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

КЛАССЫ C++

МЕТОДИЧЕСКИЕ УКАЗАНИЯ

к лабораторной работе № 2

по дисциплине «ООП»

Киев 2015

СОДЕРЖАНИЕ

1 ЦЕЛЬ ЛАБОРАТОРНОЙ РАБОТЫ

Цель работы – изучить основные концепции объектно-ориентированного программирования. Изучить особенности использования классов и объектов, а также особенности применения конструкторов и деструкторов.

2 ТЕОРЕТИЧЕСКИЕ ПОЛОЖЕНИЯ

2.1. Классы

Целью введения концепции классов в C++ является предоставление программисту средств создания новых типов, которые настолько же удобны в использовании, как и встроенные. Тип является конкретным представлением некоторой концепции. Например, встроенный тип `float` вместе с операциями +, -, * и т.д. представляет собой конкретное воплощение математической концепции вещественного числа.

Определение 1. Класс - разновидность абстрактного типа данных в объектно-ориентированном программировании, характеризуемый способом своего построения. Наряду с понятием «**объекта**» класс является ключевым понятием в ООП.

Определение 2. Класс – это определяемый пользователем тип. Класс должен быть объявлен до того, как будет объявлена хотя бы одна переменная этого класса. Конкретный экземпляр класса называется **объектом**. Формат описания класса включает в себя:

– типы данных, являющихся элементами-данными класса;

– спецификациями доступа к данным;

– функции, определяющие методы обработки элементов-данных;

– конструкторы и деструктор.

В общем случае класс имеет следующий формат:

```
class имя_класса {
      спецификация_доступа:                    //public, protected, private
             тип переменная_1;                 //данные-элементы класса
             тип переменная_2;
              …
             тип имя_функции (параметры);      //функции-элементы
              …
      имя_класса(параметры);                    //конструктор
      ~имя_класса();                           //деструктор
};
```

Обычно описание класса производится в заголовочном файле (C++ Header file), имеющим расширение *.h.

2.2. Доступ к данным. Инкапсуляция

Рассмотрим реализацию концепции даты с использованием структуры Date. Эта структура представляет собой дату и содержит набор функций, осуществляющих манипуляции с переменными-датами:

```
struct Date {
   int d,m,y;
};
void init_date(Date &d, int, int, int);     //инициализация даты d
void add_year(Date &d, int n);             //прибавить n лет к d
void add_month(Date &d, int n);             //прибавить n месяцев к d
void add_day(Date &d, int n);              //прибавить n дней к d
```

Здесь нет явной связи между типом данных и функциями. Такую связь можно установить, объявив функции в качестве членов структуры:

```
struct Date {
   int d,m,y;
   void init_date(int dd, int mm, int yy);  //инициализация даты
   void add_year(int n);                    //прибавить n лет
   void add_month(int n);                           //прибавить n месяцев
   void add_day(int n);                     //прибавить n дней
};
```

Функции, объявленные внутри определения структуры (класса) называются функциями-элементами, и их можно вызвать только для переменной соответствующего типа, используя стандартный синтаксис доступа к членам структуры. Например:

```
Date my_birthady;
Date today;
togay.init(12,11,2012);
Date tomorrow = today;
tomorrow.add_day(1);
```

Такое объявление `Date` предоставляет набор функций для работы с `Date`. Однако оно не указывает, что только эти функции непосредственно зависят от представления `Date`, и только они могут непосредственно осуществлять доступ к объектам класса `Date`. Эти ограничения можно отразить, воспользовавшись ключевым словом `class` и спецификациями доступа:

```
class Date {
   int d,m,y;
  public:
   void init_date(int dd, int mm, int yy);   //инициализация даты
   void add_year(int n);                    //прибавить n лет
   void add_month(int n);                           //прибавить n месяцев
   void add_day(int n);                     //прибавить n дней
};
```

Метка public разделяет тело класса на две части. Имена в первой могут использоваться только функциями-элементами. Вторая же образует открытый интерфейс объектов класса. Ограничение доступа имеет несколько преимуществ: например, ошибка, в результате которой Date приняла неверное значение, может быть вызвана только кодом соответствующей функции-элемента. Из этого следует, что локализация ошибки может быть завершена ещё до запуска программы. Защита закрытых данных базируется на ограничении использования имён членов класса, для чего функции

элементы и структуры данных, определяющие некоторые свойства данного класса, рассматриваются в качестве единого целого. Такой подход называется инкапсуляцией.

В общем случае, в разных языках программирования термин «**инкапсуляция**» относится к одной из или обеим одновременно следующим нотациям:

– языковая конструкция, позволяющая связать данные с методами, предназначенными для обработки этих данных;

– механизм языка, позволяющий ограничить доступ одних компонентов программы к другим.

Инкапсуляция подразумевает "защиту" данных в пределах класса таким образом, что только элементы класса получают к ним доступ. То есть, для того чтобы получить значение одного из элементов данных класса, нужно вызвать функцию элемент этого класса, который возвращает необходимое значение. Для присвоения элементу значения, вызывается соответствующая функция элемент данного класса. В объектном программировании считается хорошим тоном закрывать все данные и функции элементы описываемого класса для доступа "извне".

C++ предоставляет программистам три уровня доступа к элементам объектов:

– public (общий/открытый),

– private (приватный/закрытый),

– protected (защищенный).

Элементы, объявленные общими, будут доступны любому внешнему элементу класса, любой функции элементу или выражению в программе, когда объект является видимым.

Приватные элементы доступны только другим элементам своего же класса. Они не доступны извне, за исключением специальных функций, называемых "дружественными".

