Лабораторная работа № 3

**Программирование с использованием перегруженных функций и операторов.**

**Цель работы:** Получить практические навыки работы с перегруженными операторами и методами класса.

**Теоретические сведения**

**Перегрузка функций и операторов.**

Перегрузка является статическим полиморфизмом (полиморфизм времени компиляции)

Статический полиморфизм – гибкость и комфорт при программировании.

Полиморфи́зм — возможность объектов с одинаковой спецификацией иметь различную реализацию.

Перегрузка функций – это использование одного имени для нескольких функций.

Например функции перегружаются для выполнения одинаковых по смыслу действий над объектами различных типов.

**Перегружаемые функции.**

Функции с одним и тем же названием должны различаться сигнатурой (количеством или типами параметров) Возвращаемое значение в сигнатуру не входит

Повторные объявления (или ошибки) при перегрузке функций.

1. typedef не вводит нового типа

typedef double NALOG;

повторное объявление функции

NALOG calc(NALOG );

double calc( double );

1. Спецификаторы const или volatile не принимаются во внимание, если параметры передаются по значению, так как не влияют на их интерпретацию.

// повторное объявлене одной и той же функции

int f ( int );

void f ( const int );

1. Eсли спецификатор const или volatile применяется к указателю или ссылке, то при сравнении объявлений они учитываются.

// функции перегружаются

void f( int\* );

void f( const int\* );

void f( int& );

void f( const int& );

Пример : Функции для печати значений разного типа данных.

…

// печать целого числа

void print ( int i )

{ printf ("\n%d",i);

}

// печать дробного числа

void print ( double d )

{ printf ("\n%lf",d);

}

// печать строки

void print ( char \* s )

{ printf ("\n%s",s);

}

В вызывающей функции компилятор сам выберет одну из функций print(), основываясь на анализе типов аргументов

int i=5;

double pi=3.141592;

print (i);

print(pi);

print ("перегрузка функций");

Три приведенные функции print в объектном модуле будут соответствовать трем функциям с различными именами, модифицированные транслятором имена функций содержат информацию о количестве и типах параметров, причем модификацию имен компилятор выполняет для всех функций, а не только перегружаемых. Для того, чтобы функцию С++ можно было бы вызвать из программы на С, необходимо запретить модификацию имен, для чего функцию нужно описать с описателем extern "C".

extern "C" // отдельная функция

extern "C" // несколько функций

{ int fun2(int);

double fun3(double);

}

Очевидно, что функции описанные как extern "C" не могут быть перегружаемыми.

Область видимости и перегрузка.

Блок, функция, класс задают свои области видимости.

Все перегруженные функции объявляются в одной и той же области видимости.

Локально объявленная функция не перегружает, а скрывает глобальную

using namespace std;

void print( int r ) {cout<< r<<endl;};

void print( double r ) {cout<< r<<endl;};

void print( const char\* s ) {cout<< s<<endl;};

void f (double val)

{ void print (double k); // маскируются все print

print("строка"); // ошибка при трансляции

print (val);

};

int \_tmain(int argc, \_TCHAR\* argv[])

{ print("строка");

print (5);

f (3.14);

system ("pause");

return 0;

}

Каждый класс определяет собственную область видимости, поэтому функции, являющиеся членами двух разных классов, не перегружают друг друга.

class A

{int q;

public:

void show(const char\* s){cout<<s<<q<<endl;};

};

class B

{double w;

public:

void show(){cout<<"class B"<<w<<endl;};

};

Перегрузка конструкторов:

Создание инициированных и неинициированных объектов, конструкторов копирования.

class Int

{ int i;

public:

// Перегрузка конструкторов

Int(){i=0;} // без инициализации

Int (int n){i=n;} // с инициализацией

Int (const Int& k){i=k.i;}//конструктор копирования

void print(){cout<<"int:"<<i<<endl;}

void set (int k){i=k;}

int get (){return i;}

voiv show\_m()

};//----------конец определения класса Int

const int N=5;

void main()

{ Int a,b(10),c(b),\*pi;

int i;

a.print(); b.print(); c.print();

Int A[N]; // массив без инициализации

for (i=0;i<N;i++)

{A[i].set(i+1);A[i].print();} // инициализация

Int B[]={10,20,30};// массив с инициализацией

for (i=0;i<3;i++) B[i].print();

pi=new Int[N];// конструктор без инициализации

// иниициализация данными массива А

for(i=0;pi+i<pi+N;i++)

{(pi+i)->set(A[i].get());

(pi+i)->print();

}

system ("pause");

return 0;

}

**Указатель this**

До сих пор мы не задавали вопроса, как метод определяет с полями какого именно объекта надо работать, если объектов много, (например объекты Int a,b,c)

В тексте метода set () на этот счет нет никаких указаний!!!!

