Основы программирования

Статические массивы в C/C++

Описание и индексация

Общий формат описания одномерного статического массива: **`тип имя_массива[длина];`**

тип – тип отдельных элементов

имя_массива – идентификатор (имя переменной)

длина – число элементов (константа)

Примеры описания и использования:

```
int i, j, k, a[100]; double x[20], y[10];
j = a[i]; cin >> a[5]; a[i+1] = k + 1;
a[j++] = a[k++];
if (a[k] < 0) a[k] = 0;
cout << x[i]; y[k+2] = x[i-1] * 2.71;
if (y[i] >= a[i]) cout << y[i] << endl;
```

Пример ввода и вывода массива

```
#define N 10
…
…
int i, arr[N];
for (i = 0; i < N; i++)
    cin >> arr[i];
i = 0;
while (i < N)
    cout << a[i++] << ” ”;
```

Генерация случайных чисел

Компьютер – детерминированная система, он в принципе не допускает никаких случайностей, случайности он может лишь имитировать!

Для получения последовательности псевдослучайных чисел (выглядит, как случайная) в C++ определены:

- константа **RAND_MAX** (обычно она равна 32767)
- стандартная функция **rand()** , которая генерирует случайные целые числа по формуле:

  $x_{i+1} = (a \cdot x_i + c) \bmod (\text{RAND\_MAX}+1)$,

  где a, c – особые константы (целые числа),

  x_i , x_{i+1} – предыдущий и последующий элементы последовательности, $0 \le x_i \le \text{RAND\_MAX}$.
- стандартная функция **srand(rand_value)**, которая задает начальный элемент последовательности

Примеры генерации

```
#include <ctime>
int k, mas[20]; double arr[50];
```

1. Целые числа в диапазоне [0, RAND_MAX] :

```
for (k = 0; k < 20; k++)
   mas[k] = rand();
```

2. Целые числа в диапазоне [10, 30]:

```
srand(7);
for (k = 0; k < 20; k++)
   mas[k] = rand() % 21 + 10;
```

3. Вещественные числа в диапазоне [0.0, 1.0]:

```
srand(time(0));
for (k = 0; k < 50; k++)
   arr[k] = (double)(rand()) / RAND_MAX;
```

Использование части массива

```
#define N 10000
#include <ctime>
…
int n, x[N], a, b;
cout << ”Input array length: ”;
cin >> n;
cout << ”Input value range: ”;
cin >> a >> b;
srand(time(0));
for (int k = 0; k < n; k++)
    x[k] = rand() % (b – a + 1) + a;
```

Пример: сумма элементов массива

Сумму S_n n элементов массива x можно представить рекуррентным соотношением:

$$\begin{cases} S_0 = 0, \\ S_i = S_{i-1} + x_{i-1}, \quad i = 1, 2, ..., n. \end{cases}$$

Алгоритм (i – число элементов в сумме):

```
for (S = 0, i = 1; i <= n; i++)
    S += x[i-1];
```

или эквивалентный:

```
S = i = 0;
while (i < n)
{ S += x[i]; i++; }
```

Трудоемкость $T(n) = O(n)$

Пример: поиск минимума в массиве

Для **минимального элемента** $minval_n$ в массиве x длины n выполняется рекуррентное соотношение:

$$\begin{cases} minval_1 = x_0, \\ minval_i = \min(minval_{i-1}, x_{i-1}), \quad i = 2, \ldots, n. \end{cases}$$

Алгоритм (переменная `minval` содержит текущее значение минимума):

```
minval = x[0];
for (i = 1; i < n; i++)
    if (minval > x[i]) minval = x[i];
```

Трудоемкость $T(n) = O(n)$

Примеры тестов

Тесты по методу ***черного ящика*** (внутренняя структура программы неизвестна):

- минимальное **n=1**, например, **x[1]=10;**
- **n** на 1 больше минимального, **n=2**, например:
  1) **x[0]=10, x[1]=5;**
  2) **x[0]=5, x[1]=10;**
- **n** большее, например, **n=4**, например:
  1) **x[0]=10, x[1]=5, x[2]=1, x[3]=-2;**
  2) **x[0]=-5, x[1]=0, x[2]=1, x[3]=7;**
  3) **x[0]=5, x[1]=5, x[2]=5, x[3]=5;**

  (значения по убыванию, по возрастанию, одинаковые)

Примеры тестов

Тесты по методу ***белого ящика*** (на основе известной внутренней структуры программы):

- такое **n**, чтобы цикл ни разу не выполнялся, **n=1**;
- **n=2**, чтобы цикл выполнился 1 раз, такой массив, чтобы:
  1) условие **minval > x[i]** было истинным;
  2) условие **minval > x[i]** было ложным;
- **n** большее, например, **n=4**, такой массив, чтобы:
  1) условие **minval > x[i]** всегда было истинным;
  2) условие **minval > x[i]** всегда было ложным;

Пример: поиск номера минимального элемента

Данный алгоритм практически совпадает с алгоритмом поиска минимального значения. Нужно только учесть связь номера **nmin** и значения **minval** минимального элемента массива **x**: **minval = x[nmin]**.