К защищенным элементам имеют доступ лишь некоторые из объектов. Они доступны только элементам своего класса и любым его потомкам. Поэтому защищенные элементы занимают промежуточное положение между общими и приватными.

Примечание: приватные элементы недоступны потомкам своего класса. Поэтому и понадобились защищенные элементы.

Умелое использование уровней доступа повышает надежность программ и их способность к изменениям, ослабляя взаимозависимость между объектами. Правильно

описанными функциями элементами типа `public` можно изменять приватные элементы, не затрагивая программный код других объектов. По умолчанию в классах предполагается спецификация `private`.

2.3. Элементы-данные и статические элементы-данные

Элементы-данные могут быть любого типа, в том числе и классового; не могут быть представителями определяемого класса (но могут быть указателями или ссылками на представителя определяемого класса).

Если элемент данных объявлен как private, то доступ к нему осуществляется только через функции-элементы этого же класса. Если элемент данных объявлен как public, то доступ может осуществляться через объект, или указатель на объект:

```
имя_объекта.имя_элемента или имя_указателя→имя_элемента.
```

Например:

```
class point{
  int x=0,y=0;
  public:
    int getx(){return x;}
};
void main(){
 point k;
 cout<<k.getx();
}
```

```
class point{
  public:
    int x,y;
    int getx(){return x;}
};
void main(){
 point k;
 point *ptr_k = new point;
 cout<<k.x<<endl;
 cout<<ptr_k->x;
}
```

Обычно каждый объект класса имеет свою собственную копию всех данных-элементов класса. Но в определённых случаях во всех объектах класса должна фигурировать только одна копия некоторых данных:

– счётчик числа созданных объектов класса,

– если в классе имеются некоторые константы, одинаковые для всех объектов класса, то нерационально хранить в каждом объекте собственные копии этих констант.

Для введения в класс подобных данных используются статические данные, содержащие информацию «*для всего класса*». Таким образом, статическая переменная является частью класса, но не является частью объектов. Формат объявления статического элемента:

```
static тип_данных имя_переменной;
```

Использование статических элементов сокращает затраты памяти и гарантирует единство данных во всех объектах. Также как и обычные элементы, могут быть открытыми, закрытыми или защищёнными. Доступ к открытым статическим

переменным класса возможен посредством любого объекта класса или посредством имени класса с помощью бинарной операции разрешения области действия. Закрытые и защищённые статические элементы класса должны быть доступны открытым функциям-элементам этого класса и друзьям класса.

Статические элементы класса существуют даже тогда, когда не существует никаких объектов этого класса. Начальные значения статических элементов (открытых и закрытых) должны задаваться вне объявления класса. Задать начальное значение статического элемента можно только один раз в файле.

```
int  Myclass::D=10;
```

2.4. Функции-элементы (методы)

Функции-элементы класса имеют доступ к любым другим функциям-элементам и к любым элементам-данным. Разделяют встроенные (определённые внутри класса) и определённые вне класса функции-элементы. Формат определения функции вне класса:

```
тип_возвращаемого_значения имя_класса::имя_функции(параметры) {тело}
```

Например:

```
void Date::init(int dd, int mm, int yy){
  d=dd;
  m=mm;
  y=yy;
}
```

В теле функции имена элементов-данных можно использовать без явного указания объекта.

Простые функции чаще определяют внутри класса; фактически они являются встроенными функциями `inline`. Рекомендуется реализацию функций-элементов размещать в отдельном файле реализации, в объявлении класса должны содержаться только прототипы функций. Такая организация программы обеспечивает независимость всех модулей, использующих заголовочный файл с объявлением класса, от каких-то изменений в реализации функций-элементов класса.

Наиболее часто используются функции, реализующие арифметические методы, методы отображения и сравнения данных. Особым видом функций-элементов являются конструкторы.

Исходя из принципа инкапсуляции, элементы-данные всегда должны быть защищены от несанкционированного доступа. Как правило, доступ к ним осуществляется только через функции, включающие методы чтения и записи полей (**т.н.**

getter и setter). В этих функциях записи должна осуществляться проверка данных, чтобы не записать случайно в элементы неверные данные и чтобы не допустить их неверной трактовки. Функции чтения позволяют не переписывать всю программу, если происходит изменение в типе, способе хранения и размещения элементов данных в классе.

Данные всегда целесообразно объявлять в разделе `private`, в редких случаях – в раздел `protected`, чтобы потомки данного класса имели к ним доступ. Рассмотрим пример:

```
class MyClass{
  public:
   void SetA(int); //функция записи
  int GetA();     //функция чтения
  private:
      int FA;
      double B,C;};
```

```
void MyClass::SetA(int Value){
   if(…)      //проверка корректности
     FA=Value;
}
int MyClass::GetA() {return FA;}
```

В данном примере функция чтения просто возвращает значение поля, однако в более сложных классах может потребоваться какая-то предварительная обработка данных.

2.5. Конструкторы

Использование функций типа init() для инициализации объектов класса неэлегантно и подвержено ошибкам. Так как нигде не сказано, что объект должен быть проинициализирован, программист может сделать это дважды или забыть об этом. Наилучшим подходом будет предоставление программисту возможности объявить функцию, имеющую явное назначение – инициализация объекта. Такая функция называется конструктором и распознаётся по имени, которое совпадает с именем самого класса.

