в заданном формате, генерировать исключительную ситуацию при несоответствии формата.

20.Определить корни квадратного уравнения $ax^2+bx+c=0$. Если уравнение не является квадратным (a=0) или не имеет действительных корней генерировать исключение.

21.Вычислить значение выражения, генерировать исключение при невозможности вычисления и при выходе вычисленного значения за диапазон, заданный пользователем:

$$\frac{x}{\sqrt{x^2-5x+6}}$$

22.Осуществить поиск заданной записи в структуре данных. При отсутствии требуемой записи генерировать исключительную ситуацию.

23.Определить класс int, работающий как стандартный, но генерирующий исключительную ситуацию при превышении разрешённого диапазона значений.

24.Разработать тестовую программу, использующую все возможные форматы оператора throw и демонстрирует последовательность передачи на соответствующий обработчик catch().

25.Разработать тестовую программу, демонстрирующую работу стандартного обработчика исключений (unexpected(), terminate(), abort(), set_unexpected(), set_terminate()).

26.Разработать класс «Линейный список», имеющий методы добавления в заданную позицию, удаления из заданной позиции, удаления заданного элемента, поиска заданного элемента. Генерировать исключительные ситуации при отсутствии заданной позиции, заданного элемента и т.д.

27.Определить класс long long int, работающий как стандартный, но генерирующий исключительную ситуацию при превышении разрешённого диапазона значений.

28.Вычислить значение выражения, генерировать исключение при невозможности вычисления и при выходе вычисленного значения за диапазон, заданный пользователем:

$$\left(\frac{m}{n}\right)^{\pm\frac{k}{p}}$$

29.Определить класс Matrix, работающий как стандартная матрица, но генерирующий исключительную ситуацию при обращении к несуществующему элементу.

30.Определить класс char, работающий как стандартный, но генерирующий исключительную ситуацию при превышении разрешённого диапазона значений.

Уровень Б (+0,5 балла)

Для задания лабораторной работы №4 (уровень Б), реализовать обработку всех ошибок и особых случаев при помощи иерархии классов исключений. При необходимости обработать некорректную инициализацию элементов данных в конструкторе класса. Вводимые данные могут быть абсолютно любыми.

1. Спроектировать класс «Matrix_diag», который содержит: размерность матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операцию унарный минус «-», операцию умножения матрицы на число «*», и ее же в сокращенной форме, операцию деления матрицы на число «/», и ее же в сокращенной форме, операцию сведения к диагональному виду «~», если возможно. Добавить метод нахождения определителя матрицы. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

2. Спроектировать класс «Binary», который содержит число в двоичной системе счисления. Для него определить: операцию присвоения «=», операцию преобразования в обратный код «!» и операцию преобразования в дополнительный код «~». Реализовать алгоритмы двоичного сложения, операция «+» и двоичного вычитания, операция «-». Продемонстрировать каждую операцию.

3. Спроектировать класс «Complex», который содержит: вещественную и мнимую часть комплексного числа. Для него определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/», эти же операции в сокращенной форме, операцию унарный минус «-», операцию сопряжения «~». Добавить метод нахождения модуля комплексного числа. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

4. Спроектировать класс «Binary_heap», который содержит: двоичную кучу (пирамиду), заданную в виде дерева или массива. Для него определить: операцию добавления элемента «<<», операцию удаления элемента «>>», операцию инвертирования кучи «!».При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

5. Спроектировать класс «Set», который содержит множество целочисленных элементов, обязательно уникальных и упорядоченных. Для него определить: операцию объединения «+», операцию пересечения «*», операцию разности «/». Продемонстрировать каждую операцию.

