

Методические указания для СРС по выполнению лабораторных работ по дисциплине Объектно-ориентированное программирование

1 Содержание и оформление отчета по лабораторной работе

Отчет по лабораторной работе должен содержать

- титульный лист (пример приведен в приложении А)
- постановку задачи (текст задания согласно варианту)
- описание интерфейса класса (class {} и комментарии ко всем полям, методам и функциям)
- описание тестов для проверки классов (main с комментариями, какие действия выполнялись, и полученные результаты)
- листинг реализации класса (реализация методов и функций, отступы, без комментариев)

Отчеты сдаются в электронном виде в формате doc, docx или odt.

Пример отчета приведен в приложении Б.

2 Лабораторная работа №1

Тема: Определение класса с динамически размещаемыми данными

Задание:

I Запрограммировать класс согласно варианту

II Запрограммировать main с тестами (создание объекта и выполнение действий с ним)

III Написать отчет

Варианты заданий для лабораторной работы №1:

1. Ассоциативная таблица для представления списков вида имя=значение и словарей fish рыба

```
class ATable
```

```
{
```

```
public:
```

```
    ATable(int maxsize=100);
```

```
    ~ATable();
```

```
    void add(char *name, char *value);
```

```
    void remove(char *name);
```

```
    void setValue(char *name, char *value); // заменить значение
```

```
    char *getValue(char *name);           // если не нашли вернуть NULL
```

```
    void print(); // распечатать состояние объекта
```

```
};
```

2. Множество целых от а до b

Использовать массив флагов, флаг *i* указывает наличие числа *i+a*

```
class ASet
```

```
{
```

```
public:
```

```
    ASet(int a, int b);
```

```
    ASet(int b);    // а по умолчанию 0
```

```
    ~ASet();
```

```
    void incl(int); // включить число в множество
```

```
                // если такое число уже есть, то ничего не делать
```

```
    void excl(int); // исключить
```

```
    bool contain(int); // содержится в множестве?
```

```
    void print(); // распечатать состояние объекта
```

```
};
```

3. Сортированный набор строк

```
class SList
```

```
{
```

```
public:
```

```
    SList(int maxsize=100);
```

```
    ~SList();
```

```
    void add(char *str); // добавить строку
```

```
    int count();        // количество строк
```

```
    int find(char *str); // найти строку, вернуть -1 если нет
```

```
    void remove(int); // удалить строку
```

```
    char *item(int); // строка по номеру
```

```
    void print(); // распечатать состояние объекта
```

```
};
```

4. Список

```
typedef int Data;
```

```
class AList
```

```
{
```

```
public:
```

```
    AList();
```

```
    ~AList();
```

```
    void first(); // перейти на первый элемент списка
```

```
    void next(); // перейти на следующий элемент
```

```
    void insert(Data value); // вставка перед текущим
```

```
    void remove(); // удалить текущий, текущим становится следующий
```

```

Data get(); // получить данные
bool eol(); // список кончился, стоим на NULL
void print(); // распечатать состояние объекта
};

```

5. Очередь событий

```

typedef int Event;
class TQueue
{
public:
    TQueue(int maxsize=100);
    ~TQueue();
    void add(int dt, Event obj); // добавить событие в момент (сейчас+dt)
    Event get(int &dt);         // получить событие (dt- превышение интервала)
    bool isReady();             // подошло время обработать к-л события
    bool isEmpty();             // очередь пуста
    bool isFull();              // очередь полна
    void print(); // распечатать состояние объекта
};

```

Для получения текущего времени использовать time(0)

6. Игра 15

```

class DGame
{
public:
    DGame(int n=4, int m=4); // n,m - размеры доски, при DGame(4,4) - игра 15
    ~DGame();
    void up(); // сдвигаем верхнюю фишку вниз
    void down();
    void left();
    void right();
    bool ready(); // решено!
    void random(int n=100); // n случайных ходов
    void print(); // распечатать состояние объекта
};

```

7. Карточная колода

```

typedef int Card;
class CStock
{
public:

```

```

CStock(int maxsize, bool init=false);
    // если init==true, заполнить числами от 1 до maxsize
    // если init==false, то количество карт в колоде=0
~CStock();
void add(Card card); // добавить карту
void remove(int i); // удалить i карту
Card & get(int i); // i-я карта
int count();
void mix(); // перемешать карты
void print(); // распечатать состояние объекта
};

```

8. Окно (использовать conio.h)

```

class AWindow
{
public:
    AWindow(int left, int top, int right, int bottom);
    ~AWindow();
    void select(); // Сделать окно верхним и текущим
    void gotoxy(int x, int y);
    void putch(char);
    void puts(char *);
    void textattr(int);
    void clear();
};

```

9. Класс "Экзаменационная ведомость (по ООП)"

```

class XList
{
public:
    XList(int maxsize=100);
    ~XList();
    void add(char *name); // Добавить студента
    void remove(char *name);
    void setMark(char *name, int mark); // записать отметку
    int getMark(char *name); // если нет оценки или студента
                                // вернуть 0
    void print(); // распечатать состояние объекта
};

```

10. Класс матрица

```

class Matrix

```

```

{
public:
    Matrix(int n, int m);
    ~Matrix();
    double &get(int i, int j);
    void transp(); // транспонировать
    void add(double x); // прибавить ко всем элементам матрицы x
    void multiply(double x); // умножить все на x
    void print(); // распечатать состояние объекта
};

```

11. Множество вещественных чисел

```

class ASet
{
public:
    ASet(int size); // size - максимальное количество чисел в множестве
    ~ASet();
    void incl(double); // включить число в множество,
                        // если такое число уже есть, то ничего не делать
    void excl(double); // исключить
    bool contain(double); // содержится в множестве?
    void print(); // распечатать состояние объекта
};