Таким образом, конструктором класса называется открытая функция-элемент, которая вызывается в момент создания класса и должна инициализировать данные указанными в вызове значениями или значениями по умолчанию. Например:

```
class MyClass{
    int A;
  public:
    MyClass(void);
};
MyClass::MyClass(void) {A=0;}
```

Формат определения конструктора:

– внутри класса `имя_класса(параметры) {//тело конструктора};`
– за телом класса `имя_класса::имя_класса(параметры) {//тело конструктора}.`

Простое задание в конструкторе значений данных в общем случае не гарантирует их целостность, обычно нужна ещё и проверка допустимости данных. Лучше для инициализации данных использовать соответствующие функции записи.

Создание класса в программе может осуществляться объявлением соответствующей переменной или динамическим размещением переменной в памяти:

```
MyClass MC;
MyClass *PMC = new MyClass;
```

В момент выполнения каждого из этих операторов неявным образом выполняется вызов конструктора, устанавливающего начальные значения данных. Недостатком таких конструкторов является то, что все начальные значения данных задаются конструктором. Вызывающая функция никак не может задать другое значение.

Для устранения этого недостатка используются конструкторы, в которых все начальные значения задаются как параметры. В этом случае прототип конструктора имеет вид:

```
MyClass(int);
MyClass:: MyClass(int a){A=a;}
```

В этом случае создание объекта выполняется операторами:

```
MyClass MC(1);
MyClass *PMC = new MyClass(1);
```

Также при определении конструктора такого вида может использоваться список инициализации:

```
имя_класса(параметры):имя_элемента1(значение_элемента1),…{}.
```

Для рассмотренного выше конструктора список инициализации будет иметь следующий вид:

```
MyClass(int a):A(a);
```

Список инициализации обязателен, если элементами-данными являются константы или ссылки, либо объект, для которого определён конструктор с параметрами. Используя список инициализации можно изменять значения констант в момент создания объекта, например:

```
class MyClass{
    int A;
    const int MaxA;
   const int MinA;
  public:
    MyClass(int=0);
   void SetA(int);        //функция чтения
};
MyClass::MyClass(int a):MaxA(10),MinA(1){SetA(a);}
```


Использование конструктора, в котором программно задаются значения всех элементов класса может быть довольно громоздким. Поэтому реально используются конструкторы с параметрами по умолчанию, в этом случае объявление и реализация конструктора могут иметь такой вид:

```
MyClass(int =0);
MyClass::MyClass(int a) {SetA(a);}
```

Объект такого класса можно задавать любым из рассмотренных ранее операторов создания объекта. Если при создании указывается аргумент, то его значение присваивается полю. Если аргумент не указывается, то присваивается значение по умолчанию.

При помощи конструктора по умолчанию можно предоставить пользователю возможность изменять значения констант в момент создания объекта. В этом случае в конструкторе с умолчанием надо предусмотреть для констант соответствующие значения по умолчанию:

```
MyClass(int A=0, int MaxA=10, int MinA=1);
MyClass::MyClass(int a, int j, int i):MaxA(i),MinA(j){SetA(a)};
```

Также используют метод объявления нескольких конструкторов с разными наборами параметров:

```
class Date{
    int d,m,y;
    public:
       Date(int, int, int);       //день, месяц, год
       Date(int, int);          //день, месяц, текущий год
       Date(int);                     //день, текущий месяц и год
       Date();                  //текущая дата
       Date(const char*);             //дата в строковом представлении
};
```

Правила использования конструктора:

– конструктор имеет тоже имя, что и его класс;

– конструктор не содержит оператора return и не возвращает никакого значения (даже void);

– конструктор может определяться явно пользователем или неявно компилятором. Неявно происходит объявление конструктора в таких случаях: определение нового объекта класса, при копировании объекта, при динамическом создании объекта с помощью оператора new;

– определение конструктора может производиться внутри и вне класса;

– конструктор может иметь, а может не иметь параметры. Конструктор с параметрами инициализирует объект при объявлении. Конструктор без параметров инициализирует пустой объект;

– конструкторы не могут быть вложенными;

– конструктор не может быть виртуальным;

– ошибки, генерируемые конструктором, обрабатываются только механизмом обработки исключительных ситуаций;

– конструктор не наследуется.

Различают конструкторы инициализации, копии и преобразования.

1) конструктор-инициализатор – в теле конструктора элементам данных присваиваются значения, инициализация возможна через ввод, присваивание, список инициализации.
2) конструктор-копии – создаёт новый объект на основе существующего. В качестве параметра такой конструктор использует константную ссылку на существующий объект. Формат объявления:

```
имя_класса(имя_класса const & имя_объекта){}
```

Например:

```
class point{
      int x, y;
public:
      point();
      point(const point &);
      void setx(int px){ x = px; }
      int getx() { return x; }
      int gety() { return y; }
};

point::point(){  //конструктор-инициализатор
      cout << "Input ";
      cin >> x >> y;
}
point::point(const     point     &k):x(k.x),y(k.y){}
//конструктор-копии

void main(){
  const point a;
  point p = a;
  cout << endl << p.getx() << "  "<<p.gety();
  p.setx(25);
  point c(p);
  cout << endl << c.getx() << "  " << c.gety();
}
```


3) конструктор-преобразователь – любой конструктор с параметрами является преобразователем из одного типа в другой. Конструктор, принимающий скалярные параметры, выполняет преобразования к классовому типу; принимающий параметры классового типа – выполняет преобразования из одного класса в другой.

