Arrays in C

Data and File Structures Laboratory

http://www.isical.ac.in/~dfslab/2017/index.html

What is an array?

- Sequence of n *contiguous* memory locations
- *Length* of the array $= n$
- *Elements* of the array $\equiv$ each of the n memory locations
- Elements numbered 0 through $n-1$

Syntax

```
char charArray[128], c;
int intArray[64], i, j;
...
charArray[i] = c; // 0 <= i <= 127
intArray[0] = i; j = intArray[63];
```

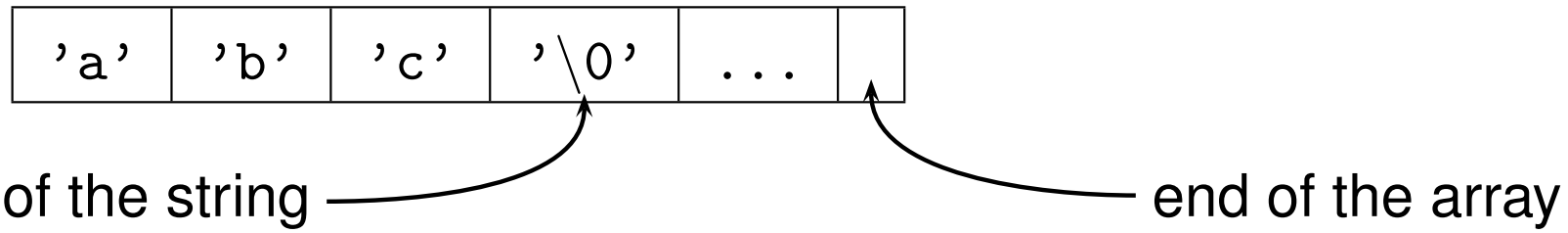
Strings

Definition

Strings are character arrays, but the end of the string is marked by the first occurrence of '\0' in the array (not the last element of the array)

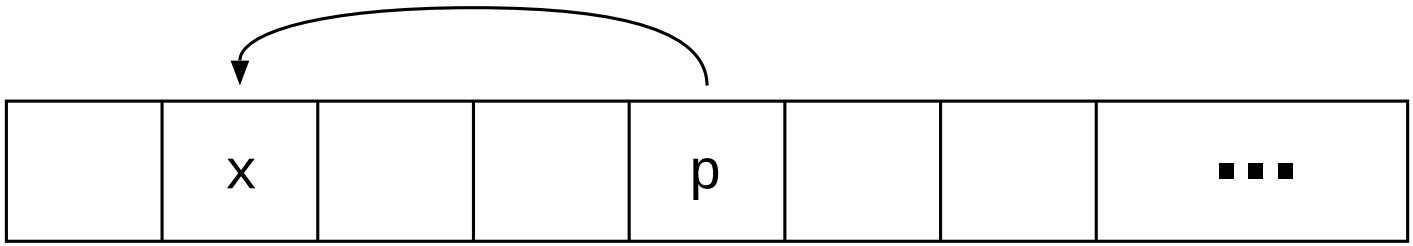
Example:

```
char str1[16];
str1[0] = 'a'; str1[1] = 'b'; str1[2] = 'c';
str1[3] = '\0'; // NOT '0'
/* str1 now holds the string "abc" */
```



Pointers

- Memory = consecutively numbered storage cells (*bytes*)
- Variable can occupy one or more bytes
- *Address* of a variable = serial number of “first” byte occupied by the variable
- *Pointer* holds the address of a variable



Operations

`&` – address / location operator
`*` – dereferencing operator

```
char c, *cp;
int i, *ip;

cp = &c;  ip = &i;
*cp = 0; /* same as c = 0 */
```

Operations

`&` – address / location operator
`*` – dereferencing operator

```
char c, *cp;
int i, *ip;

cp = &c;  ip = &i;
*cp = 0; /* same as c = 0 */
```

`ip + n` – points to n-th object after what `ip` is pointing to
`ip - n` – points to n-th object before what `ip` is pointing to

Pointers and arrays

An array name is synonymous with the address of its first element.

