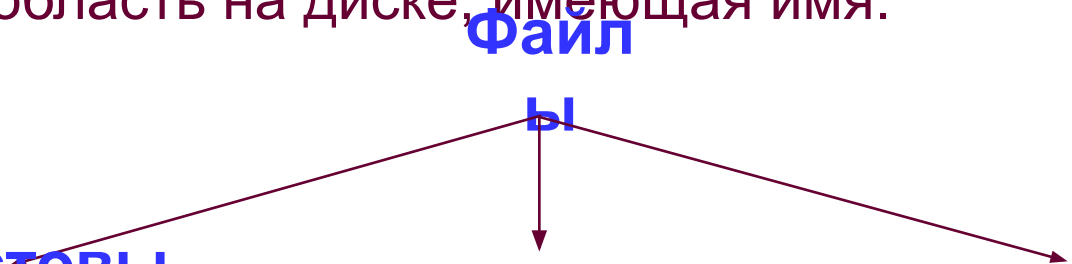
Лекция 2

Файлы (текстовые).
Массивы.

Файлы

Файл – это область на диске, имеющая имя.



только текст без оформления, не содержат управляющих символов (с кодами < 32)

ACSII (1 байт на символ)

UNICODE (2 байта на символ)

`*.txt, *.log,`
`*.htm, *.html`

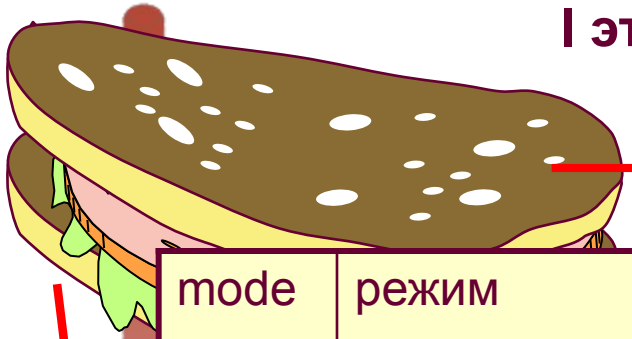
могут содержать любые символы кодовой таблицы

`*.doc, *.exe,`
`*.bmp, *.jpg,`
`*.wav, *.mp3,`
`*.avi, *.mpg`

Принцип сэндвича

Переменная типа файл:

```
FILE *fp;
```



I этап. открыть файл :

- связать переменную **f** с файлом и открыть его

```
fp= fopen(char *name, char *mode);
```

mode	режим	Действие	
		Файл существует	Файл не существует
“r”	чтение	Открытие файла	Ошибка!
“w”	запись	Содержимое файла уничтожится	Файл будет создан
“a”	добавление	Запись в конец файла	Файл будет создан

II этап: работа с файлом

```
Читать из файла или записывать в файл
```

III этап: закрыть файл

```
fclose(fp);
```

Работа с файлами

Особенности:

- имя файла упоминается только в команде **`open`**, обращение к файлу идет через файловую переменную
- файл, который открывается на чтение, должен **существовать**
- если файл, который открывается на запись, существует, старое содержимое **уничтожается**
- данные записываются в файл в текстовом виде, *если не включен режим записи в двоичном виде*
- при завершении программы все файлы закрываются автоматически
- после закрытия файла переменную **`f`** можно использовать еще раз для работы с другим файлом

Вывод в файл символа и строки

Запись символа в файл:

```
int fputc(int c, FILE *fp);
```

c - символ, который необходимо записать

fp - файл, в который производится запись

Возвращаемое значение: записанный символ или EOF в случае ошибки записи.

Запись строки в файл:

```
int fputs(char* s, FILE *fp)
```

s - строка, которую необходимо записать в файл

fp – файл, в который производится запись

Возвращаемое значение: неотрицательное значение или EOF в случае ошибки записи.

