Классы

Конструкторы (constructors)

Конструктор предназначен для инициализации объекта и вызывается автоматически при его создании. Конструктор – это особая функция, являющаяся членом класса. Её имя всегда совпадает с именем класса.

Основные свойства конструкторов:

- **Конструктор не возвращает значение, даже типа void. Нельзя получить указатель на конструктор.**
- **Класс может иметь несколько конструкторов с разными параметрами для разных видов инициализации (при этом используется механизм перегрузки).**
- **Параметры конструктора могут иметь любой тип, кроме этого же класса. Можно задавать значения параметров по умолчанию. Их может содержать только один из конструкторов.**
- **Конструктор, вызываемый без параметров, называется конструктором по умолчанию.**
- **Если программист не указал ни одного конструктора, компилятор создает его автоматически. Такой конструктор вызывает конструкторы по умолчанию для полей класса и конструкторы по умолчанию базовых классов. В случае, когда класс содержит константы или ссылки, при попытке создания объекта класса будет выдана ошибка, поскольку их необходимо инициализировать конкретными значениями, а конструктор по умолчанию этого делать не умеет.**
- **Конструкторы не наследуются.**
- **Конструкторы нельзя описывать с модификаторами const, virtual и static.**
- **Конструкторы глобальных объектов вызываются до вызова функции main. Локальные объекты создаются, как только становится активной область их действия. Конструктор запускается и при создании временного объекта (например, при передаче объекта из функции).**

см.продолжение

Классы
Конструкторы

продолжение

- **Конструктор вызывается, если в программе встретилась какая-либо из синтаксических конструкций:**

```
имя_класса   имя_объекта [(список параметров)];
// Список параметров не должен быть пустым
имя_класса (список параметров);
// Создается объект без имени (список может быть пустым)
имя_класса   имя_объекта = выражение;
// Создается объект без имени и копируется
```

Примеры:

```
monstr Super(200, 300), Vasia(50), Z;
monstr X = monstr(1000);
monstr Y = 500;
```

В первом операторе создаются три объекта. Значения не указанных параметров устанавливаются по умолчанию.

Во втором операторе создается безымянный объект со значением параметра health=1000 (значение второго параметра устанавливается по умолчанию). Выделяется память под объект X, в которую копируется безымянный объект.

В последнем операторе создается безымянный объект со значением параметра health = 500 (значение второго параметра устанавливается по умолчанию). Выделяется память под объект Y, в которую копируется безымянный объект. Такая форма создания объекта возможна в том случае, если для инициализации объекта допускается задать один параметр.

Классы

Конструкторы

Инициализаторы в конструкторах

Главной задачей конструктора класса является инициализация данных создаваемых им объектов. Однако производить инициализацию данных при помощи оператора присваивания в теле конструктора не рекомендуется!

Инициализация осуществляется при помощи списка инициализации, расположенного между двоеточием, которое следует за закрывающей круглой скобкой в заголовке конструктора, и телом конструктора. Присваиваемые при инициализации данных значения записываются в круглых скобках после идентификаторов членов-данных и отделяются друг от друга запятыми.

Порядок следования идентификаторов в списке инициализации не имеет значения.

Для понимания дальнейших примеров вспомним пример описания класса CBook

```
// Файл Book.h – спецификация класса CBook
#pragma once  /*Чтобы компилятор включал объявление типа только один раз при трансляции программы */
class CBook
{
private:
   char m_author [50];        // автор
   char *m_pTitle;            // указатель на название
   int m_year;                // год издания
public:
// методы установки значений
    void setAuthor  const char*);
    void setTitle (const char*);
    void setYear (const int);
// методы возврата значений
        char* getAuthor  void ;
        char* getTitle (void);
        int getYear (void);
}
```

Классы
Конструкторы

Конструктор по умолчанию

Конструктор по умолчанию – конструктор, не требующий параметров. Этот конструктор всегда должен быть определен для любого класса. Конструктор по умолчанию может не выполнять никаких действий, но чаще всего он инициализирует данные класса нулевыми значениями.

Объявление конструктора по умолчанию имеет следующий формат:

```
public: имя_класса ( ) ;
```

Например, для класса CBook прототип конструктора по умолчанию записывается со спецификатором открытого доступа

```
public: CBook ( ) ;
```

Пример реализации конструктора по умолчанию:

```
CBook :: CBook ( )   : m_year ( 0 ), m_pTitle (  "" )
 {
  m_author [ 0 ] = '\0';
 }
```

В результате работы этого конструктора будет построен объект-книга, у которого данное year получит нулевое значение, массив author начнется с нулевого байта точно так же, как и название книги, на которое указывает pTitie. Для инициализации автора не может быть использован инициализатор, так как этот член-данное объявлен в классе как символьный массив.

