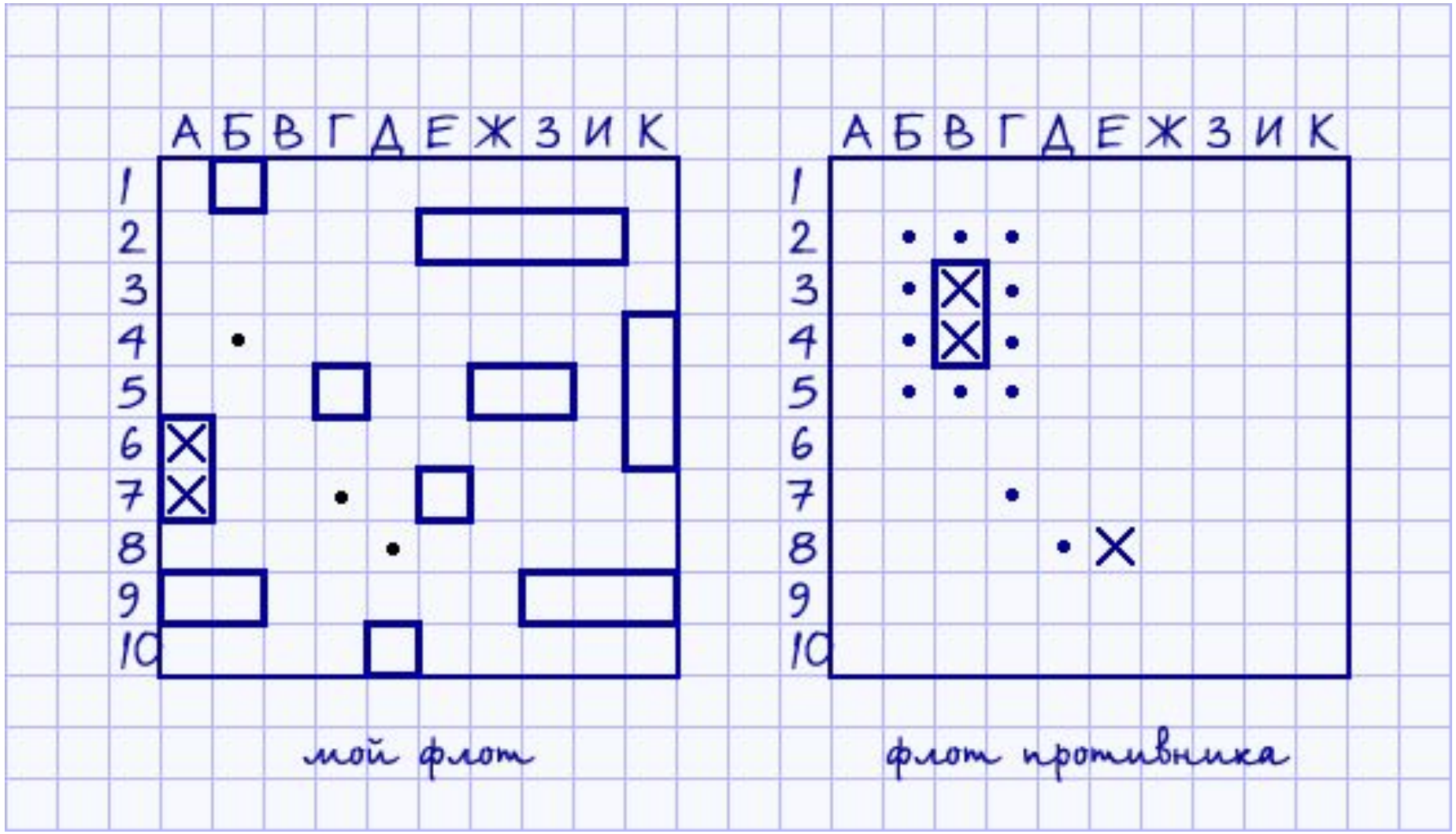
Многомерные массивы

Примеры двумерных массивов в окружающем мире

Примеры двумерных массивов в окружающем мире

Примеры двумерных массивов в окружающем мире



Статический двумерный массив

тип имя[кол-во_строк][кол-во_столбцов];

```
double dmas[3][4];
```

* удобно представлять как матрицу, но в памяти расположен как непрерывная строка значений

