Лабораторная работа N 3

Классы. Протокол класса. Конструкторы и деструкторы

1. Введение

Проведем краткий обзор некоторых ключевых принципов и терминологии ООП. ООП *инкапсулирует* данные (атрибуты) и функции (варианты поведения) в совокупности, называемые *объектами;* данные и функции объекта тесно связаны друг с другом. Объекты обладают свойством *скрытия информации.* Это значит, что хотя объекты могут знать, как связываться друг с другом посредством хорошо определенного *интерфейса,* им обычно не позволено знать, как реализуются другие объекты — детали реализации спрятаны внутри самих объектов. Несомненно, можно ездить на автомобиле, не зная технических деталей его внутреннего функционирования — трансмиссии, выхлопной трубы и др. Мы увидим, почему скрытие информации так важно для разработки хорошего программного обеспечения.

В С и других *процедурно-ориентированных языках* программирование стремится быть *ориентированным на действия,* тогда как в идеале программирование на *С++ объектно-ориентированное.* В С единицей программирования является функция. В С++ единицей программирования является *класс,* на основе которого в конечном счете создаются объекты.

Программисты на С основное внимание уделяют написанию функций. Группы действий, выполняющие некоторую задачу, объединяются в функции, а функции группируются, чтобы образовать программу. Данные несомненно важны в С, но с точки зрения ООП, данные существуют в первую очередь для поддержки действий, выполняемых функциями.

Программисты на С++ основное внимание уделяют созданию своих собственных *определяемых пользователем типов,* называемых *классами.* Класс - это определяемый программистом тип. Каждый класс содержит данные и набор функций, манипулирующих с этими данными. Компоненты-данные класса называются *данными-членами.* Компоненты-функции класса называются *функциями-членами.* Подобно тому как сущность встроенного типа, такого, как **int,** называется переменной, сущность определяемого пользователем типа (т.е. класса) называется *объектом.* Центром внимания в С++ являются не функции, а объекты.

Целью лабораторной работы является получение практических навыков при создании простых классов и при работе с конструкторами и деструкторами.

2. Общие сведения.

2.1 Классы и методы в языке С++

В общем случае, класс представляется в следующей форме:

class *имя_класса*{ *список членов*};

Описание класса начинается с ключевого слова class. Список членов класса определяет собственные элементы класса. При описании членов классов возможно указание спецификаторов управления доступом к элементам классов. Такими спецификаторами являются:

– *public:* члены класса видны извне класса

– *private:* соответствующие элементы могут использоваться только внутри класса

– *protected:* соответствующие элементы могут использоваться только внутри класса и в его потомках.

По умолчанию элементы класса имеют тип *private*. Указанный в описании спецификатор доступа распространяется на все последующие определения, пока не встретится другой спецификатор.

Конструкторы и деструкторы

После создания объекта его элементы могут быть инициализированы с помощью специальной функции - *конструктор*. ***Конструктор*** –функция-член, имющая то же имя, что и класс, частью которого он является, и не имеет возвращаемого значения. Программист предусматривает конструктор, который затем автоматически вызывается при создании объекта (создании экземпляра класса). *Данные-элементы класса не могут получать начальные значения в определении класса.* Они либо должны получить эти значения в конструкторе класса, либо их значения можно установить позже, после создания объекта. Конструкторы можно *перегружать*, чтобы обеспечить множество начальных значений объектов класса.

Когда объявляется объект класса, между именем объекта и точкой с запятой можно в скобках указать *список инициализации элементов.* Эти начальные значения передаются в конструктор класса. Если у конструктора нет параметров, то он называется *конструктором по умолчанию*.

Пример 3.1 Класс с конструктором.

```
#include <iostream>
using namespace std;
class myclass {
        int a;
public:
        myclass(); // конструктор по умолчанию
        void show();
};
myclass::myclass()
{
        cout « "В конструкторе\n" ;
        a=10;
}
void myclass::show()
{
        cout« a;
}
int main()
{
        myclass ob;
        ob.showQ ;
        return 0;
}
```

Значение **a** инициализируется конструктором **myclass**(). Конструктор вызывается тогда, когда создается объект **ob**. Объект, в свою очередь, создается при выполнений инструкции объявления объекта. При программировании на С объявление переменных понимаются просто как создание переменных. В С++, поскольку объект может иметь конструктор, инструкция объявления переменной может вызывать выполнение записанных в конструкторе действий.

Обратите внимание, как определяется конструктор **myclass**(). В соответствии с формальными правилами синтаксиса С++ конструктор не должен иметь возвращаемого значения.