Классы
Конструкторы

Конструктор с параметрами

Конструктор с параметрами инициализирует значения данных объекта значениями полученных параметров. Параметров будет столько, сколько данных необходимо проинициализировать.

Прототип такого конструктора имеет формат:

public:имя_класса (список формальных параметров) ;

Для класса CBook конструктор с параметрами может иметь следующий прототип:

CBook (char*, char*, int) ;

Пример реализации конструктора с параметрами:

```
CBook :: CBook ( char *author, char *title, int year )
: m_year ( year ), m_pTitle ( title )
{
strncpy_s ( m_author, 50, author, 49 ) ;
if ( strlen ( author ) > 49 ) m_author [ 49 ] = '\0' ;
}
```

В результате работы этого конструктора будет построен объект-книга, у которого данное m_yeag получит значение параметра year, в массив m_author будет скопировано не более 49 байтов из строки author, и указатель m_pTitie будет содержать адрес, по которому скопировано значение title.

Классы
Конструкторы

Конструктор копирования

Конструктор копирования создает копию объекта в оперативной памяти с помощью другого объекта того же класса.

В качестве параметра этот конструктор получает ссылку на объект, копию которого необходимо создать.

Прототип конструктора копирования:

```
public:имя_класса ( имя_класса & ) ;
```

Для класса CBook конструктор копирования объявляется в спецификации класса со следующим прототипом:

```
CBook ( CBook & ) ;
```

Пример конструктора копирования:

```
CBook :: CBook ( CBook &o )  : m_year ( o.m_year )
 {
strcpy_s ( m_author, strlen ( o.m_author ) + 1, o.m_author ) ;
m_pTitle = new char [strlen ( o.m_pTitle ) + 1 ] ;
strcpy_s ( m_pTitle, strlen ( o.m_pTitle ) + 1, o.m_pTitle ) ;
}
```

В результате работы конструктора копирования будет построен новый объект-книга, все члены-данные которого получат значения данных, принадлежащих копируемому объекту o, переданному по ссылке. Чтобы присвоить значение названия книги вновь создаваемому объекту, нужно сначала выделить необходимый блок памяти при помощи оператора new.

Классы

Деструкторы (destructors)

Деструктор предназначен для удаления объекта. Это может потребоваться для освобождения памяти или закрытия открытого ранее файла.

Деструктор – это особая функция-антипод конструктора. Её имя совпадает с именем класса с тильдой (~) перед ним. Деструктор вызывается автоматически при выходе объекта из области видимости (для локальных объектов – при выходе из блока, где они были объявлены; для глобальных – при выходе из main; для объектов, заданных через указатели – неявно при использовании операции delete).

В отличие от конструктора, деструктор не перегружается и может быть в классе только один. Лучше всегда определять деструктор класса, даже если он не производит никаких действий и имеет пустое тело.

Объявление деструктора имеет следующий формат:

```
public:~имя_класса ( ) ;
```

Например, для класса CBook прототип деструктора в спецификации класса имеет вид:

```
~CBook ( ) ;
CBook :: ~CBook ( )
{ delete [ ] m_pTitle ; }
```

Результатом действия деструктора ~CBook() является освобождение блока памяти с начальным адресом из m_pTitle, выделенного ранее конструктором класса.

Классы

Вызов Конструктора и Деструктора

Конструктор и деструктор вызываются в программе автоматически при объявлении объектов класса. Не имеет значения, объявляется ли объект явно или создается динамически с помощью new.

Вызов того или иного конструктора зависит от формы описания объекта:

- Если после имени объекта или типа класса в операторе new ничего не указано или стоят пустые круглые скобки, вызовется конструктор по умолчанию.
- Если в скобках записаны параметры, вызывается конструктор с параметрами.
- Если в скобках находится имя объекта, то происходит вызов конструктора копирования.

Кроме того, конструктор вызывается внутри функций для создания копии фактического параметра, если функция получает этот параметр по значению. Известно, что при выходе из функции копия уничтожается, следовательно, автоматически будет вызван деструктор для разрушения копии.

Конструкторы вызываются в программе в порядке объявления объектов.

Когда объявляется массив объектов, конструктор по умолчанию будет вызван столько раз, сколько элементов указано в объявлении массива.

При объявлении массива указателей на тип класса или единичного указателя никакие конструкторы не вызываются. Они будут вызваны позже при инициализации указателей оператором new.