Алгоритм (переменная **nmin** содержит текущее значение **номера** минимального элемента):

```
nmin = 0;
for (i = 1; i < n; i++)
    if (x[nmin] > x[i]) nmin = i;
```

Трудоемкость $T(n) = O(n)$

Полином от x степени n в виде формулы Горнера

$$P_n(x) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0 =$$

$$= (\ldots(a_n x + a_{n-1})x + \ldots + a_1)x + a_0$$

где $a_n, a_{n-1}, \ldots, a_1, a_0$ – коэффициенты

Рекуррентное соотношение:

$$\begin{cases} P_0 = a_n, \\ P_i = P_{i-1} x + a_{n-i}, \quad i = 1, 2, \ldots, n. \end{cases}$$

Алгоритм:

```
P = a[n];
for (i = 1; i <= n; i++)
  P = P * x + a[n-i];
```

Трудоемкость $T(n) = O(n)$

Позиционные системы счисления

Любое целое неотрицательное число имеет различные представления в разных позиционных системах счисления.

В каждой системе счисления определены:

- основание системы p
- “цифры” – целые числа в диапазоне $[0, p\text{-}1]$.

Если целое число $V \geq 0$ представляется в системе с основанием p последовательностью цифр $a_n a_{n-1} \ldots a_1 a_0$, $0 \leq a_i < p$, то это значит, что значение числа равно:

$$V = a_n p^n + a_{n-1} p^{n-1} + \cdots + a_1 p + a_0.$$

Примеры систем счисления

Десятичная: $p = 10$, цифры 0,1,…,8,9

Двоичная: $p = 2$, цифры 0,1

Шестнадцатеричная: $p = 16$,

цифры 0,1,…, 8, 9, A, B, C, D, E, F (буквенные обозначения для “цифр” от 10 до 15)

Примеры представления чисел:

Десятичные	Двоичные (8 бит)	16-ричные	Константы в С
10	00001010	A	0x0A
30	00011110	1E	0x1E
64	01000000	40	0x40
127	01111111	7F	0x7F
240	11110000	F0	0xF0

Целые неотрицательные числа

- Числа типа `int` занимают 4 байта. Число является неотрицательным, если старший (самый левый) бит в его записи равен 0. **Максимально возможное число** содержит 1 нуль и 31 единицу, его значение:

  $2^{31}-1 = 2147483647 = 0x7FFFFFFF$

- Числа типа `unsigned int` всегда неотрицательные. **Максимальное значение**:

  $2^{32}-1 = 4294967295 = 0xFFFFFFFF$

- Числа типа `short int` аналогичны `int`, но занимают 2 байта. **Максимальное значение**:

  $2^{15}-1 = 32767 = 0x7FFF$

Отрицательные целые числа

- Если старший бит числа типа **`int`** равен 1, то считается, что число является отрицательным и заданным в дополнительном коде.

Перевод числа N = 32 в дополнительный код для получения представления **–N** (в пределах байта):

- в двоичном виде N = 00100000
- инвертируем все биты числа и получаем 11011111
- прибавляем 1 к этому значению, получим 11100000

Минимальное целое число = –2147483648, единица и 31 нуль или 0x80000000 в дополнительном коде

Максимальное отрицательное число = –1, 32 единицы или 0xFFFFFFFF в дополнительном коде

Вычисление цифр $a_0, a_1, \ldots, a_n$ целого числа $V_n > 0$ в системе счисления с основанием p

Рекуррентное соотношение:

$$\begin{cases} a_i = V_{n-i} \bmod p, \\ V_{n-i-1} = V_{n-i} / p, \qquad i = 0, 1, \ldots, n, \ V_{n-i} > 0, \end{cases}$$

Алгоритм:

```
for (i = 0; V > 0; i++)
{
    a[i] = V % p;
    V /= p;
}
n = i - 1;
```

Двумерные статические массивы

Общий формат описания двумерного статического массива (матрицы):

```
тип имя_массива[число_строк][число_столбцов];
```

тип – тип отдельных элементов

имя_массива – идентификатор (имя переменной)

число_строк и **число_столбцов** – константы

Примеры описания и использования:

```
int i, j, k, a[10][10]; double x[20][5], z;
k = a[i][j]; cin >> a[5][0];
a[i][i+2] = k + 1; a[i][j] = a[j][i];
if (a[k][k] < 0) a[k][k] = 0;
cout << x[i+1][4]; z = x[i+j][j+2] * 2.71;
if (y[i] >= a[i]) cout << y[i] << endl;
```

Использование части массива

```
#define ROW 100
#define COL 100
…
int n, m, x[ROW][COL], a, b, i, j;
cout << "Number of rows and cols: ";
cin >> n >> m;
cout << "Value range: ";
cin >> a >> b;
for (i = 0; i < n; i++)
  for (j = 0; j < m; j++)
    x[i][j] = rand() % (b-a+1) + a;
```

Транспонирование квадратной матрицы

Исходная	***Транспонированная матрица***
11 12 13 14	11 21 31 41
21 22 23 24	12 22 32 42
31 32 33 34	13 23 33 43
41 42 43 44	14 24 34 44

Меняем элементы выше и ниже главной диагонали:

```
for (i = 0; i < n - 1; i++)
    for (j = i+1; j < n; j++)
    {
        z=x[i][j]; x[i][j]=x[j][i];
        x[j][i] = z;
    }
```

Трудоемкость: $T(n) = O(n^2)$