Инициализация статического двумерного массива

Полная:

```
double dmas[3][2]={{1,2},{3,4},{5,6}};
```

1	2
3	4
5	6

Частичная:

```
double dmas[3][2]={{1,2},{3,4}};
```

1	2
3	4
0	0

Еще частичная:

```
double dmas[3][2]={1,2,3,4};
```

1	2
3	4
0	0

И еще частичная:

```
double dmas[3][2]={{1},2,3,{4}};
```

1	0
2	3
4	0

Обращение к ячейке массива (индексация)

имя [номер_строки][номер_столбца]

```
const n = 5, m = 3;
float fmas[n][m];
/* здесь должно быть заполнение ячеек массива
  значениями*/

//замена значения в ячейке
fmas[2][1] = 1.4;
//запись значения ячейки в переменную t
float t = fmas[3][2];
```

* поскольку выделяется как строка, то операция индексации [][] вычисляет смещения исходя из количества столбцов и номера строки

Пример создания массива-алфавита

```
char alphabet[2][13] = {
  {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k',
  'l', 'm'},
  {'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w', 'x',
  'y', 'z'}
};
```

Ввод – вывод двумерного массива

```
const int n = 5, m = 3;
float fmas[n][m];
for (int i = 0; i < n; i++)
  for (int j = 0; j < m; j++)
    cin >> fmas[i][j];

… //работа с двумерным массивом

for (int i = 0; i < n; i++){
  for (int j = 0; j < m; j++)
    cout << fmas[i][j] << " ";
   cout << endl;
   }
```

Поиск минимального значения в двумерном массиве

```
const n = 5, m = 3;
float fmas[n][m];
for (int i = 0; i < n; i++)
   for (int j = 0; j < m; j++)
      cin >> fmas[i][j];
float min = fmas[0][0];
for (int i = 0; i < n; i++)
   for (int j = 0; j < m; j++)
      if (fmas[i][j] < min)
         min = fmas[i][j];
cout << min;
```

Посчитать средний балл сессии и число получаемых стипендий в группе

```
const int studCount = 20, exCount = 4;
int sum = 0, count = 0;
bool money;
unsigned marks[studCount][exCount];
for (int i = 0; i < studCount; i++){
   cout << "input marks for student №" << i << endl;
   money = true;
   for (int j = 0; j < exCount; j++){
      cin >> marks[i][j];
      sum += marks[i][j];
      if (money && marks[i][j] <= 3) money = false;
      }
   if (money) count++;
   }
cout << "avg:" << sum / exCount / studCount << endl
     << "lucky ones count: " << count;
```

Проход по диагоналям матриц

```
const int n = 4;
int s = 0, p = 1, mas [n][n];
/*…инициализация массива…*/
for (int i = 0; i < n; i++){
  s += mas[i][i];

  p *= mas[i][n – 1 – i];
  }
```

	j=0	j=1	j=2	j=3
i=0	1	2	3	4
i=1	5	6	7	8
i=2	9	10	11	12
i=3	13	14	15	16

Обход треугольников главной диагонали матрицы

```
const n = 4;
int s = 0, mas [n][n] = {/*инициализация*/};
for (int i = 0; i < n; i++)
  for (int j = i; j < n; j++)
    s += mas[i][j];
//задаем треугольник в if
for(int i = 0; i < n; i++)
  for(int j = 0; j < n; j++)
    if(i <= j)
          s += mas[i][j];
//задаем треугольник в for
for (int i = 0; i < n; i++)
  for (int j = 0; j <= i; j++)
    s += mas[i][j];
```