Conversely, a pointer can be regarded as an array of elements starting from wherever it is pointing.

Pointers and arrays

An array name is synonymous with the address of its first element.

Conversely, a pointer can be regarded as an array of elements starting from wherever it is pointing.

```
int a[10] = {...}, *p;

p = a;       /* same as p = &(a[0]); */
*p = 5;      /* same as a[0] = 5; */
p[2] = 6;    /* same as a[2] = 6; */
*(a+3) = 7; /* same as a[3] = 7; */
```

Pointers and arrays

An array name is synonymous with the address of its first element.

Conversely, a pointer can be regarded as an array of elements starting from wherever it is pointing.

```
int a[10] = {...}, *p;

p = a;       /* same as p = &(a[0]); */
*p = 5;      /* same as a[0] = 5; */
p[2] = 6;    /* same as a[2] = 6; */
*(a+3) = 7; /* same as a[3] = 7; */
```

But:

`p = a; p++;`	RIGHT	`a = p; a++;`	WRONG

Allocating memory

Syntax:

```
(type *) malloc(n * sizeof(type))
(type *) calloc(n, sizeof(type))
(type *) realloc(ptr, n * sizeof(type))

free(ptr)
```

Allocating memory

Syntax:

```
(type *) malloc(n * sizeof(type))
(type *) calloc(n, sizeof(type))
(type *) realloc(ptr, n * sizeof(type))

free(ptr)
```

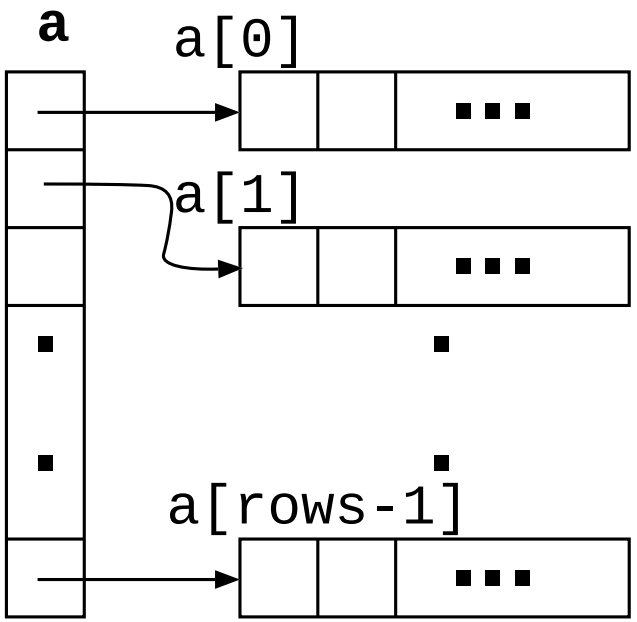
Convenient macros:

```
#define Malloc(n,type) (type *) malloc( (unsigned) ((n)*sizeof(type)))
#define Realloc(loc,n,type) (type *) realloc( (char *)(loc), \
                            (unsigned) ((n)*sizeof(type)))
```

