

Богатов Р.Н.

Программирование на языке высокого уровня

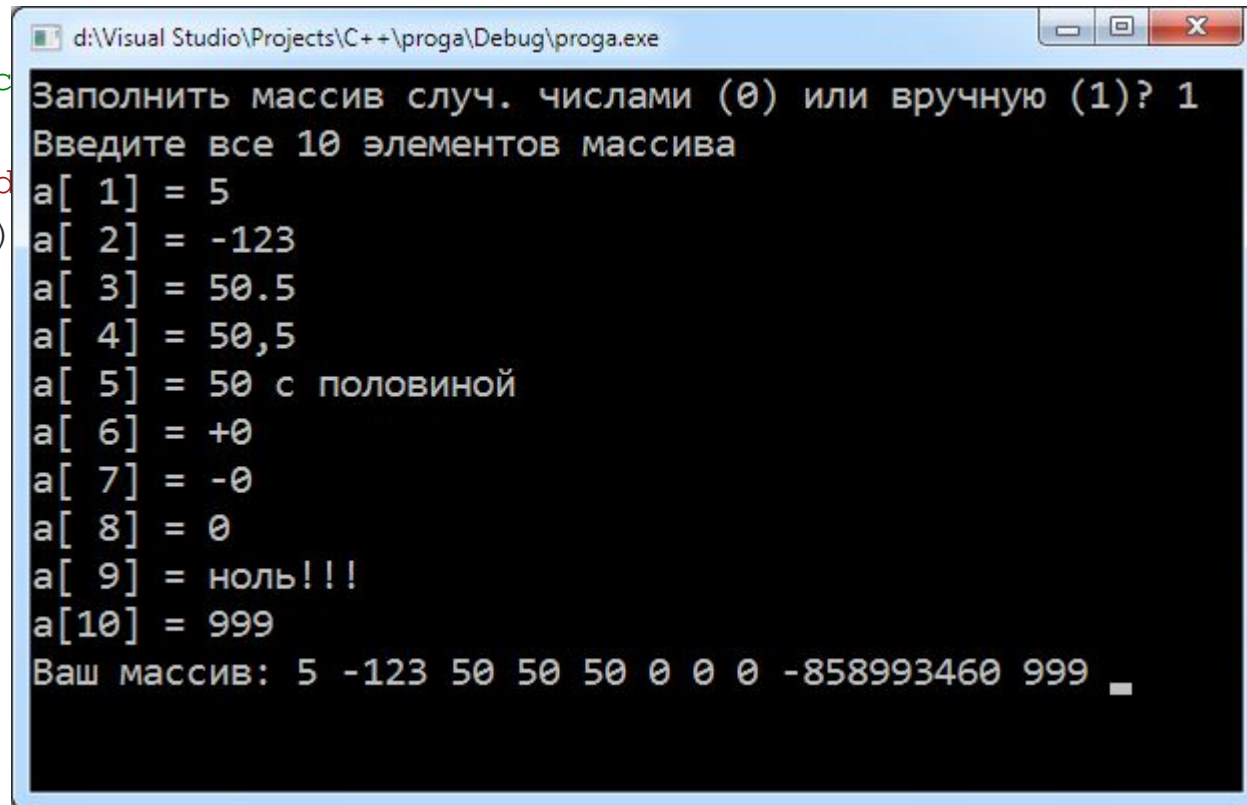
C++ ► Лекция 4.5 ► Примеры задач, которые
решаются с помощью массивов

Кафедра АСОИУ ОмГТУ, 2016

Инициализация массива

```
...
if (mode==0) // случайное заполнение массива
{
    srand( (unsigned)time(NULL) );
    for(int i=0; i<N; i++)
        a[i] = rand();
}
else // ручной ввод массива
{
    printf("Введите все %d элементов массива\n", N);
    for(int i=0; i<N; i++)
    {
        printf("a[%2d] = ", i);
        scanf("%d", &a[i]);
        fflush(stdin);
    }
}

// вывод массива
printf("Ваш массив: ");
for(int i=0; i<N; i++)
    printf("%d ", a[i]);
...
```



```
d:\Visual Studio\Projects\C++\proga\Debug\proga.exe
Заполнить массив случ. числами (0) или вручную (1)? 1
Введите все 10 элементов массива
a[ 1] = 5
a[ 2] = -123
a[ 3] = 50.5
a[ 4] = 50,5
a[ 5] = 50 с половиной
a[ 6] = +0
a[ 7] = -0
a[ 8] = 0
a[ 9] = ноль!!!
a[10] = 999
Ваш массив: 5 -123 50 50 50 0 0 0 -858993460 999
```

Задача про пятаки и трёшки

// прямое решение на основе остатка от деления

```
int A, B; // искомое число пятак и трешек
```

```
for (int i = 0; i < 5; i++)  
    if ((n - i * 3) % 5 == 0)  
        B = i;
```

```
A = (n - B * 3) / 5;
```

```
int A, B; // искомое число пятак и трешек
```

```
for (B = 0; ; B++)  
    if ((n - B * 3) % 5 == 0)  
        break;
```

```
A = (n - B * 3) / 5;
```

```
int A, B; // искомое число пятак и трешек
```

```
int G[5] = { 0, 2, 4, 1, 3 };  
B = G[n % 5];  
A = (n - B * 3) / 5;
```

```
else
```

```
{
```

```
    A = (n - 9) / 5;
```

```
    B = 3;
```

```
}
```

ег (натур

ного чис

// искомое

% 5;

0:

= n / 5;

0;

ak;