```

12. Очередь с приоритетами

```

struct Data
{
    long priorit;
    char info[50];
};

class PQueue
{
public:
    PQueue(int maxsize=100);
    ~PQueue();
    void add(const Data &dat); // добавить данные
    const Data &first(); // получить данные с наименьшим приоритетом
    void next(); // убрать данные с наименьшим приоритетом
    bool isEmpty(); // очередь пуста
    bool isFull(); // очередь полна
    void print(); // распечатать состояние объекта
};

```

```
};
```

13. Класс последовательность целых чисел

```
class Seq
{
public:
    Seq(); // пустая последовательность
    ~Seq();
    void add(int x); // добавить число
    void add(const Seq&); // добавить другую последовательность
    int count();
    int &get(int i); // получить элемент последовательности
    void print(); // распечатать состояние объекта
};
```

14. Класс "Общая коллекция CD"

```
class CDCollection
{
public:
    CDCollection(int nPerson, int nCD);
    ~CDCollection();
    void add(char *CDname, int idPerson); // добавить новый диск
    void tranfer(char *CDname, int idPerson); // дать диск idPerson
    int where(char *CDname); // у кого диск
    int amount(int idPerson); // сколько дисков у idPerson
    char *nameCD(int idPerson, int i); // название i-го диска у idPerson
    void print(); // распечатать состояние объекта
};
```

15. Класс упорядоченная последовательность целых чисел (числа могут повторяться).

```
class SortSeq
{
public:
    SortSeq(); // пустая последовательность
    ~SortSeq();
    void add(int x); // добавить число
    int count();
    int get(int i); // получить элемент последовательности
    void remove(int i); // удалить i-й элемент
    void print(); // распечатать состояние объекта
};
```

16. Класс "Очередь на получение дефицита"

```
class TQueue
{
public:
    TQueue(int maxsize=100);
    ~TQueue();
    void add(long pr, char *name); // добавить персону с указанным приоритетом
    char *first();                // имя первой персоны в очереди (без удаления)
    void next();                  // удалить первую персону
    bool isEmpty();               // очередь пуста
    bool isFull();                // очередь полна
    void print(); // распечатать состояние объекта
};
```

17. Класс "Дек (двухсторонняя очередь) для целых чисел"

```
class Deque
{
public:
    Deque(int maxsize=100);
    ~Deque();
    void addleft(int); // добавить слева
    void addright(int); // добавить справа
    int getleft(); // левый элемент
    int getright(); // правый элемент
    void delleft(); // удалить левый элемент
    void delright(); // удалить правый элемент
    bool isEmpty(); // очередь пуста
    bool isFull(); // очередь полна
    void print(); // распечатать состояние объекта
};
```

18. Класс упорядоченная последовательность вещественных чисел (числа могут повторяться).

```
class DSortSeq
{
public:
    DSortSeq(); // пустая последовательность
    ~DSortSeq();
    void add(double x); // добавить число
    int count();
    double get(int i); // получить элемент последовательности
};
```

```

    void remove(int i); // удалить i-й элемент
    void print(); // распечатать состояние объекта
};

```

19. Класс последовательность вещественных чисел

```

class DSeq
{
public:
    DSeq(); // пустая последовательность
    ~DSeq();
    void add(double x); // добавить число
    void add(const DSeq&); // добавить другую последовательность
    int count();
    double &get(int i); // получить элемент последовательности
    void print(); // распечатать состояние объекта
};

```

20. Класс "Дек (двухсторонняя очередь) для вещественных чисел"

```

class DDeque
{
public:
    DDeque(int maxsize=100);
    ~DD deque();
    void addleft(double); // добавить слева
    void addright(double); // добавить справа
    double getleft(); // левый
    double getright(); // правый
    void delleft(); // удалить левый элемент
    void delright(); // удалить правый элемент
    bool isEmpty(); // очередь пуста
    bool isFull(); // очередь полна
    void print(); // распечатать состояние объекта
};

```

21. Класс "Игра НИМ"

Представьте, что перед вами и вашим партнером по игре на столе лежит несколько кучек камешков. Правила игры разрешают забрать за один ход любое количество камешков - но только из одной кучки. Выигрывает тот из двух игроков, кто забирает со стола последний камешек.

```

class NIM
{
public:

```



```

NIM(int nheap=10);
~NIM();
void setheap(int i, int nmatch); // задать количество спичек в i кучке
void take(int i, int nmatch); // взять из i кучки nmatch спичек
int amount(int i); // количество спичек в кучке i
int count(); // количество кучек
bool iswin(); // Все кучки пусты
void print(); // распечатать состояние объекта
};

```

22. Игра 15 для букв

```

class DGame
{
public:
    DGame(int n=4, int m=4); // n,m - размеры доски, при DGame(4,4) - игра
    ~DGame();
    void settex(char *text); // задаем начальный текст
        // для 4x4 "СЛОНСПИТСТОЯАВЫ "
    void up(); // сдвигаем верхнюю фишку вниз
    void down();
    void left();
    void right();
    bool ready(); // решено!
    void random(int n=100); // n случайных ходов
    void print(); // распечатать состояние объекта
};

```

23. Окно (использовать conio.h)

```

class AWindow
{
public:
    AWindow(int left, int top, int right, int bottom);
    ~AWindow();
    void select(); // Сделать окно верхним и текущим
    void gotoxy(int x, int y);
    void putch(char);
    void puts(char *);
    void textattr(int);
    void clear();
};

```

24. Ассоциативная таблица

для представления списков имя=значение

и словарей fish рыба

class ATable

```
{
public:
    ATable(int maxsize=100);
    ~ATable();
    void add(char *name, char *value);
    void remove(char *name);
    void setValue(char *name, char *value); // заменить значение
    char *getValue(char *name);           // если не нашли вернуть NULL
    void print(); // распечатать состояние объекта
};
```

