КОНТРОЛЬНАЯ РАБОТА

Задача №1. ВВЕДЕНИЕ В ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ ПРОГРАММИРОВАНИЕ. КЛАСС КАК РАСШИРЕНИЕ ПОНЯТИЯ СТРУКТУРЫ

Цель работы: Ознакомиться с основополагающими концепциями объектно-ориентированного программирования. Произвести сравнение двух основных подходов к проектированию программ: структурного программирования и объектно-ориентированного. Изучить построение класса на основе построения структур. Научиться создавать простейшие программы с использованием классов.

Теоретические сведения:

Структура как простейший класс

Для реализации основополагающих концепций в языке C++ используются классы. Тип C++ class является расширением типа C struct. Во многом структура в C++ напоминает простейший вид класса. Для ее определения используется ключевое слово struct.

В этом примере имеется описание структуры с одним элементом данных data_value и функцией-членом, которая возвращает значение синуса угла.

```
//1TSTRUC.CPP
//Данная программа использует структуру с одним
//элементом данных data_value и функцией-членом,
// которая возвращает значение синуса угла.
//#include "stdafx.h"
#include <iostream>
#include <math.h>
using namespace std;

const double DEG_TO_RAD=0.0174532925;
struct degree
```

```
{
 doubledata_value;
 voidset_value(double);
 double get_sine(void);

} deg;

void degree::set_value(double ang)
{
 data_value=ang;
}
double degree::get_sine(void)
{
 double answer;
 answer=sin(DEG_TO_RAD*data_value);
 return(answer);
}
int main()
{
 deg.set_value(25.0);
 cout<< "The sine of the angle 25.0 is: "
      <<deg.get_sine() <<endl;
return(0);
}
```

Заметим, что определение структуры содержит прототипы функций-членов. Переменная deg связана со структурным типом degree.

Сразу же за описанием структуры создаются и перечисляются различные функции-члены, которые связываются со структурой посредством операции расширения доступа к элементам структуры (scopeoperator ::). За исключением этой операции функции-члены ничем не отличаются от обычных функций.

Рассмотрим начало функции main():

```
 deg.set_value(25.0);
```

В этом операторе значение 25.0 передается как аргумент функций set_value.

Функция получает аргумент и присваивает его переменной класса data_value. Это один из способов инициализации переменных структуры. После этого момента переменная data_value становится доступной и к ней можно обращаться при помощи любой из шести функций-членов, которые вычисляют значения синуса.

Для доступа к функциям-членам структуры обычно используется операция "точка" (.). Структуре или классу можно назначить переменные-указатели — тогда используется операция "стрелка" (->).

Синтаксис и правила для классов C++

Описание класса начинается с ключевого слова class, за которым следует имя класса (тег). В следующем примере теговое имя класса — car (автомобиль). Если класс содержит переменные-члены, то они определяются в начале класса. По умолчанию они объявляются как private. В примере описываются три переменных-члена: mileage (пробег), tire_pressure (давление в шинах) и speed (скорость). За списком переменных следуют функции-члены (методы) класса. Обычно функции-члены объявляются как public. Объявление private делает переменные-члены видимыми только для функций-членов данного и класса. Часто это явление называется сокрытием данных (datahiding). Объявление public обеспечивает доступность функций-членов за пределами класса и к ним можно обращаться из внешних для класса процедур. Все функции-члены класса имеют доступ к любым разделам этого класса: public, private и protected:

```
class car {
    int mileage;
    inttire_pressure;
```

```
    float speed;
 public:
    intmaintenace(int) ;
    intwear_record(int);
    intair_resistance(float);
    } mycar;
```

Простой класс C++

По умолчанию элементы класса C++ имеют локальную видимость. Это означает, что переменные-члены доступны только функциям-членам этого класса. Если функции-члены должны иметь видимость за пределами класса, то это нужно указать явно.

