|  |
| --- |
| **ТЕХНОЛОГИИ ПРОГРАММИРОВАНИЯ** |
|  |
|  |
|  |
| **Практическое занятие №4.**  **Тема: Проектирование классов и создание объектов** |
|  |
|  |
|  |

**Цель:** Получение практических навыков разработки классов, создания объектов и использования дружественных функций для обработки данных.

**Основные теоретические положения**

**Общие сведения о классах и объектах**

Ключевым понятием ООП является объект. Под **объектом** понимается переменная особого типа. Этот особый тип в Си++ создан на базе типа данных **структура** (struct) и называется **класс** (class). *Класс* отличается от *структуры* в языке С++ тем, что в него помимо собственно данных различных типов входят также функции, обрабатывающие эти данные. Класс – это тип, группирующий данные и функции в единое целое с точки зрения их последующей обработки. Объявленная переменная типа class есть объект. Такое объединение данных и обрабатывающих их функций называется *инкапсуляцией.* Она защищает данные от внешнего вмешательства или неправильного использования. В то же время объект может оснащаться средствами программной связи с другими объектами.

Данные и функции, входящие в класс, называются *представителями* или *членами* этого класса. Данные класса ещё называют полями, а функции – методами класса. Члены класса разделяются на закрытые (**private**) и открытые (**public**). К закрытым членам обращение возможно только внутри класса и из других частей программы они недоступны. Обычно таким образом защищаются данные от внешнего вмешательства. К открытым членам класса может производиться обращение извне.

Синтаксис объявления *класса* похож на синтаксис объявления *структуры*. Пример объявления.

**#include <iostream>**

**using namespace std;**

**class myclass** // класс под именем myclass

**{**

**private:**

**int a;** // закрытая переменная

**public:** // ключевое слово (для открытых членов класса)

/\* функции – открытые члены класса для доступа к закрытой переменной \*/

**void set\_a(int n){a=n;}**

**int get\_a() {return a;}**

**};**

Объявление класса myclass не задает ни одного объекта типа myclass. Оно определяет только **тип** объекта (шаблон). Чтобы создать объект, необходимо указать имя класса - как спецификатор типа данных, после которого записать имя объекта. Например, в следующей строке заданы два объекта типа myclass .

**myclass obj1, obj2;**

Создание объекта связано с выделением для него памяти. При объявлении же класса память не выделяется.

После того как объекты созданы, можно обращаться к открытым членам класса. Как и при работе со структурами, обращение к открытым членам класса производится через точку **“ . ”**. В следующем фрагменте показано обращение к функциям- членам класса myclass.

**obj1.set\_a(10);** // обращение к функции set\_a( ) первого объекта

**obj2.set\_a(22);** // обращение к функции set\_a( ) второго объекта

**cout<<obj1.get\_a( )<<’\t’;**  /\*обращение к функции get\_a( ) первого объекта \*/

**cout<<obj2.get\_a( )<<’\n’**; /\*обращение к функции get\_a( )второго объекта \*/

Функции set\_a( ) первого объекта в качестве аргумента передаётся численное значение 10, которое присваивается переменной **а** этого объекта. Аналогично, переменной**а** второго объекта присваивается значение 22.

Каждый объект имеет свою переменную **а**, и использует принадлежащие классу функции set\_a( ) и get\_a( ). Поэтому объекты еще называют экземплярами классов. В третьей строке фрагмента на экран выводится значение, возвращаемое функцией get\_a( ) первого объекта, а в четвертой – второго объекта. После выполнения указанных операций на экране будет: 10 22.

**Конструкторы и деструкторы**

Среди членов класса могут быть две особенные функции. Их особенность заключается в том, что обращение к ним производится по умолчанию. Первая функция называется *конструктор*. Она вызывается автоматически при создании объекта. Её назначение состоит в том, чтобы присвоить начальные значения (инициализировать) переменным объекта. Вторая функция называется *деструктор*. Она служит для ликвидации последствий использования объекта (например, для освобождения памяти). Поэтому она автоматически вызывается при удалении объекта. Конструктор имеет то же имя, что и класс, частью которого он является. Имя деструктора также совпадает с именем класса, но перед ним ставится символ **~** (тильда). Конструктор и деструктор не возвращают никаких значений.