ВВОД И ВЫВОД СИМВОЛОВ

Посимвольный ввод	Параметры	**Результат**
int **getc**(FILE *fd)	Явно указанный файл	Код символа или EOF
int **getchar**(void)	Стандартный ввод (клавиатура)	
int **ungetc**(int ch, FILE *fd)	Возвратить символ в файл (повторно читается)	
Посимвольный вывод		
int **putc**(int ch, FILE fd)	Символ (переменная или константа) Явно указанный файл	Код символа или EOF
int **putchar**(int ch)	Стандартный вывод	

Пример ввода и вывода символов

```
void main()
{char c;
 FILE *fd1=fopen(«in.txt","r"); // открыть для чтения
 FILE *fd2=fopen(“out.txt","w");
// создание и открытие для записи
 if (fd1==NULL || fd2==NULL) return;
  while((c=getc(fd1))!=EOF)
   {  //читать символ, пока не конец файла
       putc(c,fd2);    // Вывести символ в файл
    }
     fclose(fd1); fclose(fd2);  // закрыть файлы
      }
```

Построчный ввод-вывод

Построчный ввод		**Параметры и результат**
char *fgets(char *str, int n, FILE *fd)	Явно указанный файл	n - максимальная длина строки str или NULL(ошибка)
char *gets(char *str)	Стандартный ввод	
Построчный вывод		
char *fputs(char *str, FILE *fd)	Явно указанный файл	str или NULL(ошибка)
char *puts(char *str)	Стандартный вывод	

Строка символов

Строка в памяти – массив символов заданной размерности, ограниченной символом '\0' – конец строки

```
ффф 12 ert   5       45t  gyy 67 56\0
```

Строка в файле (потоке) последовательность символов произвольной длины, ограниченной символом '\n' – конец строки

```
ффф 12 ert   5       45t  gyy 67 56●
```

- при вводе-выводе строк символов '\n' уничтожается при стандартном вводе-выводе

gets - не записывает в строку, а puts автоматически добавляет при выводе '\n' сохраняется в строке при вводе-выводе из файла

fgets - записывает в строку,

fputs - выводит имеющийся в строке (сам не добавляет);

- при построчном вводе необходимо обеспечить соответствие между длиной строки в файле и размерностью массива символом.

Если строка короче, то она будет иметь в массиве два ограничителя – символы ‘\n’ и ‘\0’, если же нет, то только символ ‘\0’. Если этот факт игнорировать, то длинные строки при чтении из файла будут «порезаны» на части.

Пример считывания строк из файла

Данный пример выводит на экран содержимое файла

```
#include <stdio.h>
void main()
 {       char s[100]; /* считываемая строка */
     FILE *fp = fopen("hello.txt", "r");
    /* проверяем на ошибки */
    if (fp == NULL) {
        printf("Ошибка открытия файла\n");
        return;
                    }
    /* читаем построчно файл */
    while (fgets(s, 100, fp) != NULL)
       {
       /* выводим считанную строку на экран */
       puts(s);
       }
    fclose(fp);
 }
```

Последовательный доступ

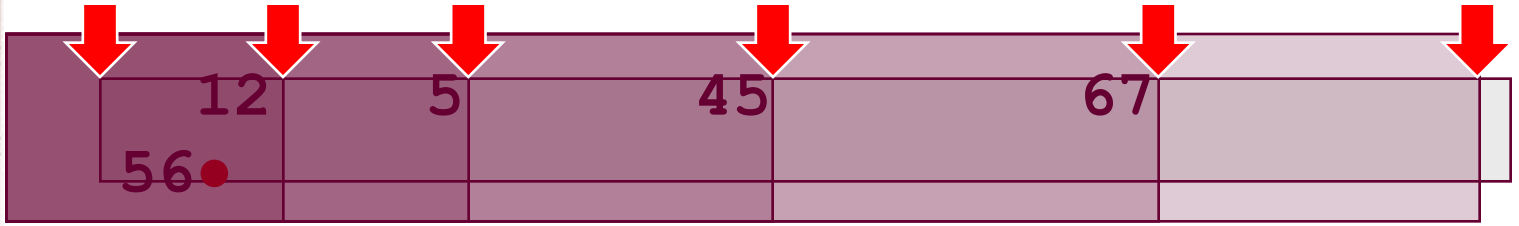
- при открытии файла курсор устанавливается в начало

```
fp=fopen (  “qq.dat”,”r” );
```