2.6. Деструкторы

Деструктор – это специальная функция-элемент, срабатывающая при уничтожении динамически размещённого объекта класса и освобождающая занимаемую им память. Имя деструктора совпадает с именем класса, но перед ним записывают символ ~:

```
~имя_класса(){}.
```

Деструкторы необходимы, если конструктор или какие-то функции-элементы класса динамически распределяют память, создавая в ней какие-то объекты. В остальных случаях можно обойтись без деструктора; если деструктор явным образом в классе не объявлен, то компилятор сам генерирует необходимые коды освобождения памяти. Пример:

```
class String{
  char *data;
  int size;
  public:
    string(int sz) {data=new char[size=sz];}
    ~string(){delete data;}
};
```

Ограничение на использование деструктора:

– деструктор не имеет параметров;

– не содержит return;

– не наследуется;

– не может быть объявлен как const, static, void.

3 ЗАДАНИЯ

Разработать пошаговый алгоритм решения задачи. 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. Определить класс «Точка» с закрытыми элементами X и Y (координаты точки). Определить методы класса, обеспечивающие доступ к этим переменным. Задать для класса три конструктора:

– конструктор-инициализатор с инициализацией через ввод данных с клавиатуры;

– конструктор-преобразователь;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Точка», и создающий зеркальную точку относительно оси абсцисс.

2. Определить класс «Точка» с закрытыми элементами X и Y (координаты точки). Определить методы класса, обеспечивающие доступ к этим переменным. Задать для класса такие конструкторы:

– конструктор-инициализатор с инициализацией данных при помощи передачи параметров (по умолчанию точку создавать в начале координат);

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Точка», и создающий зеркальную точку относительно оси ординат.

3. Определить класс «Точка» с закрытыми элементами X и Y (координаты точки). Определить методы класса, обеспечивающие доступ к этим переменным. Задать для класса такие конструкторы:

– конструктор-инициализатор с инициализацией данных при помощи передачи параметров (по умолчанию точку создавать в координатах (10,10));

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Точка», и создающий зеркальную точку относительно оси абсцисс и ординат.

4. Определить класс «Точки» с закрытыми элементами X1, Y1 и X2, Y2 (координаты точек). Определить методы класса, обеспечивающие доступ к этим переменным. Определить метод находящий длину отрезка, который образовывают эти точки. Задать для класса такие конструкторы:

– конструктор по умолчанию, создающий точки с координатами 0,0 и 5,5;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Точки», и создающий отрезок, удаленный от исходного (X1, Y1 и X2, Y2) на 3.

5. Определить класс «Точка» с закрытыми элементами X и Y (координаты точки). Определить методы класса, обеспечивающие доступ к этим переменным. Задать для класса такие конструкторы:

– конструктор по умолчанию, создающий точку с координатами 5,5;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Точка», и создающий точку, сдвинутую относительно исходной на 5 и 8 точек.

6. Определить класс «Круг» с закрытыми элементами X, Y (координаты центра) и R (радиус круга). Определить методы класса, обеспечивающие доступ к этим переменным. Создать метод, определяющий площадь круга. Задать для класса конструктор копии, принимающий в качестве параметра ссылку на

объект класса «Круг», и создающий круг с центром в той же точке, но имеющим радиус в два раза больше. По умолчанию создавать круг с единичным радиусом в начале координат.

7. Определить класс «Круг» с закрытыми элементами X, Y (координаты центра) и R (радиус круга). Определить методы класса, обеспечивающие доступ к этим переменным. Создать метод, определяющий площадь круга. Задать для класса конструктор копии, принимающий в качестве параметра ссылку на объект класса «Круг», и создающий круг с центром в точке, зеркальной относительно оси абсцисс, и таким же радиусом. По умолчанию создавать круг с единичным радиусом в точке (10,10).

8. Определить класс «Окружность» с закрытыми элементами X, Y (координаты центра) и R (радиус окружности). Определить методы класса, обеспечивающие доступ к этим переменным. Создать метод, определяющий длину окружности. Задать для класса конструктор копии, принимающий в качестве параметра ссылку на объект класса «Окружность», и создающий окружность с центром в той же точке, но имеющей радиус в два раза меньше. Задать конструктор-инициализатор с инициализацией данных через ввод.

9. Определить класс «Окружность» с закрытыми элементами X, Y (координаты центра) и R (радиус окружности). Определить методы класса, обеспечивающие доступ к этим переменным. Создать метод, определяющий длину окружности. Задать для класса конструктор копии, принимающий в качестве параметра ссылку на объект класса «Окружность», и создающий окружность с центром в точке, зеркальной относительно оси ординат, и таким же радиусом. Задать конструктор-инициализатор с инициализацией данных через ввод.

10.Определить класс «Прямоугольник» с закрытым элементами a и b – стороны прямоугольника. Определить методы класса, обеспечивающие доступ

к этим переменным. Создать метод, определяющий площадь прямоугольника. Задать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных при помощи передачи параметров (по умолчанию задавать прямоугольник со сторонами 2 и 4);

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Прямоугольник», и создающий прямоугольник со сторонами, равными меньшей стороне копируемого объекта.

11.Определить класс «Прямоугольник» с закрытым элементами a и b – стороны прямоугольника. Определить методы класса, обеспечивающие доступ к этим переменным. Создать метод, определяющий площадь прямоугольника. Задать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных через ввод;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Прямоугольник», и создающий прямоугольник со сторонами, равными большей стороне копируемого объекта.

12.Определить класс «Прямоугольник» с закрытым элементами a и b – стороны прямоугольника. Определить методы класса, обеспечивающие доступ к этим переменным. Создать метод, определяющий диагональ прямоугольника. Задать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных через ввод;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Прямоугольник», и создающий прямоугольник со сторонами, равными диагонали копируемого объекта.

13.Определить класс «Строка» с закрытыми элементами s (указатель на строку) и len(длина строки). Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор, принимающий в качестве параметра строку;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Строка», и создающий перевёрнутую строку.