При вызове методы класса получают:

* список параметров
* неявный константный указатель this c адресом объекта, вызвавшего метод

void set (int k){this->i=k;}

Указатель this константный, то есть всегда указывает на один и тот же объект, адресные операции с ним запрещены.

int a;

this=&a; // ошибка!!!

this++; // ошибка!!!

Указатель this необходим когда необходимо работать с адресами объектов

1. Необходимо получить доступ ко всему объекту, а не к отдельным его полям
2. Объект является параметром функции или возвращаемым значением.

Напомним, что друзья класса – это внешние функции и классы, у которых доступ такой же, как у методов класса.

class Int

{ int i;

public:

. . .

friend void show\_m(Int\*, int);

};//----------конец определения класса Int

void show\_m (Int\* pI, int n)

{int i;

cout<<"-------mass ["<<n<<"]----------"<<endl;

for (i=0;i<n;i++,pI++) cout<<pI->i<<'\t';

cout<<endl;

}

void main()

{ Int A[N]; // массив без инициализации

for (i=0;i<N;i++)

{A[i].set(i+1);A[i].print();} // инициализация

show\_m(A,N);

system ("pause");

return 0;

}

**Перегрузка операторов**

Для выполнения действий над абстрактными (пользовательскими) типами в классическом С используются функции.

В С++ можно использовать операции

то есть можно создавать свои операции над экземплярами класса (новыми, пользовательскими типами).

Об операторе, который работает с абстрактными типами (объектами классов), говорят, что он перегружен или переопределен

Например, можно определить оператор умножения для матриц или комплексных чисел.

Однако одной функции на оператор зачастую оказывается недостаточно. Например, матрицу можно умножить на другую матрицу, а можно на константу.

**При перегрузке операторов существует несколько ограничений:**

* нельзя создавать новые символы операций
* нельзя переопределять операции для стандартных типов
* Нельзя поменять правила, определенные в трансляторе, например, бинарный оператор сделать унарным и наоборот.
* нельзя переопределять операции :

:: разрешение области видимости

. выбор элемента

\* (разыменование)

?: троичный оператор

* Следующие операторы должны быть реализованы только в виде нестатических методов класса:

= [] () ->

* переопределение операций не меняет их приоритетов, а также порядок их выполнения (слева направо или справа налево);
* для параметров в операторных функциях нельзя задавать значения по умолчанию.

Например :

<< только для бинарной операции,

! только для унарной,

\* только для бинарной;

Стандартное поведение перегруженных операторов не определено, например

* коммутативный закон для сложения:

а + b равно b + a;

* комбинированная операция, например: i=i+j; эквивалентно i += j

и некоторые другие правила.

Соблюдение подобных правил для перегруженных операторов лежит на программисте. Например, необходимо явно написать одинаковое поведение для операторов + и +=

**Возвращаемые значения**

Очень немного операторов жестко требуют какого-то определенного типа возвращаемого значения. В большинстве своем вы абсолютно свободны в выборе типа результата. Однако правила языка и контекст использования операторов зачастую дают рекомендации на тип значения (так, например, желательно, чтобы оператор присваивания возвращал тот же тип, что и у своего объекта).

**Два способа реализации перегружаемых операторов**

Операторы реализуются (перегружаются):

* в виде методов класса,
* в виде дружественных функций.

Реализация в виде методов класса

Оператор перегружается с помощью операторной функции

Также бывают противоположные ситуации - приходится использовать не метод, а

дружественную функцию.