	j=0	j=1	j=2	j=3
i=0	1	2	3	4
i=1	5	6	7	8
i=2	9	10	11	12
i=3	13	14	15	16

	j=0	j=1	j=2	j=3
i=0	1	2	3	4
i=1	5	6	7	8
i=2	9	10	11	12
i=3	13	14	15	16

Обход треугольников побочной диагонали матрицы

```
const n = 4;
int s = 0, mas [n][n] = {/*инициализация*/};
for (int i = 0; i < n; i++)
  for (int j = 0; j <= n - 1 - i; j++)
    s += mas[i][j];
```

	i=0	i=1	i=2	i=3
i=0	1	2	3	4
i=1	5	6	7	8
i=2	9	10	11	12
i=3	13	14	15	16

```
for (int i = 0; i < n; i++)
  for (int j = n - 1 - i; j < n; j++)
    s += mas[i][j];
```

	i=0	i=1	i=2	i=3
i=0	1	2	3	4
i=1	5	6	7	8
i=2	9	10	11	12
i=3	13	14	15	16

Найти номер столбца, в котором находится наибольшее количество отрицательных элементов

```
const int n = 4, m = 5;
int i, j, b[n][m];
/*… ввод массива …*/
int col = 0, maxCount = 0;
for (i = 0; i < m; i++){ //просмотр по столбцам
  int count = 0;
  for (j = 0; j < n; j++)
    if (b[j][i] < 0) count++;
  if (count > maxCount){
    col = i ;
    maxCount = count;
    }
  }
if (maxCount)
  cout << col + 1;
else
  cout << "no negative elements";
```

Отсортировать столбцы матрицы размера 5x4 по возрастанию значений первой строки

4	3	1	2
0	1	2	3
-1	1	1	1
-2	2	2	2
-3	3	3	3

```
const int n = 5, m = 4;
int i, j, k, jmax, b[n][m];
/*… ввод массива …*/
for (i = 0; i < m - 1; i++){
  jmax = 0;
  for (j = 1; j < m - i; j++)
    if (b[0][j] > b[0][jmax]) jmax = j;
  if (jmax == m - 1 - i) continue;
  for (k = 0; k < n; k++){
    /*swap column jmax and (m-i-1)*/
    int tmp = b[k][m - 1 - i];
    b[k][m - 1 - i] = b[k][jmax];
    b[k][jmax] = tmp;
    }
  }
/*… вывод массива …*/
```

альтернатива

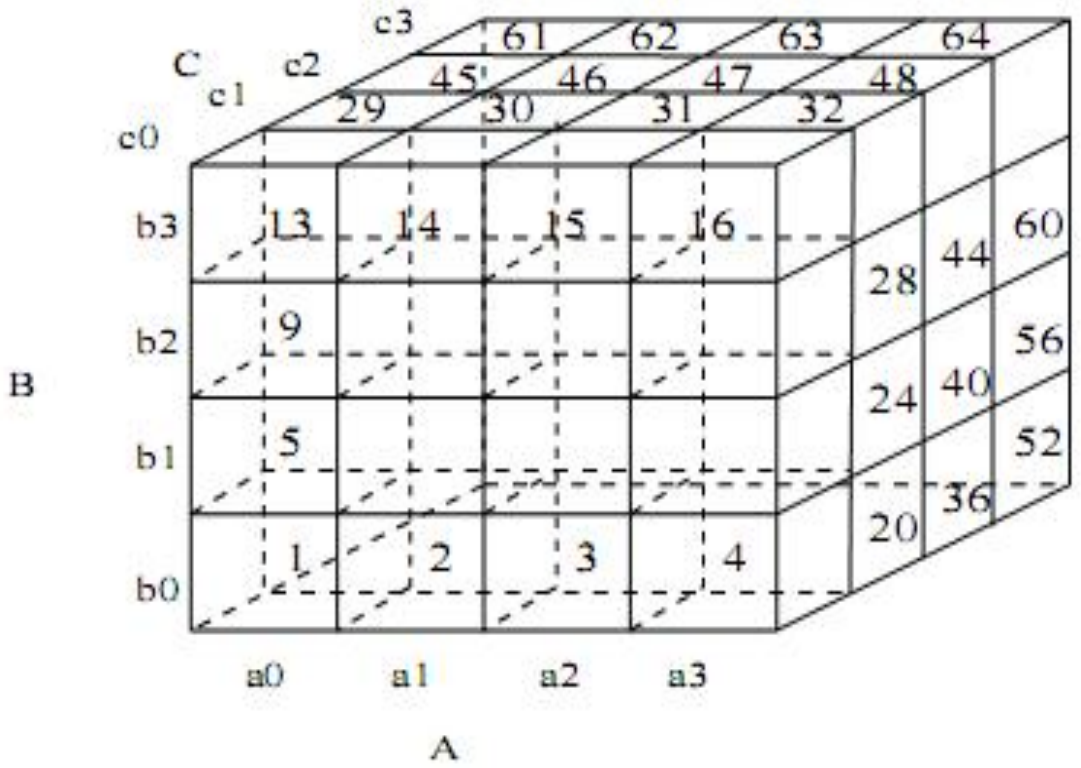
```
int tmp[n];
for(k = 0; k < n; k++)
   tmp[k] = b[k][m-1-i];
for(k = 0; k < n; k++)
   b[k][n-1-i] = b[k][jmax];
for(k = 0; k < n; k++)
   b[k][jmax] = tmp[k];
```

