

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

ОПЕРАТОРНЫЕ И ДРУЖЕСТВЕННЫЕ ФУНКЦИИ C++
МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторной работе № 4

по дисциплине «ООП»

Киев 2016

СОДЕРЖАНИЕ

| | | |
|------|-------------------------------|----|
| 1 | Цель лабораторной работы..... | 3 |
| 2 | Теоретические положения | 4 |
| 2.1. | Операторные функции..... | 4 |
| 2.2. | Дружественные функции | 7 |
| 3 | Задания..... | 10 |
| 4 | Требования к отчету | 31 |
| 5 | Контрольные вопросы..... | 32 |

1 ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Цель работы – изучить особенности операторных и дружественных функций. Освоить принципы написания: функций преобразования типов объектов, перегрузки операций и дружественных функций.

2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1. Операторные функции

Операторные функции применяются для задания новых правил выполнения стандартных операций для объектов. Различают операторные функции перегрузки операций и преобразования типов:

а) **преобразование типов объектов:**

- скалярных – для преобразования используется конструктор с параметрами (данными скалярного типа);
- преобразование объекта одного класса в объект другого класса; в классе 2 используются конструктор, параметрами которого являются ссылки на объект класса 1.

Формат определения и вызова функции преобразования:

```
operator имя_сводимого_типа {return ...;}  
имя_сводимого_класса (имя_приводимого_объекта)
```

Например:

```
class real{  
    float x;  
public:  
    real (float x1):x(x1){};  
    void write() {cout<<x<<endl;}  
};  
class value{  
public:  
    int n,m;  
    value(int n1, int m1):n(n1),m(m1){};  
    operator float () {return (float) n/m;}  
    operator int () {return (int) n/m;}  
    operator real(){return real(float(*this));}  
};  
  
void main(){  
    value v(65,6);  
    cout<<v.n<<"/"<<v.m<<endl;  
    cout<<int(v)<<endl;  
    cout<<float(v)<<endl;  
    real(v).write();  
}
```

Часто возникает необходимость вызова классовой функции из другой функции этого же класса. Функции определяют действия над одним и тем же объектом, в этом случае вложенный вызов функции понимается как передача сообщений объекта самому себе. Указатель, определяющий объект, для которого вызвана вложенная функция-элемент, имеет имя `this`. Любая функция-элемент всегда в качестве первого параметра имеет неявный параметр, указатель на объект, для которого определена функция: `const класс *this`. Обращение к элементам-данным или конкретному объекту через указатель `this` имеет следующий формат:

```
this->имя_элемента
```

Функции преобразования не имеют аргументов, не указывается тип возвращаемого значения, хотя `return` обязателен, не может быть виртуальной, не наследуется.

б) перегрузка операций.

Все операции в C++ могут быть перегружены, кроме операций точка (`.`), разыменование (`*`), разрешение области действия (`:`), условная (`?:`) и `sizeof`. Операции `=`, `[]`, `()` и `->` могут быть перегружены только как нестатические функции-элементы, не могут быть перегружены для перечислимых типов.

Все остальные операции можно перегружать, чтобы применять их к каким-то новым типам объектов, вводимым пользователем. Операции перегружаются путём составления описания функции, как это делается для любых функций, за исключением того, что в этом случае имя функции состоит из ключевого слова `operator`, после которого записывается перегружаемая операция, т.е.

```
имя_класса operator операция (параметры) {тело}
class MyClass{
    int *T;
    int n;
public:
    MyClass(int size=8){
        n=size;
        T=new int[n];
    }
    MyClass operator=(int B*){return T};
};
```

```
MyClass a(5),b;
b=a;
```

Чтобы использовать операцию над объектами классов, эта операция должна быть перегружена, однако есть два исключения:

- операция = может быть использована любым классом без явной перегрузки. По умолчанию операция присваивания сводится к побитовому копированию элементов-данных класса. Такое копирование опасно для классов с элементами, которые указывают на динамически выделенные области памяти; для таких классов следует явно перегружать операцию присваивания.

- операция адресации & также может быть использована с объектами любых классов без перегрузки, она просто возвращает адрес объекта в памяти.

