Язык программирования C++

Лекция 8

Объектно-ориентированный подход:

- дизайн и реализация классов

Класс SimpleFloatArray

пример класса, созданного программистом

Пример дизайна и реализации класса массива чисел со свойствами:

- задание размера массива во время выполнения
- доступ к элементам с помощью x[i]
- автоматическое управление памятью
- автоматическое копирование элементов массива
- автоматическое копирование при присвоении
- присвоение определенного значения всем элементам
- определение текущего размера массива
- динамическое изменение размера массива

Каждое свойство реализуется с помощью метода класса (функции)

Класс SimpleFloatArray

зачем создавать такой класс ?

Пример стандартного использования массива в языке C++ :

```
int n;
cin >> n;
float *x = new float[n];
// ... some operations
sx += x[i];
delete [] x;
```

Можно заменить на :

```
int n;
cin >> n;
SimpleFloatArray x(n);
// ... some operations
sx += x[i];
delete [] x;
```

- уход от явного использования указателей
- автоматическое удаление
- возможность оперирования массивом как единым объектом :

```
SimpleFloatArray x(n);
SimpleFloatArray y = x;
SimpleFloatArray z;
//
z = x;       // copy array
x = 0.0;   // clears the array
```

Класс SimpleFloatArray

определение класса

Заголовочный файл (SimpleFloatArray.h) с объявлением класса :

```
class SimpleFloatArray {
public:
        SimpleFloatArray(int n);                    // init to size n
        SimpleFloatArray();                         // init to zero size
        SimpleFloatArray(const SimpleFloatArray&);
        ~SimpleFloatArray();                        // destroy
        float& operator[ ](int i);                  // subscript
        int numElems();
        SimpleFloatArray& operator=(const SimpleFloatArray&);
        SimpleFloatArray& operator=(float);   // set values
        void setSize(int n);
private:
        int num_elems;
        float *ptr_to_data;
        void copy(const SimpleFloatArray& a);
};
```

~SimpleFloatArray() – деструктор класса – метод, который вызывается автоматически при уничтожении объекта

float& operator[ ](int i) – метод класса вызывается при использовании оператора [ ]

operator=() – метод класса вызывается при операции присвоения

Класс SimpleFloatArray

конструкторы

```
SimpleFloatArray::SimpleFloatArray(int n) {
        num_elems = n;
        ptr_to_data = new float[n];
}

SimpleFloatArray::SimpleFloatArray() {
        num_elems = 0;
        ptr_to_data = 0;  // set pointer to null
}

SimpleFloatArray::SimpleFloatArray(const SimpleFloatArray& a) {
        num_elems = a.num_elems;
        ptr_to_data = new float[num_elems];
        copy(a);            // copy a elements
}
```

Конструктор по умолчанию (default constructor) — это конструктор класса, который объявляется без параметров. Он обеспечивает выделение памяти и корректную инициализацию объекта перед его первым использованием.

Класс SimpleFloatArray

реализация закрытой функции copy()

Пример 1:

```
void SimpleFloatArray::copy(const SimpleFloatArray& a) {
        // copy elements of ‘a’ into the elements of our array
        float* p = ptr_to_data + num_elems;
        float* q = a.ptr_to_data + num_elems;
        while (p > ptr_to_data) *--p = *--q;
}
```

Пример 2:

```
void SimpleFloatArray::copy(const SimpleFloatArray& a) {
        // copy elements of ‘a’ into the elements of *this
        for (int i = 0; i < num_elems; i++) {
                ptr_to_data[i] = a.ptr_to_data[i];
        }
}
```

Класс SimpleFloatArray

пример реализации деструктора

```
SimpleFloatArray::~SimpleFloatArray() {
        delete [ ] ptr_to_data;
}
```

- один и только один деструктор в классе
- имеет то же самое имя, что и класс, с символом ~ (тильда) впереди
- нет ни аргументов, ни возвращаемого типа
- вызывается автоматически, когда объект выходит из области видимости, т.е. уничтожается
- вызывается автоматически, когда объект уничтожается явным образом
- обычно используется для очистки динамически выделяемой памяти или других ресурсов

Класс SimpleFloatArray

пример метода operator [ ]

```
float& SimpleFloatArray::operator [ ] (int i) {
        return ptr_to_data[i];
}
```

- перегружает оператор [ ] для объектов данного класса
- возвращает *ссылку* на элемент массива
- т.к. это ссылка, то она может быть использована и справа и слева от оператора присвоения

```
int n;
cin >> n;
SimpleFloatArray  x(n);
SimpleFloatArray  y(n);

for (int i=0; i < n; i++) {
        cin >> x[i] >> y[i];
}
```

Класс SimpleFloatArray

пример метода operator=

```
SimpleFloatArray& SimpleFloatArray::operator=(const SimpleFloatArray& rhs) {
        if ( ptr_to_data != rhs.ptr_to_data ) {
                setSize ( rhs.num_elems );
                copy ( rhs );
        }
        return *this;
}
```

- if ( ) проверяет, что объект не присваивается сам себе
- this – это указатель на объект, с которым этот метод класса был вызван

Класс SimpleFloatArray

пример функции присвоения значений всем элементам массива

```
SimpleFloatArray& SimpleFloatArray::operator=(float rhs) {
        float* p = ptr_to_data + num_elems;
        while (p > ptr_to_data) *--p = rhs;
        return *this;
}
```

вызывается при такой записи :

```
SimpleFloatArray a(10);
a = 0.0;
```

но не при такой :

```
SimpleFloatArray a(10) = 0.0;
```

в которой смешиваются конструктор и присвоение.
Но при необходимости, можно реализовать дополнительно конструктор вида :

```
SimpleFloatArray a(10, 0.0);
```

Практическое задание (№ С8)

Разработать дизайн и написать реализацию оставшихся методов класса SimpleFloatArray (см. слайд 4):

- int numElems() - вывод количества элементов в данном объекте класса
- void setSize(int n) - изменение текущего размера объекта (если размер уменьшается по сравнению с первоначальным, то лишние элементы отбрасываются, а оставшиеся сохраняют свои значения. Если размер увеличивается, то добавляемые элементы зануляются)
- еще один конструктор, который создает объект класса заданного размера и присваивает всем элементам определенное значение

Проверить сделанную реализацию, написав программу, которая будет использовать класс SimpleFloatArray, сначала создав объект (т. е. массив) определенного размера, а потом изменив его размер в сначала большую, и потом в меньшую стороны. Напечатать содержимое оригинального и измененных объектов.