Функцией, обратной конструктору, является *деструктор*. ***Деструктор*** –это функция-член, которая вызывается при удалении объекта. Обычно при работе с объектом в момент его удаления должны выполняться некоторые действия. Например, при создании объекта для него выделяется память, которую необходимо освободить при его удалении. Имя деструктора совпадает с именем класса, но с символом ~ (тильда) в начале.

Пример 2 Класс с деструктором

```
#include <iostream>
using namespace std;
class myclass {
int a;
public:
myclass(); // конструктор
-myclass'O; // деструктор
void show();
};

myclass::myclass()
{
cout« "Содержимое конструктора\n" ;
a =10;
}
myclass:: ~myclass ()
{
cout« "Удаление...\n";
}
void myclass::show()
I
cout « a « "\n";
}
int main()
I
myclass ob;
ob.show();
return 0;
}
```

Деструктор класса вызывается при удалении объекта. Локальные объекты удаляются тогда, когда они выходят из области видимости. Глобальные объекты удаляются при завершении программы. Адреса конструктора и деструктора получить невозможно.

2.2 Разновидности конструкторов.

2.2.1 Конструктор с параметрами

Конструктору можно передавать аргументы. Для этого нужно добавить необходимые параметры в объявление и определение конструктора. Затем при объявлении объекта задайте параметры в качестве аргументов.

Пример 3 Конструктор с параметрами.

```
#include <iostream>
using namespace std;
class myclass
{
int a;
public:
myclass(int x); // конструктор
void show();
};
myclass::myclass(int x)
{
cout« "В конструкторе\n";
a = x;
void myclass::show()
{
```

```
cout « a « "\n";
}
int main()
{
myclass ob(4);
ob.showQ;
return 0;
}
```

Здесь конструктор класса **myclass** имеет один параметр. Значение, передаваемое в **myclass(),** используется для инициализации переменной a. Обратите особое внимание на то, как в функции **main()** объявляется объект ob. Число 4, записанное в круглых скобках, является аргументом, передаваемым параметру x конструктора **myclassQ,** который используется для инициализации переменной a. Вполне допустимо передавать конструктору несколько аргументов.

Фактически синтаксис передачи аргумента конструктору с параметром является сокращенной формой записи следующего, более длинного выражения:

myclass ob = myclass(4);

Однако большинство программистов C++ пользуются сокращенной формой записи. На самом деле, с технической точки зрения между этими двумя формами записи имеется небольшое отличие, связанное с конструктором копий (copy constractor), о котором будет рассказано позже.

2.2.2 Конструктор копирования

Объект класса без конструкторов можно инициализировать путем присваивания ему другого объекта этого класса. Это можно делать и тогда, когда конструкторы описаны. Например:

date d = today; // инициализация посредством присваивания

Вообще, имеется конструктор по умолчанию, определенный как почленная (в ранних версиях C++ - побитовая) копия объекта того же класса. Если для класса X такой конструктор по умолчанию нежелателен, его можно переопределить конструктором с именем X(X&). Это будет конструктор копирования. Описание конструктора копирования:

stack::stack(const stack&);

По умолчанию компилятор производит копирование путем почленной инициализации. Это не всегда применимо. В некоторых случаях в конструкторах происходит, например, выделение динамической памяти, поэтому применение почленной инициалиции приведет к фатальной ошибке при попытке обратится к динамической памяти или при удалении объекта. Поэтому важно, чтобы класс имел свою собственную явно определенную копию конструктора.

Пример 4 конструктор копирования.

```
Stack::stack(const stack& str) {
s=new char [str.max_len];
max_len=str.max__len;
top=str.top;
memcpy(s,str.s,max_len);
}
```

Конструктор копирования вызывается в случае инициализации при объявлении.

```
Stack s(10); // конструктор с одним параметром
Stack d=s; // конструктор копирования.
```

Все хорошо разработанные классы должны иметь конструктор копирования, особенно те, которые имеют дело с динамической памятью, открытыми файлами, любыми указателями.

2.3 Встраиваемые функции (inline-функции)

В C++ можно задать функцию, которая на самом деле не вызывается, а ее тело встраивается в программу в месте ее вызова. Преимуществом встраиваемых (**inline**) функций является то, что они не связаны с механизмом вызова функций и возврата ими своего значения. Следовательно, встраиваемые функции выполняются гораздо быстрее обычных.