Перегрузка не может изменять старшинство и ассоциативность операций, нельзя изменить число операндов операции. Перегруженные функции не наследуются. Бинарные операции перегружаются функцией с одним параметром, указателем на другой аргумент является this. Унарные операции перегружаются функцией без параметров. Рассмотрим пример создания класса с перегруженными операциями:

```
#include "stdafx.h"
#include <cstdlib>
#include <iostream>
#include <cstring>

using namespace std;

class stroka{
    int len; //длина строки
    char * str; //символы
public:
    stroka(){ str = new char('\0'); }; //конструктор по умолчанию
    stroka(char *); //конструктор
    //инициализатор
    stroka(const stroka &cstr); //конструктор копии
    stroka operator =(const stroka &s); //перегруженный оператор присвоения
    stroka operator +(const stroka &s); //перегруженный оператор +
    stroka operator !(); //перегруженный оператор !
    int Get_len() { return len; }
    void show(){ cout << str << endl; }
    ~stroka(){ if (str) delete[]str; } //деструктор
};

stroka::stroka(char *tstr){
    len = strlen(tstr)+1;
    str = new char[len];
    strcpy_s(str, len, tstr);
}

stroka::stroka(const stroka &cstr){
    len = cstr.len;
    str = new char[len];
    strcpy_s(str, len, cstr.str);
}

stroka stroka::operator=(const stroka &s){
```

```

if (strlen(s.str) > strlen(str))      //если строка s больше той что объявили
{
    delete[]str;
    len = strlen(s.str) + 1;
    str = new char[len];
}
strcpy_s(str, len, s.str);
return *this;      //
}
stroka stroka::operator+(const stroka &s){
    char * tmpstr = new char[len + s.len - 1];
    int tmpflen = strlen(tmpstr);
    strcpy_s(tmpstr, tmpflen, str);
    strcat_s(tmpstr, tmpflen, s.str);
    stroka tmp(tmpstr);
    return tmp;
}
stroka stroka::operator!(){
    stroka tmp(str);
    strcpy_s(tmp.str, len, _strrev(str));
    return tmp;
}

int _tmain(int argc, _TCHAR* argv[])
{
    stroka a = "str 1";
    a.show();
    stroka a1;
    a1 = !a;
    a1.show();
    stroka b("str 2");
    b.show();
    stroka c;
    c = a + b;
    c.show();
}

```

2.2. Дружественные функции

Обычное объявление функции-элемента гарантирует три логически разные вещи:

- функция имеет право доступа к закрытой части объявления класса;
- функция находится в области видимости класса;
- функция должна вызываться для объекта класса, т.е. имеется указатель `this`.

В некоторых случаях желательно обеспечить доступ к закрытым элементам (наделить свойством 1) функции, не являющиеся элементами данного класса. Это можно сделать, объявив соответствующую функцию как друга класса с помощью спецификатора `friend`. Использование дружественных функций необходимо если:

- одна и та же функция является элементом двух разных классов;

- функция объявляется как дружественная для двух классов, но ни в одном не определяется;
- необходимо неявное преобразование типов.

Пример: имеем оператор, который умножает матрицу на вектор.

```
class Matrix;
class Vector{
    float v[4];
    friend Vector operator *(const Matrix&, const Vector&);
};
class Matrix{
    Vector v[4];
    friend Vector operator *(const Matrix&, const Vector&);
};
Vector operator *(const Matrix &m, const Vector &v){
    Vector r;
    for(int i=0;i<4;i++){
        r.v[i]=0;
        for (int j=0;j<4;j++) r.v[i]+=m.v[i].v[j]*v.v[j];
    }
    return r;
}
```

Правила использования дружественных функций:

- тип возвращаемого значения – имя класса, если функция является функцией преобразования типов, для расчётных функций тип любой;
- должна быть объявлена дружественной в том классе, для которого должна быть дружественной;
- для дружественной функции несущественны спецификации доступа;
- дружественные функции не взаимны;
- не наследуются.

Иногда можно видеть, что переопределенная операция также описывается, как дружественная, но, обычно, дружественные функции используются редко. Их присутствие в программе обычно означает то, что иерархия классов нуждается в видоизменении.

Рассмотрим пример. Возьмем наш класс `sber_bank`, с приватным элементом `big_bucks` и добавим в него "дружественную" функцию вычисления налога - `irs`:

```
class sber_bank {
private:                                // Начало раздела private
    double big_bucks;                  // элемент private
public:                                // Начало раздела public
    void deposit(double bucks);        // Элемент public
```



```
double withdraw(double bucks); // Элемент public
friend void irs(void);         // Дружественная функция irs
};
```

Дружественную функцию `irs` определим следующим образом:

```
void irs(void)
{
    big_bucks -= big_bucks * 0.10; // Взять 10% от итога
}
```

Отметим, что хотя мы объявили `irs()` внутри класса, но она не является функцией элементом! Это достигается благодаря ключевому слову `friend`. Но даже хотя этот не функция элемент, `irs()` может выполнить указанную операцию с нашими данными, имеющими тип `private`.