6. Спроектировать класс «List», который содержит двусвязный список целочисленных элементов. Для него определить: операцию соединения двух списков «+», ее же в сокращенной форме, операцию обращения списка «!». Добавить методы поиска элемента по значению и по индексу, методы удаления и добавления новых элементов в начало и в конец списка. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

7. Спроектировать класс «Numeral_system», который содержит: идентификатор системы счисления (2,8,16) и число в заданной системе счисления. Для него определить: оператор преобразования к целочисленному типу «int» и оператор преобразования к действительному типу «double». Продемонстрировать каждую операцию.

8. Спроектировать класс Date, который содержит дату в формате дд/мм/гг. Для него определить: операцию «+», прибавляющую к дате другую дату, операцию «-» отнимающую от даты другую дату и оператор «int», который возвращает UNIX-время, для заданной даты. Продемонстрировать каждую операцию.

9. Спроектировать класс «Matrix_rank», который содержит: высоту и ширину матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операцию унарный минус «-» и операцию сведения к треугольному виду «~». Добавить метод нахождения ранга матрицы. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

10.Спроектировать класс «Bignum_arithmetic», который содержит: длинное число, т.е. число, значения которого превышает максимально (минимально) допустимые значения целочисленного типа (MIN_INT и MAX_INT). Для него определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/». Предусмотреть возможность применять данные операции для базового типа int и объекта класса Bignum_arithmetic в любом порядке (**Bignum_arithmetic+int** или **int+Bignum_arithmetic**). Продемонстрировать каждую операцию.

11.Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для класса определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/». При возможности, обязательно сокращать дроби. Для нахождения наибольшего общего делителя использовать бинарный алгоритм. При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

12.Спроектировать класс «Algebraic_expressions», который содержит строку, представляющую собой алгебраическое выражение. Для него определить: операцию унарный минус «-», которая меняет знак выражения; операцию «+», которая складывает два выражения; операцию «-», которая вычитает из первого выражения второе. При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

13.Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для класса определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/».Предусмотреть возможность применять данные операции для базового типа int и объекта класса Fraction в любом порядке (**Fraction+int** или **int+Fraction**). При возможности, обязательно сокращать дроби. Для нахождения наибольшего общего делителя использовать бинарный алгоритм. При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

14.Спроектировать класс «Linear_system», который содержит: размерность СЛАУ, двойной указатель на тип **double** (матрица коэффициентов), указатель на тип **double** (вектор свободных членов). Для него определить: операцию нахождения обратной матрицы «!» и операцию умножения матриц «*». При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать решение СЛАУ матричным методом.

15.Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для класса определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/».Предусмотреть возможность применять данные операции для базового типа double и объекта класса Fraction в любом порядке (**Fraction+double** или **double+Fraction**). При возможности, обязательно сокращать дроби. Для нахождения наибольшего общего делителя использовать бинарный алгоритм. При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

16.Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операцию нахождения транспонированной матрицы «~». При

необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

17.Спроектировать класс «Linear_system», который содержит: размерность СЛАУ, двойной указатель на тип **double** (матрица коэффициентов), указатель на тип **double** (вектор свободных членов). Для него определить: операцию нахождения решения СЛАУ любым методом «~».

18.Спроектировать класс «Matrix_symmetric», который содержит: высоту и ширину матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операцию нахождения транспонированной матрицы «~», операцию умножения матриц «*», ее же в сокращенной форме и операцию сведения к симметричному виду «!». При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

19.Спроектировать класс «Stack», который содержит стек целочисленных элементов. Для него определить: операцию добавления элемента «<<» и удаления элемента «>>». При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

20.Спроектировать класс «Vector», который содержит координаты вектора. Для него определить: операцию сложения «+», операцию вычитания «-», операцию векторного произведения «*», операцию умножения на константу «*». При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

21.Определить класс «Binary_search_tree», который содержит: двоичное дерево поиска. Для него определить: операцию добавления элемента «<<», операцию удаления элемента «>>», операцию объединения двух деревьев «+», операцию разбиения по ключу «>>=» ($\geq K_0$) и «<<=» ($<K_0$). При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

22.Спроектировать класс «Queue», который содержит очередь целочисленных элементов. Для него определить: операцию добавления элемента «<<» и удаления элемента «>>». При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

23.Спроектировать класс «Multi_set», который содержит множество упорядоченных символьных элементов. Для него определить: операцию добавления «<<» и удаления «>>» элементов из множества, операцию «=» для инициализации множества при помощи строки и для инициализации другим множеством. При необходимости разрешается определять другие операции. Продемонстрировать каждую операцию.