конец файла
(*end of file*, EOF)

- чтение выполняется с той позиции, где стоит курсор
- после чтения курсор сдвигается на первый непрочитанный символ

```
fscanf ( fp,”%i” ,&x );
```

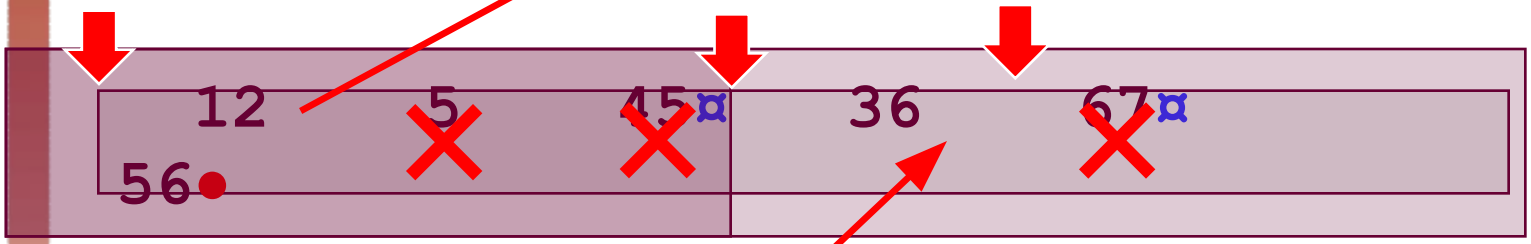


Последовательный доступ

- чтение до конца строки (остаток строки после считывания X игнорируется)

```
fcanf ( fp,"%i\n", &x );
```

конец строки




```
fcanf ( fp,"%i\n", &x );
```

```
fclose ( fp );
```

Вывод форматируемого текста в файл

```
int fprintf(FILE *fp, char* fmt, …);
```

fp – файл, в который производится запись

fmt – форматная строка

... – форматируемые значения

Примечание: fprintf идентична функции printf с той лишь разницей, что в качестве первого параметра передается файл, в который необходимо вывести значения.

Пример записи в файл

```
#include <stdio.h>
void main()
 {
    /* открываем файл на запись */
    FILE *fp = fopen("hello.txt", "w");
    /* проверяем на ошибки */
    if (fp == NULL) {
        printf("Ошибка создания файла\n");
        return;
                    }
    /* записываем в файл строку */
    fprintf(fp, "Hello, file world!\n");
    /* закрываем файл */
    fclose(fp);
 }
```

Стандартный ввод/вывод

В языке Си есть три преопределенных стандартных файла:

- stdin – стандартный ввод (клавиатура)
- stdout – стандартный вывод (дисплей)
- stderr – вывод сообщений об ошибках (дисплей)

puts(<строка>) аналогично fputs(<строка>, stdout)

scanf(<формат>, …) аналогично fscanf(stdin, <формат>, …)

Недопустимо:

fputs(<строка>, stdin);

fgets(<строка>, <длина>, stdout);

Дополнительные функции работы с файлами

Определение наличия ошибки в файле:

```
int ferror(FILE *fp)
```

fp - файл

Возвращает ненулевое значение, если при работе с данным файлом произошла ошибка.

Определение достижения конца файла:

```
int feof(FILE *fp)
```

fp - файл

Возвращает ненулевое значение, если достигнут конец файл.