25. Сортированный набор строк

class SList

```
{
public:
    SList(int maxsize=100);
    ~SList();
    void add(char *str); // добавить строку
    int count();         // количество строк
    int find(char *str); // найти строку, вернуть -1 если нет
    void remove(int);    // удалить строку
    char *item(int);     // строка по номеру
    void print(); // распечатать состояние объекта
};
```

26. Класс матрица

class Matrix

```
{
public:
    Matrix(int n, int m);
    ~Matrix();
    double &get(int i, int j);
    void assign(const Matrix &); // скопировать данные из др. матрицы
    void add(double x); // прибавить ко всем элементам матрицы x
    void multiply(double x); // умножить все на x
    void print(); // распечатать состояние объекта
};
```

3 Лабораторная работа №2

Тема: Перегрузка операций

Задание:

I Запрограммировать класс согласно варианту

Операции (если есть в задании) =, [], +=, -=, *=, /=, ++, -- определить как методы.

Для ввода переопределить >> , для вывода - << .

Формат ввода-вывода объектов делать так, как указано в задании.

Запись [текст] означает, что текст может отсутствовать, например, конструктор ИмяКласса(a[,b[,c]]) может быть вызван с 1, 2 или 3 аргументами.

Пример:

```
Vector a(0,0),b(1.5,0.3);
```

```
cout<<"Введите вектор a:"<<endl;
```

```
cin>>a;
```

```
//нужно ввести (1.2,3.2) вместе со скобками и запятой
```

```
cout<<a<<"+"<<b<<"="<<(a+b)<<endl;
```

```
// печатается (1.2,3.2)+(1.5,0.3)=(2.7,3.5)
```

II Запрограммировать main с тестами (создание объекта и выполнение действий с ним)

III Написать отчет

Варианты заданий для лабораторной работы №2:

1. Рациональные числа (дроби) Rational

это числа вида a/b , где a и b не имеют общих делителей,
 $b > 0$, $|a|, |b| < 2^{31}$

Конструктор: Rational(a,b)

Операции:

$x+y$, $x-y$, $x*y$, x/y ,

$x+=y$, $x-=y$, $x*=y$, $x/=y$,

$x>y$, $x<y$, $x>=y$, $x<=y$, $x==y$, $x!=y$

!x (x равно нулю)

где x, y - рациональные числа или long

(определить преобразование long->Rational)

вывод, ввод в виде 3/5

Методы:

long getp(); (числитель)

long getq(); (знаменатель)

2. Вектор на плоскости (x,y) Vector

Конструктор: `Vector(x,y)`

Операции:

`a+b`, `a-b`,

`a+=b`, `a-=b`,

`a*b` (скалярное произведение),

`a==b`, `a!=b`

`!a` (`a` является нулевым вектором),

`a/z`, `z*a` ("масштабирование" вектора)

где `a,b` - вектора, `z` - `double`

Вывод, ввод в виде (1.4,-2.5)

Методы:

`double getX();`

`double getY();`

3. Комплексное число `a+bi` `Complex`

Конструктор: `Complex(a,b)`

Операции:

`x+y`, `x-y`, `x*y`, `x/y`,

`x+=y`, `x-=y`, `x*=y`, `x/=y`,

`x==y`, `x!=y`,

`!x` (`x` равно 0)

где `x,y` - комплексные числа или `double`

(определить преобразование `double->Complex`)

Вывод, ввод в виде 1.5+2.2i

Методы:

`double getReal();` (реальная часть)

`double getImaginary();` (мнимая часть)

4. Вектор в полярных координатах (`r,phi` в градусах) `Polar`

Конструктор: `Polar(r,phi)`

Операции:

`x+y`, `x-y`,

`x+=d`, `x-=d` (поворот вектора на `d` градусов),

`x*=d`, `x/=d` (увеличение/уменьшение вектора в `d` раз),

`d*x` (увеличенный в `d` раз вектор `x`, `x` не изменять!),

`x==y`, `x!=y`,

`!x` (`x` является нулевым вектором)

где `x,y` - вектора в полярных координатах, `d` - `double`

Вывод, ввод в виде (5.1,30.5)

Методы:

`double getLength();`

`double getAngle();`

5. Прямоугольник [l,t,r,b] Rect координаты целые

Конструктор: Rect(l,t,r,b)

Операции:

$x * y$ (пересечение, friend),
 $x + y$ ("объединение", охватывающий прямоугольник минимального размера, friend),
 $x += y$ (расширить x , заменив охватывающим)
 $x *= y$ (заменить пересечением)
 $x == y, x != y$,
 $x \leq y$ (x полностью лежит в y), $x \geq y$ (наоборот, y в x),
 $!x$ (пустой прямоугольник)
где x, y - прямоугольники
вывод, ввод в виде [0,5,40,20]

Методы:

int getLeft();
int getTop();
int getRight();
int getBottom();

6. Множество целых чисел от 0 до 31 Set32

При определении хранить в виде целого числа и
использовать битовые операции над long
 $a|b$ -объединение $a \& b$ -пересечение и т.п.