**Бинарные перегруженные операторы**

(реализация через метод класса) имеют 1 параметр

Через указатель this передается левый операнд, правый операнд – через параметр

class Int

{ int i;

public:

// перегрузка операторов

// оператор+ не меняют значения параметров

const Int operator + (const Int& right)

{cout<<"op+"<<endl;

return Int(this->i+right.i);

}

// оператор+= меняет левый параметр

Int& operator += (const Int& right)

{cout<< "op+="<<endl;

i+=right.i;

return \*this;

}

};//----------конец определения класса Int

Void main()

{ Int a(1),b(2),c(3);

c+=a+b; c.print();

}

На экране:

Ор+

Ор+=

Int: 6

Void main()

{ Int a(1),b(2),c;

c=a+10; c.print(); // На экране:

} Ор+

Int: 11

c=10+a; // ошибка трансляции

При перегрузке в виде методов порядок операндов фиксирован: левый операнд передается неявно через this, а правый – явно через параметр

Чтобы реализовать коммутативность оператора +, нужна реализация через дружественную функцию.

Перегрузим оператор + для работы с константой

class Int

{ int i;

public: . . .

friend const Int operator + (int k, const Int& right);

};//----------конец определения класса Int

У независимой функции нет указателя , поэтому все параметры задаются явно :

1-ый параметр – левый операнд, 2-ой – правый

const Int operator + (int k, const Int& right)

{cout<<"op+"<<endl;

return Int(right.i+k);

}

void main()

{ Int a(1),b(2),c;

c=a+10; c.print();

b=1+a; b.print();

}

**Перегрузка унарного оператора ++**

class Int

{ int i;

public: . . .

// префиксная форма

friend const Int& operator ++ (Int& q);

// постфиксная форма

friend const Int operator ++ (Int& q, int);

};//----------конец определения класса Int

// префиксная форма возвращает значение после инкремента

const Int& operator ++ (Int& q)

{ cout<<"++op"<<endl;

q.i++;

return q;

}

// постфиксная форма возвращает значение до инкремента

const Int operator ++ (Int& q, int)

{cout<<"op++"<<endl;

Int t(q); // сохранение значения

q.i++;

return t;

}

void main()

{ Int a(1),b(2),c;

Int d=++a; d.print();

Int e = b++; e.print();

}

**Аргументы и возвращаемые значения**

1. Аргумент только читается, но не изменяется – передаем как ссылку на const

const Int operator + (const Int& right);

Int a(1),b(2),c(3);

Int d=a+b;(a+b).show()

а+b – временный объект всегда константный

1. Изменяется левосторонний аргумент

передаем как ссылку, такой аргумент может быть изменен

Int& operator += (const Int& right)

Int d+=a

1. Если оператор должен создавать новое значение, то нужно создавать новый объект

const Int operator + (int k, const Int& right)

{cout<<"op+"<<endl;

return Int(right.i+k);

}

возвращаемое значение – новый объект, т.к. этот оператор может генерировать временные объекты.

Возвращаемое значение константное, что запрещает изменять левый операнд.

1. Все операции присваивания изменяют левое значение, чтобы было возможно множественное присваивание

Int a(1),b(2),c(3);

a=b=c;

(a=c).print();

**Пример :**

class Byte

{unsigned char b;

public:

Byte (unsigned char bb='a'): b(bb){};

void print ()

{cout<<":"<<b<<endl;

}

// перегрузка операторов

const Byte& operator++ ()

{cout<<"++Byte"<<endl;

b++;

return \*this;

}

const Byte& operator-- ()

{cout<<"--Byte"<<endl;

b--;

return \*this;

}

const Byte operator+ (const Byte& right) const

{cout<<"Byte+"<<endl;

return Byte(b+right.b);

}

friend int operator>(const Byte& left, const Byte& right);

};//-------- конец определения Byte

int operator>(const Byte& left, const Byte& right)

{cout<<"Byte>"<<endl;

return left.b>right.b;

};

int \_tmain(int argc, \_TCHAR\* argv[])

{ Byte f;

f.print();

++f; f.print();

Byte d=f+10; d.print();

Byte e=(f+d+10);e.print();

if (f>d) f.print();

else d.print();

system("pause");

return 0;

}

**Перегрузка присваивания**

Вновь создаваемый объект проходит инициализацию (вызывается конструктор), в других случаях вызывается

operator =

Byte d;

Byte t=d; //работает конструктор копирования

t=d; //работает оператор присваивания

1. operator = может быть только функцией класса и неразравно связана с «левым» объектом
2. Глобальное переопределение operator = запрещено!

operator = обязательно должна быть функцией

класса

Иными словами область видимости – класс

class Numb

{ int a,b;

double c;

public:

// конструкторы

Numb (int aa=0, int bb=0, double cc=0):

a(aa),b(bb),c(cc){};

// перегрузка присваивания

Numb& operator= (const Numb& right)

{ a=right.a;

b=right.b;

c=right.c;

return \*this;

}

// перегрузка оператора вывода <<

friend ostream& operator<<

(ostream& os, const Numb& right)

{return os<<"a="<<right.a<<",b="<<right.b<<",c="<<right.c;

}

};

void main()

{ Numb q1, q2(1,5,6.45);

cout<<"q1:"<<q1<<endl;

cout<<"q2:"<<q2<<endl;

system("pause");

return 0;

}

.