**#include<iostream>**

**using namespace std;**

**class abc** // объявление класса abc

**{**

**int a;** // закрытая переменная

**public:**

**abc(){ cout<<"I am constructor \n"; a=10;}** // конструктор

**~abc(){ cout<<"I am destructor \n";}** // деструктор

**void show(){ cout<<a<<"\n";}** // функция-членкласса

**};**

**int main()**

**{**

**abc obj;** // создание объекта

**obj.show();**

**return 0;**

**}**

В этом примере конструктор не содержал параметров. Но обычно параметры присутствуют, т.к. основное назначение конструкторов – производить инициализацию переменных объекта. В следующем примере используется конструктор с параметрами.

**#include<iostream>**

**using namespace std;**

**class black** // объявление класса black

**{**

**int a,b;** // закрытые переменные

**public:**

**black(int x, int y**) // конструктор с параметрами

**{a=x; b=y;}**

**void show()**

**{cout<<a<<'\t'<<b;}**

**};**

**int main()**

**{**

// создание объекта с инициализацией переменных

**black ob(20,100);**

**ob.show();**

**return 0;**

**}**

**Массивы объектов**

Объекты – это переменные, и они имеют те же возможности и признаки, что и переменные любых других типов. Поэтому объекты могут объединяться в массивы. Синтаксис объявления массива объектов совершенно аналогичен тому, который используется для объявления массива переменных любого другого типа. То же касается и доступа к элементам массива.

Пример с массивом объектов.

**#include<iostream>**

**using namespace std;**

**class one** // имя класса

**{**

**int a;** // закрытая переменная

**public:**

**void set\_a(int n){a=n;}** // функция доступа к переменной

**int get\_a( ) {return a;}** // функция доступа к переменной

**};**

**int main(**

**{**

**one obj[8];** // объявление массива из 8-и объектов

**int i;**

**for(i=0; i<8; i++)** // инициализация переменных объектов

**obj[i].set\_a(i);**

**for(i=0;i<8;i++)** // вывод значений переменных объектов

**cout<<obj[i].get\_a()<<' ';**

**return 0;**

**}**

В данном примере объявлен класс **one**, в котором имеется закрытая переменная и две функции доступа к ней. В главной функции создан массив из 8-и объектов, в котором циклической операцией производится инициализация закрытой переменной каждого элемента массива с помощью функции **set\_a( ).** Во второй циклической операции осуществляется вывод на экран значения переменной каждого объекта с использованием функции **get\_a( ).**

**Дружественные функции**

Дружественные функции введены для прямого доступа к private-членам объектов класса. Дружественная функция определяется в программе как обычная функция. Чтобы показать, к какому классу она дружественна, её объявляют в этом классе с ключевым словом *friend*. Особенность дружественной функции состоит в следующем. Дружественная функция не является членом класса. К закрытым членам класса она не может обращаться непосредственно. Она это может сделать только через объекты этого класса. Поэтому в качестве параметра у дружественной функции должен указыватьсяобъект со спецификатором дружественного класса.

**#include<iostream>**

**using namespace std;**

**class data**

**{**

**int a,b;**

**public:**

**data(int x,int y)** // конструктор с параметрами x и y

**{ a=x; b=y;}**

**friend int diff(data ob);** // дружественнаяфункция

**};**

**int diff(dataob)** // описание дружественной функции

**{ return ob.a-ob.b;}**

**int main()**

**{**

**int d;**

**data obj(8,5);**

**d=diff(obj);** // вызов дружественной функции

**cout<<"Difference = "<<d;**

**return 0;**

**}**

**Практические задания**

*Выберите один из вариантов задания.*

**Вариант А.**

**Задание 1**. Спроектировать класс **point,** обозначающий координаты точки на плоскости x, y (закрытые переменные), и функции доступа к закрытым переменным, а также конструктор и деструктор.

Динамически выделить память под массив из 4-х объектов так, чтобы они соответствовали точкам вершин прямоугольника. Разработать функцию, вычисляющую длины сторон и гипотенузы.