Конструктор: Set32() (первоначально пустое)

Операции:

$x * y$ (пересечение), $x += y$,
 $x + y$ (объединение), $x *= y$,
 $x += z$ (включение числа z в множество),
 $x - y$ (разность множеств), $x -= y$,
 $x -= z$ (исключение числа z из множества),
 $x == y, x != y$,
 $x \leq y, x \geq y$ (подмножество),
 $x \geq z, z \leq x$ (проверка вхождения числа z в множество x)
 $!x$ (x - пустое множество)
где x, y - множества, z - целое число от 0 до 31
вывод, ввод в виде {1,2,5}

7. Отрезок [a,b] (на координатной прямой) Segment

Конструктор: Segment(a,b)

Операции:

$x * y$ (пересечение),

$x + y$ ("объединение", охватывающий отрезок минимального размера),

$x == y$, $x != y$,

$x \leq y$ (x полностью лежит в y), $x \geq y$ (наоборот, y в x),

$!x$ (x - пустой отрезок)

$x += d$ (сдвинуть правый край отрезка на d),

$x -= d$ (сдвинуть левый край отрезка на d),

$x \geq d$, $d \leq x$ (проверка попадания числа d в отрезок x)

где x,y - отрезки, d - double

вывод, ввод в виде [0,5.5]

Методы:

double getLeft();

double getRight();

8. Угол в градусах,минутах,секундах [0,360) Angle

Конструктор: Angle(g[,m[,s]])

Операции:

$x + y$, $x - y$, (увеличить/уменьшить угол на соответствующий угол)

$x += y$, $x -= y$,

$z * x$, x / z , (увеличить/уменьшить угол в z раз)

$x * = z$, $x /= z$,

$x == y$, $x != y$,

$!x$ (x равен нулю)

где x,y - углы, z - целое число ≥ 1

вывод, ввод в виде 5^16'25"

Методы:

int getGrad(); [0..359]

int getMin(); [0..59]

int getSec(); [0..59]

9. Вектор в пространстве (x,y,z) Vector

Конструктор: Vector(x,y,z)

Операции:

$a + b$, $a - b$,

$a += b$, $a -= b$,

$a * b$ (скалярное произведение),

$a == b$, $a != b$

!a (a является нулевым вектором),
a/z, z*a ("масштабирование" вектора)
где a,b - вектора, z - double
вывод, ввод в виде (1.4,-2.5,3.1)

Методы:

double getx();
double gety();
double getz();

10. Дата год,месяц,день от 01.01.0001 до 31.12.2999 Date

Високосными являются года кратные 4 и 400, но не кратные 100
Рекомендуется хранить дату в виде количества дней от 01.01.0001

Конструктор: Date(y,m,d)

Операции:

x+z, z+x (добавить к дате число дней z),
x+=z,
x-z (убавить дату на число дней z),
x-=z,
x-y (количество дней между датами),
x==y, x!=y, x<=y, x<y, x>=y, x>y,
++x, --x, x++, x--,

где x,y - даты, z - целое число

вывод, ввод в виде 31/12/2004

Методы:

int getYear(); [1..2999]
int getMonth(); [1..12]
int getDay(); [1..31]

11. Длина в ярдах, футах, дюймах, точках Length

1 ярд=3 фута=36 дюймов

1 дюйм=72 точки

Конструктор: Length(y[,f[,i[,p]]])

Операции:

x+y, x-y, (увеличить/уменьшить длину на соответствующую длину)
x+=y, x-=y,
z*x, x/z, (увеличить/уменьшить длину в z раз)
x*=z, x/=z,
x==y, x!=y, x<y, x>y, x<=y, x>=y,
!x (x равна нулю)

где x, y - длины, z - целое число ≥ 1
вывод, ввод в виде 5.2.4.30

Методы:

`int getYard();` [0..oo]
`int getFoot();` [0..2]
`int getInch();` [0..11]
`int getPoint();` [0..71]

12. Рациональные числа (дроби) Rational

это числа вида a/b , где a и b не имеют общих делителей,
 $b > 0$, $|a|, |b| < 2^{31}$

Конструктор: `Rational(a,b)`

Операции:

$x+y$, $x-y$, $x*y$, x/y ,
 $x+=y$, $x-=y$, $x*=y$, $x/=y$,
 $x>y$, $x<y$, $x\geq y$, $x\leq y$, $x==y$, $x!=y$
!x (x равно нулю)

где x, y - рациональные числа или `long`
(определить преобразование `long->Rational`)

вывод, ввод в виде 3/5

Методы:

`long getp();` (числитель)
`long getq();` (знаменатель)

13. Комплексное число $a+bi$ Complex

Конструктор: `Complex(a,b)`

Операции:

$x+y$, $x-y$, $x*y$, x/y ,
 $x+=y$, $x-=y$, $x*=y$, $x/=y$,
 $x==y$, $x!=y$,
!x (x равно 0)

где x, y - комплексные числа или `double`
(определить преобразование `double->Complex`)

вывод, ввод в виде (1.5,2.2)

Методы:

`double getReal();` (реальная часть)
`double getImaginary();` (мнимая часть)

14. Длина в милях, ярдах, футах, дюймах Length

1 миля=1760 ярдов

1 ярд=3 фута=36 дюймов

Конструктор: Length(m[,y[,f[,i]]])

Операции:

$x+y$, $x-y$, (увеличить/уменьшить длину на соответствующую длину)

$x+=y$, $x-=y$,

$z*x$, x/z , (увеличить/уменьшить длину в z раз)

$x*=z$, $x/=z$,

$x==y$, $x!=y$, $x<y$, $x>y$, $x<=y$, $x>=y$,

!x (x равна нулю)

где x, y - длины, z - целое число ≥ 1

вывод, ввод в виде 2m 5y 2f 4i

Методы:

int getMile(); [0.. ∞];

int getYard(); [0..1759]

int getFoot(); [0..2]

int getInch(); [0..11]

15. Вектор в полярных координатах (r,phi в градусах) Polar

Конструктор: Polar(r,phi)

Операции:

$x+y$, $x-y$,

$x+=d$, $x-=d$ (поворот вектора на d градусов),

$x*=d$, $x/=d$ (увеличение/уменьшение вектора в d раз),

$d*x$ (увеличенный в d раз вектор x , x не изменять!),

$x==y$, $x!=y$,

!x (x является нулевым вектором)

где x, y - вектора в полярных координатах, d - double

вывод, ввод в виде 5.1@30.5

Методы:

double getLength();

double getAngle();

16. Множество целых чисел от 0 до 31 Set32

При определении хранить в виде целого числа и использовать битовые операции над long

$a|b$ -объединение $a\&b$ -пересечение и т.п.

