```
            params[1] = rand() % 20 + 5;
            fs[i]->set_parameters(params);
        }
        for (int i = 0; i < N; i++)
            printf("%s. S = %.0lf\n", fs[i]->get_figure_name_and_params(),
                                      fs[i]->calculate_square());
}
```

4. Лабораторная работа «Параметризованные классы»

Ключевые понятия: полиморфизм, параметрический полиморфизм, параметризованный класс

Цель работы: изучить механизм параметрического полиморфизма на основе создания и использования параметризованных классов.

Задание кафедры: реализовать на языке C++ параметризованный класс «Матрица», типы элементов которого могут быть заданы в соответствии с вариантом (табл. 4).

Таблица 4

Типы данных для матриц и возможные действия над матрицами

Вариант	Тип 1	Тип 2	Возможность класса
1	Комплексное число	Вектор в полярных координатах	Сложение двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A+B
2			Разность двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A-B
3			Умножение матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A*b, где b – вещественное число
4			Умножение вещественного числа на матрицу и присвоение результата третьей переменной-матрице осуществляется одной строкой C=b*A, где b – вещественное число
5			Деление матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A/b, где b – вещественное число

Продолжение табл. 4

Вариант	Тип 1	Тип 2	Возможность класса
6	Множество	Строка (массив символов)	Сложение двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A+B
7			Разность двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A-B
8			Сложение матрицы с символом и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A+b, где b – символ
9			Сложение символа с матрицей и присвоение результата третьей переменной-матрице осуществляется одной строкой C=b+A, где b – символ
10	Вектор в декартовой системе N координат	Вектор в полярных координатах	Сложение двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A+B
11			Разность двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A-B
12			Умножение матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A*b, где b – вещественное число
13			Умножение вещественного числа на матрицу и присвоение результата третьей переменной-матрице осуществляется одной строкой C=b*A, где b – вещественное число
14			Деление матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A/b, где b – вещественное число
15	Комплексное число	Интервальное число	Сложение двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A+B
16			Разность двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A-B
17			Умножение матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A*b, где b – вещественное число
18			Умножение вещественного числа на матрицу и присвоение результата третьей переменной-матрице осуществляется одной строкой C=b*A, где b – вещественное число
19			Деление матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A/b, где b – вещественное число

Окончание табл. 4

Вариант	Тип 1	Тип 2	Возможность класса
20	Вектор в декартовой системе N координат	Комплексное число	Сложение двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A+B
21			Разность двух матриц и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A-B
22			Умножение матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A*b, где b – вещественное число
23			Умножение вещественного числа на матрицу и присвоение результата третьей переменной-матрице осуществляется одной строкой C=b*A, где b – вещественное число
24			Деление матрицы на вещественное число и присвоение результата третьей переменной-матрице осуществляется одной строкой C=A/b, где b – вещественное число

Содержание отчета: сведения об авторе и работе, задание кафедры, цель работы, текст программы (указать в комментариях: назначение всех элементов параметрического полиморфизма, назначение всех методов и перегруженных операторов, необходимых для реализации параметрического полиморфизма), выводы.

Контрольные вопросы:

1. Зачем нужны параметризованные классы?

2. Какой из механизмов ООП реализуется с использованием параметризованных классов?

3. Что такое шаблон класса?

4. Во сколько раз сокращается исполняемый машинный код программы при использовании параметризованных классов?

5. Какого вида могут быть параметры при задании шаблона класса?

