

4.6. Составные типы данных в C++

4.6.1. Массивы

В языке C/C++, кроме базовых типов, разрешено вводить и использовать производные типы, полученные на основе базовых. Стандарт языка определяет три способа получения производных типов:

- 1) массив элементов заданного типа;
- 2) указатель на объект заданного типа;
- 3) функция, возвращающая значение заданного типа.

Массив – это упорядоченная последовательность переменных одного типа. Каждому элементу массива отводится одна ячейка памяти. Элементы одного массива занимают последовательно расположенные ячейки памяти. Все элементы имеют одно имя – имя массива и отличаются индексами – порядковыми номерами в массиве. *Количество элементов в массиве называется его размером.*

В математике понятию массив соответствуют понятия вектора и матрицы. Различают одно- и многомерные массивы. Двумерный массив данных (матрица) – это таблица, которая состоит из двух строк. Чтобы отвести в памяти нужное количество ячеек для размещения массива, надо заранее знать его размер. Резервирование памяти для массива выполняется на этапе компиляции программы.

Определение массива в C/C++

Для заданного типа *type* объявление массива: **type array_name[size];**

формирует в памяти массив из *size* элементов типа *type*.

Элементы индексируются от 0 до **size-1**, т.е.

v[10] – это *v[0]...[9]*.

int a[100]; // массив из 100 элементов целого типа

Операция *sizeof(a)* даст результат 400, т.е. 100 элементов по 4 байта.

Для обращения к значению элемента массива задаётся имя массива (*array_name*) и индекс элемента массива (*index*) в квадратных скобках: **array_name[index]**

a[0] – индекс задается как константа,

a[55] – индекс задается как константа,

a[I] – индекс задается как переменная,

*a[2*I]* – индекс задается как выражение.

Элементы массива можно задавать при его определении:

int v1[4] = {1, 2, 3, 4};

char v2[4] = {'a', '', 'c', 'l'};*

Когда массив объявлен без указания размера, но при этом инициализируется списком, размер массива вычисляется путём подсчёта числа элементов этого списка:

int v3[] = {-10, 2, -35, 4};

char v4[] = {'(', 'b', '=', 'd'};

Здесь *v3* и *v4* - массивы из четырех символов. Длина массива вычисляется компилятором по количеству значений, перечисленных при инициализации.

Если в списке инициализации недостаёт элементов, всем остальным элементам присваиваются нулевые значения. Например:

```
int v5[8] = {1, 2, 3, 4};    равнозначно    int v5[8] = {1, 2, 3, 4, 0, 0, 0, 0};
```

Если размер задан явно, присваивать большее число элементов нельзя.

Массив символов можно задавать в виде строки инициализатора:

```
char alpha[] = "abcdefghijklmnopqrstuvwxyz";
```

Это удобный и единственный способ подобного применения строк. Не нужно указывать размер массива *alpha*: компилятор установит его, подсчитав число символов в строке, заданной в качестве инициализатора.

Присваивание строки массиву недопустимо, поскольку в языке присваивание массивам не определено, например:

```
char v[9];  
v = "a string";    // ошибка!
```

Многомерные массивы представляются как массивы массивов.

```
int d2[10][20];    // d2 является массивом из 10 элементов по 20 элементов каждый  
char v[2][5] = { {'a', 'b', 'c', 'd', 'e'}, {'0', '1', '2', '3', '4'} };
```

Однако нельзя при задании граничных значений индексов использовать, как это делается в некоторых языках, запятую. Например:

```
int v[5][2];        // правильно  
int badv[4, 1];     // ошибка!
```

Пример 45. Ввод-вывод массива

```
int main()  
{  
    SetConsoleOutputCP(1251);  
    const int k = 20;  
    int a[k];  
    //Ввод массива  
    cout << "Введите элементы массива :\n";  
    for (int i = 0; i < k; i++)  
        cin >> a[i];  
    //Вывод массива  
    cout << "Массив:\n";  
    for (int i = 0; i < k; i++)  
        cout << a[i] << " ";  
}
```

Пример 46. Копирование 10-ти элементов из одного массива в другой:

```
void another_function()  
{  
    int v1[10];  
    int v2[10];  
    for (int i = 0; i < 10; ++i)  
        v1[i] = v2[i];  
}
```

