Определение классов. Конструкторы, деструкторы, указатель this

Тема 2

Класс – это

абстрактный тип данных, содержащий данные (поля класса) и функции (методы), манипулирующие этими данными

```
class Rational      //Заголовок класса
{
  …                 //Тело класса
};
```

Управление доступом

- `public`
- `protected`
- `private`

Задача

Разработать класс «Рациональная дробь»

Заголовочный файл Rational.h

```
class Rational
{
public: //открытый интерфейс класса
   Rational():numerator(1), denominator(1){}
   Rational(int num, int denom);
   int GetNum(){return numerator;}
   int GetDenom(){return denominator;}
   void SetNum(int num){numerator = num; Shorten();}
   void SetDenom(int denom);
   void Print();
   void GetRational(int & num, int & denom);
   void SetRational(int num, int denom);
private: //закрытый раздел класса
   int numerator, denominator;
   int NOD(int x, int y);
   void Shorten();
};
```

Конструктор - это

специальный член класса, призванный инициализировать объект класса в момент создания.

Синтаксически конструктор подобен функции, поэтому дальше будем называть его функцией.

Имя конструктора совпадает с именем класса.

Виды конструкторов:

- Конструктор по умолчанию (default constructor)
- Конструктор копирования (copy constructor)
- Конструктор перемещения (move constructor)
- Конструктор инициализации (initial constructor)

Конструктор по умолчанию

```
Rational():numerator(1), denominator(1){}
```

список инициализации

тело функции

Конструктор инициализации

```
Rational(int num, int denom);
```

Конструктор по умолчанию

- Предопределен в классе, пока программист не определит хотя бы один конструктор
- Необходимо иметь собственный конструктор по умолчанию, если в классе есть:
  - Ссылки
  - Константы
  - Объекты других классов, не имеющих конструктора по умолчанию

Список инициализации

Необходим для инициализации:

- Ссылок
- Констант
- Объектов других классов, не имеющих конструктора по умолчанию

inline функции класса

```
int GetNum(){return numerator;}


int GetDenom()
{return denominator;}


void SetNum(int num) {
   numerator = num;
   Shorten();
}
```

Заголовочный файл Rational.h

Определение остальных функций класса

Файл исходного кода Rational.cpp

```
Rational::Rational(int num, int denom)
{
    numerator = num;
    if(denom)
        this->denominator = denom;
    else
        this->denominator = 1;
    Shorten();
}
void Rational::SetDenom(int denom)
{
    if(denom)
        denominator = denom;
    Shorten();
}
```

Определение остальных функций класса

Файл исходного кода Rational.cpp

```
void Rational::Print()
{
   std::cout << numerator << "/" << denominator
             << std::endl;
}
void Rational::GetRational(int &num, int & denom)
{
   num = numerator;
   denom = denominator;
}
void Rational::SetRational(int num, int denom)
{
   numerator = num;
   if(denom)
       denominator = denom;
   Shorten();
}
```

Определение остальных функций класса

Файл исходного кода Rational.cpp

```
int Rational::NOD(int x, int y)
{
   if(x%y==0)
       return y;
   return NOD(y, x%y);
}

void Rational::Shorten(){
   int s = NOD(numerator, denominator);
   numerator/=s;
   denominator/=s;
}
```

Программа для тестирования класса

```
int main(){
   Rational r1(4,5), r2(3,9);
   r1.Print();
   r2.Print();
   r1.numerator = 15;      //ошибка
   r1.denominator = 3;     //ошибка
   r1.SetRational(15,3);
   Print();                //ошибка
   r1.Print();
   Rational *pr = &r2;
   pr->SetDenom(0);
   pr->Print();
   Rational arr[5];
   for(int i = 0; i < 5; ++i)
       arr[i].Print();
   return 0;
}
```

Файл исходного кода test.cpp

Конструктор копирования

- Каждый класс имеет предопределенный конструктор копирования (пока программист не определит свой собственный)
- Работает в следующих случаях:
  - один объект явно инициализирует другой объект (например, в объявлении);
  - копия объекта передается параметру функции;
  - генерируется временный объект (например, в качестве значения, возвращаемого функцией)

Перемещающий конструктор

- Позволяет повысить производительность приложения
- Вместо полного копирования позволяет просто изменять владельца ресурса
- Имеет смысл, когда в классе есть ссылка на внешний ресурс (файл, память и т.д.)

Деструктор

- Деструктор выполняет необходимые действия при разрушении объекта
- Программист должен определять деструктор тогда, когда необходимо освободить ресурсы (системные объекты, память, файловые дескрипторы) при уничтожении объекта

Более сложная задача

Разработать класс «Вектор»

Заголовочный файл Vector.h

```
class Vector
{
public:
   Vector();
   Vector(int s);
   Vector(double * v, int s);
   Vector(const Vector& ref);     //copy constructor
   Vector(Vector&& ref);          //move constructor
   ~Vector();                     //destructor
   void Print();
private:
   double * arr;
   int size;
};
```

Файл исходного кода Vector.cpp

```
Vector::Vector():arr(0), size(0)
{}

Vector::Vector(int s):size(s)
{
   arr = new double[size];
   for(int i = 0; i < size; i++)
       arr[i] = 0;
}

Vector::Vector(double *v, int s):size(s)
{
   arr = new double[size];
   for(int i = 0; i < size; i++)
       arr[i] = v[i];
}
```

```
Vector::Vector(const Vector &ref): size(ref.size)
{
   arr = new double[size];
   for(int i = 0;i < size; i++)
       arr[i] = ref.arr[i];
}
Vector::Vector(Vector &&ref): size(ref.size),
                              arr(std::move(ref.arr))
{
   ref.arr = nullptr;
   ref.size = 0;
}
```

Файл исходного кода Vector.cpp

Файл исходного кода Vector.cpp

```
Vector::~Vector()
{
   delete [] arr;
}
void Vector::Print(){
   if(!size){
       std::cout << "Empty array" << std::endl;
       return;
   }
   std::cout.setf(std:ios::fixed);
   std::cout.precision(2);
   for(int i = 0;i < size; i++)
   {
       std::cout.width(6);
       std::cout << arr[i];
   }
   std::cout << std::endl;
}
```

Файл исходного кода
test.cpp

```
int main() {
    srand(time(0));
    const int size = 10;
    double v[size];
    for(int i = 0;i < size; i++)
        v[i] = (double)(rand())/(rand()+1);
    Vector v1(v,size), v2(size);
    Vector v3 = v1;                 //copy constructor work
    Vector v4(v2);                  //copy constructor work
    Vector v5(std::move(v2));       //move constructor work
    v1.Print();
    v2.Print();
    v3.Print();
    v4.Print();
    return 0;
}
```

Ключевое слово `explicit`

- Запрет неявного вызова конструктора

Если так определен конструктор копии и в классе нет конструктора перемещения, то:

- нельзя передавать объект в функцию;
- нельзя возвращать объект из функции.

Указатель this

Каждый объект содержит указатель на самого себя – называемый указателем **`this`**. Этот указатель неявно присутствует как аргумент во всех ссылках на элементы внутри объекта. Чтобы функции класса знали, для какого объекта они вызываются, первый неявный параметр каждой функции – указатель **`this`**

Конец