Преобразовать структуру из последнего примера в истинный класс C++ очень просто. Во-первых, ключевое слово structзаменяется на словоclass. Во-вторых, функции-члены, которые должны иметь глобальную видимость, отделяются от частной переменной класса при помощи объявления public. Рассмотрим законченную программу:

```
//2TCLASS.CPP
//Данная программа выполняет задачу программы 1TSTRUC.CPP с
 // преобразованием структуры в класс
//#include "stdafx.h"

#include <iostream>
#include <math.h>
using namespace std;

const double DEG_TO_RAD=0.0174532925;

class degree
{
 doubledata_value;
public:
 voidset_value(double);
 double get_sine(void);
```

```
} deg;

void degree::set_value(double ang)
{
 data_value=ang;
}

double degree::get_sine(void)
{
 double answer;
 answer=sin(DEG_TO_RAD*data_value);
 return(answer);
}
 int main()
{
 deg.set_value(25.0);
 cout<< "The sine of the angle 25.0 is: "
      <<deg.get_sine() <<endl;
 return(0);
}
```

В этом примере тело программы осталось прежним. Описание структуры преобразовано в описание полноценного, хотя и простого, класса с разделами private и public.

Заметим, что переменная data_value является частной (private) для класса (по умолчанию) и результат можно получить только при помощи методов класса. Сами методы имеют глобальную видимость (public) и доступны извне класса. Однако, любой элемент класса — как частный, так и общий — имеет доступ ко всем остальным элементам класса.

Этот класс имеет тип (имя тега) degree. Локальная переменная data_value хранит значения углов, доступные для различных функций-членов. Функция get_sine() реализует метод класса. С данным классом связано имя deg. В отличие от этого примера, имена переменной и класса чаще всего связываются в функции main().

Разве описание класса не кажется знакомым? По сути, это — определение структуры из предыдущего примера, преобразованное в истинное описание класса.

Повторим, функции-члены класса обычно описываются в программе непосредственно после объявления класса и перед функцией main(). Функции, не относящиеся к классам, по-прежнему описываются после функции main() и имеют прототипы обычного вида.

Контрольные вопросы:

1. Произведите сравнение двух основных подходов к проектированию программ: структурного программирования и объектно-ориентированного.
2. Как определить структуру?
3. Как определить класс?
4. Чем отличается определение класса от определения структуры?
5. Расскажите о двух способах задания объекта класса?
6. Как получить доступ к элементам данных класса?
7. Как описать функцию-член класса?

Задания:

1. Продолжить программу, определяющую синус, косинус, тангенс, котангенс заданного угла с помощью классов. Выведите данные в табличном виде с помощью соответствующего метода класса.

2. Создать структуру, которая будет содержать данные о студентах института (фамилия, имя, группа, средний балл). С помощью функции-члена класса необходимо вывести на экран фамилию, имя студента и номер группы. Создать аналогичный класс.

3. Создать структуру, которая будет содержать данные о книге (название, издательство, автор, количество страниц). С

помощью функции-члена класса необходимо вывести на экран название, издательство и автора книги. Создать аналогичный класс.

4. Написать программу, вычисляющую с помощью функции-члена класса средний балл студента за сессию. Количество экзаменов является величиной переменной изадается пользователем. Список студентов обрабатывается как класс.

5. Написать программу, вычисляющую с помощью функции-члена класса суммарную величину продаж книжного магазина. Количество продаж является величиной переменной и задается пользователем. Список всех продаж, состоящий из товара и цены, обрабатывается как класс.

6. Создать класс оперирующий информацией о сотруднике фирмы, и содержащий имя, фамилию, отчество, табельный номер, количество отработанных часов за месяц, почасовой тариф. Рабочее время свыше 144 часов считается сверхурочным и оплачивается в двойном размере. С помощью функции-члена класса вывести размер заработной платы каждого сотрудника фирмы за вычетом подоходного налога, который составляет 12% от суммы заработка.

7. Создать класс для хранения информации о студентах института (фамилия, имя, группа, средний балл). Необходимо вывести на экран фамилию, имя студента и средний балл студентов одной группы, отсортированных по алфавиту.