Определить деструктор класса.

14.Определить класс «Строка» с закрытыми элементами s и q (указатели на строку) и len1 и len2 (длины строк). Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор, принимающий в качестве параметров строки;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Строка», и меняющий местами строки s и q.

Определить деструктор класса.

15.Определить класс «Строка» с закрытыми элементами s (указатель на строку) и len (длина строки). Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных через ввод;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Строка», и создающий строку, состоящую только из чётных символов исходной строки.

Определить деструктор класса.

16.Определить класс «Строка» с закрытыми элементами s (указатель на строку) и len(длина строки). Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных через ввод;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Строка», и создающий строку, состоящую только из нечётных символов исходной строки.

Определить деструктор класса.

17.Определить класс «Треугольник» с закрытыми элементами a, b и c (стороны треугольника). Определить методы класса, обеспечивающие доступ к этим переменным. Определить метод, вычисляющий площадь треугольника. Создать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных через ввод;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Треугольник», и создающий треугольник, стороны которого в два раза больше сторон исходного треугольника.

18.Определить класс «Треугольник» с закрытыми элементами a, b и c (стороны треугольника). Определить методы класса, обеспечивающие доступ к этим переменным. Определить метод, вычисляющий периметр треугольника. Создать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных при помощи передачи параметров;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Треугольник», и создающий треугольник, стороны которого в два раза меньше сторон исходного треугольника.

19.Определить класс «Треугольник» с закрытыми элементами a, b и c (стороны треугольника). Определить методы класса, обеспечивающие доступ к этим переменным. Определить метод, определяющий, является ли треугольник прямоугольным. Создать такие конструкторы для класса:

– конструктор-инициализатор (по умолчанию создавать равносторонний треугольник со сторонами равными единице);

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Треугольник», и создающий треугольник, стороны которого равны максимальной стороне исходного треугольника.

20.Определить класс «Треугольник» с закрытыми элементами a, b и c (стороны треугольника). Определить методы класса, обеспечивающие доступ к этим переменным. Определить метод, определяющий, является ли треугольник равнобедренным. Создать такие конструкторы для класса:

– конструктор-инициализатор (по умолчанию создавать равносторонний треугольник со сторонами равными единице);

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Треугольник», и создающий треугольник, стороны которого равны среднему арифметическому сторон исходного треугольника.

21.Определить класс «Треугольник» с закрытыми элементами a, b и c (стороны треугольника). Определить методы класса, обеспечивающие доступ к этим переменным. Определить метод, определяющий, является ли треугольник Египетским. Создать такие конструкторы для класса:

– конструктор-инициализатор (по умолчанию создавать равносторонний треугольник со сторонами равными единице);

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Треугольник», и создающий треугольник, стороны которого равны модулю разности сторон исходного треугольника, но не 0 (в этом случае стоит принимать стороны равные 1).

22.Определить класс «Одномерный массив чисел» с закрытыми элементами mas – массив чисел и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией n через ввод, а mas – при помощи генератора случайных чисел;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Одномерный массив чисел», и создающий перевёрнутый массив.

Определить метод, определяющий максимальный элемент в массиве.

23.Определить класс «Одномерный массив чисел» с закрытыми элементами mas – массив чисел и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией данных при помощи передачи параметров;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Одномерный массив чисел», и создающий массив, состоящий только из чётных элементов исходного массива.

Определить метод, определяющий минимальный элемент в массиве.

24.Определить класс «Одномерный массив чисел» с закрытыми элементами mas – массив чисел и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор с инициализацией n через ввод (по умолчанию n=10), а mas – при помощи генератора случайных чисел;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Одномерный массив чисел», и создающий массив, состоящий только из нечётных элементов исходного массива.

Определить метод, определяющий среднее арифметическое элементов в массиве.

25.Определить класс «Одномерный массив чисел» с закрытыми элементами mas – массив чисел и n – число элементов массива. Определить

методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-преобразователь;

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Одномерный массив чисел», и создающий массив, элементы которого являются суммой смежных элементов исходного массива.

Определить метод, определяющий число элементов в массиве, которые меньше числа заданного пользователем.

26.Определить класс «Одномерный массив чисел» с закрытыми элементами mas – массив чисел и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор-инициализатор (по умолчанию задавать массив из 10 элементов, каждый элемент в диапазоне от 20 до 80);

– конструктор копии, принимающий в качестве параметра ссылку на объект класса «Одномерный массив чисел», и создающий массив, элементы которого являются произведением смежных элементов исходного массива.

Определить метод, определяющий число элементов в массиве, которые больше числа заданного пользователем.

27.Определить класс «Меню» с закрытыми элементами str_men – массив названий пунктов меню и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор по умолчанию (создающий массив из пунктов меню «Открыть», «Закрыть», «Сохранить»);

– конструктор копии, получающий в качестве параметра ссылку на объект класса «Меню», и добавляющий к нему пункты «Выход» и «О программе».

Реализовать возможность добавления нового пункта меню.

28.Определить класс «Множество» с закрытыми элементами mas – массив с элементами множества и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор инициализатор (по умолчанию создавать пустое множество);

– конструктор копии, получающий в качестве параметра ссылку на объект класса «Множество», и создающий множество, в котором ни один элемент не принадлежит исходному множеству.

Реализовать методы добавления в множество новых элементов.