Классы

Вызов Конструктора и Деструктора

Деструктор вызывается автоматически при разрушении объявленного по имени объекта, поэтому нет необходимости в явном вызове деструктора для разрушения локального объекта или для разрушения объектов перед выходом из программы.

Деструктор также вызывается автоматически, когда с помощью оператора delete освобождается указатель на объект класса. Без оператора delete объект разрушен не будет.

Разрешается и явный вызов деструктора для объекта, который выполняется, как и вызов любого другого метода класса.

Деструкторы вызываются в порядке, обратном порядку объявления именованных объектов в программе.

Вызов деструкторов объектов, объявленных через указатели, определяется следованием операторов delete для этих указателей.

Классы

Примеры вызова Конструктора и Деструктора

```
// конструктор по умолчанию
CBook :: CBook ( )  : m_year ( 0 ), m_pTitle ( "" )
{
m_author [ 0 ] = '\0' ;
cout << "default CONSTRUCTOR\nthis = " << this ;
}
// конструктор с параметрами
CBook :: CBook ( char *author, char *title, int year )
: m_year ( year ),  m_pTitle ( title )
{
strncpy_s ( m_author, 50, author, 49 ) ;
if ( strlen ( author ) > 49 ) m_author [ 49 ] = '\0' ;
cout << "with parameters CONSTRUCTOR\nthis = " <<this ;
}
// конструктор копирования
CBook :: CBook ( CBook &o )  : m_year ( o.m_year )
{
strcpy_s ( m_author, strlen ( o.m_author ) +1, o.m_author ) ;
m_pTitle = new char [strlen ( o.m_pTitle ) + 1 ] ;
strcpy_s ( m_pTitle, strlen ( o.m_pTitle ) + 1, o.m_pTitle ) ;
cout << "CONSTRUCTOR of copying\nthis = " << this;
}
```

Классы

Примеры вызова Конструктора и Деструктора

```
// деструктор
CBook :: ~CBook ( )
{ delete [ ] m_pTltle ; cout << "DESTRUCTOR\this " << this <<endl ; }
// главная функция
#include <iostream>
using namespace std ;
#include "Book.h"
void view ( char*, CBook& ) ;
int main ( )
{
// объявление с вызовом конструктора по умолчанию
CBook book ;
book.setAuthor ( "Robert Lafore" ) ;
book.setTitle ( "Object-Oriented Programming in C++" ) ;
book.setYear ( 2004 ) ;
view ( "book", book ) ;
// объявление с вызовом конструктора с параметрами
CBook obj  ( "Carrol L.",  "Alice in Wonderland", 2002 ) ;
view ( "obj", obj ) ;
// объявление с вызовом конструктора копирования
CBook copy ( obj ) ;
view ( "copy obj", copy ) ;
// объявление через указатель с вызовом конструктора с параметрами
CBook *p ;.
p = new CBook ( "Herbert Schildt", "n> "C++ The Complete Reference", 2003 ) ;
view ( "pointer p", *p ) ;
// разрушение объекта, объявленного через указатель
delete p ;      }
```

Классы

Примеры вызова Конструктора и Деструктора

```
// функция вывода состояния объекта класса CBook
void view ( char *s, CBook  &o )
{
cout << "\nState of object \' "  << s << "\'\n";
cout << "Author:\t"  << o.getAuthor()  o.getTitle()  << endl;
cout << "Title\t"  <<  o.getTitle()  << endl;
cout << "Year:\t"  <<  o.getYear()  << endl << endl;
}
```

В связи с тем, что эти методы работы конструкторов и деструктора класса CBook. вызываются автоматически, в реализацию каждого из них добавлен вывод сообщения с названием метода, а также вывод адреса начала объекта в оперативной памяти. В примере определена дополнительная функция view ( ), при помощи которой выводится состояние объекта, чей адрес получает функция.

В главной функции первым вызывается конструктор по умолчанию, когда объявляется объект book. Далее для объекта book при помощи методов класса setAuthor ( ), setTitie ( ) и setYear ( ) устанавливаются значения данных m_author, m_pTitie и m_Year, после чего выводится состояние объекта.

Вторым вызывается конструктор с параметрами, создающий объект obj с инициализацией данных этого объекта.

Третий вызываемый конструктор, который порождает являющийся копией obj объект copy – это конструктор копирования. После вывода состояния copy при помощи оператора динамического выделения памяти и конструктора с параметрами создается объект, адрес которого возвращает оператор new.

Оператор delete в конце программы освобождает указатель p и вызывает деструктор класса, который разрушает последний созданный объект. Далее деструктор автоматически вызывается три раза, разрушая объекты copy, obj и book.