Практическое занятие №4.
Тема: Перегрузка операторов и проектирование виртуальных функций

Цель: получение практических навыков разработки перегруженных операторов и исследования виртуальных функций.

Перегрузка операторов

Перегрузка операторов позволяет использовать знакомые операторы для работы с новыми типами данных, такими, например, как класс. Когда оператор перегружается, то старое его значение сохраняется, и дополнительно к этому он приобретает новое значение, связанное с классом, для которого он определен.

Для перегрузки оператора создается **оператор-функция**, которая является членом класса, для которого она определена. Имя оператор-функции –

`operator#( ),`

где вместо знака **`#`** следует подставить перегружаемый оператор (например: **`operator*`**). В скобках указывается список аргументов (параметров).

В C++ можно перегружать большинство операторов, кроме: **`. :: ?`**.

Оператор-функция (за исключением оператора =) наследуется производным классом. Тем не менее, для производного класса тоже можно перегружать любой оператор (в том числе уже перегруженный в базовом классе).

Если в классе перегружается бинарный оператор, то необходимо указать, где брать оба операнда. Однако в оператор-функции, которая перегружает этот оператор, указывается только один параметр. Этим параметром будет объект, стоящий справа от оператора; объект, стоящий слева генерирует вызов оператор-функции и передается неявно, с помощью указателя ***`this`***.

Пример.

```
#nclude <iostream>
using namespace std;

 class coord
 {int x, y;
   public:
   coord() {x=0; y=0;}
   cord(int i, int j) {x=i; y=j;}
   int get_x() {return x;}
   int get_y() {return y;}
   coord operator+(coord ob2);
 };
 coord coord:: operator+(coord ob2)
 {
   coord temp;
```

```
  temp.x=x+ob2.x;
  temp.y=y+ob2.y;
  return temp;
}
int main ()
{
coord obj1(10, 5),
          obj2(5,3),
          obj3;
obj3=obj1+obj2;
cout <<”obj1+obj2: \n” <<”x=” <<obj3.get_x()<<”\t”;
cout <<”y=” <<obj3.get_y() <<”\n”;
return 0;
}
```

В описанном в примере классе: две закрытые переменные, перегруженный конструктор, функции получения значений закрытых переменных и оператор-функция.

Оператор-функция возвращает объект типа ***coord***; чтобы его иметь, он создается внутри функции. У оператора-функции – один параметр, вторым является ***this*** и его переменные ***x*** и ***y*** внутри оператор-функции указываются ***открыто***. Локальный объект ***temp*** позволяет хранить результат сложения, оставляя неизменными слагаемые и обеспечивая присваивание результата сложения.

Виртуальные функции

Виртуальные функции являются инструментом для поддержки динамического полиморфизма. Основой для использования виртуальных функций выступают указатели на производные классы.

Виртуальная функция является членом класса. Она объявляется внутри базового класса и **переопределяется** в производном. Только в базовом классе перед ней ставится слово ***virtual***. Динамический полиморфизм поддерживается только в том случае, когда вызов виртуальной функции **производного класса** производится через **указатель базового**. Какая версия виртуальной функции будет вызвана, определяется **типом объекта,** на который ссылается указатель. Пример.

```
#nclude <iostream>
using namespace std;
 class base
 {
 public:
  int i;
  base (int x) {i=x;}
  virtual void func()
  {cout <<”virtual of base ” <<i <<’\n’;}
 };
```

```
class derived1: public base
{public:
 derived1(int x): base(x) {}
 void func()
 {cout <<”virtual of derived1 ” <<i+i <<’\n’;}
};
class derived2: public base
{public:
 derived2(int x): base(x) {}
 void func()
 {cout <<”virtual of derived2” <<i*i <<’\n’;}
};

int main()
{
    base*p;
     base ob(10);
     derived1 d_ob1(10), derived2 d_ob2(10);
     p=&ob;    p→func();
     p=&d_ob1; p→func();
     p=&d_ob2; p→func();
 return 0;
}
```

Статические члены класса

Применяя ключевое слово **static** к члену класса, тем самым утверждается, что существует только одна копия этого **static-**члена, которая используется всеми объектами класса. Все статические данные при первом создании объекта инициализируются нулевыми значениями, если не представлено других значений инициализации.

При объявлении статического члена данных в классе программист не должен его определять. Необходимо обеспечить его глобальное определение вне этого класса. Это реализуется путем повторного объявления этой статической переменной с помощью оператора разрешения области видимости, который позволяет идентифицировать, к какому классу она принадлежит. Только в этом случае для этой статической переменной будет выделена память.