**Задание 2**. В проект добавить дружественную функцию, вычисляющую площадь прямоугольника, заданного точками его вершин.

**Задание 3.** Спроектировать класс в соответствии с *индивидуальным заданием* (см. таблица 6). Создать массив из 6-ти объектов спроектированного класса. Рассчитать *Вычисляемый показатель*.

Интерфейс программы оформить в виде меню.

**Вариант В.**

**Задание 1**. Спроектировать класс **point,** обозначающий координаты точки на плоскости x, y (закрытые переменные), и функции доступа к закрытым переменным, а также конструктор и деструктор.

Создать массив из 4-х объектов так, чтобы они соответствовали точкам вершин прямоугольника. Разработать функцию, вычисляющую длины сторон и гипотенузы.

**Задание 2**. В проект добавить дружественную функцию, вычисляющую площадь прямоугольника, заданного точками его вершин.

**Задание 3.** Спроектировать класс в соответствии с *индивидуальным заданием* (см. таблица 6). Создать массив из 6-ти объектов спроектированного класса. Рассчитать *Вычисляемый показатель*.

**Вариант С.**

**Задание 1**. Спроектировать класс **point,** обозначающий координаты точки на плоскости: x,y. Разработать:

а) конструктор с параметрами - для инициализации объектов класса **point**;

б) функции доступа к закрытым членам класса;

в) деструктор, обозначающий момент уничтожения объекта.

Создать два объекта, соответствующих двум точкам, и определить расстояние между ними.

**Задание 2.** Спроектировать класс в соответствии с *индивидуальным заданием* (см. таблица 6). Создать несколько объектов. Рассчитать *Вычисляемый показатель*.

Отчет оформляется по общеустановленным правилам в *электронном виде* со следующим содержанием:

1. титульный лист,
2. тема и цель практической работы,
3. задание на практическую работу,
4. текст программы с комментариями,
5. результаты работы программы (вид экрана) и
6. выводы по созданному проекту и использованию средств языка программирования.

Таблица 6

|  |  |  |
| --- | --- | --- |
| **№ п\п** | ***Класс*** | ***Вычисляемый показатель*** |
|  | Компьютер | Экземпляр с наибольшей оперативной памятью |
|  | Локальная сеть | Минимальная стоимость монтажа |
|  | Транспортное средство | Максимальный пробег на полном бензобаке |
|  | Программный продукт | Последняя версия |
|  | Документ | Количество документов, выданных в прошлом году |
|  | Диета | Максимальное дневное количество белков |
|  | Периферийное устройство компьютера | Минимальная цена устройства |
|  | Строительный товар | Сумма покупки |
|  | Представитель университета | Количество студентов, обучающихся у конкретного преподавателя |
|  | Предприятие малого бизнеса | Название предприятия с максимальным числом сотрудников |
|  | СУБД | Количество СУБД заданного производителя |
|  | Страховой полис | Количество полисов на заданную фамилию |
|  | Наушники | Тип с максимальным частотным диапазоном |
|  | Недвижимость | Максимальная жилая площадь |
|  | Часы | Самый дорогой экземпляр |
|  | Телефон | Самая новая модель |
|  | Бытовая техника | Количество товаров заданной фирмы |
|  | Магнитная карта для проезда на транспорте | Количество карт без поездок |
|  | Насекомое | Максимальный размер крыльев |
|  | Канцелярские товары | Количество товаров заданной фирмы |
|  | Товар | Сумма покупки |
|  | Учащийся | Учащийся с максимальным IQ (коэффициентом интеллекта) |
|  | Путешествие (тур) | Самый дешевый тур на 7 и более дней |
|  | Мебель | Количество предметов из дерева |
|  | Программное обеспечение | Продукт с максимальным объемом памяти |
|  | Представление | Минимальное число зрителей в зале |
|  | Запоминающее устройство | Экземпляр с минимальным размером |
|  | Принтер | Экземпляр с наибольшей производительностью |
|  | Книга | Количество книг одного автора |
|  | Телефон | Самая новая модель |