Поиск минимального значения в строке с максимальной суммой элементов

```
const int n = 5, m = 6;
int a [n][m] = {/*инициализация*/};
int maxsum = 0, sum, imaxsum = 0;
for (int i = 0; i < m; i++)
  maxsum += a[0][i];
for (int i = 1; i < n; i++){
    sum = 0;
    for (int j = 0; j < m; j++)
        sum += a[i][j];
    if (sum > maxsum){
        maxsum = sum;
        imaxsum = i;
        }
    }
int min = a[imaxsum][0];
for (int j = 1; j < m; j++)
    if (a[imaxsum][j] < min)
        min = a[imaxsum][j];
cout << min;
```

Трехмерный массив

```
const int A = 4, B = 4, C = 4;
int mas[C][B][A] = {
{ { 1, 2, 3, 4}, { 5, 6, 7, 8},
  { 9,10,11,12}, {13,14,15,16} },
{ {…}, {…},
  {…}, {…} },
{ {…}, {…},
  {…}, {…} },
{ {…}, {…},
  {…}, {…} }
};
```



Трехмерный массив

```
short m[3][4][2] =
{
  { {17,21}, {18,22}, {19,23}, {20,24} },
  { { 9,13}, {10,14}, {11,15}, {12,16} },
  { { 1, 5}, { 2, 6}, { 3, 7}, { 4, 8} }
};
```