Пример 47. Ввод-вывод матрицы

```
int main()
{
    SetConsoleOutputCP(1251);
    const int k = 3;
    int a[k][k];
    cout << "Введите элементы матрицы " << k << "x" << k << ":\n";
    for (int i = 0; i < k; ++i)
        for (int j = 0; j < k; ++j)
            cin >> a[i][j];
    cout << "Матрица:\n";
    for (int i = 0; i < k; ++i)
    {
        for (int j = 0; j < k; ++j)
            cout << a[i][j] << " ";

        cout << '\n';
    }
}
```

Обработка одномерных массивов

При работе с массивами часто требуется одинаково обрабатывать все элементы или часть элементов массива. Для этого организуется перебор массива.

Перебор элементов массива характеризуется:

- 1) направлением перебора;
- 2) количеством одновременно обрабатываемых элементов;
- 3) характером изменения индексов.

По направлению перебора массивы обрабатывают:

- 1) слева направо (от начала массива к его концу);
- 2) справа налево (от конца массива к началу);
- 3) от обоих концов к середине.

Индексы могут меняться:

- 1) линейно (с постоянным шагом);
- 2) нелинейно (с переменным шагом).

Перебор массива по одному элементу

Элементы можно перебирать:

1. Слева направо с шагом 1, используя цикл с параметром:

```
for (int I=0; I<n; I++)    { обработка a[I]; }
```

2. Слева направо с шагом отличным от 1, используя цикл с параметром:

```
for (int I=0; I<n; I+=step)    { обработка a[I]; }
```

3. Справа налево с шагом 1, используя цикл с параметром:

```
for (int I=n-1; I>=0; I--)    { обработка a[I]; }
```

4. Справа налево с шагом отличным от 1, используя цикл с параметром:

```
for (int I=n-1; I>=0; I-=step) { обработка a[I]; }
```

5. От обоих концов к середине:

```
for (int I=0, J=n-1; I<J; I++, J--) { обработка a[I]; }
```

Перебор массива по два элемента

1) Элементы массива можно обрабатывать по два элемента, двигаясь с обеих сторон массива к его середине:

```
int I=0, J=N-1;
while( I<J)
{обработка a[I] и a[J];
 I++;J--;}

```

2) Элементы массива можно обрабатывать по два элемента, двигаясь от начала к концу с шагом 1 (т.е. обрабатываются пары элементов a[1]и a[2], a[2]и a[3] и т.д.):

```
for (I=0;I<N-1;I++)
{обработка a[I] и a[I+1]}
```

3) Элементы массива можно обрабатывать по два элемента, двигаясь от начала к концу с шагом 2 (т.е. обрабатываются пары элементов a[1]и a[2], a[3]и a[4] и т.д.)

```
int I=0;
while (I<N-1 )
{обработка a[I] и a[I+1];
 I+=2;}

```

Формирование псевдодинамических массивов

Псевдодинамические массивы реализуются следующим образом:

1) при определении массива выделяется достаточно большое количество памяти:

```
const int MAX_SIZE=100;           //именованная константа
int mas[MAX_SIZE];

```

2) пользователь вводит реальное количество элементов массива меньшее MAX_SIZE:

```
int n;
cout<<"\nEnter the size of array<"<<MAX_SIZE<<": ";
cin>>n;

```

3) дальнейшая работа с массивом ограничивается заданной пользователем размерностью n (рис. 19).



Рис. 19. Представление псевдодинамического массива

Таким образом, используется только часть массива.

Использование датчика случайных чисел для формирования массива

Простейший ДСЧ работает следующим образом:

1) Берется большое число K и произвольное $x_0 \in [0,1]$.

2) Формируются числа $x_1 = \text{дробная_часть}(x_0 * K)$;

$x_2 = \text{дробная_часть}(x_1 * K)$; ... $x_i = \text{дробная_часть}(x_{i-1} * K)$.

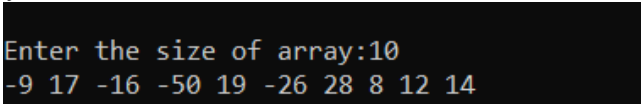
В результате получается последовательность чисел x_0, x_1, x_2, \dots беспорядочно разбросанных по отрезку от 0 до 1. Их можно считать случайными, а точнее псевдослучайными. Реальные ДСЧ реализуют более сложную функцию $f(x)$.