Однако, если встраиваемые функции слишком большие и вызываются слишком часто, объем программ сильно возрастает. Из-за этого их применение ограничивается короткими часто вызываемыми функциями. Для объявления встраиваемой функции нужно вставить спецификатор inline перед определением функции.

Пример 5. Использование встраиваемых функций:

```
#include <iostream>
using namespace std;
inline int even(int x)
      { return ! (x%2);
      }
int main() {
      if (even(10)) cout«"10 является четным \n";
       if (even(ll)) cout«; "11 является четным\n";
return 0;}
```

В этом примере функция **even(),** которая возвращает истину при четном аргументе, объявлена встраиваемой. Это означает, что строка

```
if (even(10)) cout« "10 является четным\n";
```

функционально идентична строке

```
if (! (10%2)) cout « "10 является четным\n";
```

2.4 Ключевое слово *this*

Ключевое слово *this* обозначает специальную локальную переменную, доступную в теле любой функции-члена класса. Переменная *this* не требует описания и является указателем на самого себя. Каждый объект может определить свой собственный адрес с помощью ключевого слова *this*.

– *this->имя члена* указывает на объект, членом, которого он является.

– **this* представляет собой сам объект и, в зависимости от контекста, может быть лево- или правосторонней величиной

– *this* представляет собой адрес объекта.

Пример 6. *Ключевое слово this*

```
class C {
int c1,c2;
public:
void init(int b) { c2=b; c1=b+1; }
C & inc() { c1++; c2++; return *this; }
void *adress() { return this; }
void print() { cout << c1 << c2; }
```

```
};
void main(void) {
C a;
a.init(10);
a.print();
cout << " adress=" << a.adress<< " inc " << a.inc().print() << endl;
}
```

Указатель на объект, для которого вызвана функция-член, является скрытым параметром функции. На этот неявный параметр можно ссылаться явно как на *this*. В каждой функции класса x указатель *this* неявно описан как

```
x* this;
```

и инициализирован так, что он указывает на объект, для которого была вызвана функция-член. *this* не может быть описан явно, так как это ключевое слово. Класс x можно эквивалентным образом описать так:

```
class x {
int m;
public:
int readm() { return this->m; }
};
```

```
class x {
int m;
public:
int readm() { return m; }
};
```

2.5 Отделение интерфейса от реализации

Один из наиболее фундаментальных принципов разработки хорошего программного обеспечения состоит в отделении интерфейса от реализации. Это облегчает модификацию программ. Что касается клиентов класса, то изменения в реализации класса не влияют на клиента класса до тех пор, пока интерфейс класса, изначально предназначенный для клиента, остается неизменным.

Для отделения интерфейса от реализации необходимо поместить объявление класса в заголовочный файл (*.h), чтобы оно было доступно любому клиенту класса. Это открытый интерфейс класса. А определения функций-элементов класса помещаются в исходный файл (*.cpp). Это закрытая от клиента реализация класса. Но на самом деле заголовочные файлы содержат некоторую часть реализации. Встраиваемые функции-элементы, например, должны находиться только в заголовочном файле.

При построении программы на C++ каждое определение класса обычно помещается в *заголовочный файл,* а определения функций-элементов этого класса помещаются в *файлы исходных кодов* с теми же базовыми именами. Заголовочные файлы включаются (посредством директивы #include) в каждый файл, в котором используется класс, а файлы с исходными кодами компилируются и компонуются с файлом, содержащим главную программу.