8. На склад игрушек хранятся игрушки. Игрушки имеют название, артикул, имя фирмы, цену и количество дней хранения на складе. Создать класс для хранения вышеперечисленной информации с функцией-членом, которая выводит на печать список игрушек (название игрушки, артикул, имя фирмы, цену – 20% и количество дней хранения на складе), хранящиеся на складе больше 20 дней. Список должен быть отсортирован в порядке уменьшения количества дней хранения на складе.

9. Составить описание класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров с помощью методов класса.

10. Составить описание класса треугольниковв декартовой системе координат. Предусмотреть возможность вращение

треугольника вокруг начала координат на заданный угол с помощью методов класса.

Задача №2

КОНСТРУКТОРЫ И ДЕСТРУКТОРЫ

Цель работы: Ознакомиться и на практике научиться работать сконструкторами и деструкторами, освоить их использование для инициализации переменных-членов класса, выделения и освобождения свободной памяти.

Теоретические сведения:

Конструктор (constructor) — это метод класса. Конструкторы используются для инициализации переменных класса или для выделения памяти. Они всегда имеют такое же имя, как и класс, в котором они определены. Конструкторы располагают дополнительными возможностями: они могут иметь аргументы и их можно перегружать. Конструктор выполняется автоматически при создании объекта типа class. Объекты свободной памяти — это такие объекты, которые генерирует операция new, они служат для выделения памяти создаваемым объектам. Если конструкторы не определены явно, то они генерируются компилятором MicrosoftVisual C/C++.

Деструктор (destructor) — это метод класса, использующийся обычно для освобождения памяти, выделенной из пула свободной памяти. Деструктор — так же как и конструктор — имеет такое же имя, как и класс, в котором он описан; ему предшествует символ "тильда" (~). Деструкторы являются дополнениями конструкторов. Деструктор вызывается автоматически при использовании операции delete по отношению к указателю класса или в случае, когда программа выходит из области действия объекта класса. Деструкторы, в отличие от конструкторов, не могут иметь параметров и их нельзя

перегружать. Они также создаются компилятором MicrosoftVisual C/C++, если не определены явно.

Создание простых конструкторов и деструкторов

В первом примере конструктор и деструктор используются для оповещения о начале и окончании программы преобразования денежных единиц. Эта программа иллюстрирует автоматический вызов конструкторов и деструкторов:

```
// 3COINS.CPP
// программа,
// иллюстрирующая использование конструкторов и деструкторов
// Эта программа пересчитывает центы в эквивалентную сумму,
// состоящую из монет 25центов

//#include "stdafx.h"
#include <iostream>
#include <tchar.h>

using namespace std;

constint QUARTER= 25;

class coins
{
 int number;
public:
 coins() {cout<< "Begin Conversion!\n";} // конструктор - началопреобразования
 ~coins() {cout<< "\nFinished Conversion!\n";} // деструктор - окончание преобразования
 voidget_cents(int);
 intquarter_conversion(void);

 };

void coins::get_cents(int cents)
{
```

```
    number= cents;
    cout<< number << " cents, converts to: "  <<endl;
// … центовпересчитываютсяв …
   }
   int coins::quarter_conversion()
   {
    cout<<   number/QUARTER   <<   "   quarter(s),   ";
//монетыв 25 центов
    return(number%QUARTER);
   }
   int _tmain(intargc, _TCHAR* argv[])
   {
    intc,d;
    cout<< "Enter the cash, in cents, to convert:  ";
// введитесуммудляпересчетавцентах
    cin>> c;
    coinscash_in_cents;
    cash_in_cents.get_cents(c);
    d= cash_in_cents.quarter_conversion();
    cout<<    d    <<    "quarter_conversion.";    //
монетыводинцент
    return 0;
   }
```

Обратите особое внимание на то, где в описании класса располагаются конструктор и деструктор. Функции конструктора и деструктора содержат только сообщение, выводимое на экран. Программы непосредственно не вызывают конструкторы. Их появление на экране свидетельствует о том, что конструктор и деструктор автоматически вызывались при создании и удалении объекта.