24.Спроектировать класс «List», который содержит двусвязный список целочисленных элементов. Для него определить: операцию удаления элементов из первого списка, которые входят во второй «-». При необходимости разрешается определять другие операции (например «=»). Продемонстрировать каждую операцию.

25.Спроектировать класс «LList», который содержит кольцевой список целочисленных элементов. Для него определить: операцию добавления элемента «<<» и удаления элемента «>>».При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

26.Спроектировать класс «Tree», который содержит: бинарное дерево целочисленных элементов. Для него определить: операцию добавления элемента «<<», операцию удаления элемента «>>».При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

27.Спроектировать класс «String», который содержит строку заданной длины. Для него определить: операцию присвоения «=», операцию

конкатенации строк «+», сокращенную операцию конкатенации строк «+=», операцию «++», которая приводит строку к верхнему регистру и операцию «--», которая приводит строку к нижнему регистру. Продемонстрировать каждую операцию.

28.Спроектировать класс «List», который содержит двусвязный список целочисленных элементов. Для него определить: операцию сортировки элементов списка по возрастанию «++».При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

29.Спроектировать класс «List», который содержит двусвязный список целочисленных элементов. Для него определить: операцию сортировки элементов списка по убыванию «--».При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

30.Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операции нахождения максимального «++» и минимального «--» элементов. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

Уровень B (+1 балл)

Для задания лабораторной работы №4 (уровень B), реализовать обработку всех ошибок и особых случаев при помощи иерархии классов исключений. При необходимости обработать некорректную инициализацию элементов данных в конструкторе класса. Вводимые данные могут быть абсолютно любыми.

1. Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операцию сложения матриц «+», операцию вычитания матриц «-»,

операцию умножения матриц «*», операцию деления матрицы на число «/», эти же операции в сокращенной форме, операцию нахождения обратной матрицы «!», операцию нахождения определителя «~». При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию, программа должна выполнять заданные операции за приемлемое время для матриц размерности не менее 100x100.

2. Спроектировать класс «Roman_numerals», который содержит: число в римской форме записи (в виде строки). Для него определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/», эти же операции в сокращенной форме, операцию унарный минус «-» и операции постфиксного и префиксного сложения/вычитания: «++» и «--». При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

3. Спроектировать класс «Set», который содержит множество целочисленных элементов, обязательно уникальных и упорядоченных. Для него определить: операцию объединения «+», ее же в сокращенной форме, операцию пересечения «*», ее же в сокращенной форме, операцию разности «-», ее же в сокращенной форме. Определить операции поэлементного добавления и удаления: «<<» и «>>» соответственно. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию. Множество вводится пользователем в виде строки с элементами разделенными запятыми, и преобразовывается в корректное множество.

4. Спроектировать класс «Numeral_system», который содержит: идентификатор системы счисления (2,8,16) и число в заданной системе счисления. Для него определить: операцию сложения «+», операцию вычитания

«-», операцию умножения «*», операцию деления «/», эти же операции в сокращенной форме, операцию унарный минус «-», и операции постфиксного и префиксного сложения/вычитания: «++» и «--», все логические операции сравнения. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

5. Спроектировать три класса «Numeral_2», «Numeral_8», «Numeral_16» каждый из них содержит: число в соответствующей системе счисления (целое или действительное). Определить для каждого из этих классов операторы преобразования к другим классам и к базовым типам (по возможности). Определить для классов: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/», эти же операции в сокращенной форме. Операции должны выполняться между объектами разных классов, и между объектами и базовыми типами. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию и преобразования.