**Контрольные вопросы**

1. Какие классы и функции называются дружественными?
2. Как осуществляется перегрузка операций?
3. Сколько аргументов требуется для определения перегруженной унарной (бинарной) операции?
4. Чем отличается действие перегруженной операции ++ при ее использовании в префиксной форме от использовании в постфиксной форме?
5. Какие ограничения существуют при перегрузке операторов?
6. Какую роль играет указатель this при перегрузке операторов?
7. Какую роль играет указатель this при вызове метода класса?
8. Приведите примеры повторных объявлений функций (ошибок перегрузки).

**Общие требования**

1. Для подготовки к лабораторной работе используйте данные из папок «Справочная информация» и «Теория»
2. Начать работу над новым проектом
3. В каждой базе должна быть статическая переменная для учета общего числа элементов в базе данных.
4. Данные размещаются в динамической памяти.
5. Для выполнения функций, указанных в задании, написать диалоговый интерфейс, позволяющий выполнять функции в произвольном порядке многократно
6. При выполнении функции «выход из программы» нужно сохранить базу на диске
7. Первичное создание базы – ввод данных с клавиатуры
8. Если программа уже запускалась, то данные загружаются из файла перед выходом на диалог. Иными словами вносятся изменения и дополнения в уже существующую базу данных.
9. Обязательные функции для всех вариантов:

* добавить новый элемент в базу
* распечатка данных в табличном виде
* выход из программы

1. Остальные функции для работы с базой указаны в задании индивидуально.
2. Примеры диалогового интерфейса и табличного вывода смотрите в лабораторной работе №1
3. Перегруженные операторы реализовывать как с помощью дружественной функции (ДФ), так и с помощью метода класса (МК). Если в задании не указан метод реализации – решаете по своему усмотрению.
4. В классе должны быть все необходимые элементы для работы тестовой программы.