Пример программы

```
//класс вещество, содержащий температуру и массу
class Substance {
        double Mass;
        double Temperature;
public:
//требуется перегрузка оператора сложения для класса Substance, так как данная операция совершается в классе Matrix. Без данной перегрузки компилятор выдаст ошибку
        Substance operator + (Substance c) {
                Substance res;
                res.Mass = this->Mass + c.Mass;
                res.Temperature = (this->Temperature*this->Mass +
                                        c.Temperature*c.Mass) / (this->Mass + c.Mass);
                return res;
        }
//так как метод rand_val вызывается из класса Matrix, то данный метод должен обязательно присутствовать в классе Substance
        static Substance rand_val() {
                Substance res;
                res.Mass = (rand() % 2000) / 100.0 + 5;
                res.Temperature = (rand() % 2000) / 100.0 + 10;
                return res;
        }
//так как метод to_str вызывается из класса Matrix, то данный метод должен обязательно присутствовать в классе Substance
        void to_str(char* bufer) {
                sprintf(bufer, "M=%.2lf;T=%.2lf", Mass, Temperature);
        }
};

//определение шаблонного класса Matrix, который позволяет хранить матрицу из объектов, которые имеются тип T
template <class T>
class Matrix {
//определение массива элементов типа T
        T m[4][4];
```

```
public:
    Matrix() {
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
                m[i][j] = T::rand_val();
    }
```
//запись T::rand_val() в шаблонном классе предполагает наличие у класса T публичного статического метода rand_val. Программа не может быть собрана, если у класса T нет данного метода.
```
    void print() {
        char *bufer = new char[100];
        printf("Matrix\n");
        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 4; j++) {
                m[i][j].to_str(bufer);
                printf("%s\t", bufer);
            }
            printf("\n");
        }
        printf("\n");
        delete bufer;
    }
```
//запись m[i][j].to_str(bufer) для элемента (он имеет тип T) массива m предполагает наличие метода to_str у класса T. Программа не может быть собрана, если у класса T нет данного метода.
```
    Matrix operator + (Matrix &b) {
        Matrix<T> res;
        for (int i = 0; i < 4; i++)
            for (int j = 0; j < 4; j++)
                res.m[i][j] = this->m[i][j] + b.m[i][j];
        return res;
    }
```
//операция сложения между экземплярами класса T (это - this->m[i][j] и b.m[i][j]) предполагает наличие у класса T перегруженного оператора сложения. Программа не может быть собрана, если у класса T нет данного перегруженного оператора.
```
};
```

```
//класс Substance имеет статический метод rand_val(), метод to_str() и перегруженный оператор сложения, поэтому он может быть использован в качестве параметра шаблонного класса Matrix.
void main() {
    Matrix<Substance> m1;
    m1.print();
    Matrix<Substance> m2;
    m2.print();
    Matrix<Substance> m3 = m1 + m2;
    m3.print();
}
```

5. Лабораторная работа «Диаграмма классов»

Ключевые понятия: сущность, отношение между сущностями, ассоциация, агрегация, композиция, наследование, зависимость, диаграмма классов.

Цель работы: научиться формализовывать предметную область в виде классовой диаграммы.

Задание кафедры: для заданной предметной области (табл. 5) придумать возможную задачу, которую можно автоматизировать созданием информационной системы. Необходимо выделить сущности предметной области, которые можно описать классами. Каждую сущность следует описать ее свойствами и методами. Необходимо построить диаграмму классов с указанием их свойств и методов, а также связей между классами. На итоговой диаграмме классов (не менее 10 классов) должны быть представлены все основные виды связей (ассоциация, агрегация, наследование, композиция, зависимость). Необходимо добавить краткое описание каждого класса, каждого свойства/метода класса и каждой связи в диаграмме.

Таблица 5

Исследуемые предметные области

Вариант	Предметная область	Вариант	Предметная область
1	Животные	10	Ландшафт
2	Растения	11	Мебель
3	Автомобили	12	Страны
4	Бытовая техника	13	Ювелирные изделия
5	Строения	14	Морские суда
6	Продовольственные товары	15	Видеоигры
7	Музыкальные инструменты	16	Географические карты
8	Условные знаки	17	Одежда
9	Обувь	18	Книги

Содержание отчета: сведения об авторе и работе; задание кафедры; цель работы; описание задачи, поставленной для информационной системы; диаграмма классов; описание сущностей диаграммы классов (с их свойствами и методами); описание связей между сущностями на диаграмме классов; выводы.

Контрольные вопросы:

1. Какие бывают отношения между классами?
2. Как обозначаются отношения на диаграмме классов?
3. Что обозначает отношение «ассоциация»? Приведите пример.
4. Что обозначает отношение «агрегация»? Приведите пример.
5. Что обозначает отношение «композиция»? Приведите пример.
6. Чем отличается отношение «агрегация» от отношения «композиция»?
7. Что обозначает отношение «зависимость»? Приведите пример.