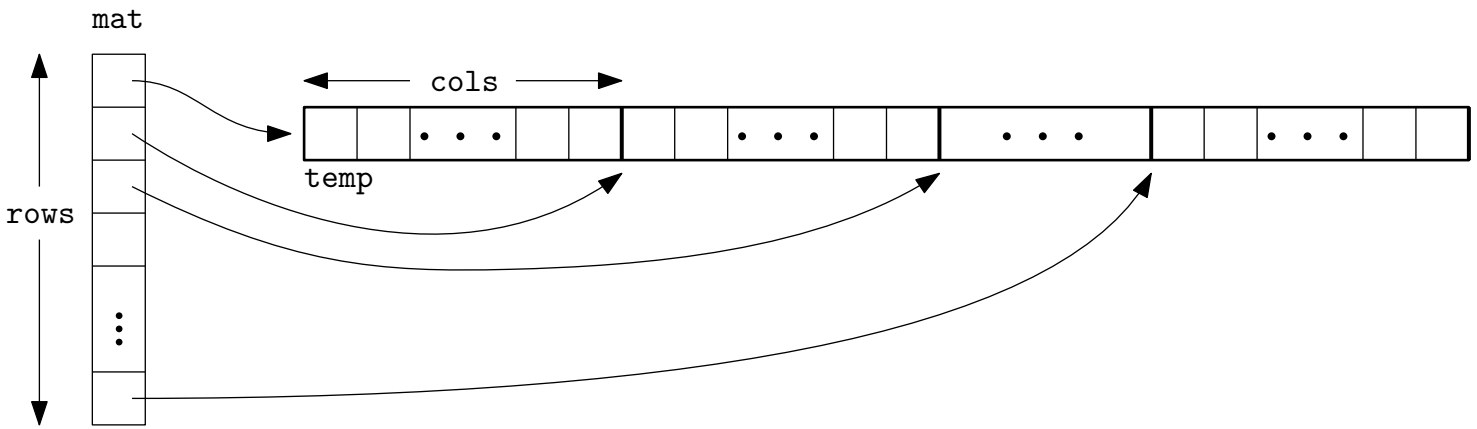
Multi-dimensional arrays

Multi-dimensional array = array of arrays = pointer to pointer

```
int **a, i;
a = (int **) malloc(rows * sizeof(int *));
for (i = 0; i < rows; i++)
    a[i] = (int *) malloc(cols * sizeof(int));
```



Multi-dimensional arrays




```
int ii;
int *temp;
if (NULL == (temp = (int *) malloc(rows*cols*sizeof(int))) ||
    NULL == (mat = (int **) malloc(rows * sizeof(int *))))
    ERR_MESG("Out of memory");
for (ii = 0; ii < rows; temp += cols, ii++)
    mat[ii] = temp;
```

Review questions

1. Suppose `s` and `t` are strings. What does the following do?

```
while (*s++ = *t++);
```

2. What output is generated by the following code?

```
for (i=0; i < 10; i++)
   printf("abcdefghijklmnop\n" + i);
```

Review questions — Solutions

1 String copying

```
do {
  *s = *t;
  s++; t++;
} while (*t != '\0');
```

Review questions — Solutions

1 String copying

```
do {
  *s = *t;
  s++; t++;
} while (*t != '\0');
```

```
do {
  *s++ = *t++;
} while (*t != '\0');
```

Review questions — Solutions

1 String copying

```
do {
  *s = *t;
  s++; t++;
} while (*t != '\0');
```

```
do {
  *s++ = *t++;
} while (*t != '\0');
```

```
while ((*s++ = *t++) != '\0');
```

Review questions — Solutions

2 Think of the problem this way:

```
p = "abcdefghijklmnop\n";
printf(p);
```

Review questions — Solutions

2. Think of the problem this way:

```
p = "abcdefghijklmnop\n";
printf(p);
```

```
p = "abcdefghijklmnop\n";
printf(p + 2);
```

Review questions — Solutions

2 Think of the problem this way:

```
p = "abcdefghijklmnop\n";
printf(p);
```

```
p = "abcdefghijklmnop\n";
printf(p + 2);
```

```
p = "abcdefghijklmnop\n";
printf(p + i);
```

Programming problems - Day 2

1. Consider 2 sequences of integers, A and B, stored in arrays.
   (a) Write a program to find the number of (possibly overlapping) occurrences of the sequence B in A.
   (b) Write a program to find whether the multisets corresponding to A and B are equal.
2. Write a program to find the highest and second highest value in an array of n integers using less than $2n - 3$ comparisons.