Программа на рис. 6.5 состоит из заголовочного файла time.h, в котором объявляется класс Time, файла time.cpp, в котором описываются функции-элементы класса Time, и файла main.cpp, в котором описывается функция main.

```
// TIME.H
// Объявление класса Time.
// Функции-элементы определены в TIME.CPP
// Предотвращение многократного включения заголовочного файла
#ifndef TIME_H
#define TIME_H
// Определение абстрактного типа данных Time
class Time {
public:
```

```
Time(); // конструктор по умолчанию
void setTime(int, int, int);    // установка часов, минут, секунд
void printMilitary();    //печать времени в военном формате
void printStandard();    //печать времени в стандартном формате
private:
int hour;       // 0 – 23
int minute;      // 0 – 59
int second;     // 0 - 59 };
#endif
```

```
// TIME.CPP
// Определения функций-элементов для класса Time.
#include <iostream.h>
#include "timel.h"
// Конструктор Time присваивает нулевые начальные значения каждому
//элементу данных. Обеспечивает согласованное начальное состояние
//всех объектов
Time Time::Time()
{ hour = minute = second = 0; }
// Задание нового значения Time в виде военного времени.
// Проверка правильности значений данных. Обнуление неверных значений.
void Time::setTime(int h, int m, int s)
{
hour = (h >= 0 && h < 24) ? h : 0;
minute = (m >= 0 && m < 60) ? m : 0;
second = (s >= 0 && s < 60) ? s : 0;
}
// Печать времени в военном формате
void Time::printMilitary()
{
cout « (hour < 10 ? "0" : "") « hour « ":"« (minute < 10 ? "0" : "") « minute « ":" « (second < 10 ? "0" : "")
« second;
// Печать времени в стандартном формате
void Time::printStandard()
{
cout « ((hour == 0 | | hour == 12 ) ? 12 : hour % 12) « ":" « (minute < 10 ? "0" : "") « minute « ":" « (second
< 10 ? "0" : ""} « second « (hour < 12 ? " AM" : " PM");
}
```

```
// MAIN.CPP
// Основная программа
// ЗАМЕЧАНИЕ: Компилируется вместе с TIME.CPP
#include <iostream.h>
#include "timel.h"
// Проверка простого класса Time
main ( )
{
Time t;       // определение экземпляра объекта t класса Time
cout << "Начальное значение военного времени равно ";

t.printMilitary();
cout <<endl << "Начальное значение стандартного времени равно";
t.printStandard();
t.setTime(13, 27, 6);
cout « endl « "Военное время после setTime равно ";
t.printMilitary();
cout « endl « "Стандартное время после setTime равно ";
 t . printStandard ( ) ;
t.setTime(99, 99, 99);     // попытка установить неправильные значения
cout « endl « "После попытки неправильной установки: " « endl « "Военное время: ";
t.printMilitary() ;
cout « endl « "Стандартное время: ";
t . printStandard ( ) ;
return 0; }
```

Заметим, что объявление класса заключено в следующие директивы препроцессора:

```
// Предотвращение многократного включения заголовочного файла
#ifndef TIME_H
#define TIME_H
…
#endif
```

При построении больших программ в заголовочные файлы будут помещаться также и другие определения и объявления. Приведенные выше директивы препроцессора предотвращают включение кода между **#ifndef и #endif,** если определено имя **TIME_H.** Если заголовок еще не включался в файл, то имя **TIME_H** определяется директивой **#define** и операторы заголовочного файла включаются в результирующий файл. Если же заголовок уже был включен ранее, **TIME_H** уже определен и операторы заголовочного файла повторно не включается. Попытки многократного включения заголовочного файла обычно случаются в больших программах с множеством заголовочных файлов, которые могут сами включать другие заголовочные файлы. Замечание: по негласному соглашению в приведенных выше директивах используется имя символической константы, представляющее собой просто имя заголовочного файла с символом подчеркивания вместо точки.

2.6 Констукторы и массивы объектов

Чтобы описать вектор объектов класса, имеющего конструктор, этот класс должен иметь конструктор, который может вызываться без списка параметров. Нельзя использовать даже параметры по умолчанию. Например:

```
table tblvec[10];
```

будет ошибкой, если нет конструктора без параметров так как для **table::table(int sz=10)** требуется целый параметр. Нет способа задать параметры конструктора в описании вектора. Чтобы можно было описывать вектор таблиц table, можно модифицировать описание table, например, так:

```
class table {
void init(int sz); // как старый конструктор
public:
table(int sz) { init(sz); } // как раньше, но без по умолчанию
table() { init(10); } // по умолчанию
};
```

Когда вектор уничтожается, деструктор должен вызываться для каждого элемента этого вектора. Для векторов, которые не были размещены с помощью new, это делается неявно. Однако для векторов в свободной памяти это не может быть сделано неявно, поскольку компилятор не может отличить указатель на один объект от указателя на первый элемент вектора объектов. Например:

```
void f() {
table* t1 = new table;
table* t2 = new table[10];
table* t3 = new table[10];
delete t1; // одна таблица
delete t2; // неприятность: 10 таблиц
delete[] t3;
}
```

Компилятор не может найти число элементов вектора из объема выделенной памяти, потому, что распределитель свободной памяти не является частью языка и может быть задан программистом.