Загрузка информации из файла в массив

```
int Input_Vector
(int  *n,  int x[],char *NameMatrix)
{int i,j;
 FILE *f;
 f=fopen(NameMatrix,"r");
 fscanf(f,"%i\n",n);
 for(i=0; i < *n; i++)
  {     fscanf(f,"%i",&x[i]);
  }
        fclose(f);
return 0;
}
```

Выходной параметр – n – указатель на число элементов

Строка – имя файла с массивом

Файловая переменная

Передача параметра по адресу

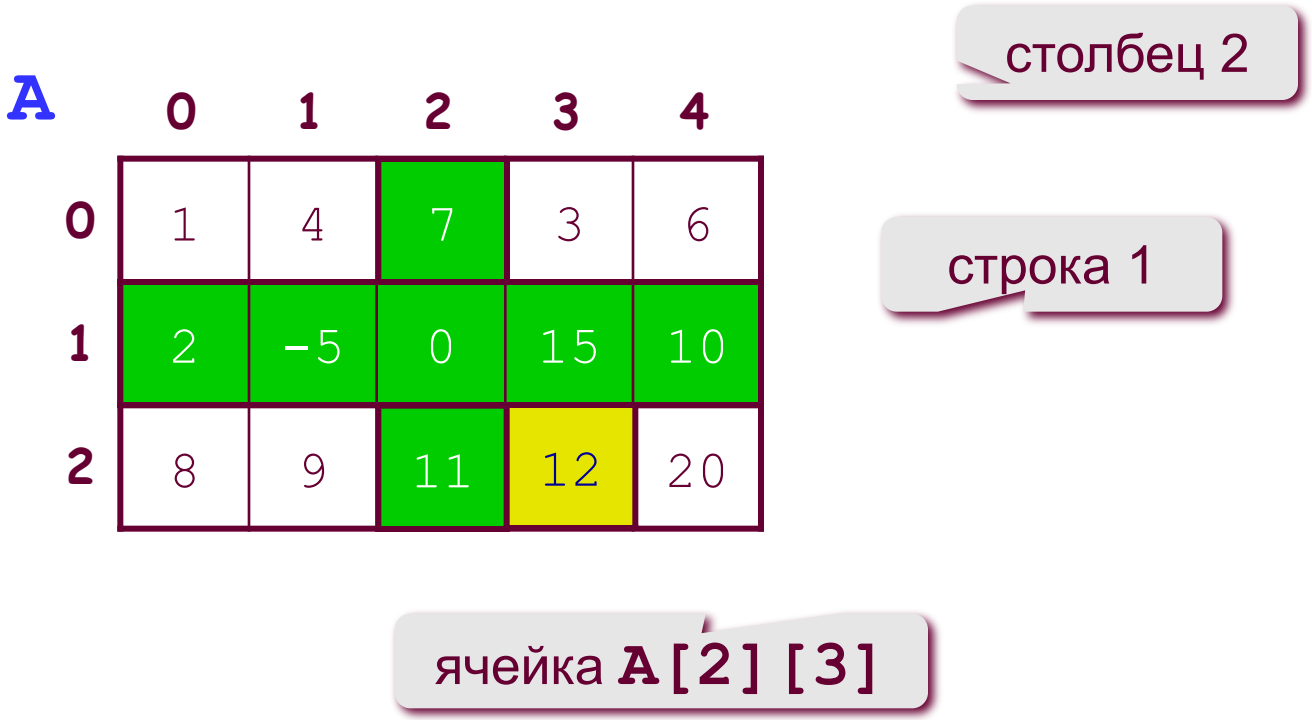
Вызов функции ввода из главной программы

```
… int a[100], na;
Input_Vector(&na,a,"veca.txt");
```

Матрицы

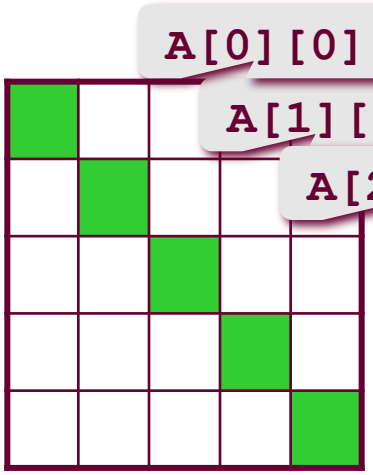
Матрица – это прямоугольная таблица чисел.