29.Определить класс «Множество» с закрытыми элементами mas – массив с элементами множества и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор инициализатор (по умолчанию создавать множество из 10 случайных элементов);

– конструктор копии, получающий в качестве параметра ссылку на объект класса «Множество», и создающий множество, в котором все элементы сдвинуты на одну позицию по отношению к исходному множеству (использовать побитовую операцию <<).

Реализовать методы удаления из множества указанного элемента .

30.Определить класс «Множество» с закрытыми элементами mas – массив с элементами множества и n – число элементов массива. Определить методы класса, обеспечивающие доступ к этим переменным. Создать такие конструкторы для класса:

– конструктор инициализатор с инициализацией через ввод;

– конструктор копии, получающий в качестве параметра ссылку на объект класса «Множество», и создающий множество, в котором все элементы сдвинуты на одну позицию по отношению к исходному множеству (использовать побитовую операцию >>).

Реализовать метод поиска заданного элемента в множестве.

Уровень Б (+1 балл)

Выполнить задание уровня А, используя в конструкторах списки инициализации.

Уровень В (+4 балла)

1. Определить класс «Квадратная матрица», обладающий такими элементами: размерность матрицы указатель на матрицу. Определить метод класса, вычисляющий детерминант матрицы. Вычисление детерминанта должно проводится за приемлемое время для матриц размерности (100x100). Определить такие конструкторы:

– конструктор-инициализатор (по умолчанию создавать единичную матрицу);

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Квадратная матрица» и коэффициент пропорциональности, и создающий матрицу, являющую произведением коэффициента и исходной матрицы.

Определить метод отображения матрицы на экран и записи в файл. Определить деструктор класса.

2. Разработать класс «Комплексные числа». Предусмотреть в классе несколько конструкторов: с возможностью задания только действительной части, только комплексной, конструктор по умолчанию, и конструктор инициализатор с инициализацией при помощи списка инициализации для пары

комплексных чисел. Предусмотреть возможность выведения комплексного числа в стандартном виде.

Реализовать для класса возможности: сложения, вычитания, умножения, деления, возведения в степень (целую, положительную) комплексных чисел.

3. Определить класс «Человек» с заданием атрибутов: рост, вес, возраст. Предусмотреть для класса конструктор-инициализатор (с инициализацией при помощи списка инициализации), конструктор копии (создающий объект с «идеальным весом» при заданном росте и возрасте).

Разработать метод класса, который рассчитывает идеальный вес для указанного возраста и роста. Реализовать метод, отображающий разницу между фактическим и идеальным весом.

4. Определить класс «Список». Реализовать для класса:

– конструктор-инициализатор, с инициализацией через передачу параметров;

– конструктор по умолчанию, создающий пустой список;

– конструктор копии, создающий список, обратный исходному.

Реализовать в классе методы добавления нового элемента (в указанную позицию), поиска заданного элемента, вывод на печать элемента с заданным индексом, вывод списка в прямом и обратном порядке.

5. Определить класс «Длинная арифметика», с двумя закрытыми элементами *a* и *b*. *a* и *b* – длинные целые числа, т.е. числа, значения которых превышают максимально (минимально) допустимое значение целочисленного типа (MIN_INT и MAX_INT). Предусмотреть в классе:

– конструктор инициализатор, для ввода чисел с клавиатуры (по умолчанию числа равны нулю).

– конструктор копии, создающий числа с обратным знаком.

Реализовать в классе методы нахождения результатов операций: +,-,*,/ для чисел *a* и *b*, возвести в степень, найти квадратный корень для числа *a*.

Определить деструктор класса.

6. Определить класс «Система линейных уравнений», обладающий такими элементами: размерность системы, указатель на матрицу коэффициентов, указатель на матрицу свободных членов. Предусмотреть в классе:

– конструктор по умолчанию (размерность системы 3, коэффициенты и свободные члены задаются случайным образом);

– конструктор-инициализатор с инициализацией через ввод.

Реализовать в классе метод нахождения корней системы уравнений и метод отображения результатов. Определить деструктор класса.

7. Определить класс «Системы счислений», обладающий такими закрытыми элементами: число a – целое число в некоторой системе счисления (2,8,10,16), q – основание системы счисления. Предусмотреть в классе:

– конструктор инициализатор, для ввода числа с клавиатуры (по умолчанию число равно нулю, система десятичная).

– конструктор копии, создающий число с обратным знаком.

Реализовать в классе методы доступа к числу a и преобразования числа в систему счисления с любым другим основанием от 2 до 16 включительно.

Определить деструктор класса.

8. Определить классы «Каталог игр для PC», «Клиенты», и «Заказы». В классе «Каталог игр для PC» реализовать структуру для хранения информации об: идентификаторе игры, названии игры, годе выпуска, системных требованиях (одной строкой), цене. В классе «Клиенты» реализовать структуру для хранения информации об: идентификаторе клиента, ФИО, номер телефона. В классе «Заказы» реализовать структуру для хранения информации об: игре (идентификатор), клиенте (идентификатор) и дате заказа. Определить такие конструкторы:

– конструктор по умолчанию, создающий пустые структуры.

Определить методы добавления данных в структуру каждого из классов. Предусмотреть проверки правильности добавления. Определить методы отображения данных структур каждого класса.

Определить метод позволяющий отобразить данные о: названии игры, ФИО пользователя и дате заказа игры пользователем.

Определить деструкторы классов.

9. Определить класс «Слова», с двумя закрытыми элементами-строками s1 и s2. Предусмотреть в классе:

– конструктор инициализатор, для ввода строк с клавиатуры (по умолчанию строки пусты).

Реализовать в классе метод проверки можно ли из букв, входящих в одно слово, составить другое (каждая буква используется только один раз). Словарь задать самостоятельно или подключить существующий.