6. Спроектировать класс «Date», который содержит дату в некоторых допустимых форматах (в виде строки). Форматов должно быть не менее 3-х. Для него определить: операцию «+», прибавляющую к дате заданное число дней или другую дату, операцию «-» отнимающую от даты заданное количество дней или другую дату. Дату можно передавать в любом допустимом формате. Определить все логические операции сравнения. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

7. Спроектировать класс «Set», который содержит множество символьных элементов, обязательно уникальных и упорядоченных. Для него определить: операцию объединения «+», ее же в сокращенной форме, операцию

пересечения «*», ее же в сокращенной форме, операцию разности «-», ее же в сокращенной форме. Определить операции поэлементного добавления и удаления: «<<» и «>>» соответственно. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию. Множество вводится пользователем в виде строки с элементами разделенными запятыми, и преобразовывается в корректное множество. Множество не должно содержать строки.

8. Спроектировать класс «Bignum_arithmetic», который содержит: длинное число, т.е. число, значения которого превышает максимально (минимально) допустимые значения целочисленного типа (MIN_INT и MAX_INT). Для него определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/», эти же операции в сокращенной форме, операцию унарный минус «-», операцию возведения в степень «^» и операции постфиксного и префиксного сложения/вычитания: «++» и «--», все логические операции сравнения. Добавить метод извлечения квадратного корня. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

9. Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для того чтобы избежать потерь точности числитель (m) и знаменатель (n) следует сделать длинными числами (объектами соответствующего класса с переопределенными операциями). Для класса «Fraction» определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления «/», эти же операции в сокращенной форме, операцию унарный минус «-», операцию возведения в степень «^» целый и дробный, и операции постфиксного и префиксного сложения/вычитания: «++» и «--», все логические операции сравнения. При

возможности, обязательно сокращать дроби. Для нахождения наибольшего общего делителя использовать бинарный алгоритм. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

10.Спроектировать класс «Point», который содержит точку на плоскости и методы доступа к ней. Спроектировать класс «Points», который содержит: множество точек на плоскости – *A* (объектов класса Point), количество точек – *n*. Для него определить: операцию добавления новой точки «<<» и операцию удаления точки «>>». Для него определить: операцию объединения «+», ее же в сокращенной форме, которая находит объединение точек без дубликатов; операцию пересечения «*», ее же в сокращенной форме, которая находит пересечение точек без дубликатов; операцию разности «-», ее же в сокращенной форме, которая находит разность точек без дубликатов. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

11.Спроектировать класс «Bignum_arithmetic», который содержит: длинное число, т.е. число, значения которого превышает максимально (минимально) допустимые значения целочисленного типа (MIN_INT и MAX_INT). Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для того чтобы избежать потерь точности числитель (m) и знаменатель (n) следует сделать длинными числами (объектами класса Bignum_arithmetic). Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, а сама матрица состоит из объектов класса «Fraction». Для класса «Matrix» определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления на число «/» (причем это может быть как объект класса «Fraction» или «Bignum_arithmetic» так и простое число), эти же операции в сокращенной

форме, операцию унарный минус «-», операцию нахождения транспонированной матрицы «~». Элементы матрицы при возможности стоит сокращать. При сокращении, для нахождения наибольшего общего делителя использовать бинарный алгоритм. Для классов «Bignum_arithmetic» и «Fraction» достаточно определить только необходимые операции. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию класса «Matrix».