6. Лабораторная работа «Сериализация в xml файл»

Ключевые понятия: множественное наследование, иерархия классов, формат xml файла, сериализация.

Цель работы: научиться использовать механизм сериализации в формат xml для сохранения данных в структурированном виде.

Задание кафедры: реализовать на языке C++ ведение справочников для заданной предметной области (табл. 5): создание, удаление, редактирование объектов. Необходимо реализовать возможность сохранения в файл / загрузку из файла данных с использованием механизма сериализации (использовать формат xml). Реализовать иерархию классов минимум из трех уровней с множественным наследованием. Операции сериализации осуществляться с классом, агрегирующим данные.

Содержание отчета: сведения об авторе и работе, задание кафедры, цель работы, текст программы, сохраненные данные в формате xml, выводы.

Контрольные вопросы:

1 Что такое иерархия классов?

2. Что представляет собой xml файл?

3. Что такое сериализация? Для каких целей она может быть использована?

4. Для чего нужно множественное наследование?

5. Какой порядок вызовов конструкторов и деструкторов при множественном наследовании?

7. Лабораторная работа «Обработка собственных событий»

Ключевые понятия: делегат, событие, обработчик события

Цель работы: научиться создавать и использовать собственные события для возможности отслеживания другими классами наступления определенных условий в текущем классе

Задание кафедры: реализовать на языке C++ программу, требования для которой представлены в табл. 6, с использованием механизма событий.

Таблица 6

Задачи для реализации механизма событий

Вариант	Описание задачи	События
1	Программа-редактор групп студентов. Имеются классы: группа, студент. Программа должна предоставлять возможность редактировать списки групп и студентов в группах в рамках одной кафедры. Необходим класс, генерирующий сообщения по событиям. Необходим класс для сохранения сообщений в файл о сработавших событиях	1) количество студентов в группе X = 0 (после удаления студента из группы), 2) был удален студент из группы X с фамилией Y
2	Программа-редактор групп студентов. Имеются классы: группа, студент. Программа должна предоставлять возможность редактировать списки групп и студентов в группах в рамках одной кафедры. Необходим класс, генерирующий сообщения по событиям. Необходим класс для сохранения сообщений в файл о сработавших событиях	1) создание студента 2) удаление студента 3) редактирование студента
3	Программа-эмулятор игры. Имеются классы: комната, игрок. Игроки находятся по два в комнате, обладают некоторым запасом здоровья и по очереди наносят друг другу случайной степени урон. Победа у того, кто первым исчерпал запас здоровья противника. Необходим класс, собирающий статистику по победам из всех комнат, а также класс для сохранения сообщений в файл о сработавших событиях	1) победа в игре, 2) нанесение урона за один ход больше заданного значения
4	Программа-редактор групп студентов. Имеются классы: группа, студент. Программа должна предоставлять возможность редактировать списки групп и студентов в группах в рамках одной кафедры. Необходим класс, генерирующий сообщения по событиям. Необходим класс для сохранения сообщений в файл о сработавших событиях	1) количество студентов в группе X > 5 (после добавления студента в группу), 2) Был добавлен новый студент в группу X с фамилией Y