Определить деструктор класса.

10. Определить класс «Рифмы», с закрытым элементом-строкой s1. Предусмотреть в классе:

– конструктор инициализатор, для ввода строки с клавиатуры (по умолчанию строка пуста);

– конструктор копии, создающий перевернутую строку.

Реализовать в классе метод вывода на экран всех пар слов, образующих рифму.

Определить деструктор класса.

11. Определить класс «Квадратная матрица», обладающий такими элементами: размерность матрицы указатель на матрицу. Определить метод класса, вычисляющий ранг матрицы. Определить такие конструкторы:

– конструктор-инициализатор (по умолчанию создавать единичную матрицу);

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Квадратная матрица», и создающий матрицу с обратным знаком.

Определить метод отображения матрицы на экран и записи в файл. Определить деструктор класса.

12. Определить класс «Криптография», с закрытым элементом *t* – который представляет собой некоторый текст. Предусмотреть в классе:

– конструктор инициализатор, для ввода текста с клавиатуры (задать некоторый текст по умолчанию).

Класс должен содержать методы для шифровки и расшифровки текста по трем алгоритмам или более. Сымитировать атаку криптоаналитика (имеется зашифрованный текст, делается предположение о методе, которым он зашифрован, подбирается ключ и расшифровывается текст). Сравнить время, затраченное на подбор ключей для разных методов.

Определить деструктор класса.

13. Определить класс «Алгебраическое выражение», с закрытым элементом-строкой *f* – которая представляет собой алгебраическое выражение. Предусмотреть в классе:

– конструктор инициализатор, для ввода строки с клавиатуры (по умолчанию строка пуста).

Реализовать в классе метод проверки, допустимым ли образом расставлены скобки в алгебраическом выражении *f*, учитывать минимум два типа скобок («(…)» и «[…]»).

Определить деструктор класса.

14. Определить класс «Система линейных уравнений», обладающий такими элементами: размерность системы, указатель на матрицу коэффициентов, указатель на матрицу свободных членов. Предусмотреть в классе:

– конструктор по умолчанию (размерность системы 3, коэффициенты и свободные члены задаются случайным образом);

– конструктор-инициализатор с инициализацией через ввод.

Реализовать в классе метод нахождения количества решений системы. Определить деструктор класса.

15. Определить класс «Точки», обладающий такими элементами: массив точек на плоскости – *A*, количество точек – *n*. Определить такие конструкторы:

– конструктор-инициализатор;

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Точки», и создающий массив зеркальных точек.

Определить метод для нахождения уравнения прямой, которая делит это множество точек на два равномощных подмножества.

Определить деструктор класса.

16. Определить класс «Квадратная матрица», обладающий такими элементами: размерность матрицы указатель на матрицу. Определить метод класса, приводящий матрицу к треугольному виду. Определить такие конструкторы:

– конструктор-инициализатор (по умолчанию создавать единичную матрицу);

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Квадратная матрица» и создающий транспонированную матрицу к исходной.

Определить метод отображения матрицы на экран и записи в файл. Определить деструктор класса.

17. Определить класс «Пятнашки», с закрытым элементом: массив ячеек – *A* (4x4). Определить такие конструкторы:

– конструктор-инициализатор случайным образом расставляющий 15 фишек от 1 до 15 по ячейкам, одна ячейка остается свободной.

Определить метод для расставления фишек по возрастанию их номеров, при том, что передвигать фишки можно только на соседнюю свободную ячейку.

18. Определить класс «Морской бой», с закрытыми элементами: два массива клеток – *A* (10x10). Определить такие конструкторы:

– конструктор-инициализатор случайным образом расставляющий 4 корабля по 1 клетке, 3 корабля по 2 клетки, 2 корабля по 3 клетки, 1 корабль в 4 клетки (вертикально, горизонтально или по диагонали) для двух игроков, с тем условием, что между кораблями должна быть пустая клетка.

Определить методы для имитации игры в морской бой между двумя игроками.

Доказать или опровергнуть теорию, о том что, при случайной расстановке кораблей и случайном выборе клеток для атаки противников процент побед будет равным 50.

Клетки для атаки противника выбираются случайно, однако если есть фактор попадания во вражеский корабль, то следующей берется случайная соседняя клетка, а если корабль уничтожен, то соседние с кораблем клетки больше не выбираются.

Определить деструктор класса.

19. Определить класс «Алгебраическое выражение», с закрытым элементом-строкой *fx* – которая представляет собой алгебраическое выражение одной переменной. Предусмотреть в классе:

– конструктор инициализатор, для ввода строки с клавиатуры (по умолчанию строка пуста).

Реализовать в классе метод нахождения значения алгебраического выражения при заданной переменной. Для упрощения выполнения использовать список (стек).

Определить деструктор класса.

20. Определить класс «Квадратная матрица», обладающий такими элементами: размерность матрицы указатель на матрицу. Определить метод класса, вычисляющий обратную матрицу, за приемлемое время для размерности (100x100). Определить такие конструкторы:

– конструктор-инициализатор (по умолчанию создавать единичную матрицу);

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Квадратная матрица» и коэффициент пропорциональности, и создающий матрицу, являющую произведением коэффициента и исходной матрицы.

Определить метод отображения матрицы и обратной матрицы на экран и записи в файл. Определить деструктор класса.

21. Определить класс «Быки и коровы», с закрытым элементом: *r* – случайное число. Определить такие конструкторы:

– конструктор-инициализатор случайным образом задающий пятизначное целое число с разными цифрами.