N = 5a+3b

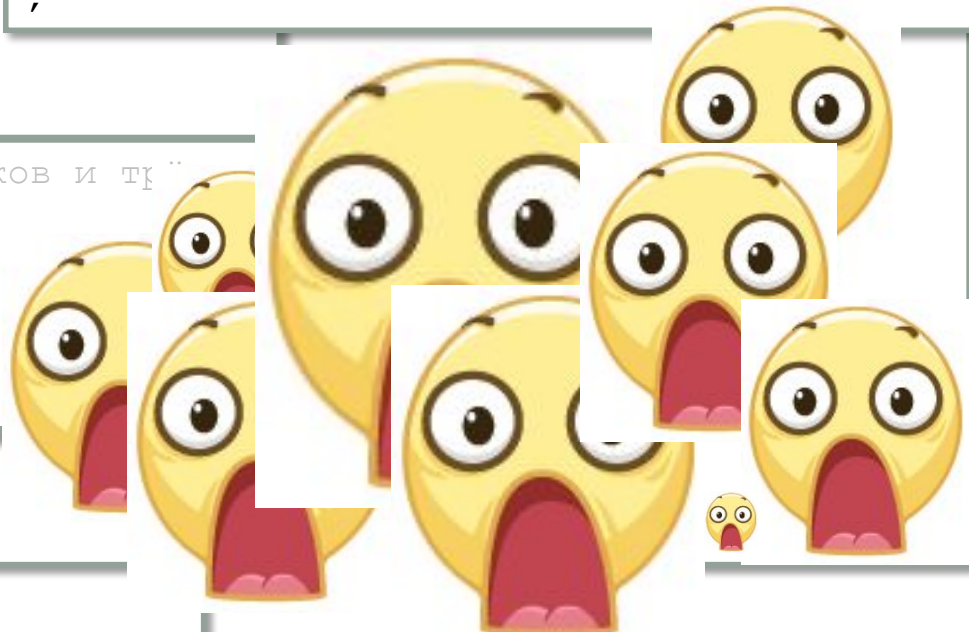
Введите сумму денег (>7):
107

Выдать

Итог: 19 пятак + 4 трешек

=

```
for (B = 0; (n - B * 3) % 5 != 0; B++)  
    ;
```



Произведение матриц

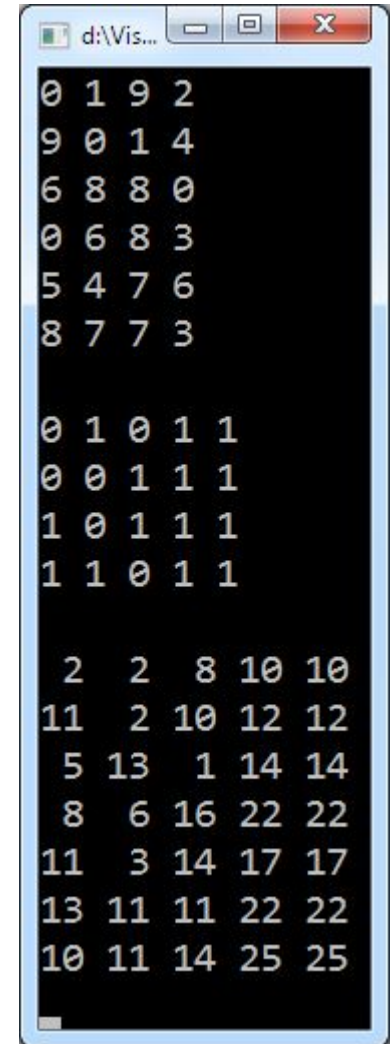
```
// произведение прямоугольных матриц константного размера
const int N=7, M=4, L=5;
int a[N][M], b[M][L], c[N][L];

// инициализация массива a
...
// вывод массива a на экран
...
// инициализация массива b
...
// вывод массива b на экран
...

// произведение матриц
for(int i=0; i<N; i++)
    for(int j=0; j<L; j++)
    {
        int s = 0;
        for(int k=0; k<M; k++)
            s += a[i][k]*b[k][j];

        c[i][j] = s;
    }

// вывод результата на экран
...
```



0	1	9	2
9	0	1	4
6	8	8	0
0	6	8	3
5	4	7	6
8	7	7	3
0	1	0	1
0	0	1	1
1	0	1	1
1	1	0	1
2	2	8	10
11	2	10	12
5	13	1	14
8	6	16	22
11	3	14	17
13	11	11	22
10	11	14	25

Перебор всех расстановок

Математическая постановка задачи: перестановки единиц в множестве из M элементов

Постановка задачи для программиста: перебор всех способов заполнения массива длины N нулями и единицами

```
// частный случай: получение всех расстановок единиц  
// в массиве длиной N<31
```

```
...  
for(int n=0; n<(1<<N); n++)  
{  
    for(int i=0; i<N; i++)  
        a[i] = (n & (1<<i))>0 ? 1 : 0;  
  
    // вывод массива  
    printf("\r%d: ", n);  
    for(int i=0; i<N; i++)
```

Вопросы:

- 1) Почему алгоритм для общего случая работает быстрее?
- 2) Почему во втором алгоритме массив нужно выводить до обработки массива?
- 3) Какой из алгоритмов удобнее взять за основу для решения задачи расстановки трёхзначных чисел?

```
// общий случай: получение всех расстановок  
// единиц в массиве произвольной длины
```

```
#include <stdio.h>  
#include <conio.h>  
  
const int N=10;  
int a[N];  
  
void main()  
{  
    for(int n=0; n<(1<<N); n++)  
    {  
        // вывод массива  
        printf("\r%d: ", n);  
        for(int i=0; i<N; i++)  
            printf("%d", a[i]);  
  
        for(int i=0; i<N; i++)  
            if (a[i]==1) // переполнение  
                a[i] = 0;  
            else // прибавление единицы  
            {  
                a[i] = 1;  
                break;  
            }  
    }  
    _getch();  
}
```