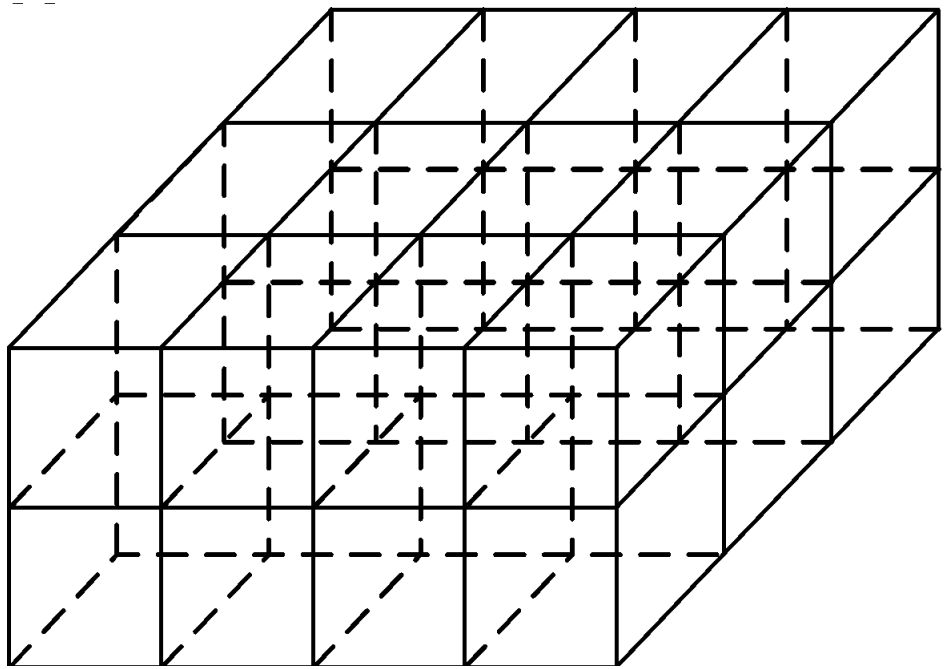
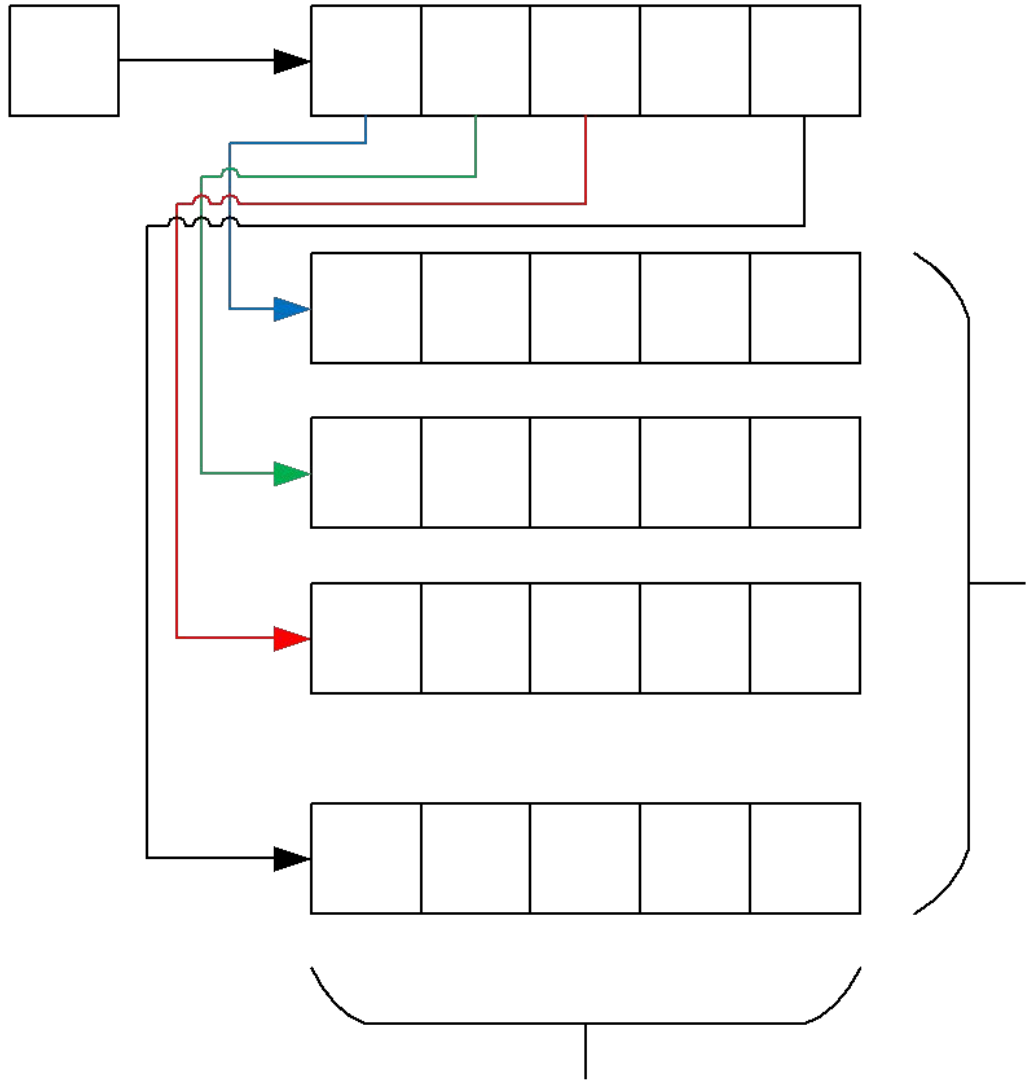


