C++: СТРУКТУРЫ, ПЕРЕОПРЕДЕЛЕНИЕ ОПЕРАТОРОВ, ШАБЛОНЫ

Школа::Кода

Олимпиадное программирование

2020-2021 Таганрог

Что такое структура?

- Структура – это пользовательский тип данных, который хранит в себе переменные других типов и функции для работы с ними. Создаётся следующим образом:
  struct ‘имя структуры’
  {‘список участников (членов) структуры’}
- Переменные, являющиеся членами структуры, называют полями. Поля объявляются как обычные переменные (но в старых стандартах C++ инициализация при объявлении поля не поддерживается). Обращение к полям происходит через точку.
- Функции, являющиеся членами структуры, называют методами. Объявление методов ничем не отличается от объявления функции. Из метода есть доступ ко всем полям структуры. Вызов методов происходит через точку.
- Конструктор – метод без возвращаемого значения и с именем, совпадающим с именем структуры, который вызывается при создании экземпляра структуры для инициализации полей стартовыми значениями.

Примеры структуры

```
struct MyStruct
{
    int field1;
    double field2;

    MyStruct(int _field1)
    {
        field1 = _field1;
        field2 = 0;
    }

    MyStruct(double field2)
        :field1(0), field2(field2)
    {}

    double product()
    {
        return field1 * field2;
    }
};
```

```
struct Node
{
    bool was;
    vector<int> dist;

    Node(int n)
        :was(false)
    {
        dist.resize(n);
    }
};
```

Имя структуры

Поля

Конструктор

Конструктор с использованием списков инициализации

Метод

Переопределение операторов

- Если оператор не работает с нужными вам параметрами, его можно переопределить.
- Синтаксически переопределение оператора практически не отличается от определения функции:
  ‘тип возвращаемого значения’ operator’символы операции’ (‘параметры’) {‘тело оператора’}
- Тип возвращаемого значения не может быть void.
- Количество параметров у оператора определяется его смысловой нагрузкой и является постоянным (у ‘+’ всегда два параметра, у ‘-’ может быть и один, т.к. минус бывает унарным).
- При переопределении операторов для собственной структуры стоит использовать ключевое слово friend и переопределять оператор внутри неё, чтобы иметь доступ к закрытым членам структуры.

Примеры переопределения операторов

```
bool operator<(vector<int>& a, vector<int>& b)
{
    return a.size() < b.size();
}
struct MyStruct
{
    int field1;
    double field2;

    MyStruct(int _field1) { ... }

    MyStruct(double field2) { ... }

    double product() { ... }

    friend bool operator>(MyStruct& a, MyStruct& b)
    {
        return a.product() > b.product();
    }
};

bool operator<(MyStruct& a, MyStruct& b)
{
    return a.product() < b.product();
}
```

Переопределение операторов потокового ввода и вывода

- Требуется #include <sstream>.
- Возвращаемое значение istream& или ostream& имя – operator >> или operator << для ввода или вывода соответственно.
- Первый параметр: istream& in или ostream& out.
- Второй параметр: ‘тип’& ‘имя’.
- В теле оператора ввести/вывести всё необходимое из(в) поток(а) in(out).
- Вернуть поток: return in; или return out;

Пример переопределения операторов потокового ввода и вывода

```
istream& operator >> (istream& in, pair<int, int>& p)
{
    in >> p.first >> p.second;
    return in;
}

ostream& operator << (ostream& out, pair<int, int>& p)
{
    out << p.first << " " << p.second;
    return out;
}
```

```
friend istream& operator >> (istream& in, MyStruct& a)
{
    in >> a.field1 >> a.field2;
    return in;
}
```

Шаблоны

- Шаблон — это конструкция, которая создает обычный тип или функцию во время компиляции на основе аргументов, предоставленных пользователем для параметров шаблона.
- Шаблоны служат основанием для универсального программирования на C++. В качестве строго типизированного языка C++ требует, чтобы все переменные имели конкретный тип, либо явно объявленный программистом, либо выведенный компилятором. Однако многие структуры данных и алгоритмы выглядят одинаково независимо от типа, на котором они работают. Шаблоны позволяют определить операции класса или функции и предоставить пользователю указание конкретных типов, с которыми должны работать эти операции.

Конструкция, задающая шаблон

- Строка template <‘параметры шаблона’> указывается перед функцией/структурой, которая будет использовать данный шаблон.
- В качестве параметров шаблона могут указываться типы, которые следует использовать, или константы. Например: template <typename T, class C, size_t L>
- Ключевое слово typename позволяет задать T любой примитивный тип данных.
- Ключевое слово class позволяет задать C не только примитивные типы данных, но и пользовательские.
- Далее при описании функции/структуры вместо конкретных типов данных или определённых констант в рассмотренном примере следует использовать T, C и L.

Примеры использования шаблонов

```
template<typename T1, typename T2>
istream& operator >> (istream& in, pair<T1, T2>& p)
{
    in >> p.first >> p.second;
    return in;
}
template<typename T1, typename T2>
ostream& operator << (ostream& out, pair<T1, T2>& p)
{
    out << p.first << " " << p.second;
    return out;
}
```

```
template<typename T>
struct Node
{
    bool was;
    vector<pair<int, T>> to;
    T dist;
};
```