Конструктор: Set32() (первоначально пустое)

Операции:

$x * y$ (пересечение), $x += y$,

$x + y$ (объединение), $x * = y$,

$x += z$ (включение числа z в множество),

$x - y$ (разность множеств), $x -= y$,

$x -= z$ (исключение числа z из множества),

$x == y$, $x != y$,

$x <= y$, $x >= y$ (подмножество),

$x >= z$, $z <= x$ (проверка вхождения числа z в множество x)

$!x$ (x - пустое множество)

где x, y - множества, z - целое число от 0 до 31

вывод, ввод в виде (1 2 5)

17. Время часы, минуты, секунды Time

Конструктор: Time(h[,m[,s]])

Операции:

$x + y$, $x - y$, (увеличить/уменьшить время на соответствующий время)

$x += y$, $x -= y$,

$z * x$, x / z , (увеличить/уменьшить время в z раз)

$x * = z$, $x /= z$,

$x == y$, $x != y$, $x < y$, $x > y$, $x <= y$, $x >= y$,

$!x$ (x равно нулю)

где x, y - время, z - целое число ≥ 1

вывод, ввод в виде 5:16:25

Методы:

int getHour(); [0.. ∞];

int getMin(); [0..59]

int getSec(); [0..59]

18. Отрезок [a,b] (на координатной прямой) Segment

Конструктор: Segment(a,b)

Операции:

$x * y$ (пересечение),

$x + y$ ("объединение", охватывающий отрезок минимального размера),

$x += y$, $x * = y$,

$x == y$, $x != y$,

$x <= y$ (x полностью лежит в y), $x >= y$ (наоборот, y в x),

!x (x - пустой отрезок)
x>=d, d<=x (проверка попадания числа d в отрезок x)
где x,y - отрезки, d - double
Вывод, ввод в виде [0,5.5]

Методы:

double getLeft();
double getRight();

19. Строка String

Конструкторы: String(char * =""), копий

Операции:

x+y, x+=y,
x=y,
x[i],
x>y, x<y, x>=y, x<=y, x==y, x!=y,

!x (x - пустая строка)

где x,y - строка или char *
Вывод, ввод в виде "te\nt"

Методы:

int length();
const char *c_str(); (строка)

20. Вектор из n значений Vector

Конструктор: Vector(n), копий

Операции:

x+y, x+=y,
x=y,
x[i],
x==y, x!=y,
где x,y - векторы
Вывод, ввод в виде {1.0,2.0,3.5}

Методы:

int size();

21. Прямоугольник [l,t,r,b] Rect в целых координаты

Конструктор: Rect(l,t,r,b)

Операции:

$x * y$ (пересечение, friend),
 $x + y$ ("объединение", охватывающий прямоугольник минимального размера, friend),
 $x += y$ (расширить x , заменив охватывающим)
 $x *= y$ (заменить пересечением)
 $x == y$, $x != y$,
 $x \leq y$ (x полностью лежит в y), $x \geq y$ (наоборот, y в x),
 $!x$ (пустой прямоугольник)
где x, y - прямоугольники
вывод, ввод в виде [0 5 40 20]

Методы:

```
int getLeft();  
int getTop();  
int getRight();  
int getBottom();
```

22. Вектор в пространстве (x,y,z) Vector

Конструктор: Vector(x,y,z)

Операции:

$a + b$, $a - b$,
 $a += b$, $a -= b$,
 $a * b$ (скалярное произведение),
 $a == b$, $a != b$
 $!a$ (a является нулевым вектором),
 a / z , $z * a$ ("масштабирование" вектора)
где a, b - вектора, z - double
вывод, ввод в виде (1.4 -2.5 3.1)

Методы:

```
double getx();  
double gety();  
double getz();
```

23. Время часы, минуты, секунды Time

Конструктор: Time(h[,m[,s]])

Операции:

$x + y$, $x - y$, (увеличить/уменьшить время на соответствующий время)
 $x += y$, $x -= y$,
 $++x$, $--x$, $x++$, $x--$ (изменить время на секунду)

$z+x$, $x+z$, $x-z$, (изменить время на z секунд)
 $x+=z$, $x-=z$,
 $x==y$, $x!=y$, $x<y$, $x>y$, $x<=y$, $x>=y$,
 $!x$ (x равно нулю)
где x, y - время, z - целое число ≥ 1
вывод, ввод в виде 5:16:25

Методы:

`int getHour(); [0..23]`
`int getMin(); [0..59]`
`int getSec(); [0..59]`

24. Вектор на плоскости (x,y) Vector

Конструктор: `Vector(x,y)`

Операции:

$a+b$, $a-b$,
 $a+=b$, $a-=b$,
 $a*b$ (скалярное произведение),
 $a==b$, $a!=b$
 $!a$ (a является нулевым вектором),
 a/z , $z*a$ ("масштабирование" вектора)
где a, b - вектора, z - double
вывод, ввод в виде (1.4 -2.5)

Методы:

`double getx();`
`double gety();`

25. Вектор из n значений Vector

Конструктор: `Vector(n)`, копий

Операции:

$x+y$, $x+=y$,
 $x=y$,
 $x[i]$,
 $x==y$, $x!=y$,
где x, y - векторы
вывод, ввод в виде (1.0 2.0 3.5)

Методы:

`int size();`

26. Строка String

Конструктор: String(char * =""), копий

Операции:

x+y, x+=y,

x=y,

x[i],

x>y, x<y, x>=y, x<=y, x==y, x!=y,

!x (x - пустая строка)

где x,y - строка или char *

вывод, ввод в виде 'te'xt' //апострофы удваиваются

Методы:

int length();

const char *c_str(); (строка)

4 Лабораторная работа №3

Тема: Виртуальные методы, наследование

Задание:

I Базовый класс для всех вариантов:

class Figure

{

int c; // цвет

bool visible;

protected:

int x,y; // базовая точка

virtual void draw();

public:

Figure(int c, int x, int y);

~Figure();

void move(int dx, int dy); // сместить фигуру на (dx,dy)

// видимая фигура гасится, затем рисуется в другом месте

// у невидимой просто меняются поля x,y

void setcolor(int c); // установить цвет фигуры

// видимая фигура рисуется новым цветом

// у невидимой просто меняется поле c

int getcolor(); // получить цвет

void hide(); // спрятать: нарисовать черный прямоугольник

// по размерам area()

void show(); // показать

bool isvisible(); // видима?

virtual void area(int &x1,int &y1,int &x2,int &y2);

// получить размеры прямоугольной области, содержащей фигуру

};

Определить реализацию методов класса Figure.

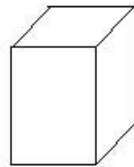
Методы area и draw нужно определить как чисто виртуальные.

7. Правильный треугольник

Triangle(цвет линий, x и y центра, длина стороны, угол поворота)

8. 3D кирпич (в проекции)

Brick(цвет линий, x и y левого нижнего угла, длина, высота, толщина)



9. Квадрат

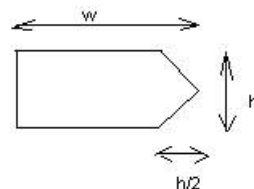
Square(цвет линий, x и y центра, длина стороны, угол поворота)

10. Шестиугольник

Hexagon(цвет линий, x и y центра, длина стороны)

11. Стрелка

Arrow(цвет линий, x и y левого нижнего угла, длина, высота)

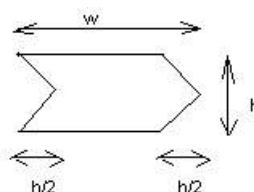


12. Пятиугольная звезда

Pentagram(цвет линий, x и y центра, радиус, угол поворота)

13. Стрелка

Arrow(цвет линий, x и y левого нижнего угла, длина, высота)

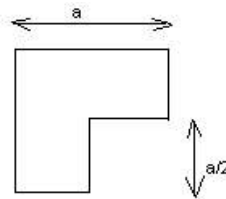


14.Бублик

Boublik(цвет линий, x и y центра, радиус1, радиус2)
радиус1<радиус2

15.3/4 Квадрата

Square34(цвет линий, x и y центра, длина стороны, N вырезанного сектора
1..4)

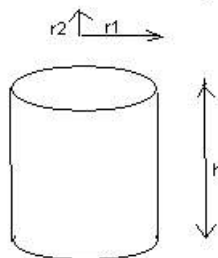


16. Шестиугольная звезда

Star(цвет линий, x и y центра, радиус, угол поворота)

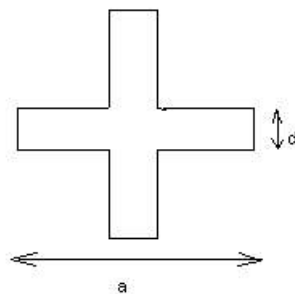
17.Цилиндр (в проекции)

Cylinder(цвет линий, x и y центра, радиус1, радиус2, высота)



18.Крест

Cross(цвет линий, x и y центра, длина, толщина полосок)



19.Пятиугольник

Pentagon(цвет линий, x и y центра, длина стороны)

20.Параллелограмм

Parallelogram(x и y левого нижнего угла, длина, высота, угол наклона)

21.Полукруг

HalfCircle(цвет линий, x и y левого нижнего угла, радиус, угол поворота)

22.Радуга (полубублик)

Rainbow(цвет линий, x и y центра, радиус1, радиус2)

23.Равнобедренная трапеция

Trapezium(цвет линий, x и y левого нижнего угла, основание1, основание2, высота)

24.Капля (состоит из полукруга и равнобедренного треугольника)

Drop(цвет линий, x и y центра, высота треугольника, радиус, угол поворота)

25.Восьмиугольник

Octagon(цвет линий, x и y центра, длина стороны)

26.Прямоугольник с одним загнутым углом (лист бумаги)

Rectangle(цвет линий, x и y левого нижнего угла, длина, высота, величина сгиба)

4 Лабораторная работа №4

Тема: Шаблоны, исключительные ситуации

Задание:

- I Определить класс-шаблон с использованием динамического распределения памяти согласно варианту и необходимые конструкторы и операции, включая конструктор копий, операция присваивания и если указано операцию индексации. При выходе за границу, переполнении и т.п. вызвать исключительную ситуацию (определить собственные классы) для информирования программы, вызвавшей метод.