Продолжение табл. 6

Вариант	Описание задачи	События
5	Программа-редактор книг на полках и в картотеке. Имеются классы: книжная полка, книга. Программа должна предоставлять возможность редактировать списки полок и книг на них в библиотеке. Необходим класс картотеки, в который книги будут попадать по событиям, а также необходим класс для сохранения сообщений в файл о сработавших событиях	1) создание книги, 2) удаление книги, 3) редактирование книги
6	Программа-эмулятор игры. Имеются классы: комната, игрок. Игроки находятся по два в комнате, обладают полем 3x3 ячейки с пятью расставленными мишенями и по очереди стреляют по мишеням в случайную клетку. Победа у того, кто первым сбил все мишени. Необходим класс, собирающий статистику по победам из всех комнат, а также класс для сохранения сообщений в файл о сработавших событиях	1) победа в игре, 2) поражение подряд двух мишеней, 3) промахи более 5 раз подряд
7	Программа-редактор групп студентов. Имеются классы: группа, студент. Программа должна предоставлять возможность редактировать списки групп и студентов в группах в рамках одной кафедры. Необходим класс, генерирующий сообщения по событиям. Необходим класс для сохранения сообщений в файл о сработавших событиях	1) количество студентов на кафедре < 7 (при добавлении и удалении студентов из групп), 2) количество студентов в группе X > 5 (после добавления студента в группу)
8	Программа-редактор товаров на складах. Имеются классы: склад, товар. Программа должна позволять редактироваться списки складов и товаров в них. Необходим класс с общим перечнем товаров (с указанием расположения), в который товары будут попадать по событиям, а также класс для сохранения сообщений в файл о сработавших событиях	1) создание товара, 2) удаление товара, 3) редактирование товара

Окончание табл. 6

Вариант	Описание задачи	События
9	Программа-эмулятор игры. Имеются классы: комната, игрок. Игроки находятся по два в комнате, обладают некоторым запасом здоровья и по очереди наносят друг другу случайной степени урон. Победа у того, кто первым исчерпал запас здоровья противника. Необходим класс, собирающий статистику по победам из всех комнат, а также класс для сохранения сообщений в файл о сработавших событиях	1) добавление игрока в комнату, 2) удаление игрока из комнаты, 3) начало игры, 4) окончание игры
10	Программа-редактор групп студентов. Имеются классы: группа, студент. Программа должна предоставлять возможность редактировать списки групп и студентов в группах в рамках одной кафедры. Необходим класс, генерирующий сообщения по событиям. Необходим класс для сохранения сообщений в файл о сработавших событиях	1) количество групп студентов равно нулю (после удаления группы), 2) количество студентов в группе X > 5 (после добавления студента в группу)
11	Программа-редактор гаражей и транспортных средств в нем. Имеются классы: гараж (более чем на 1 место), транспортное средство. Программа должна предоставлять возможность редактировать списки гаражей и транспортных средств в них. Необходим класс с общим перечнем транспортных средств (с указанием расположения), в который транспортные средства будут попадать по событиям. Необходим класс для сохранения сообщений в файл о сработавших событиях	1) создание транспортного средства, 2) удаление транспортного средства, 3) редактирование транспортного средства
12	Программа-эмулятор игры. Имеются классы: комната, игрок. Игроки находятся по два в комнате, обладают полем 3x3 ячейки с пятью расставленными мишенями и по очереди стреляют по мишеням в случайную клетку. Победа у того, кто первым сбил все мишени. Необходим класс, собирающий статистику по победам из всех комнат, а также класс для сохранения сообщений в файл о сработавших событиях	1) добавление игрока в комнату, 2) удаление игрока из комнаты, 3) начало игры, 4) окончание игры

Контрольные вопросы:

1. Что такое делегат?
2. Что такое событие? Как его можно создать?
3. Как передать в событие один или несколько параметров?
4. Что такое обработчик события?
5. Как отслеживать наступление события?

Пример программы

```
//для данной программы необходимо включить поддержку clr (проект -> свойства -> свойства конфигурации -> общие -> поддержка общеязыковой среды выполнения (CLR) -> поддержка clr среды /clr)

//объявление с использованием делегата формата метода для обработки события
delegate void MethodContainer(int count);
//класс, в котором производится счет
ref class ClassCounter  {
public:
//объявление события onCount с типом делегата MethodContainer.
    event MethodContainer^ onCount;
    void Count() {
        for (int i = 0; i < 100; i++) {
            if (i == 71)  {
                onCount(i);
            }
        }
    }
};
//классы Handler_I и Handler_II реагируют на событие класса ClassCounter записью строки в консоль. Классы среагируют, когда значение счетчика будет равно 71
ref class Handler_I {
public:
    void Message(int count) {
        printf("Пора действовать, ведь уже %d!\n", count);
```

```
    }
};
ref class Handler_II {
public:
    void Message(int count) {
        printf("Точно, уже %d!\n", count);
    }
};
void main() {
    ClassCounter^ Counter = gcnew ClassCounter();
    Handler_I^ Handler1 = gcnew Handler_I();
    Handler_II^ Handler2 = gcnew Handler_II();
//методы Message() классов Handler_I и Handler_II экземпляров Handler1 и
Handler2 подписываются на событие onCount класса Counter. Метод
Message() выполнится у Handler1 и Handler2 при свершении события onCount
класса Counter
    Counter->onCount +=
        gcnew MethodContainer(Handler1, &Handler_I::Message);
    Counter->onCount +=
        gcnew MethodContainer(Handler2, &Handler_II::Message);
//старт счета в объекте Counter. При достижении значения счетчика 71 вы-
полнятся методы Message объектов Handler1 и Handler2
    Counter->Count();
    getchar();
}
```

