10. Символьные строки. Структуры. Массивы структур

В языке C/C++ **строка** представляется как массив символов (элементов типа char), завершающийся специальным символом с кодом 0 (нуль-символом, не имеющем графического изображения), который записывается как ’\0’ и является маркером конца строки. По его положению может быть определена длина строки. Конкретные строки-константы заключаются в двойные кавычки (например, ”Moscow”), одиночные символы — в апострофы (например, ’F’, ’9’, ’+’). Каждый символ занимает в памяти 1 байт. Строки-переменные объявляются как обычные массивы, но при их инициализации могут использоваться конструкции строк-констант:

char str[10]=”Russia”;

Здесь для строки str выделено 10 байт, из которых 6 занято под значащие символы, а 7‑й — под нуль-символ. Если строка описывается с инициализацией, то размер можно не указывать (компилятор сам выделит необходимый объём памяти):

char str2[]=”Russia”;

(выделяется и заполняется 7 байт).

Как и для массивов, имя строки является указателем на её первый байт. Выделение динамической памяти под строки делается по тем же правилам, что и для массивов. Однако если для ввода и вывода массивов требовались циклы, то ввод и вывод строк может быть реализован одним действием. Если, к примеру, имеется описание

char st[12];

то для чтения строки st (в стиле C) применима операция

scanf(”%s”,st); либо scanf(”%s”,&st[0]);

Возможно применение модификаторов формата:

scanf(”%5s”,st);

Тогда будет прочитано только 5 первых значащих символов строки. Однако в любом случае **операция scanf читает строку только до первого пробела**.

Для печати строки применяется операция

printf(”%s”,st);

с возможным использованием модификаторов формата.

Допускается также чтение и печать строк производить с использованием потоков C++, через операции cin и cout:

cin>>st; и cout<<st;

**Чтение в этом случае идёт также только до первого пробела.** Чтобы при чтении строка воспринималась вместе с пробелами, следует использовать операцию gets из заголовочного файла stdio.h. Для печати в этом файле имеется «парная» ей операция puts. Возможно также подключение заголовочного файла srting.h (или cstring), содержащего ряд полезных функций для работы со строками в стиле C. В частности, там имеется функция strlen, возвращающая длину строки, и функция strcpy, копирующая одну строку в другую (простое присваивание здесь применять нельзя, поскольку строка — это массив). Подробнее обо всех функциях этого файла можно прочитать в [1, 2].

Однако в рамках нашего курса строки будут использоваться только в простейшем варианте, в основном в качестве полей структур.

**Структурой** в языке C/C++ называется комбинированный объект, компоненты которого, в отличие от массивов, могут быть разных типов. Компоненты структуры называются **полями**, и каждое поле должно обладать именем (по этому имени происходит к нему доступ — см. далее). Поля структуры могут иметь любой тип, кроме типа этой же структуры, но могут быть указателями на этот тип.

Описание структуры задаёт новый тип, который может быть поименован и использоваться в дальнейшем:

struct *имя\_типа* {

*тип\_1* *поле\_1*;

*тип\_2* *поле\_2*;

...

*тип\_N* *поле\_N*;};

Если несколько подряд идущих полей относятся к одному и тому же типу, допускается перечисление их через запятую (после наименования этого типа).

Допускается также *имя\_типа* не указывать (оставить тип структуры безымянным), однако в этом случае одновременно с описанием типа структуры следует объявить конкретные структуры этого типа (хотя бы одну). Так, к примеру, определим структуру типа Worker (работник) с полями: fio (фамилия — строка, до 15 символов), yb (год рождения — целое число), sal (зарплата — целое число) и два конкретных работника w1 и w2. Допустим любой из двух способов:

struct Worker {

char fio[16];

int yb,sal;};

Worker w1,w2;

Либо так:

struct {

char fio[16];

int yb,sal;} w1,w2;

Для **описания** структуры **с инициализацией** значения полей перечисляют в фигурных скобках в порядке описания:

struct {

char fio[16];

int yb,sal;} w1={”Ivanov”,1975,50000};