В данном примере для конструктора и деструктора имеются описанные функции. Когда методы класса включены в функции, такое описание называется явным. Методы можно определять обычным образом (то есть после определения класса и перед функцией main()) или объявлять явно через встроенные функции.

Использование конструкторов и деструкторов для выделения и освобождения свободной памяти

Возможно, в первую очередь конструкторы необходимы при использовании свободной памяти. В следующем примере конструктор выделяет память для указателя string1 при помощи операции new. Также используется деструктор, который возвращает системе выделенную память при уничтожении объекта. Этовыполняетсяпосредствомоперации delete.

```
    classstring_operation
    {
        char *string1;
        intstring_len;
    public:
        string_operation
{string1=new char[string_len];}
        ~ string_operation() {delete
        string1;} void input_data(char *);
        voidoutput_data(char *);
    };
```

Память, выделяемую указателю string1 при помощи операции new, можно освободить только посредством последующего вызова операции delete. Поэтому память для указателей обычно выделяют конструкторы, а освобождают — деструкторы. Это гарантирует, что если переменная, связанная с классом, выходит из области действия, то занимаемая память возвращается системе. Эти операции обеспечивают динамическое распределение памяти и наиболее удобны в программах, работающих со связанными списками.

Память, занимаемая такими типами данных, как int и float, автоматически возвращается системе.

Контрольные вопросы:

1. Для каких целей используются конструктор и деструктор?
2. Почему деструктор не передает параметры?
3. Какими дополнительными возможностями располагают конструкторы?
4. Указывается ли конструктору тип возвращаемого значения? Поясните ту или иную точку зрения.
5. Имеет ли параметры деструктор? Возвращает ли он значения? Поясните ту или иную точку зрения.
6. Может ли быть несколько деструкторов?
7. Как ведет себя компилятор программы, если конструктор не определен в классе?
8. Расскажите синтаксис использования конструктора и деструктора.

Задания:

1. Напишите программу, которая пересчитывает центы в эквивалентную сумму, состоящую из монет различного достоинства (25, 10, 5 центов и 1 центов). Экземпляры классов необходимо создавать динамически.
2. Написать программу, которая создает и инициализирует 2 объекта разработанного класса, вычисляет и выводит значения площадей основания, боковой и полной поверхностей. Разработать класс Parallel, который должен содержать закрытые переменные (private), a, b – стороны основания прямоугольника, H – высота параллелепипеда. Класс должен содержать конструктор инициализирующий указанные переменные, а также методы, вычисляющие и возвращающие значения площадей основания getSo(), боковой getSb() и полной getSp().
3. Написать программу, которая создает и инициализирует 2 объекта разработанного класса, вычисляет и выводит значения площадей основания, боковой и полной

поверхностей. Для создания и выполнения второго объекта используйте указатель. Разработать класс Prisma (прямая призма, в основании – правильный шестиугольник, вписанный в окружность), который должен содержать закрытые переменные (private), R – радиус описанной окружности, H – высота призмы. Класс должен содержать конструкториницализирующий указанные переменные, а также методы, вычисляющие и возвращающие значения площадей основания getSo(), боковой getSb() и полной getSp().

4. Написать программу, которая создает и инициализирует 2 объекта класса прямоугольников со сторонами, параллельными осям координат. Предусмотреть возможность перемещения прямоугольников на плоскости, изменение размеров, расчет площади и периметра с помощью методов класса.Использование конструкторов и деструкторов обязательно. Экземпляры классов необходимо создавать динамически.

5. Написать программу, которая создает и инициализирует 2 объекта класса треугольников в декартовой системе координат. Предусмотреть возможность вращение треугольника вокруг начала координат на заданный угол, расчет площади и периметра с помощью методов класса.Использование конструкторов и деструкторов обязательно. Экземпляры классов необходимо создавать динамически.

6. Определить оптимальный подбор банкнот для выдачи задаваемой суммы в рублях для банкомата (купюры -1000, 5000, 10000, 20000, 50000). Использование конструкторов и деструкторов обязательно. Экземпляры классов необходимо создавать динамически.