12. Спроектировать класс «Bignum_arithmetic», который содержит: длинное число, т.е. число, значения которого превышает максимально (минимально) допустимые значения целочисленного типа (MIN_INT и MAX_INT). Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для того чтобы избежать потерь точности числитель (m) и знаменатель (n) следует сделать длинными числами (объектами класса Bignum_arithmetic). Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, а сама матрица состоит из объектов класса «Fraction». Для класса «Matrix» определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления на число «/» (причем это может быть как объект класса «Fraction» или «Bignum_arithmetic» так и простое число), эти же операции в сокращенной форме, операцию унарный минус «-», операцию нахождения верхней треугольной матрицы «~». Добавить метод нахождения определителя. Элементы матрицы при возможности стоит сокращать. При сокращении, для нахождения наибольшего общего делителя использовать бинарный алгоритм. Для классов «Bignum_arithmetic» и «Fraction» достаточно определить только необходимые операции. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию класса «Matrix».

13.Спроектировать класс «Bignum_arithmetic», который содержит: длинное число, т.е. число, значения которого превышает максимально (минимально) допустимые значения целочисленного типа (MIN_INT и MAX_INT). Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для того чтобы избежать потерь точности числитель (m) и знаменатель (n) следует сделать длинными числами (объектами класса Bignum_arithmetic). Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, а сама матрица состоит из объектов класса «Fraction». Для класса «Matrix» определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления на число «/» (причем это может быть как объект класса «Fraction» или «Bignum_arithmetic» так и простое число), эти же операции в сокращенной форме, операцию унарный минус «-», операцию нахождения обратной матрицы «!». Элементы матрицы при возможности стоит сокращать. При сокращении, для нахождения наибольшего общего делителя использовать бинарный алгоритм. Для классов «Bignum_arithmetic» и «Fraction» достаточно определить только необходимые операции. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию класса «Matrix».

14.Спроектировать класс «Algebraic_expressions», который содержит строку, представляющую собой алгебраическое выражение. Для него определить: операцию унарный минус «-», которая меняет знак выражения, операцию «+», которая складывает два выражения и операцию «-», которая вычитает из первого выражения второе. Реализовать в классе метод нахождения значения алгебраического выражения при заданной переменной. Для упрощения выполнения использовать список (стек). При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

15.Спроектировать класс «Vector», который содержит координаты вектора в пространстве. Для него определить: операцию сложения «+», операцию вычитания «-», операцию векторного произведения «*», операцию умножения на константу «*», эти же операции в сокращенной форме, операцию унарный минус «-», логическую операцию «==», проверяющую векторы на коллинеарность. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию. Рассмотреть случай когда вектор задан на плоскости.

16.Спроектировать класс «Algebraic_expressions», который содержит строку, представляющую собой алгебраическое выражение. Для него определить: операцию унарный минус «-», которая меняет знак выражения; операцию «+», которая складывает два выражения; операцию «-», которая вычитает из первого выражения второе; операцию «~», которая упрощает выражение при возможности. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

17.Спроектировать класс «Algebraic_expressions», который содержит строку, представляющую собой алгебраическое выражение. Для него определить: операцию унарный минус «-», которая меняет знак выражения; операцию «+», которая складывает два выражения; операцию «-», которая вычитает из первого выражения второе; операцию «!», которая находит производную выражения. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

18.Спроектировать класс «Algebraic_expressions», который содержит строку, представляющую собой алгебраическое выражение. Для него определить: операцию унарный минус «-», которая меняет знак выражения;

операцию «+», которая складывает два выражения; операцию «-», которая вычитает из первого выражения второе; операцию «!», которая находит первообразную выражения. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

19.Спроектировать класс «Associative_array», который содержит массив пар ключ-значение строкового типа. Для элементов массива при необходимости можно разработать собственный класс. Для него определить: операцию добавления элемента «<<», операцию удаления элемента «>>», операцию доступа по ключу «[]» с возможностью присвоения, операцию объединения «+» двух массивов, ее же в сокращенной форме. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

20.Спроектировать класс «Multi_set», который содержит множество упорядоченных символьных элементов. Для него определить: операцию объединения «+», ее же в сокращенной форме, операцию пересечения «*», ее же в сокращенной форме, операцию разности «-», ее же в сокращенной форме. Определить операции поэлементного добавления и удаления: «<<» и «>>» соответственно. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию. Множество вводится пользователем в виде строки с элементами разделенными запятыми, и преобразовывается в корректное множество. Множество не должно содержать строки.