|  |  |
| --- | --- |
| **Номер компьютера** | **Варианты заданий** |
| 1,11,  21 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных: Банк (депозит).  Создать класс **fio,** состоящий из 3-х текстовых строк (фамилия, имя и отчество)  Создать класс **bank** со следующими элементами:  **Элементы-данные класса:**   1. Характеристики депозита **(все данные pri)**:   Клиент: объект класса **fio**  Номер счета: целое число  Сумма на счете (сумма депозита ) : дробное число  2. Cчетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поиск по номеру счета  Сортировка по уменьшению суммы на счете  **Перегрузить следующие операторы:**  **<<**  вывод объекта класса **bank** во входной поток **cout**  **+** добавить число к полю «сумма на счете»  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Вывести базу в табличном виде (использовать перегруженный оператор **<<**) 6. Добавить число к полю «сумма на счете», использовать перегруженный оператор **+** 7. При выходе из программы запомнить измененную базу на диске |
| 2,12,  22  2,12,  22 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных:Поликлиника  Создать класс **fio,** состоящий из 3-х текстовых строк (фамилия, имя и отчество)  Создать класс **clinic** со следующими элементами:  **Элементы-данные класса(все данные pri)**:  Врач : объект класса **fio**  специальность(1-терапевт, 2-кардиолог, 3- стоматолог): перечисляемый тип  квалификация (0-высшая, 1- первая, 2- вторая): перечисляемый тип  2. Cчетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поиск врачей заданной специальности и квалификации (ДФ)  Сортировка о алфавиту  **Перегрузить следующие операторы:**  **>>**  ввод объекта класса **clinic** из входного потока **cin** (с клавиатуры)  **==** сравнение объекта типа **clinic** собъектом типа **fio**  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры с использованием перегруженного оператора **>>**) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Найти все записи в базе **clinic** сопределеннымобъектом типа **fio** 6. При выходе из программы запомнить измененную базу на диске |
| 3,13,  23 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных:Склад (товары)  Создать класс **date,** для работы в числовом формате дд : мм : гг  Создать класс  **storage** со следующими элементами:  **Элементы-данные класса:**   1. Характеристики изделия **(все данные pri)**:   Название товара : текстовая строка  Цена : дробное число  Количество : целое число  Номер секции : целое число   1. Cчетчик созданных объектов класса (СД)   **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поиск товара по номеру секции  Сортировка по убыванию количества  **Перегрузить следующие операторы:**  **> (сравнение)** объекта типа **storage** собъектом типа **date**  **=** (присваивание) для объектов **storage**  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Поиск товара с датой больше, чем заданная, использовать перегруженный оператор **> , з**апомнить все найденные данные в массиве, используя перегруженный оператор **=** 6. При выходе из программы запомнить измененную базу на диске |
| 4,14,  24 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных: **Поезд (заказ билетов)**  Создать класс **fio,** состоящий из 3-х текстовых строк (фамилия, имя и отчество)  Создать класс **ticket** со следующими элементами:  **Элементы-данные класса (все данные pri):**  - пассажир : объект класса **fio**  - пункт отправления : текстовая строка  - пункт прибытия : текстовая строка  - цена билета : дробное число  **Элементы-функции класса**   1. Создать обязательные функции, указанные в общих требованиях (МК). 2. Создать функции для данного варианта:   - Поиск билета по пункту назначения (МК)  - Сортировать по алфавиту (поле пункт отправления)  **Перегрузить следующие операторы:**  **=**  присваивание для объектов класса **ticket**  **>>**  ввод объекта класса **ticket** из входного потока **cin** (с клавиатуры)  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Выполнить все реализованные функции (поиск, сортировка) 4. Найти все объекты по пункту назначения, найти и запомнить в массиве, используя перегруженный оператор **=** 5. Добавить несколько новых элементов, ввод с клавиатуры (использовать перегруженный оператор **>>**) 6. При выходе из программы запомнить измененную базу на диске |
| 5,15,  25 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных:Работник (зарплата)  Создать класс **fio,** состоящий из 3-х текстовых строк (фамилия, имя и отчество)  Создать класс  **worker** со следующими элементами:  **Элементы-данные класса:**   1. Характеристики работника **(все данные pri)**:   Работник: объект класса **fio**  Номер отдела: целое число  Оклад : дробное число  2. Cчетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поикс по номеру отдела  Сортировка по возрастанию оклада  **Перегрузить следующие операторы:**  **==** сравнение объекта типа **worker** собъектом типа **fio**  **>>**  ввод объекта класса **worker** из входного потока **cin**  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (использовать перегруженный оператор **>>**) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Найти всех работников с с заданной фамилией, используя перегруженный операттор **==** 6. При выходе из программы запомнить измененную базу на диске |

|  |  |
| --- | --- |
| 6,16,  26 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных:База данных: Почта (ценное письмо).  Создать класс **fio,** состоящий из 3-х текстовых строк (фамилия, имя и отчество)  Создать класс  **letter** со следующими элементами:  **Элементы-данные класса:**   1. Характеристики письма **(все данные pri)**:   - адрес получателя: текстовая строка  - адрес отправителя: текстовая строка  - имя получателя : объект класса **fio**  - стоимость письма : дробное число  2. Cчетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поиск письма по отправителю (МК)  Сортировать по увеличению стоимости (ДФ)  **Перегрузить следующие операторы:**  **==** (проверка на равенство) дляобъекта класса **letter** иобъекта  типа **fio** (при сравнении учитывать все три поля)  **!** сортировать по алфавиту (по полю адрес отправителя)  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Найти все письма определенного получателя, используя перегруженный оператор **==** 6. Отсортироватьпо алфавиту (по полю адрес отправителя), используя перегруженный оператор ! 7. При выходе из программы запомнить измененную базу на диске |
| 7,17,27  7,17,27 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных:Производство (поставщики)  Создать класс **date,** для работы в формате дд(число) мм(строка) г(число)  Создать класс **supplier** со следующими элементами:  **Элементы-данные класса (все данные pri)**:  1. **Название фирмы** поставщика: текстовая строка  **юридический адрес** : текстовая строка  **тип поставщика**( 1- поставщик сырья, 2- поставщик оборудования):  перечисляемый тип  **сальдо** (положительное - должны поставщику, отрицательное-наоборот):  дробное число  2. Cчетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поиск поставщика по названию фирмы (ДФ)  Сортировка по уменьшению сальдо (МК)  **Перегрузить следующие операторы:**  **=** присваивание для объектов типа **supplier**  **>>**  ввод объекта класса **supplier** из входного потока **cin** (с клавиатуры)  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Найти данного поставщика (введите данные используя перегруженный оператор **>>**), запомнить все найденные данные в массиве, используйте используя перегруженный оператор **=** 6. При выходе из программы запомнить измененную базу на диске |
| 8,18,  28 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных:Студент  Создать класс **man,** состоящий из 2-х текстовых строк (фамилия и имя)  Создать класс **student** со следующими элементами:  **Элементы-данные класса(все данные pri)**:   1. Объект класса **man**   Факультет: строка текста  Номер группы: целое число  2. Cчетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поиск по факультету  Сортировка по возрастанию номера группы  **Перегрузить следующие операторы:**  **<<** вывод объекта класса student в выходной **поток**  **==** сравнение объекта типа **student** собъектом типа **man**  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Вывести базу в табличном виде (использовать перегруженный оператор **<<**) 6. Найти студента по имени и фамилии используя перегруженный оператор **==** 7. При выходе из программы запомнить измененную базу на диске |