Если бы функция `big_bucks` была элементом другого класса (например, класса `free_shop`), то в описании `friend` нужно использовать операцию разрешения области действия:

```
friend void free_shop::irs(void);
```

Дружественным для описанного класса можно также сделать целый класс, для чего в описании используется ключевое слово `class`:

```
friend class check_bucks;
```

После этого любая функция элемент класса `check_bucks` может получить доступ к приватным элементам класса `sber_bank`. Заметим, что в C++, как и в жизни, дружественность не транзитивна: если А является другом для Б, а Б является другом для И, то отсюда не следует, что А является другом для И.

Дружественные описания следует использовать только в том случае, когда это действительно необходимо, например, когда без этого пришлось бы использовать чересчур сложную иерархию классов. По своей природе дружественные описания уменьшают инкапсуляцию и модульность. В частности, если нужно определить целый класс, как дружественный для другого класса, то вместо этого лучше рассмотреть возможность построения обоюдно порождаемого класса, который можно использовать для доступа к нужным элементам.

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 (1,5 балла)

1. Спроектировать класс «String», который содержит строку заданной длины. Для него определить: операцию присвоения «=», операцию конкатенации строк «+», сокращённую операцию конкатенации строк «+=» и операцию сравнения «==». Продемонстрировать каждую операцию.

2. Спроектировать класс «Vector», который содержит массив заданной размерности. Для него определить: операцию присвоения «=», операцию поэлементного сложения «+», операцию поэлементного вычитания «-» и операцию унарный минус «-», меняющий знак элементов массива. Продемонстрировать каждую операцию.

3. Спроектировать класс «Vector», который содержит массив заданной размерности. Для него определить: операцию присвоения «=», операцию префиксного и постфиксного сложения «++», которая увеличивает все элементы на единицу, операцию префиксного и постфиксного вычитания «--», которая уменьшает все элементы на единицу. Продемонстрировать каждую операцию.

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

подстроки «-», сокращенную операцию удаления подстроки «-=» и операцию не равно «!=». Продемонстрировать каждую операцию.

5. Спроектировать класс «String», который содержит строку заданной длины. Для него определить: операцию присвоения «=», операцию обращения строки «!», и операцию сравнения «==». Продемонстрировать каждую операцию, на задаче проверки строк на палиндромы.

6. Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для него определить: операцию присвоения «=», операторы преобразования к типу **int** и **double**. Продемонстрировать каждую операцию.

7. 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 «-», эти же операции в сокращенной форме. Продемонстрировать каждую операцию.

8. Спроектировать класс «Vector», который содержит массив заданной размерности. Для него определить: операцию присвоения «=», поэлементного сложения «+» и вычитания «-», их же в сокращенной форме. Продемонстрировать каждую операцию.

9. Спроектировать класс «String», который содержит строку заданной длины. Для него определить: операцию присвоения «=», операцию добавления подстроки в конец «>>» и в начало «<<», их же в сокращенной форме «>>=» и «<<=». Продемонстрировать каждую операцию.

10. Спроектировать класс «Vector», который содержит массив заданной размерности. Для него определить: операцию присвоения «=», скалярного произведения «*» и поэлементного деления вектора на число «/», их же в сокращенной форме. Продемонстрировать каждую операцию.

11.Спроектировать класс «Binary», который содержит число в двоичной системе счисления. Для него определить: операцию присвоения «=», операцию сложения «+» ее же в сокращенной форме и операцию вычитания «-» ее же в сокращенной форме. Продемонстрировать каждую операцию.

12.Спроектировать класс «Vector», который содержит массив заданной размерности. Для него определить: операцию присвоения «=», матричного (тензорного) произведения «*» и поэлементного деления вектора на число «/», их же в сокращенной форме. Продемонстрировать каждую операцию.

13.Спроектировать класс «Fraction», который содержит: дробь в формате «±m/n», правильную или неправильную. Для него определить: операцию присвоения «=», операции «==» и «!=». Продемонстрировать каждую операцию.

14.Спроектировать класс «Binary», который содержит число в двоичной системе счисления. Для него определить: операцию присвоения «=», операцию произведения «*» ее же в сокращенной форме и операцию деления «/» ее же в сокращенной форме. Продемонстрировать каждую операцию.