21.Спроектировать класс «Set», который содержит множество целочисленных элементов, обязательно уникальных и упорядоченных. Спроектировать класс «Multi_set», который содержит множество целочисленных упорядоченных элементов. Определить операции

поэлементного добавления и удаления: «<<» и «>>» соответственно, для каждого класса. Для данных классов определить: операцию объединения «+», ее же в сокращенной форме, операцию пересечения «*», ее же в сокращенной форме, операцию разности «-», ее же в сокращенной форме. Для класса «Multi_set» определить оператор преобразования к классу «Set». При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

22.Спроектировать класс «Tree», который содержит: бинарное дерево целочисленных элементов. Для него определить: операцию добавления элемента «<<», операцию удаления элемента «>>», операцию превращения дерева в бинарную кучу (пирамиду) «~» и операцию превращения дерева в бинарное дерево поиска «!». При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию.

23.Спроектировать класс «Bignum_arithmetic», который содержит: длинное число, т.е. число, значения которого превышает максимально (минимально) допустимые значения целочисленного типа (MIN_INT и MAX_INT). Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для того чтобы избежать потерь точности числитель (m) и знаменатель (n) следует сделать длинными числами (объектами класса Bignum_arithmetic). Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, а сама матрица состоит из объектов класса «Fraction». Для класса «Matrix» определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления на число «/» (причем это может быть как объект класса «Fraction» или «Bignum_arithmetic» так и простое число), эти же операции в сокращенной форме, операцию унарный минус «-», операцию нахождения ранга матрицы

«!». Элементы матрицы при возможности стоит сокращать. При сокращении, для нахождения наибольшего общего делителя использовать бинарный алгоритм. Для классов «Bignum_arithmetic» и «Fraction» достаточно определить только необходимые операции. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию класса «Matrix».

24.Спроектировать класс «Bignum_arithmetic», который содержит: длинное число, т.е. число, значения которого превышает максимально (минимально) допустимые значения целочисленного типа (MIN_INT и MAX_INT). Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для того чтобы избежать потерь точности числитель (m) и знаменатель (n) следует сделать длинными числами (объектами класса Bignum_arithmetic). Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, а сама матрица состоит из объектов класса «Fraction». Для класса «Matrix» определить: операцию сложения «+», операцию вычитания «-», операцию умножения «*», операцию деления на число «/» (причем это может быть как объект класса «Fraction» или «Bignum_arithmetic» так и простое число), эти же операции в сокращенной форме, операцию унарный минус «-», операцию нахождения нормы Фробениуса «~». Элементы матрицы при возможности стоит сокращать. При сокращении, для нахождения наибольшего общего делителя использовать бинарный алгоритм. Для классов «Bignum_arithmetic» и «Fraction» достаточно определить только необходимые операции. При необходимости разрешается определять другие операции (например «=») и методы (например, getter, setter и прочее). Продемонстрировать каждую операцию класса «Matrix».

4 ТРЕБОВАНИЯ К ОТЧЕТУ

Отчет подается после полной сдачи и защиты лабораторной работы в электронном виде (документ Word).

Отчет должен быть оформлен согласно ДСТУ 3008-95.

В отчет должен содержать следующие пункты:

– титульный лист;

– содержание;

– цель работы;

– постановка задачи;

– аналитические выкладки;

– пошаговый алгоритм решения расчётной части задачи и подзадач (при необходимости);

– UML-диаграмму классов;

– исходный код программы с комментариями;

– примеры работы программы;

– выводы, с обоснованием результата.

Отчет оценивается максимально в 0,5 балла.

5 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое исключительная ситуация?
2. Для чего используется блок catch(…)?
3. Что происходит с необработанным исключением?
4. Как обработать исключение, генерируемое конструктором?