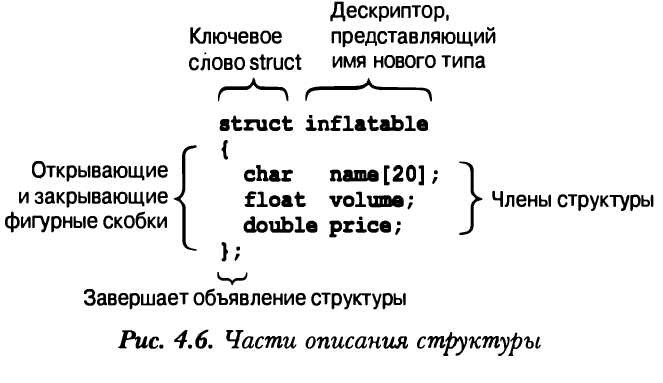
# Структура

**Структура** представляет собой определяемый пользователем тип с объявлением, описывающим свойства данных типа. После определения типа можно создавать переменные этого типа. То есть создание структуры — процесс, состоящий из двух частей. Вначале определяется описание структуры, в котором перечисляются и именуются типы данных, ранящиеся в структуре. Затем создаются структурные переменные, или, иначе говоря, структурные объекты данных, которые следуют плану, заданному

объявлением.

**Объявление структуры**

*struct inflatable // объявление структуры*

*{*

*char name[20];*

*float volume;*

*ouble price;*

*};*

***Идентификатор*** *inflatable —* ***имя****, или* ***дескриптор****,*

После определения структуры можно создавать переменные этого типа:

*inflatable hat; // hat — структурная переменная типа inflatable*

*inflatable woopie\_cushion; // переменная типа inflatable*

*inflatable mainframe; // переменная типа inflatable*

*struct inflatable goose; // ключевое слово struct требуется в С*

*inflatable vincent; // ключевое слово struct не требуется в C++*

**Использование структур в программах**

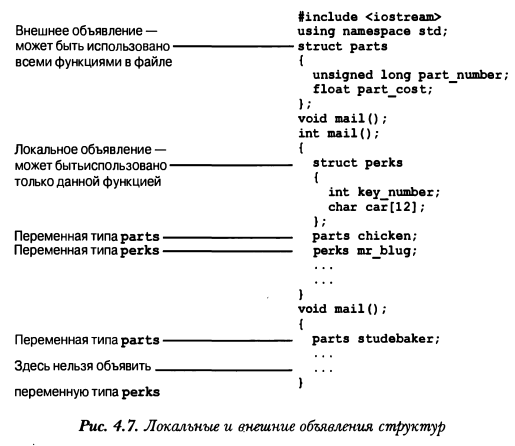
**Листинг 4.11. structur. срр**

*// structur.срр -- простая структура*

*#include <iostream>*

*struct inflatable // объявление структуры*

*{*

*char name[20];*

*float volume;*

*double price;*

*};*

*int main ()*

*{*

*using namespace std;*

*inflatable guest =*

*{*

*"Glorious Gloria", // значение name*

*1.88, // значение volume*

*2 9.99' // значение value*

*};*

*// guest — структурная переменная типа inflatable*

*// Инициализация указанными значениями*

*inflatable pal =*

*{*

*"Audacious Arthur",*

*3.12,*

*32.99*

*};*

*// pal — вторая переменная типа inflatable*

*// ПРИМЕЧАНИЕ: некоторые реализации требуют использования*

*// static inflatable guest =*

*cout « "Expand your guest list with " « guest.name;*

*cout « " and " « pal.name « "!\n"; // pal.name - член name переменной pal*

*cout << "You can have both for $";*

*cout << guest.price + pal.price << "!\n";*

*return 0;*

*}*

**Процедуру инициализации:**

*inflatable guest =*

*{*

*"Glorious Gloria", // значение name*

*1.88, // значение volume*

*29.99 // значение value*

*};*

*inflatable duck = {"Daphne", 0.12, 9.98}; //или можно инициализировать таким образом*