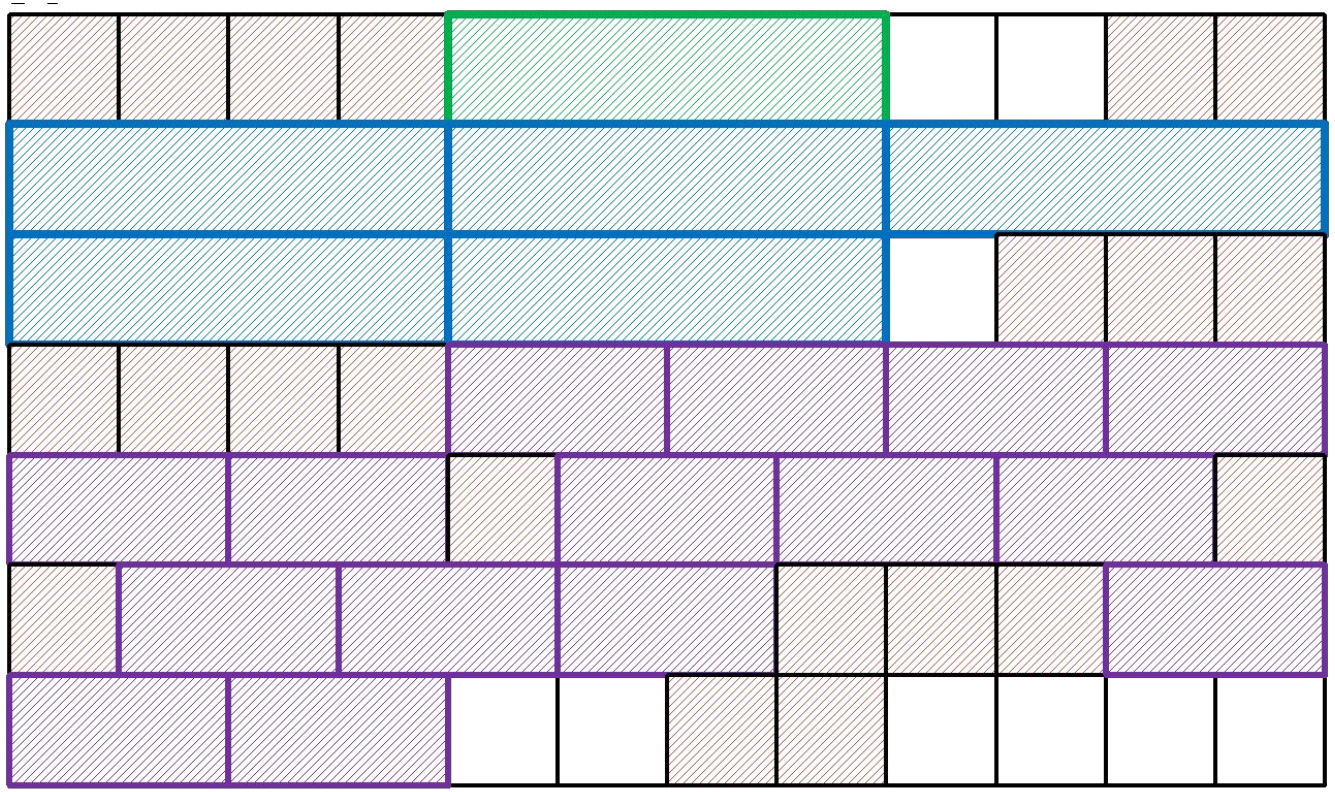
Схема расположения двумерного динамического массива в памяти