Матрица – это массив, в котором каждый элемент имеет два индекса (номер строки и номер столбца).



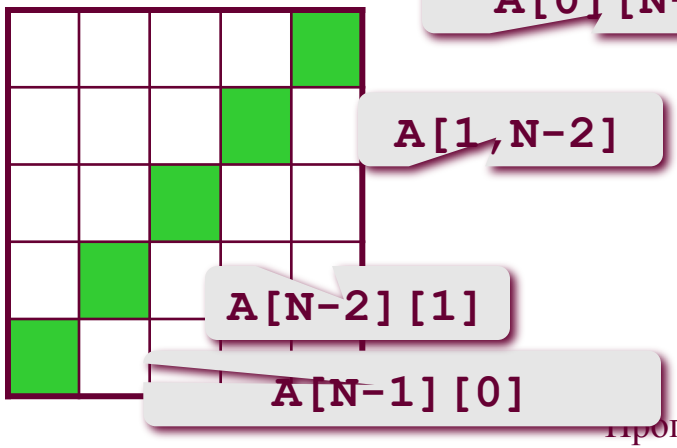
Операции с матрицами

Задача . Вывести на экран главную диагональ квадратной матрицы из N строк и N столбцов.




```
for (i=0;i< N ;i++)
  printf ( "%i ",A[i][i] );
```

Задача . Вывести на экран побочную диагональ.



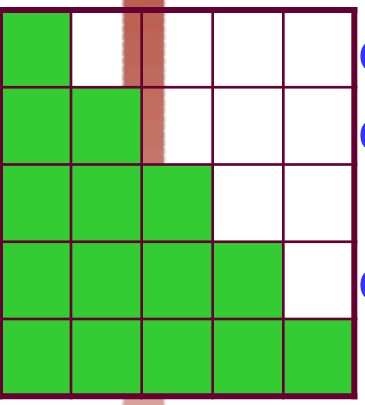
сумма номеров строки и столбца N-1

```
for (i=0;i< N ;i++)
printf ("%i ",A[i][N-1-i]) ;
```

Операции с матрицами

Задача 3. Найти сумму элементов, стоящих на главной диагонали и ниже ее.

Одиночный цикл или вложенный?



строка 0: `A[0][0]`

строка 1: `A[1][0]+A[1][1]`

`...`

строка N-1: `A[N-1,0]+A[N-1][1]+..+A[N-1][N-1]`

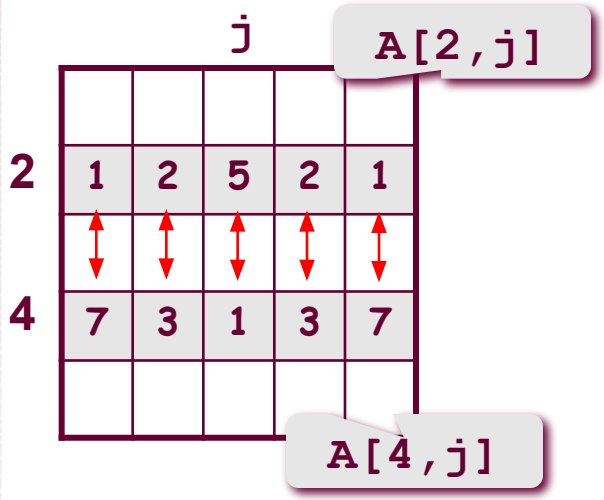
```
S = 0;
for (i=0;i<N;i++)
  for (j=0;j<=i;j++)
    S := S +
  A[i][j];
```

цикл по всем строкам

складываем нужные элементы строки **i**

Операции с матрицами

Задача . Перестановка строк или столбцов. В матрице из N строк и M столбцов переставить 2-ую и 4-ую строки.