*inflatable duck {"Daphne", 0.12, 9.98}; //в С++11 знак = можно опустить*

**Применение типа string**

*#include <string>*

*struct inflatable // определение структуры*

*{*

*std::string name;*

*float volume;*

*double price;*

*};*

**Операции со структурой**

В C++ пользовательские типы сделаны, насколько возможно, похожими на встроенные типы. Например, структуру можно передавать как аргумент функции, а функция может использовать структуру в качестве возвращаемого значения. Также можно применять операцию присваивания (=), чтобы присвоить одну структуру другой того же самого типа. Эта операция устанавливает значение каждого члена одной структуры равным значению соответствующего члена другой структуры, даже если член является массивом. Такой тип присваивания называется почленным присваиванием.

Листинг 4.12. assign\_st.cpp

*// assign\_st.срр -- присваивание структур*

*#include <iostream>*

*struct inflatable*

*{*

*char name[20];*

*float volume;*

*double price;*

*};*

*Составные типы 159*

*int main ()*

*{*

*using namespace std;*

*inflatable bouquet =*

*{*

*"sunflowers",*

*0.20,*

*12.49*

*};*

*inflatable choice;*

*cout « "bouquet: " « bouquet.name « " for $";*

*cout « bouquet .price « endl;*

*choice = bouquet;* ***// присваивание одной структуры другой***

*cout « "choice: " « choice.name « " for $";*

*cout « choice.price « endl;*

*return 0;*

*}*

Можно комбинировать определение формы структуры с созданием структурных переменных. Чтобы сделать это, сразу после закрывающей фигурной скобки понадобится указать имя переменной или нескольких переменных:

*struct perks*

*{*

*int key\_number;*

*char car[12];*

*} mr\_smith, ms\_jones; // две переменных типа perks '*

*Можно даже инициализировать созданную переменную, как показано ниже:*

*struct perks*

*{*

*int key\_number;*

*char car[12];*

*} mr\_glitz =*

*{*

*7, // значение члена mr\_glitz.key\_number*

*"Packard" // значение члена mr\_glitz.car*

*};*

**Массивы структур**

Структура inflatable содержит массив (по имени name). Также можно создавать массивы, элементами которых являются структуры. Подход здесь в точности совпадает с таковым для массивов фундаментальных типов. Например, чтобы создать массив из 100 структур inflatable, можно поступить так:

*inflatable gifts[100]; // массив из 100 структур inflatable*

Поскольку каждый элемент массива является структурой, его значение представляется инициализацией структуры. Таким образом, мы получаем следующую конструкцию:

*inflatable guests [2] = // инициализация массива структур*

*{*

*{"Bambi", 0.5, 21.99}, // первая структура в массиве*

*{"Godzilla", 2000, 5 65.99} // следующая структура в массиве*

*};*

**Листинг 4.13. arrstruc. срр**

*// arrstruc.срр -- массив структур*

*finclude <iostream>*

*struct inflatable*

*{*

*char name[20];*

*float volume;*

*double price;*

*};*

*int main ()*

*{*

*using namespace std;*

*inflatable guests [2] = // инициализация массива структур*

*{*

*{"Bambi", 0.5, 21.99}, // первая структура в массиве*

*{"Godzilla"/ 2000, 5 65.99} // следующая структура в массиве*

*};*

*cout « "The guests " << guests [0] .name « " and " << guests [1] .name*

*« "\nhave a combined volume of "*

*« guests[0].volume + guests[1].volume << " cubic feet.\n";*

*return 0;*

*\_}*

**Битовые поля в структурах**

Язык C++, как и С, позволяет указывать члены структур, занимающие определенное количество битов памяти. Это может пригодиться для создания структур данных, которые соответствуют, скажем, регистру в некотором аппаратном устройстве.

struct torgle\_register

{

unsigned int SN : 4; //4 бита для значения SN

unsigned int : 4; //4 бита не используются

bool goodln : 1; // допустимый ввод (1 бит)

bool goodTorgle : 1; // признак успешности

};