Объявление и выделение памяти для двумерного динамического массива

// объявление указателя на указатель
тип **имя;
// выделение блока памяти для массива указателей
имя = new тип *[кол-во_строк];
// инициализация каждого указателя из массива указателей
for (int i = 0; i < кол-во_строк; i++)
  // адресом начала блока памяти с данными
  имя[i] = new тип [кол-во_столбцов];

```
float **fmas = new float* [rows];
for (int i = 0; i < rows; i++)
 fmas[i] = new float [cols];
```



указатель на указатель

указатель на данные

данные

занятые ячейки

```
int n = 5, m = 3;
short **a = new short *[n];
for (int i = 0; i < n; i++)
   a[i] = new short [m];
for (int i = 0; i < n; i++)
   for (int j = 0; j < m; j++)
      a[i][j] = i * 10 + j + 1;
```

Освобождение памяти, занятой двумерным динамическим массивом

```
for (int i = 0; i < rows; i++)
  delete []fmas[i]; // освобождаем строки с данными
delete [] fmas; // освобождаем массив указателей на строки
```

Альтернативный способ выделения динамической памяти для работы с двумерным массивом

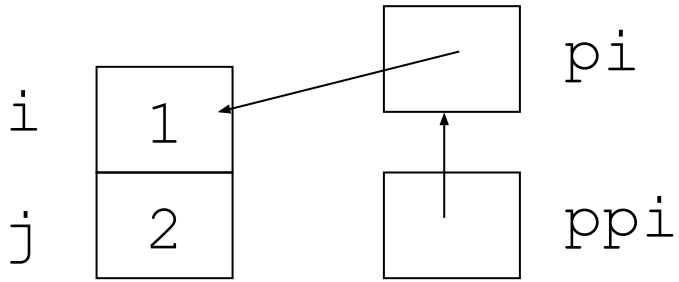
```
int ** mas = new int *[rows];
mas[0] = new int [rows * cols]; // единый блок
for (int i = 1; i < rows; i++) // инициализация указателей
   mas[i] = mas[i - 1] + cols;

//new работает 1 раз!!!

delete [] mas[0];
delete [] mas;
```

Указатель на указатель

```
int i = 1, j = 2, *pi, **ppi;
cout << i; // 1
pi = &i;
ppi = &pi;
i++;
(**ppi) = -5;
cout << i; // -5
*ppi = &j;
j++;
cout << **ppi; // 3
```



Альтернативный доступ к ячейке двумерного массива

```
int n, m, k, t;
cin >> n >> m >> k >> t;
int **imas = new int *[n];
for (int i = 0; i < n; i++)
  imas[i] = new int [m];
imas[k][t] = 0;
// *(imas[k] + t) = 0;
// *(*(imas+k) + t) = 1;
```

Многомерные массивы в языке C (malloc)

```
// первый способ – выделить строчку
int *mas = (int*) malloc(sizeof(int) * N * M);
// и вычислять индекс вручную
int a = mas[i * N + j];
//освобождать один блок
free(mas);

//второй способ – через массив указателей
int **mas = (int**) malloc(N * sizeof(int*));
for (int i = 0; i < N; i++)
   mas[i] = (int*) malloc (M * sizeof(int));
int a = mas[i][j];
for (int i = 0; i < N; i++)
   free(mas2[i]);
free(mas2);
```

Суффиксы и префиксы

"Суффикс" привязан крепче "префикса". Если при описании переменной используются одновременно префикс * (указатель) и суффикс [] (массив), то переменная интерпретируется как массив указателей, а не указатель на массив:

```
int *p[10]; // массив из 10 указателей
int *p[3][5];//матрица 3x5 указателей
```