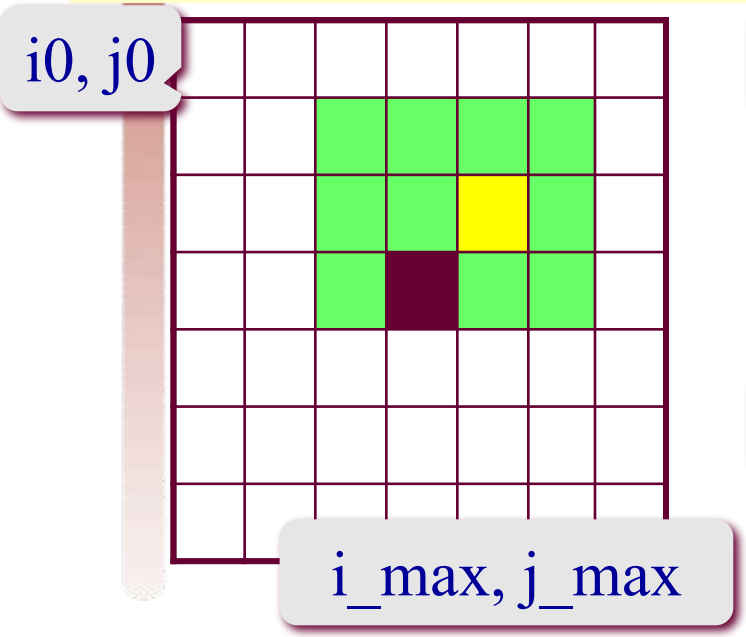

```
for (j=0;j<M;j++)
{   c = A[2][j];
    A[2][j] = A[4][j];
    A[4][j]= c;
}
```

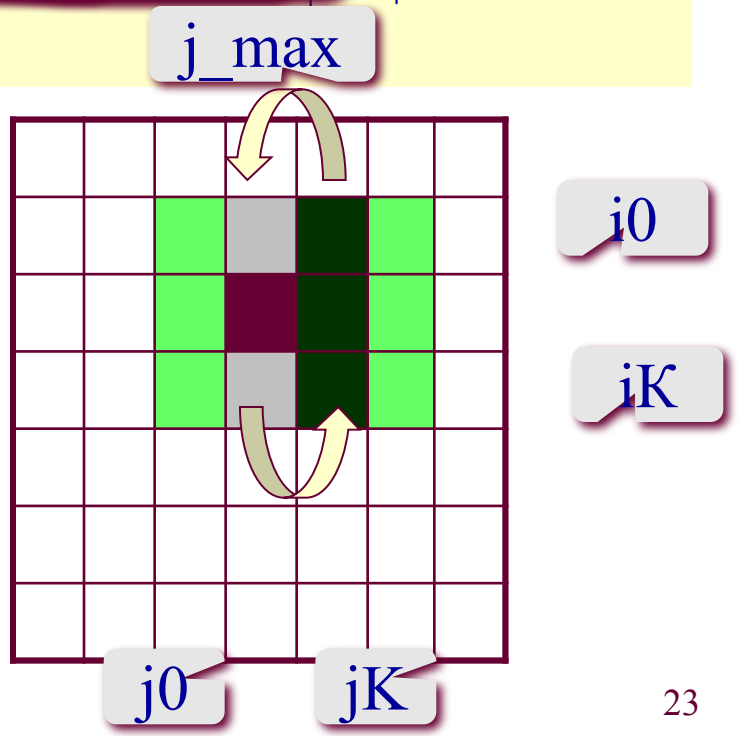
Задача . К третьему столбцу добавить шестой.

```
for (i=0;i<M;i++)
  A[i,3] := A[i,3] + A[i,6];
```

Исходная матрица находится в файле (*размер матрицы и элементы по строкам*)
Создать функции для перестановки строк и столбцов части матрицы от i0, j0 до iK,jK так, чтобы максимальный элемент частичной матрицы находился в заданной позиции (ii,jj), применить процедуру для
ирать
угол
ицы и
полученную полную матрицу.