Определить метод для угадывания пользователем числа, согласно правилам. На каждом шаге пользователь вводит пятизначное число, а программа сообщает, сколько цифр числа угадано (быки) и сколько цифр угадано и стоит не на нужном месте (коровы).

Реализовать подсчет очков: +2 за быка, +1 за корову.

22. Определить класс «Шахматная доска», с закрытым элементом: массив клеточек – *A* (8x8). Определить такие конструкторы:

– конструктор-инициализатор помещающий в заданную клеточку шахматную фигуру – конь.

Определить метод для поиска путей коня к четырем угловым клеточкам доски за наименьшее число ходов. Шахматная фигура конь перемещается на 1 клеточку по горизонтали и 2 по вертикали или на 2 клеточки по горизонтали и 1 клеточку по вертикали.

Если за начальную клеточку задана угловая, найти пути к трем оставшимся.

23. Определить класс «Ханойские башни», с закрытыми элементами: *t1, t2, t3* – массивы заданной размерности, *n* – размерность массивов. Определить такие конструкторы:

– конструктор-инициализатор динамически создающий массивы и записывающий в *t1* числа от *n* до 1 по убыванию;

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Ханойские башни», сохраняющий текущее состояние массивов *t1, t2, t3*.

Массивы *t1, t2, t3* – представляют собой три стержня; на *t1* нанизаны круги разного диаметра в виде пирамидки.

Определить метод для перекладывания кругов из *t1* на *t3* используя *t2*. При этом круги нужно класть так, чтобы круг с меньшим диаметром всегда был сверху.

Определить деструктор класса.

24. Определить класс «Тренажёр клавиатуры», с закрытым элементом-строкой s1 и закрытым элементом *p* – счет пользователя. Предусмотреть в классе:

– конструктор инициализатор, для генерации случайной строки, не менее 50-ти символов;

– конструктор копии, создающий перевернутую строку.

Реализовать в классе метод для проверки умения пользователя печатать, не глядя на клавиатуру.

За каждую ошибку пользователю начисляется штрафной балл.

Вывести итоговый счет пользователя.

Определить деструктор класса.

25. Определить класс «Ферзи», с закрытым элементом: массив клеточек – *A* (8x8). Определить такие конструкторы:

– конструктор-инициализатор помещающий в случайные 8 клеточек 8 шахматных фигур – ферзей.

Определить метод для поиска такого расположения ферзей, чтобы они не били друг друга. Шахматная фигура ферзь может бить фигуры по вертикали, горизонтали и диагонали.

За одну итерацию можно двигать только одного ферзя. Вывести все промежуточные перестановки ферзей в текстовый файл.

26. Определить класс «Неизвестное слово», с закрытым элементом-строкой s1 и закрытым элементом *p* – счет пользователя. Предусмотреть в классе:

– конструктор инициализатор, для выбора слова из наперед заданных слов или из подключенного словаря.

Реализовать в классе метод угадывания пользователем слова по буквам.

За каждую правильно названную букву пользователю начисляется от 100 до 1000 очков, за неправильно названную снимается 200. Если в слове несколько одинаковых букв число очков, при угадывании, умножается на количество этих букв.

При желании организовать подсказки.

Вывести итоговый счет пользователя.

Определить деструктор класса.

27. Определить класс «Выражения», с двумя закрытыми элементами-строками s1 и s2. Предусмотреть в классе:

– конструктор инициализатор, для ввода строк с клавиатуры (алгебраических выражений высокого порядка).

Реализовать в классе методы сложения и вычитания этих выражений.

Пользователь задает выражения в следующем виде, один из примеров: *x^5+3x^3-x^2-8+5x^2=0*.

Порядок выражений ограничен 10.

Определить деструктор класса.

28. Определить класс «Лабиринт», с открытым элементом: массив клеточек – *A* (40x40). Определить такие конструкторы:

– конструктор-инициализатор генерирует случайным образом лабиринт (*A*[*i*,*j*]=0, если клетка [*i*,*j*] "проходима"; *A*[*i*,*j*]=1, если клетка [*i*,*j*] "непроходима"), боковые строки и столбцы всегда "непроходимы", кроме одной клеточки.

Определить метод для поиска выхода из лабиринта из заданной клеточки или сообщить о его отсутствии. Лабиринт и путь отобразить в текстовом файле.

29. Определить класс «Квадратная матрица», обладающий такими элементами: размерность матрицы указатель на матрицу. Определить метод класса, транспонирующий матрицу относительно обратной диагонали. Определить такие конструкторы:

– конструктор-инициализатор (по умолчанию создавать единичную матрицу);

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Квадратная матрица» и создающий транспонированную матрицу к исходной.

Определить метод отображения матрицы на экран. Определить деструктор класса.

30. Определить класс «Квадратная матрица», обладающий такими элементами: размерность матрицы указатель на матрицу. Определить метод класса, приводящий матрицу к ортогональному виду. Определить такие конструкторы:

– конструктор-инициализатор (по умолчанию создавать единичную матрицу);

– конструктор копии, принимающий в качестве параметров ссылку на объект класса «Квадратная матрица» и создающий транспонированную матрицу к исходной.

Определить метод отображения матрицы на экран. Определить деструктор класса.

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, если необходимо;
- пошаговый алгоритм решения задачи;
- UML-диаграмму классов;
- исходный код программы с комментариями;
- примеры работы программы;
- выводы.

Отчет оценивается максимально в 1 балл.

5 КОНТРОЛЬНЫЕ ВОПРОСЫ

1. Что такое инкапсуляция данных?
2. Назначение и виды конструкторов.
3. Назначение деструктора.