8. Лабораторная работа «Обработка исключений»

Ключевые понятия: исключительная ситуация, тип исключительной ситуации, обработка исключения.

Цель работы: научиться использовать встроенное в язык средство обработки исключительных ситуаций для предотвращения аварийного завершения работы программы в случае ошибок в данных.

Задание кафедры: для класса, полученного в результате лабораторной работы №2, перегрузить еще как минимум три операции. Создать собствен-

ную иерархию класса обработки исключений. В ситуациях, при которых может возникать ошибка, генерировать собственное исключение и его обрабатывать. В программе должны быть использованы блоки обработки исключений: try, catch. В лабораторной работе описать все обрабатываемые в программе исключительные ситуации.

Контрольные вопросы:

1. Зачем нужен блок «try»?
2. Зачем нужен блок «catch»?
3. Какие исключения может обрабатывать каждый из блоков «catch»?
4. Как задать блок «catch», чтобы он мог обрабатывать любые исключительные ситуации?
5. Для чего используется блок «finally»?
6. Что будет если в программе не задан блок «catch» для возникшей исключительной ситуации?
7. Как работает обработка исключительных ситуаций, если заданы несколько вложенных блоков «try»?
8. Когда необходима обработка исключительных ситуаций? Приведите примеры.

9. Индивидуальное домашнее задание «Разработка графического редактора на языке C++ с использованием механизмов ООП»

Ключевые понятия: инкапсуляция, наследование, полиморфизм, динамический полиморфизм, интерфейс, виртуальная функция, абстрактный класс

Цель работы: закрепить навыки использования механизмов ООП на примере реализации графического редактора

Задание кафедры: реализовать на языке C++ редактор графической схемы (табл. 8). В ходе выполнения работы обязательно применение объектно-ориентированных возможностей языка C++: наследования и динамическо-

го полиморфизма. Каждый тип элемента схемы должен быть представлен в программе в виде отдельного класса, который наследован от базового класса "графический элемент" (имеющего чисто виртуальную функцию прорисовки). Также необходим один класс "поле рисования", который содержит все графические элементы и отвечает за вызов функций прорисовки. Хранение графических элементов осуществляется с использованием контейнеров стандартной библиотеки C++.

Таблица 7

Векторные редакторы

Вар	Редактор		Вар	Редактор
1	Принципиальная схема		10	Классовая диаграмма
2	Редактор выкройки		11	Позиции на шахматной доске
3	Векторный графический редактор		12	Позиции игроков на футбольном поле
4	Схема локальной вычислительной сети		13	Позиции игроков на хоккейной площадке
5	Блок-схема алгоритма		14	Планировка магазина
6	Инфологических схем		15	Планировка вокзала
7	Планировка кухни		16	Планировка микрорайона
8	Планировка спальни		17	Производственная схема
9	Планировка гостиной			

Контрольные вопросы:

1. Зачем нужен перегруженный оператор присваивания?
2. Зачем нужен механизм наследования?
3. Зачем используются модификаторы при наследовании классов? Какие это модификаторы?
4. Зачем нужен механизм полиморфизма?
5. Что понимается под динамическим полиморфизмом?
6. Что такое интерфейс класса?
7. Зачем нужен чисто виртуальный метод? Как он выглядит?
8. Какой класс называется абстрактным?

Содержание