При определении друзей класса-шаблона использовать следующий пример:

```
// предварительные объявления для дружественной функции-шаблона
template<class T> class task; // есть класс-шаблон
template<class T> task<T>* preempt(task<T>*);
    // есть функция-шаблон, у которой аргумент - класс-шаблон

template<class T> class task {
    // ...
    friend void next_time(); // друг - обычная функция
    friend void process(task<T>*); // друг - экземпляр перегруженной функции
        // (не шаблон)
            // друг - конкретная инстанция шаблона функции
            // с соответствующими аргументами;
    friend task<T>* preempt<T>(task<T>*);
        // друг - функция-шаблон с таким же аргументом, как у
класса
        // этот друг требует предварительных объявлений см. выше

template<class C> friend int func(C); // все инстанции шаблонной функции
    // с любыми аргументами
    friend class task<int>; // друг - конкретная инстанция класса-шаблона
template<class P> friend class frd; // друг - все инстанции класса-шаблона
// ...
};
```

II Запрограммировать main с тестами

(создание объектов и выполнение действий с ним,
в т.ч. действие, приводящее к возникновению исключительной ситуации,
которую необходимо перехватить)

III Написать отчет

Варианты заданий для лабораторной работы №4:

1. класс вектор (одномерный массив элементов заданного типа и размера,
указанного в аргументах конструктора)
получение i-го элемента с помощью операции индексации
14. класс вектор (одномерный массив элементов заданного типа и размера,
указанного в аргументах конструктора)
получение i-го элемента с помощью операции индексации

2. класс массив (одномерный массив элементов заданного типа),
в аргументах конструктора задаются номер первого и последнего
элемента в массиве как в языке Pascal.

получение i -го элемента с помощью операции индексации

15. класс массив (одномерный массив элементов заданного типа),
в аргументах конструктора задаются номер первого и последнего
элемента в массиве как в языке Pascal.

получение i -го элемента с помощью операции индексации

3. класс матрица (двумерный массив элементов заданного типа,
размеры задаются в аргументах конструктора)

получение (i,j) -го элемента с помощью операции $()$

получение строки как матрицы

16. класс матрица (двумерный массив элементов заданного типа,
размеры задаются в аргументах конструктора)

получение (i,j) -го элемента с помощью операции $()$

получение столбца как матрицы

4. класс очередь элементов заданного типа, размером не более
указанного в параметрах конструктора,

добавление $<<$ и извлечение $>>$ элемента

4. класс очередь элементов заданного типа, размером не более
указанного в параметрах конструктора,

добавление $<<$ и извлечение $>>$ элемента

5. класс последовательность элементов заданного типа,
вставка в любое место

последовательности, получение i -го элемента последовательности с
помощью операции индексации,

удаление подпоследовательности.

9. класс последовательность элементов заданного типа,

удаление произвольного элемента по номеру, вставка в любое место

последовательности, получение i -го элемента последовательности с помощью операции индексации.

18. класс последовательность элементов заданного типа,
вставка в любое место
последовательности, получение i -го элемента последовательности с помощью операции индексации,
сцепление последовательностей.

22. класс последовательность элементов заданного типа,
вставка в любое место
последовательности, получение i -го элемента последовательности с помощью операции индексации,
получение подпоследовательности.

6. класс упорядоченный набор элементов заданного типа (для которого заданы операции сравнения)
удаление произвольного элемента по номеру, добавление \ll нового элемента,
получение i -го элемента последовательности с помощью операции индексации.

10. класс упорядоченный набор элементов заданного типа (для которого заданы операции сравнения)
добавление \ll нового элемента,
получение i -го элемента последовательности с помощью операции индексации,
удаление подпоследовательности.

19. класс упорядоченный набор элементов заданного типа (для которого заданы операции сравнения)
добавление \ll нового элемента,
получение i -го элемента последовательности с помощью операции индексации,
слияние с помощью операции $+$ двух последовательностей.

23. класс упорядоченный набор элементов заданного типа (для которого заданы операции сравнения)
добавление \ll нового элемента,

получение i -го элемента последовательности с помощью операции индексации,
получение подпоследовательности.

7. класс множество элементов заданного типа (для которого заданы операции сравнения на равенство), размером не более указанного в параметрах конструктора. Каждое значение в множестве встречается не более одного раза. удаление произвольного значения $-$, добавление нового значения $+$, проверка принадлежности элемента множеству, одно множество является подмножеством другого (\leq).

11. класс множество элементов заданного типа (для которого заданы операции сравнения на равенство), размером не более указанного в параметрах конструктора. Каждое значение в множестве встречается не более одного раза. удаление произвольного значения $-$, добавление нового значения $+$, проверка принадлежности элемента множеству, разность $(-)$ множеств.

20. класс множество элементов заданного типа (для которого заданы операции сравнения на равенство), размером не более указанного в параметрах конструктора. Каждое значение в множестве встречается не более одного раза. удаление произвольного значения $-$, добавление нового значения $+$, проверка принадлежности элемента множеству, объединение $(+)$ множеств.

24. класс множество элементов заданного типа (для которого заданы операции сравнения на равенство), размером не более указанного в параметрах конструктора. Каждое значение в множестве встречается не более одного раза. удаление произвольного значения $-$, добавление нового значения $+$, проверка принадлежности элемента множеству, пересечение $(*)$ множеств.

8. класс ассоциативный массив, в котором можно осуществлять поиск значений заданного типа по ключу другого заданного типа, для которого определена проверка на равенство. операция доступа по ключу $val=obj[key]$,

добавление, изменение значения, соответствующее ключу.

12. класс ассоциативный массив, в котором можно осуществлять поиск значений заданного типа по ключу другого заданного типа, для которого определена проверка на равенство.
операция доступа по ключу `val=obj[key]`,
добавление, удаление значения, соответствующее ключу.