Пример использования static-члена класса.

```
#include <iostream>
using namespace std;

class ShareVar {
static int num;
```

```
public:
void setnum(int i) { num = i; };
void shownum() { cout << num << " "; }
};

int ShareVar::num; // определяется num

int main()
{
ShareVar a, b;
a.shownum(); // выводится 0
b.shownum(); // выводится 0
a.setnum(10); // устанавливается num равным 10
a.shownum(); // выводится 10
b.shownum(); // также выводится 10
return 0;
}
```

Обратите внимание на то, что статический целочисленный член num объявлен и в классе ShareVar, и определен в качестве глобальной переменной. Как было заявлено выше, необходимость такого двойного объявления вызвана тем, что при объявлении члена num в классе ShareVar память для него не выделяется. C++ инициализирует переменную num значением 0, поскольку никакой другой инициализации в программе нет.

Таким образом, при объявлении члена класса статическим обеспечивается создание только одной его копии, которая будет совместно использоваться всеми объектами этого класса.

Практикум

Задание 1. Доработать проект, выполненный для *Практикума №7*: добавить статический член в каждый класс, обозначающий общее свойство для всех объектов класса; перегрузить оператор для работы с объектами классов и использовать его для определения *Вычисляемого показателя*.

Задание 2. Добавить в проект, выполненный для *Задания 1,* виртуальную функцию в каждом классе, выводящую на экран значения полей собственного и базового классов.

Задание 3. Разработать меню для демонстрации работы программы и сделанных доработок.

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

1) титульный лист,
2) тема и цель практического занятия,
3) задание на практическое занятие,
4) текст программы с комментариями,
5) результаты работы программы и
6) выводы по разработанным элементам программы.

Приложение

№ п/п	Базовый класс	Производные классы	Вычисляемый показатель
1.	Компьютер	Настольный компьютер, ноутбук	Экземпляр с наибольшей оперативной памятью
2.	Локальная сеть	Одноранговая сеть, сеть типа клиент-сервер	Минимальная стоимость монтажа
3.	Транспортное средство	Легковой автомобиль, грузовой автомобиль	Максимальный пробег на полном бензобаке
4.	Программный продукт	Операционная система, текстовый редактор	Последняя версия
5.	Документ	Паспорт, студенческий билет	Количество документов, выданных в прошлом году
6.	Диета	Для здоровья, для похудения	Максимальное дневное количество белков
7.	Периферийное устройство компьютера	Клавиатура, мышь	Минимальная цена устройства
8.	Строительный товар	Сыпучие материалы, инструмент	Сумма покупки
9.	Представитель университета	Преподаватель, студент	Количество студентов, обучающихся у конкретного преподавателя
10.	Предприятие малого бизнеса	Магазин, парикмахерская	Название предприятия с максимальным числом сотрудников
11.	СУБД	Реляционная, иерархическая	Количество СУБД заданного производителя
12.	Страховой полис	Полис обязательного медицинского страхования, страхования жилища	Количество полисов на заданную фамилию
13.	Наушники	Проводные, беспроводные	Тип с максимальным частотным диапазоном
14.	Недвижимость	Таунхаус, квартира в многоквартирном доме	Максимальная жилая площадь
15.	Часы	Электронные часы, механические часы	Самый дорогой экземпляр
16.	Язык программирования	Процедурный, объектно-ориентированный	Последний по году разработки язык программирования
17.	Бытовая техника	Телевизор, холодильник	Количество товаров заданной фирмы
18.	Магнитная карта для проезда на транспор-	Карта общего назначения для проезда в метро, льгот-	Количество карт без поездок

	те	ная транспортная карта учащегося	
19.	Насекомое	Стрекоза, бабочка	Максимальный размер крыльев
20.	Канцелярские товары	Бумага, авторучка	Количество товаров заданной фирмы
21.	Товар	Посуда, продукты питания	Сумма покупки
22.	Учащийся	Школьник, студент	Учащийся с максимальным IQ (коэффициентом интеллекта)
23.	Путешествие (тур)	Морской круиз, отдых на море	Самый дешевый тур на 7 и более дней
24.	Мебель	Кровать, стол	Количество предметов из дерева
25.	Программное обеспечение (ПО)	Системное ПО, прикладное ПО	Представители с максимальным и минимальным объемом занимаемой памяти
26.	Представление	Театр, кино	Минимальное число зрителей в зале
27.	Запоминающее устройство	Флэш-карта, карта памяти	Экземпляры с максимальным и минимальным размером
28.	Принтер	Струйный, лазерный	Представитель с наибольшей производительностью
29.	Книга	Электронная, на бумаге	Количество книг одного автора
30.	Телефон	Смартфон, кнопочный	Самая новая модель