**Доступ к полям структуры** осуществляется с помощью операции «помещения в контекст» . (точка):

*имя\_структуры*.*имя\_поля*

Например:

cout<<w1.fio<<” ”<<w1.sal<<endl;

При использовании **указателя на структуру** вместо точки ставится «импровизированная стрелка» из двух символов -> (без пробела!):

struct Worker {

char fio[16];

int yb,sal;};

Worker \*pw=new Worker;

cin>>pw->fio>>pw->yb>>pw->sal;

delete pw;

Для переменных одного и того же типа структуры определена операция присваивания = (при этом происходит поэлементное копирование). Структуру можно передавать в функцию в качестве параметра и возвращать в качестве результата. Структурные типы активно используются при работе с **базами данных** — файлами, хранящими структурные характеристики объектов (и связей между ними), относящимся к какой-либо предметной области.

**Пример 1.** Определить структурный тип date (дата) с полями: day (день), month (месяц), year (год). Задать дату рождения и сегодняшнюю дату. Определить количество полных лет.

В случае, если в текущем году день рождения уже был, количество полных лет N равно разности годов, в противном случае из разности годов нужно вычесть единицу.

#include <iostream>

using namespace std;

int main() {

struct date {int day,month,year;} d1,d2;

int N;

cout<<”задайте дату рождения\n”;

cout<<”день: ”; cin>>d1.day;

cout<<”месяц: ”; cin>>d1.month;

cout<<”год: ”; cin>>d1.year;

cout<<”задайте сегодняшнюю дату\n”;

cout<<”день: ”; cin>>d2.day;

cout<<”месяц: ”; cin>>d2.month;

cout<<”год: ”; cin>>d2.year;

if (d2.month>d1.month ||

d2.month==d1.month && d2.day>=d1.day)

N=d2.year-d1.year;

else N=d2.year-d1.year-1;

cout<<”N=”<<N<<endl;

return 0; }

В случае, когда поле структуры само является структурой (ситуация иерархической структуры), при доступе к полям «внутренней» структуры контекстная точка используется дважды — на каждом уровне:

struct date {int day,month,year;};

struct Worker2 {

char fio[16];

date db;

int sal;};

Worker2 w;

w.db.year=1975;

(db — дата рождения работника).

Если поле структуры является массивом, то при обращении к конкретному элементу этого массива индекс ставится при наименовании поля-массива:

struct Worker3 {

char fio[16];

int yb;

int sal[6];};

Worker3 w;

w.sal[3]=32000;

(sal в данном случае — массив зарплат работника за последние 6 месяцев).

Если имеется массив структур, то при обращении к элементу этого массива индекс ставится при наименовании структуры:

struct Worker {char fio[16];int yb,sal;};

Worker w[10];

w[4].sal=70000;

**Пример 2.** В массиве из 12 структур типа Worker определить фамилии и зарплаты работников, родившихся позже 1970 года.

#include <iostream>

using namespace std;

int main() {

struct Worker {

char fio[16];

int yb,sal;};

const int N=12;

Worker w[N];

for(int i=0; i<N; i++) {

cout<<”задайте fio,yb.sal\n”;

cin>>w[i].fio>>w[i].yb>>w[i].sal; }

for(int i=0; i<N; i++)

if(w[i].yb>1970)

cout<<w[i].fio<<” ”<<w[i].sal<<endl;

return 0; }

В случае большого массива структур чтение данных удобнее производить из файла.

**Пример 3.** В массиве из 20 структур типа Worker3 вывести фамилии работников, у которых минимальная зарплата за последние 6 месяцев была меньше заданного значения. Поиск минимального элемента в массиве должен производиться с помощью функции.

При чтении данных из файла будет использоваться двойной цикл: по 20 работникам и для каждого работника по 6 месяцам с заданными зарплатами. Минимальная зарплата будет вводиться с экрана, вывод результатов производиться на экран.