В C++ имеется специальная функция

`int rand()` – возвращает псевдослучайное число из диапазона 0 ... RAND_MAX=32767, описание функции находится в файле `<stdlib.h>`.

Пример 47. Формирования и печати массива с помощью ДСЧ:

```
#include<iostream>
#include<stdlib.h>
using namespace std;
void main()
{
    int a[100];
    int n;
    cout<<"\nEnter the size of array:";
    cin>>n;
    for(int I=0; I<n; I++)
    {
        a[I]=rand()%100-50;
        cout<<a[I]<<" ";
    }
}
```



Enter the size of array:10
-9 17 -16 -50 19 -26 28 8 12 14

В этой программе используется перебор массива по одному элементу слева направо с шагом 1.

Классы задач по обработке массивов

Задачи 1-го класса

К задачам 1 класса относятся задачи, в которых выполняется однотипная обработка всех или указанных элементов массива.

Решение таких задач сводится к установлению того, как обрабатывается каждый элемент массива или указанные элементы, затем подбирается подходящая схема перебора, в которую вставляются операторы обработки элементов массива.

Пример 35. Нахождение среднего арифметического массива.

```
void main()
{
    setlocale (LC_CTYPE,"rus");
    int a[100];
    int n,I;
    // формирование массива
    cout<<"\nEnter the size of array:";
    cin>>n;
    for ( I=0; I<n; I++)
    {
        a[I]=rand()%100-50;
        cout<<a[I]<<" ";
    }
    // Нахождение суммы элементов массива
    double Sum=0;
```

```

for (I=0; I<n; I++)
Sum+=a[I];
cout<<"Среднее арифметическое="<<Sum/n;
}

```

```

Enter the size of array:5
-9 17 -16 -50 19 Среднее арифметическое=-7.8

```

Пример 36. Нахождение максимального элемента массива и его позиции.

```

a {28 5 -4 12 2 -6 25 13}
I 0 1 2 3 4 5 6 7

```

```
min -6
```

```
nMin 5
```

```

void main()
{
setlocale (LC_CTYPE,"rus");
int a[100];
int n,I;
// формирование массива
cout<<"\nEnter the size of array:";
cin>>n;
for ( I=0; I<n; I++)
{
a[I]=rand()%100-50;
cout<<a[I]<<" ";
}
// Нахождение максимального элемента массива и его позиции
int max=a[0],nMax=0;
for (I=1;I<n;I++)
if (a[I]>max) {max=a[I]; nMax=I;}
cout<<"\n Max="<<max<<" его номер в массиве "<<nMax+1;
}

```

```

Enter the size of array:10
-9 17 -16 -50 19 -26 28 8 12 14
Max=28 его номер в массиве 7

```

Пример 37. Найти сумму элементов массива с четными индексами.

```

void main()
{
int a[100];
int n;
cout << "\nEnter the size of array:";
cin >> n;
for (int I = 0; I < n; I++) //ВВОД массива

```

```

{
    a[I] = rand() % 100 - 50;
    cout << a[I] << " ";
}
int Sum = 0;
for (int I = 0; I < n; I += 2)
    Sum += a[I]; //элементы с индексами 0, 2, 4...
cout << "\nSum = " << Sum;
}

```

```

Enter the size of array:8
-9 17 -16 -50 19 -26 28 8
Sum = 22

```

//Второй способ

```

for (I = 0; I < n; I++)
    if (I % 2 == 0) Sum += a[I]; //элементы с индексами 0, 2, 4...
    cout << "\nSum = " << Sum;

```

Задачи 2-го класса

К задачам 2 класса относятся задачи, в которых изменяется порядок следования элементов массива.

