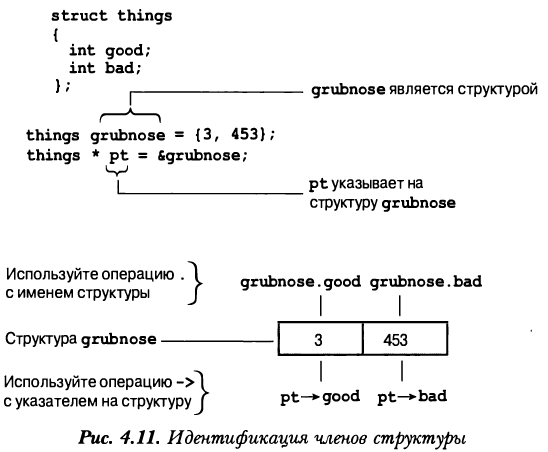
# Использование операции new для создания динамических структур

***Динамические*** *здесь снова означает выделение памяти во время выполнения, а не во время компиляции.*

*Применение new со структурами состоит из двух частей: создание структуры и обращение к ее членам. Для создания структуры вместе с операцией new указывается тип структуры. Например, чтобы создать безымянную структуру типа inflatable и присвоить ее адрес соответствующему указателю, можно поступить следующим образом:*

*inflatable \* ps = new inflatable;*

Более сложная часть — доступ к членам. Когда вы создаете динамическую структуру, то не можете применить операцию членства к имени структуры, поскольку эта структура безымянна. Все, что у вас есть — ее адрес. В C++ предусмотрена специальная операция для этой ситуации — операция членства через указатель **(->)**. Эта операция, оформленная в виде тире с последующим значком "больше", означает для указателей на структуры то же самое, что операция точки для имен структур. Например, если ps указывает на структуру типа inflatable, Tops->price означает член price структуры, на которую указывает ps.



**Листинг 4.21. news tret. epp**

*// newstrct.cpp -- использование new со структурой*

*#include <iostream>*

*struct inflatable // определение структуры*

*{*

*char name[20];*

*float volume;*

*double price;*

*};*

*int main ()*

*{*

*using namespace std;*

*inflatable \* ps = new inflatable; // выделение памяти для структуры*

*cout << "Enter name of inflatable item "; // ввод имени элемента inflatable*

*cin.get(ps->name, 20); // первый метод для доступа к членам*

*cout « "Enter volume in cubic feet: "; // ввод объема в кубических футах*

*cin >> (\*ps).volume; // второй метод для доступа к членам*

*cout « "Enter price: $"; // ввод цены*

*cin >> ps->price ;// первый метод*

*cout « "Name: " « (\*ps).name « endl;// второй метод*

*cout « "Volume: " << ps->volume << " cubic feet\n";*

*cout << "Price: $" << ps->price << endl;*

*delete ps; // освобождение памяти, использованной структурой*

*return 0;*

*}*

**Ниже показан пример выполнения программы из листинга 4.21:**

*Enter name of inflatable item: Fabulous Frodo*

*Enter volume in cubic feet: 1.4*

*Enter price: $27.99*

*Name: Fabulous Frodo*

*Volume: 1.4 cubic feet*

*Price: $27.99*

**Пример использования операций new и delete**

Использование операции new и delete для управления сохранением строкового ввода с клавиатуры. В листинге 4.22 определяется функция getname (), которая возвращает указатель на входную строку. Эта функция читает ввод в большой временный массив, а затем использует new [ ] с указанием соответствующего размера, чтобы выделить фрагмент памяти в точности такого размера, который позволит вместить входную строку. После этого функция возвращает указатель на этот блок. Такой подход может сэкономить огромный объем памяти в программе, читающей большое количество строк. (В реальности было бы проще воспользоваться классом string, в котором операции new и delete применяются внутренне.)

Предположим, что ваша программа должна прочитать 1000 строк, самая длинная из которых может составлять 79 символов, но большинство строк значительно короче. Если вы решите использовать массивы char для хранения строк, то вам понадобится 1000 массивов по 80 символов каждый, т.е. 80 000 байт, причем большая часть этого блока памяти останется неиспользованной. В качестве альтернативы можно создать массив из 1000 указателей на char и применить new для выделения ровно такого объема памяти, сколько необходимо для каждой строки. Это может сэкономить десятки тысяч байт. Вместо того чтобы создавать большой массив для каждой строки, вы выделяете память, достаточную для размещения ввода. И более того, вы можете использовать new для выделения памяти лишь для стольких указателей, сколько будет входных строк в действительности.

**Листинг 4.22. delete. срр**

*// delete.срр — использование операции delete*

*#include <iostream>*

*#include <cs.tring> // или string, h*

*using namespace std;*

*char \* getname(void); // прототип функции*

*int main()*

*{*

*char \* name; // создание указателя, но без хранилища*

*name = getname(); // присваивание name адреса строки*

*cout « name « " at " << (int \*) name << "\n";*

*delete [] name; // освобождение памяти*

*name = getname(); // повторное использование освобожденной памяти*

*cout « name « " at " « (int \*) name « "\n";*

*delete [] name; // снова освобождение памяти*

*return 0;*

*}*

*char \* getname() // возвращает указатель на новую строку*

*{*

*char temp[80]; // временное хранилище*

*cout « "Enter last name:'4"; // ввод фамилии*

*cin » temp;*

*char \* pn = new char [strlen (temp) + 1] ;*

*strcpy(pn, temp); // копирование строки в меньшее пространство*

*return pn; // по завершении функции temp теряется*

*}*

**Ниже приведен пример выполнения программы из листинга 4.22***:*

*Enter last name: Fredeldurapkin*

*Fredeldumpkin at 0x004326b8*

*Enter last name: Pook*

*Pook at 0x004301c8*

**Замечания по программе**

Функция getname (), представленную в листинге 4.22. Она использует сіn для размещения введенного слова в массив temp. Далее она обращается к new для выделения памяти, достаточной, чтобы вместить это слово. С учетом нулевого символа программе требуется сохранить в строке strlen (temp) + 1 символов, поэтому именно это значение передается new. После получения пространства памяти getname () вызывает стандартную библиотечную функцию strcpy (), чтобы скопировать строку temp в выделенный блок памяти. Функция не проверяет, поместится ли строка, но getname () гарантирует выделение блока памяти подходящего размера. В конце функция возвращает ps — адрес копии строки.

Внутри main () возвращенное значение (адрес) присваивается указателю name. Этот указатель объявлен в main (), но указывает на блок памяти, выделенный в функции getname (). Затем программа печатает строку и ее адрес. Далее, после освобождения блока, на который указывает name, функция main () вызывает getname () второй раз. C++ не гарантирует, что только что освобожденная память будет выделена при следующем вызове new, и, как видно из вывода программы, это и не происходит. Обратите внимание, что в рассматриваемом примере getname () выделяет память, a main () освобождает ее. Обычно это не слишком хорошая идея — размещать new и delete в разных функциях, потому что в таком случае очень легко забыть вызвать delete.