Программирование и алгоритмизация

Санкт-Петербургский государственный политехнический университет

07 апреля 2015

Предварительное неполное объявление класса

Такое объявление называется (forward declaration) и допускает использование ссылки или указателя на пользовательский тип до его полного объявления.

```
class A
{
  B* pB; //???
  void F(B &); //???
};
class B
{
  A a;
};
```

Предварительное неполное объявление класса

```
class B;
class A
{
  B b; //???
  B* pB; //???
  void F(B &); //???
};
class B
{
  A a;
};
```

Объектно-ориентированный список

Двухсвязный список состоит из рекурсивного пользовательского типа данных (структуры или класса) с неполным объявлением, содержащего указатель на предыдущий и следующий элементы.

```
class Node
{
  Point m_data;
  Node *pPrev;
  Node *pNext;
};
```

Вложенное объявление вспомогательного класса

```
class List
{
  private:
    class Node { ...
      friend class List;
    };
  ...
};
```

Вспомогательный класс обертка

```
class Node {
  private:
    Point m_data;
    Node *pPrev;
    Node *pNext;
    ~Node();
    Node(Node *p, const Point&);
    friend class List;
};
```

Вспомогательный класс обертка

```
Node::Node(Node *prev, const Point &pt) : m_data(pt
    )
{
  pNext = prev->Next;
  prev->pNext = this;
  pPrev = prev;
  pNext->pPrev = this;
}

Node:~Node()
{
  if(pPrev)
    pPrev->pNext = pNext;
  if(pNext)
    pNext->pPrev = pPrev;
  pPrev = nullptr;
  pNext = nullptr;
}
```

Класс List

```
class List
{
  class Node { ... };
  Node Head; //Node*
  Node Tail;
  unsigned int size;
  public:
  List();
  ~List();
  void AddToHead(const Point&);
  bool Remove(const Point&);

};
```

Класс List

```
List::List()
{
  Head.pNext = &Tail;
  Tail.pPrev = &Head;
  m_size = 0;
}
void List::AddToHead(const Point &pt)
{
  new Node(&Head, pt);
  m_size++;
}
```

Класс List. Метод Remove

```
bool List::Remove(const Point &pt)
{
  Node *ptr = Head.pNext;
  for(int i = 0; i < m_size; i++)
  {
    Node *n = ptr->pNext;
    if(ptr->m_data == pt)
    {
      delete ptr;
      return true;
    }
    ptr = n;
  }
  return false;
}
```

Статические данные

- характеризуют количество или взаимосвязь всех существующих на данный момент объектов данного типа;
- имеют отношение к классу в целом, но не являются частью объекта данного класса.

Статические данные

Специфика:

- Статические переменные класса размещаются в статической области памяти, в независимости от того где располагается объект. Существуют в единственном экземпляре.
- Статическую переменную необходимо явно определить независимо от спецификатора доступа.
- При обращении к статической переменной класса, можно обращаться к ней, как к глобальной переменной заключенной в пространство имен (без создания объекта), но т.к. она входит в класс, можно работать с ней и посредством созданного объекта.

Статические данные

- Не следует инициализировать статические переменные в конструкторе, иначе есть опасность каждый раз при обращении получать одно и тоже значение.
- В константных методах, можно можифицировать статические переменные класса.

Счетчик объектов

```
//A.h
class A {
  int m_a; //обычная переменная класса
  public:
  static int count;
  A() { count++; }
  ~A() { count--; }

};

//A.cpp
#include "A.h"

int A::cout = 0; //выделение памяти под переменную
    и ее инициализация
//если память не выделить, то будет ошибка при
    линковке
```

Счетчик объектов

```
#include "A.h"
int main()
{
  size_t n = sizeof(A); //???
  std::cout << A::count;
  A a1;
  std::cout << a1.count;
  {
    A a2=a1;
    std::cout << a2.count; //???
  }
  std::cout << A::count; //???
}
```