Обмен элементов внутри массива выполняется с использованием вспомогательной переменной:

```
int R=a[I];a[I]=a[J]; a[J]=R; // обмен a[I] и a[J] элементов массива.
```

Пример 38. Перевернуть массив.

```

//формирование массива
for (int i=0, j=n-1; i<j; i++, j--)
{int r=a[i];
a[i]=a[j];
a[j]=r;}
//Вывод массива

```

```

Enter the size of array:5
-9 17 -16 -50 19
New array:
19 -50 -16 17 -9

```

Пример 39. Поменять местами пары элементов в массиве: 1 и 2, 3 и 4, 5 и 6 и т.д.

```

//формирование массива
for (int i=0; i<n-1; i+=2)
{int r=a[i];
a[i]=a[i+1];
a[i+1]=r;}
//Вывод массива

```

```

Enter the size of array:6
-9 17 -16 -50 19 -26
New array:
17 -9 -50 -16 -26 19

```

Пример 40. Циклически сдвинуть массив на k элементов влево (вправо).

```

//формирование массива
int k,t,r;
cout<<"\nK=?";
cin>>k;
for (t=0; t<k; t++)
{
r=a[0];
for (int i=0; i<n-1; i++)
a[i]=a[i+1];
a[n-1]=r;
}
//Вывод массива

```

```

Enter the size of array:5
-9 17 -16 -50 19
K=? 1

New array:
17 -16 -50 19 -9

```

Задачи 3-го класса

К задачам 3 класса относятся задачи, в которых выполняется обработка нескольких массивов или подмассивов одного массива. Массивы могут обрабатываться по одной схеме – синхронная обработка или по разным схемам – асинхронная обработка массивов.

При синхронной обработке массивов индексы при переборе массивов меняются одинаково.

Пример 41. Заданы два массива из n целых элементов. Получить массив c , где $c[i]=a[i]+b[i]$.

```

//формирование массива a и b
for (int i=0; i<n; i++) c[i]=a[i]+b[i];
//Вывод массива c

```

```

Enter the size of array: 6

array a:
-9 17 -16 -50 19 -26
array b:
28 8 12 14 -45 -5
New array c:
19 25 -4 -36 -26 -31

```

При асинхронной обработке массивов индекс каждого массива меняется по своей схеме.

Пример 42. В массиве целых чисел все отрицательные элементы перенести в начало массива.

```

//формирование массива a
int b[12]; //вспомогательный массив
int i, j=0;
for (i=0; i<n; i++)

```



```

if (a[i]<0){b[j]=a[i]; j++;} //переписываем из а в b все отрицательные элементы
for (i=0; i<n; i++)
if (a[i]>=0){b[j]=a[i]; j++;} // переписываем из а в b все положительные элементы
for (i=0; i<n; i++) cout<<b[i]<< " "; //вывод массива b

```

```

Enter the size of array:7

array a:
-9 17 -16 -50 19 -26 28
array b:
-9 -16 -50 -26 17 19 28

```

Пример 43. Удалить из массива все четные числа

```

//формирование массива a
int b[12];
int i, j=0;
for (i=0; i<n; i++)
if (a[i]%2!=0) {b[j]=a[i]; j++;} //переписываем из а в b все нечетные элементы
for (i=0; i<j; i++) cout<<b[i]<< " "; //вывод массива b
cout<<"\n";

```

```

Enter the size of array:8

array a:
-9 17 -16 -50 19 -26 28 8
array b:
-9 17 19

```

Задачи 4-го класса

К задачам 4 класса относятся задачи, в которых требуется отыскать первый элемент массива, совпадающий с заданным значением – поисковые задачи в массиве.

В поисковых задачах требуется найти элемент, удовлетворяющий заданному условию. Для этого требуется организовать перебор массива и проверку условия. Но при этом существует две возможности выхода из цикла:

- 1) нужный элемент найден;
- 2) элемент не найден, но просмотр массива закончен.

Пример 44. Найти первое вхождение элемента K в массив целых чисел.

```

//формирование массива a
int k;
cout<<"\nK=?";
cin>>k;
int ok=0; //признак найден элемент или нет
int i,nom;
for(i=0; i<n; i++)
if(a[i]==k){ok=1; nom=i; break;}
if(ok==1)
cout<<"\nnom="<<nom;
else
cout<<"\nthere is no such element! ";

```

```
Enter the size of array: 8
```

```
array a:
```

```
-9 17 -16 -50 19 -26 28 8
```

```
K=? -50
```

```
nom=3
```

```
Enter the size of array: 8
```

```
array a:
```

```
-9 17 -16 -50 19 -26 28 8
```

```
K=? 5
```

```
there is no such element!
```