15.Спроектировать класс «Octal», который содержит число в восьмеричной системе счисления. Для него определить: операцию присвоения «=», операцию сложения «+» ее же в сокращенной форме и операцию вычитания «-» ее же в сокращенной форме. Продемонстрировать каждую операцию.

16.Спроектировать класс «Hexadecimal», который содержит число в шестнадцатеричной системе счисления. Для него определить: операцию присвоения «=», операцию сложения «+» ее же в сокращенной форме и операцию вычитания «-» ее же в сокращенной форме. Продемонстрировать каждую операцию.

17.Спроектировать класс «Fraction», который содержит: дробь в формате « $\pm m/n$ », правильную или неправильную. Для него определить: операцию присвоения «=», операции «>» и «<». Продемонстрировать каждую операцию.

18.Спроектировать класс «Octal», который содержит число в восьмеричной системе счисления. Для него определить: операцию присвоения «=», операцию умножения «*» ее же в сокращенной форме и операцию деления «/» ее же в сокращенной форме. Продемонстрировать каждую операцию.

19.Спроектировать класс «Hexadecimal», который содержит число в шестнадцатеричной системе счисления. Для него определить: операцию присвоения «=», операцию умножения «*» ее же в сокращенной форме и операцию деления «/» ее же в сокращенной форме. Продемонстрировать каждую операцию.

20.Спроектировать класс «Fraction», который содержит: дробь в формате « $\pm m/n$ », правильную или неправильную. Для него определить: операцию присвоения «=», операции «>=» и «<=». Продемонстрировать каждую операцию.

21.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 «~». Продемонстрировать каждую операцию.

22.Спроектировать класс «Fraction», который содержит: дробь в формате « $\pm m/n$ », правильную или неправильную. Для него определить: операцию присвоения «=», операцию возведения в степень целый и дробный «^». Продемонстрировать каждую операцию.

23.Спроектировать класс «Binary», который содержит число в двоичной системе счисления. Для него определить: операцию присвоения «=», оператор приведения к целому типу «int» и к действительному типу «double». Продемонстрировать каждую операцию.

24.Спроектировать класс «Octal», который содержит число в восьмеричной системе счисления. Для него определить: операцию присвоения «=», оператор приведения к целому типу «int» и к действительному типу «double». Продемонстрировать каждую операцию.

25.Спроектировать класс «Hexadecimal», который содержит число в шестнадцатеричной системе счисления. Для него определить: операцию присвоения «=», оператор приведения к целому типу «int» и к действительному типу «double». Продемонстрировать каждую операцию.

26.Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операцию умножения матриц «*», операцию деления матрицы на число «/», эти же операции в сокращенной форме. Продемонстрировать каждую операцию.

27.Спроектировать класс «Matrix», который содержит: высоту и ширину матрицы, двойной указатель на тип **double** (сама матрица). Для него определить: операцию нахождения максимального элемента «++», операцию нахождения минимального элемента «--». Продемонстрировать каждую операцию.

28.Спроектировать класс «String», который содержит строку заданной длины. Для него определить: операцию присвоения «=», операцию нахождения наибольшего по длине слова «++», операцию нахождения наименьшего по длине слова «--». Продемонстрировать каждую операцию.

29.Продемонстрировать работу перегруженных операторов «->» и «,».

30.Спроектировать класс «Vector», который содержит массив заданной размерности. Для него определить: операцию присвоения «=», операцию нахождения максимального элемента «++», операцию нахождения минимального элемента «--». Продемонстрировать каждую операцию.

Уровень Б (+1 балл)

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). Продemonстрировать каждую операцию.

5. Спроектировать класс «Set», который содержит множество целочисленных элементов, обязательно уникальных и упорядоченных. Для него определить: операцию объединения «+», операцию пересечения «*», операцию разности «/». Продemonстрировать 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). Продemonстрировать 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». Продemonстрировать 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. Продemonстрировать 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 « $\pm m/n$ », 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 « $\pm m/n$ », 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 «=»). Продemonстрировать 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). Продemonстрировать решение СЛАУ матричным методом.

15.Спроектировать класс «`Fraction`», который содержит: дробь в формате « $\pm m/n$ », 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 «=»). Продemonстрировать 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 «=»). Продemonстрировать 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 В (+3 балла)

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). Продemonстрировать 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). Продemonстрировать 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). Продemonстрировать 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 « $\pm m/n$ », 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 « $\pm m/n$ », 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 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 « $\pm m/n$ », 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.

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 операторные функции?
2. Какие бывают типы операторных функций?
3. Что такое дружественные функции?
4. Правила использования дружественных функций?
5. Какие операции не могут быть перегружены?