13. класс стек элементов заданного типа, размером не более указанного в параметрах конструктора,
добавление `<<` и извлечение `>>` элемента

26. класс стек элементов заданного типа, размером не более указанного в параметрах конструктора,
добавление `<<` и извлечение `>>` элемента

21. класс дек для элементов заданного типа, размером не более указанного в параметрах конструктора,
добавление и извлечение элементов

25. класс дек для элементов заданного типа, размером не более указанного в параметрах конструктора,
добавление и извлечение элементов

ТИТУЛЬНЫЙ ЛИСТ

ФГАОУ ВО «Южно-Уральский государственный университет» (НИУ)

Факультет «Математики, механики и компьютерных технологий»

Кафедра «Прикладная математика и программирование»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № _____

по дисциплине «Объектно-ориентированное программирование»

Автор работы

студент группы ЕТ-XXX

_____ И.О. Фамилия

_____ 20__ г.

Работа зачтена с оценкой

_____ И.О. Фамилия

_____ 20__ г.

Челябинск, 20__

1 Постановка задачи

Реализовать класс

3. Сортированный набор строк

```
class SList
{
public:
    SList(int maxsize=100);
    ~SList();
    void add(char *str);    // добавить строку
    int size();            // количество строк
    int find(char *str);    // найти строку, вернуть -1 если нет
    void remove(int);       // удалить строку
    char *item(int);        // строка по номеру
    void print();           // распечатать состояние объекта
};
```

Реализовать main с тестами для проверки класса (создание объекта и выполнение действий с ним)

2 Описание интерфейса класса

```
class SList
{
    //Максимальный размер
    int maxsize;
    //Текущий размер
    int cursize;
    //Указатель на область памяти
    char **m;
public:
    //Конструктор
    SList(int maxsize = 100);
    //Деструктор
    ~SList();
    //Добавление строки.
    void add(char *str);
    //Возврат текущего количества строк
    int size() {
        return cursize;
    };
    //Нахождение номера строки по ее названию
    //Если строки нет, возвращается -1
    int find(char *str);
    //Удаление строки по номеру
    void remove(int);
    //Возврат строки по номеру
```

```

char *item(int);
//Печать информации о состоянии объекта
void print();
};

```

3 Описание тестов для проверки классов

```

int main() {
    //Создаем объект
    SList obj(10);
    //Выводим на экран его начальное состояние
    cout << "Начальное состояние:\n";
    obj.print();
    //Добавим три строки не в порядке возрастания
    obj.add("Анна");
    obj.add("Петр");
    obj.add("Александр");
    //Выведем на экран количество элементов после добавления
    cout << "Размер после добавления строк:\n";
    cout << obj.size() << endl;
    //Выведем на экран состояние объекта после добавления
    cout << "Состояние после добавления строк:\n";
    obj.print();
    //Попробуем найти не существующую строку
    cout << "Поиск не существующей строки Олег:\n";
    int i = obj.find("Олег");
    //Выведем на экран, найдена строка, или нет
    if (i != -1)
        cout << "Олег найден\n";
    else
        cout << "Олег не найден\n";
    //Попробуем найти существующую строку
    cout << "Поиск существующей строки Анна:\n";
    i = obj.find("Анна");
    if (i != -1) {
        cout << "Анна найдена\n";
        cout << "Попробуем вывести найденную строку на экран:\n";
        cout << obj.item(i) << endl;
        //Удалим строку Анна
        obj.remove(i);
        //Выведем на экран данные после удаления строки Анна
        cout << "Данные после удаление строки Анна:\n";
        obj.print();
    }
    else

```

```

        cout << "Анна не найдена\n";
//Попробуем удалить строку по не существующему номеру
cout << "Попытка удалить строку по не существующему номеру:\n";
obj.remove(20);
return 0;
}

```

Полученные результаты:

Начальное состояние:

Максимальный размер массива: 10

Текущий размер массива: 0

Размер после добавления строк:

3

Состояние после добавления строк:

Максимальный размер массива: 10

Текущий размер массива: 3

Содержимое массива:

Александр

Анна

Петр

Поиск не существующей строки Олег:

Олег не найден

Поиск существующей строки Анна:

Анна найдена

Попробуем вывести найденную строку на экран:

Анна

Данные после удаление строки Анна:

Максимальный размер массива: 10

Текущий размер массива: 2

Содержимое массива:

Александр

Петр

Попытка удалить строку по не существующему номеру:

Невозможно удалить строку

Код завершения: 1

4 Листинг реализации класса

```

//-----
SList::SList(int maxsize) :
    maxsize(maxsize), cursize(0),
    m(new char*[maxsize]) {}

//-----
SList::~SList() {
    for (int i = 0; i < cursize; i++)

```

```

        delete[] m[i];
    delete[] m;
}

//-----
void SList::add(char *str) {
    if (cursize == maxsize) {
        cout << "Невозможно добавить строку\n";
        exit(1);
    }
    int i;
    for (i = cursize - 1; i >= 0 && strcmp(m[i], str) > 0; i--) {
        m[i + 1] = m[i];
    }
    char *s = new char[strlen(str)+1];
    strcpy(s, str);
    m[i + 1] = s;
    cursize++;
}

//-----
int SList::find(char *str) {
    if (cursize == 0)
        return -1;
    int l = 0, r = cursize - 1, mid;
    while (r - l > 1) {
        mid = (l + r) / 2;
        if (strcmp(m[mid], str) < 0)
            l = mid;
        else
            r = mid;
    }
    if (strcmp(m[r], str) == 0)
        return r;
    if (strcmp(m[l], str) == 0)
        return l;
    return -1;
}

//-----
void SList::remove(int n) {
    if (n >= cursize || n < 0) {
        cout << "Невозможно удалить строку\n";
        exit(1);
    }
}

```

```

        delete m[n];
        for (int i = n; i < cursize - 1; i++)
            m[i] = m[i + 1];
        cursize--;
    }

//-----
char *SList::item(int n) {
    if (n >= cursize || n < 0) {
        cout << "Неверный индекс\n";
        exit(1);
    }
    return m[n];
}

//-----
void SList::print() {
    cout << "Максимальный размер массива: " << maxsize << endl;
    cout << "Текущий размер массива: " << cursize << endl;
    if (cursize != 0) {
        cout << "Содержимое массива:\n";
        for (int i = 0; i < cursize; i++)
            cout << m[i] << endl;
    }
}
}

```