Конструкторы могут инициализировать массивы объектов класса таким образом, как и массивы встроенных типов. Максимальное число элементов может быть опущенно, если оно равно числу инициализирующих значений. Если максимальное число элементов больше числа значений, то оставшиеся значения инициализируются при помощи конструктора по умолчанию. Если это не конструктор по умолчанию, то все же значения должны быть указаны. Если заданы все значения для данного размера массива, то фигурные скобки при указании списка значений могут быть опущены.

```
class Phone {
int a,b,c;
public:
Phone(int a1,int b1,int c1):a(a1),b(b1),c(c1) {}
};

Phone office[]= { // Компилятор вычислит размер за нас
900, 800, 905,
678,456,546 };
Phone office[3]= { // Требуется конструктор по умолчанию, которого у Phone нет
890, 790,343,
238, 279, 564
};
```

3. Дополнительная литература

1. Герберт Шилдт. Самоучитель C++: Пер. с англ. – 3-е изд. – СПб.: БХВ-Петербург, 2003 (глава 2)

2. Харви Дейтел, Пол Дейтел Как программировать на C++: пер. с англ. -М: ЗАО "Издательство Бином", 1999. (глава 6)

2. Бьярн Страуструп. Язык программирование Си++. – М.: Бином, 2005.(глава 5)

4. Порядок выполнения работы

1. Создайте новый проект в среде MS Visual C++.

2. Для полученного варианта задания, руководствуясь описанием настоящей лабораторной работы, напишите программы на языке C++. Откомпилируйте, отладьте и выполните свою программу.

3. Подготовьте отчет о выполнении лабораторной работы

Для успешной сдачи лабораторной работы необходимо:

– представить преподавателю работающую программу для указанного варианта задания;

– ответить на вопросы по тексту программы.

5. Порядок оформления отчета

Отчет о выполнении лабораторной работы должен содержать:

1) титульный лист;

2) вариант задания;

3) текст программ;

4) результаты работы программ.

6. Варианты заданий

Для допуска к экзамену достаточно выполнить любое одно задание. Для студентов, претендующих на освобождение от экзаменов необходимо выполнить оба задания.

Вариант 1.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на int, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного int. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на int, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 2.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на float, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного float. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица. Данный класс содержит указатель на float, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 3.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на double, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного double. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на double, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 4.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на long, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного long. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на long, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 5.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на int, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного int. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на int, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 6.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на float, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного float. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на float, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 7.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на double, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного double. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на double, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 8.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на long, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного long. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на long, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 9.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на int, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного int. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на int, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 10.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на float, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного float. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на float, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 11.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на double, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного double. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на double, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 12.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на int, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного int. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на int, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 13.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на long, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного long. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на long, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 14.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на float, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного float. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на float, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого

класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 15.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на int, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного int. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на float, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 16.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на float, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного float. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на long, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого

класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 17.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на double, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного double. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на int, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 18.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на long, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного long. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на float, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого

класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 19.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на int, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного int. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на double, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 20.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на float, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного float. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на long, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого

класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 21.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на double, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного double. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на int, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 22.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на long, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного long. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на float, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого

класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 23.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на int, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного int. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на long, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 24.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на float, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного float. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на double, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого

класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.

Вариант 25.

1. Создать абстрактный тип данных - класс вектор, который имеет указатель на double, число элементов и переменную состояния. Определить конструктор без параметров, конструктор с параметром, конструктор с двумя параметрами. Конструктор без параметров выделяет место для одного элемента и инициализирует его в ноль. Конструктор с одним параметром, - размер вектора, - выделяет место и инициализирует номером в массиве, конструктор с двумя параметрами выделяет место (первый аргумент) и инициализирует вторым аргументом. Деструктор освобождает память. Определить функцию, которая присваивает элементу массива некоторое значение (параметр по умолчанию), функцию которая получает некоторый элемент массива. В переменную состояния устанавливать код ошибки, когда не хватает памяти, выходит за пределы массива. Определить функцию печати. Определить функции сложения, умножения, вычитания, которые производят эти арифметические операции с данными этого класса и встроенного double. Определить методы сравнения: больше, меньше или равно. Предусмотреть возможность подсчета числа объектов данного типа. Проверить работу этого класса.

2. Создать класс матрица Данный класс содержит указатель на int, размер строк и столбцов и состояние ошибки. Определить конструктор без параметров, конструктор с одним параметром и конструктор с двумя параметрами, деструктор. Определить методы доступа: возвращать значение элемента (i,j) и адрес этого элемента. Определить функцию печати. Определить функции сложения и вычитания (матрицы с матрицей), умножение матрицы на матрицу. Определить умножение матрицы на число. Проверить работу этого класса. В случае нехватки памяти, несоответствия размерностей, выхода за пределы устанавливать код ошибки.