#include <iostream>

#include <fstream>

using namespace std;

int Min(int n, int \*x); //прототип функции,

//находящей минимальный элемент в

//массиве x из n элементов

int main() {

struct Worker3 {

char fio[16];

int yb;

int sal[6];};

const int N=20;

Worker3 w[N];

int m; // минимальная зарплата

ifstream fin(”data.txt”);

for(int i=0; i<N; i++) {

fin>>w[i].fio>>w[i].yb;

for(int j=0;j<6;j++) fin>>w[i].sal[j]; }

cout<<”задайте минимальную зарплату\n”;

cin>>m;

for(int i=0; i<N; i++)

if(Min(6,w[i].sal)<m)

cout<<w[i].fio<<endl;

return 0; }

int Min(int n, int \*x) {

int m=x[0];

for(int i=1; i<n; i++) if(x[i]<m) m=x[i];

return m; }

**Задачи**

1. Определить структуру, представляющую момент времени (час, минута, секунда). Задача: ввести текущий момент времени, вывести момент на следующую секунду (учесть возможную смену суток).
2. Определить структуру «дата» (число, месяц, год). Задача: ввести сегодняшнюю дату, вывести завтрашнюю (с учётом високосных годов — делящихся на 4).
3. Сделать предыдущую задачу в расширенном виде: учесть григорианскую поправку (года, оканчивающиеся двумя нулями, являются високосными только в том случае, когда после отбрасывания этих двух нулей оставшееся число делится на 4).
4. Описать иерархическую структуру «горожане» с полями «фамилия» (строка) и «адрес». Поле «адрес» само является структурой с полями «город», «улица» (строковые); «дом», «квартира» (целочисленные). Задача: в файле имеется массив из 15 таких структур. Вывести на экран фамилии живущих по одинаковым адресам, но в разных городах (как в фильме «Ирония судьбы»).
5. Определить структуру с именем Tram (трамвай), содержащую следующие поля: бортовой номер трамвая, маршрутный номер, фамилия водителя. Задача: имеется массив из 10 трамваев. По заданному номеру маршрута вывести бортовые номера соответствующих трамваев и фамилии водителей (либо сообщение, что такого маршрута нет).
6. Определить структуру с именем Country (страна), содержащую следующие поля: название, площадь, количество жителей, форма правления. Задача: имеется массив из 12 стран. Вывести названия всех парламентских республик с плотностью населения ниже заданной (либо сообщение, что такой страны нет).
7. Определить структуру с именем Train (поезд), содержащую следующие поля: номер поезда, пункт отправления, пункт назначения, время отправления в формате ЧЧММ. В файле размещён массив из 15 поездов. Написать функцию F с аргументами t1, t2: функция выводит на экран информацию обо всех поездах, отправляющихся в промежуток времени от t1 до t2 (принять, что t1<t2). Проверить её работу с помощью тестовой main, задающей t1 и t2 и вызывающей функцию F.
8. Определить структуру Student с полями: ФИО, группа, успеваемость (массив из 5 оценок по 5‑балльной системе). Имеется массив из 10 студентов. Вывести ФИО и группы тех из них, у кого средний балл выше заданного значения. Если таких студентов нет, вывести соответствующее сообщение. Для вычисления среднего балла использовать функцию, вычисляющую среднее значение целочисленного массива из n элементов.
9. Определить структуру с именем Truck (грузовой автомобиль), содержащую следующие поля: гос. номер, марка, год выпуска, грузоподъёмность, габариты кузова (массив из трёх элементов — длина, ширина, высота). Имеется массив из 12 грузовиков. Вывести гос. номер и марку тех, у которых объём кузова не ниже заданного значения (либо сообщение, что таких машин нет).
10. Определить структуру с именем City (город), содержащую следующие поля: название, географические координаты центра (внутренняя структура с двумя полями — широта и долгота), среднемесячная температура (массив из 12 элементов — на каждый месяц года). Имеется массив из 6 городов. Вывести названия и координаты тех, у которых среднемесячная температура мая ниже среднемесячной температуры октября. Если таких городов нет, вывести об этом сообщение.

Более полный список задач по всем разделам настоящего пособия можно найти в [5].