7. Написать программу, которая создает и инициализирует 2 объекта класса, содержащего информацию о почтовом адресе организации, включающем индекс, страну, город, улицу, дом. Предусмотреть возможность раздельного изменения составных частей адреса, создания и уничтожения объектов этого класса. Использование конструкторов и деструкторов обязательно. Экземпляры классов необходимо

создавать динамически.

8. Составить описание класса для представления комплексных чисел с возможностью задания вещественной и мнимой частей как числами типов double, так и целыми числами. Обеспечить выполнение операций сложения, вычитания и умножения комплексных чисел. Использование конструкторов и деструкторов обязательно.

9. Составить описание класса для объектов-векторов, задаваемых координатами концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами. Использование конструкторов и деструкторов обязательно.

10. Составить описание класса для определения одномерных массивов целых чисел (векторов). Предусмотреть возможность обращения к отдельному элементу массива с контролем выхода за пределы индексов, возможность задания произвольных границ индексов при создании объекта и выполнения операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов, умножения и деления всех элементов массива на скаляр, печати (вывода на экран) элементов массива по индексам и всего массива.Использование конструкторов и деструкторов обязательно.

Задача №3

ПЕРЕГРУЗКА МЕТОДОВ КЛАССА

Цель работы: Ознакомиться и освоить на практикеиспользование перегрузки методов класса, перегруженные операции и вызовы функций.

Теоретические сведения:

Методы классов, так же как и обычные функции C++, можно перегружать. Перегрузка функций означает, что в текущей области действия одно и то же имя могут использовать несколько функций. Компилятор выбирает нужную функцию, учитывая количество и тип аргументов, использованных при ее вызове. Первый пример в этом разделе иллюстрирует перегрузку метода, названного number (). Перегружаемая функция возвращает абсолютное значение чисел типа int или double, используя математическую функцию abs(), принимающую и возвращающую значения int, и функцию fabs(), которая принимает и возвращает значения типа double. При перегрузке функции тип аргумента определяет, какой метод класса на самом деле используется.

```
// 4ABSOL.CPP
// Программа на C++, иллюстрирующая перегрузку
методов класса.
// Программа определяет абсолютное значение
чисел типа int и double

//#include "stdafx.h"
#include <iostream>
#include <tchar.h>
#include <math.h>
using namespace std;
```

```
classabsolute_value
{
public:
     int number(int);
     double number(double);
};

intabsolute_value::number(inttest_data)
{
     int answer;
     answer=abs(test_data);
     return(answer);
}

doubleabsolute_value::number(double test_data)
{
     double answer;
     answer=fabs(test_data);
     return(answer);
}

int _tmain(intargc, _TCHAR* argv[])

{
     absolute_valueneg_number;
     cout<< "The absolute value is "
          << neg_number.number(-583) <<endl; //
Абсолютное значение равно...
     cout<< "The absolute value is "
          << neg_number.number(-583.1749)  <<endl;
     return 0;
}
```

Перегрузка операций

Как было указано ранее, можно перегружать методы класса. В данном разделе вы узнаете, как перегружать операции C++.

В языке C++ можно переопределить в некотором классе такие привычные операции, как +, -, * и /. Главное ограничение для перегрузки операции состоит в том, что синтаксис и приоритет операции не должны изменяться по сравнению с первоначально

определенными. Другой важный момент — перегрузка операции возможна только в области действия того класса, в котором она выполняется.

Синтаксис перегрузки операции

Для перегрузки операции используется ключевое слово operator, за которым следует сама операция:

тип operator операция (список параметров) Например:

```
angle_value operator +(angle_argument);
```

В пределах области действия соответствующим образом описанного класса можно непосредственно складывать значения углов, выраженные в градусах, минутах и секундах:

```
angle_valueangle1("37° 15' 56\"");
angle_value angle2("10° 44' 44\"");
angle_valuesum_of_angles;
sum_of_angles =angle1+ angle2;
```

Как уже упоминалось в предыдущих примерах, для обозначения угловых секунд используется символ двойных кавычек ("). Этот же символ означает начало и конец символьной строки.