|  |  |
| --- | --- |
| 9,19,  29 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных:Преподаватель  Создать класс **fio,** состоящий из 3-х текстовых строк (фамилия, имя и отчество)  Создать класс  **prepod** со следующими элементами:  **Элементы-данные класса:**   1. Характеристики преподавателя **(все данные pri)**:   Преподаватель: объект класса **fio**  кафедра (1-ИПОВС, 2-ВМ, 3-ВТ) : перечисляемый тип  ученое звание ( 1- преподаватель, 2- доцент,3-доктор наук):  перечисляемый тип  предмет: текстовая строка  2. Cчетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:  Поиск преподавателя по заданному предмету  Поиск преподавателей по заданной кафедре  **Перегрузить следующие операторы:**  **<<** вывод объекта класса **prepod** в выходной **поток**  **==** сравнение объекта типа **prepod**  с объектом типа **fio**  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры) 4. Выполнить все реализованные функции (поиск, сортировка) 5. Вывести базу в табличном виде (использовать перегруженный оператор **<<**) 6. Найти преподавателя с заданным fio, использовать перегруженный оператор == 7. При выходе из программы запомнить измененную базу на диске |
| 10,20,  30  10,20,  30 | **При выполнении задания используйте общие требования**  Создайте проект новой базы данных: **Пищевое производство (расчеты).**  Создать класс **date,** для работы в числовом формате дд : мм : гг  Создать класс **food** со следующими элементами:  **Элементы-данные класса:**  1. Характеристики изделия **(все данные pri)**:  - название – текстовая строка  - тип (1- булочка, 2-пирожок, 3-пирожное) – перечисляемый тип  - вес – дробное число  - количество – целое число  - стоимость – дробное число  - дата изготовления – объект класса **date**  2. счетчик созданных объектов класса (СД)  **Элементы-функции класса**  1. Создать обязательные функции, указанные в общих требованиях (МК).  2. Создать функции для данного варианта:   * Поиск изделий по названию (МК) * Сортировать по уменьшению стоимости (сначала более дорогие) (ДФ)   **Перегрузить следующие операторы:**  **<<** вывод объекта класса **food** в выходной поток **cout** (на терминал)  **==** оператор сравнения (на равенство) для объекта класса **food и** объекта  типа **date** (при сравнении учитывать все три поля даты)  **Тестовая программа (функция main):**   1. Базу данных (массив объектов) располагать в динамической памяти 2. При запуске программы данные ввести с клавиатуры (первый запуск программы) или загрузить с диска (все последующие запуски) 3. Добавить несколько новых элементов (ввод с клавиатуры) 4. Вывести базу в табличном виде (использовать перегруженный оператор **<<**) 5. Выполнить все реализованные функции (поиск, сортировка) 6. Найти все изделия с определенной датой выпуска, дату ввести с клавиатуры (использовать перегруженный оператор ==) 7. При выходе из программы запомнить измененную базу на диске |