Найти номер максимального элемента из части матрицы



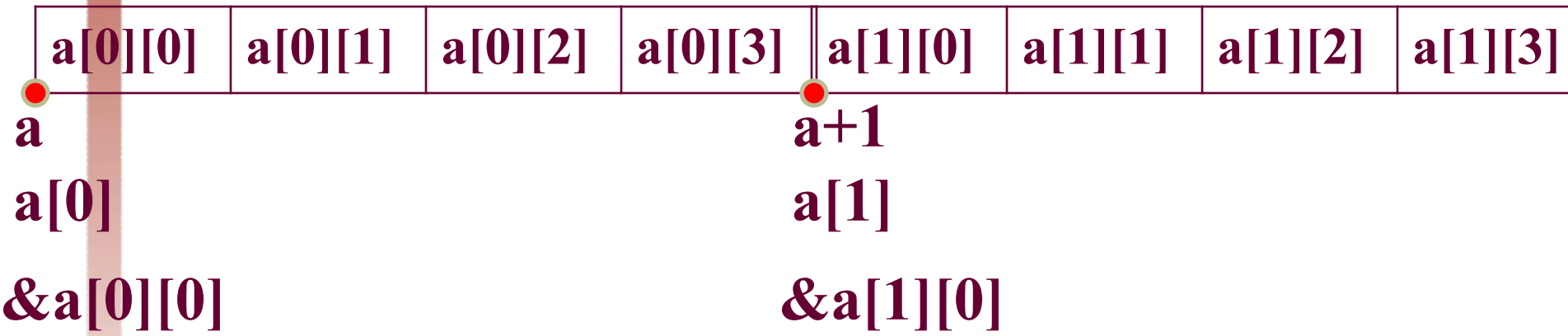


Многомерные массивы

Многомерные массивы в С++ рассматриваются как массивы массивов.

```
int a[2][3];
```

```
a[0]==&a[0][0]  // a[0] - имя первой строки
a[1]==&a[1][0]
```



При прибавлении 1 к целой переменной ее значение увеличивается на 1. При прибавлении 1 к указателю его числовое значение увеличивается на размерность типа, на который он указывает.

Ввод матрицы из файла (1)

```
#define LINES   31
#define COLUMNS 79
void Input_Matrix(int *n, int *m,
    float MATR[LINES][COLUMNS],char *file_inp)
{int i , j;
FILE *f1;  f1=fopen(file_inp,"r");
fscanf(f1,"%i %i\n", n,m);
     for(i=0; i < *n; i++)
        for(j=0; j < *m; j++)
            fscanf(f1,"%f",&MATR[i][j]);
fclose(f1);}
```

Вызов

```
 int m=0, n=0;
        float A[LINES][COLUMNS];
       Input_Matrix(&n , &m, A,"M.dat");
```

Ввод матрицы из файла (2)

```
void Input_Matrix (int *n, int *m,
   float MATR [ LINES ] [ COLUMNS ] ,char  *file_inp)
void Input_Matrix (int *n, int *m,
   float MATR [ ] [ COLUMNS ] , char  *file_inp)
```

Использование одномерного массива как двумерного

```
void Input_Matrix (int *n, int *m,
        float *MATR ,char *file_inp)
{int i , j;
FILE *f1;f1=fopen(file_inp,"r");
fscanf(f1,"%i %i\n", n,m);
     for(i=0; i < *n; i++)
        for(j=0; j < *m; j++)
           fscanf(f1,"%f", &MATR [ i *  (*m) + j )  ]    );
fclose(f1);
}
```

Пересчет индекса