Контрольные вопросы:

1. Для каких целей предназначена перегрузка? Приведите пример.
2. Почему перегрузка позволяет упростить код программы?
3. Расскажите синтаксис перегруженных функций в программе.
4. Какое удобство заключается в использовании перегруженных методов класса?
5. Какие стандартные операции в C++ допускается перегружать? Приведите пример перегрузки той или иной операции.

Задания:

1. Написать программу, которая использует перегрузку методов класса: для целых данных вычисляется куб числа, для действительных (с плавающей точкой) – синус угла.

2. Составить описание класса для объектов-векторов, задаваемых координатами концов в трехмерном пространстве. Обеспечить операции сложения и вычитания векторов с получением нового вектора (суммы или разности), вычисления скалярного произведения двух векторов, длины вектора, косинуса угла между векторами. Операции реализовать через перегрузку операторов.

3. Составить описание класса для определения одномерных массивов целых чисел (векторов). Предусмотреть возможность обращения к отдельному элементу массива с контролем выхода за пределы индексов, возможность задания произвольных границ индексов при создании объекта и выполнения операций поэлементного сложения и вычитания массивов с одинаковыми границами индексов, умножения и деления всех элементов массива на скаляр, печати (вывода на экран) элементов массива по индексам и всего массива. Использование конструкторов и деструкторов обязательно. Операции реализовать через перегрузку операторов.

4. Написать программу, выполняющую перегрузку операций для подготовки рецептов, если вес задается в килограммах, граммах, миллиграммах.

5. Написать программу, иллюстрирующуя перегрузку двух методов класса, которая позволяет вводить значение угла в десятичном формате или в формате "градусы/минуты/ секунды". Один метод класса получает данные типа double, а другой – типа строка. Программа возвращает значения синуса, косинуса и тангенса и котангенса данных углов.

6. Программа вычисляет и выводит на экран сумму двух углов в десятичном формате, а потом в формате "градусы/минуты/ секунды".

7. Написать свой класс, реализующий работу с комплексными числами. Операции сложение, вычитание, деление и умножение должны быть реализованы при помощи перегрузки соответствующих операторов.

8. Написать свой класс, реализующий матричную арифметику. Операции сложения и умножения матриц должны быть реализованы через перегрузку соответствующих операторов.

9. Написать свой класс, реализующий матричную арифметику. Операции сложения и умножения матриц, а также сложения матрицы с числом и умножения матрицы на число должны быть реализованы в виде перегруженных методов класса.

10. Составить описание класса многочленов от одной переменной, задаваемых степенью многочлена и массивом коэффициентов. Предусмотреть методы для вычисления значения многочлена для заданного аргумента, операции сложения, вычитания и умножения многочленов с получением нового объекта-многочлена, печать (вывод на экран) описания многочлена. Операции сложения, вычитания и умножения должны быть реализованы через перегрузку соответствующих операторов.

ЛИТЕРАТУРА

1. *Глушаков С.В., Коваль А.В., Смирнов С.В.* Учебный курс"Языкпрограммирования C++". – М.: Издательство "Фолио", 2001
2. *Дал У., Дейкстра Э., Хоор К.* Структурное программирование/Пер. с англ. Мир, – М.: 1975.
3. *Касаткин А.И.* Профессиональное программирование на языкеСи. Системное программирование. – Мн.: Выш. школа, 1992.
4. *Касаткин А.И.* Профессиональное программирование на языкеСи. Управление ресурсами: Справочное пособие. – Мн.: Выш.

   школа, 1992.
5. *Касаткин А.И., Вальвачев А.Н.* Профессиональноепрограммирование на языке Си. От Turbo C к Borland C++: Справочное пособие. – Мн.: Выш. школа, 1992.
6. *Культин Н.* C/C++в задачах и примерах. –Санкт-Петербург:БХВ- Петербург, 2004.
7. *Подбельский В.В.* Язык Си++:учебное пособие.- 5-е изд.- М.:Финасы и статистика, 2001.
8. *Шиманович Е.Л.* C/C++в примерах и задачах. – М.:Новоезнание, 2004.