EECS2031 AC Software Tools

F 2021

Oct 20,21, 2021 Lecture 11A

1

- Finished Ch1 – 4
  - pre-processor
  - Recursions

RECALL

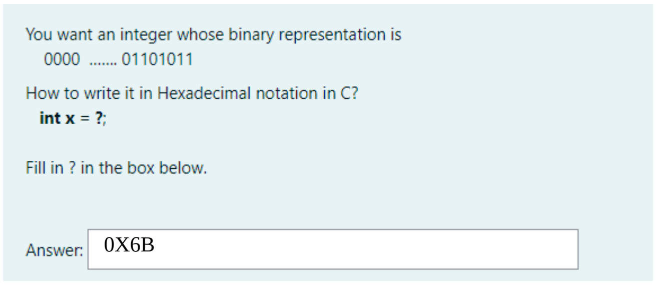
- Other C materials before pointer
  - Common library functions [Appendix of K+R]
  - 2D array, table of strings

- Ch5: Pointers

2

2

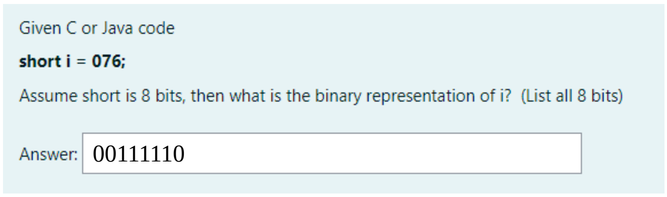
SMQ1


You want an integer whose binary representation is
0000 ……. 01101011

How to write it in Hexadecimal notation in C?
int x = ?;

Fill in ? in the box below.

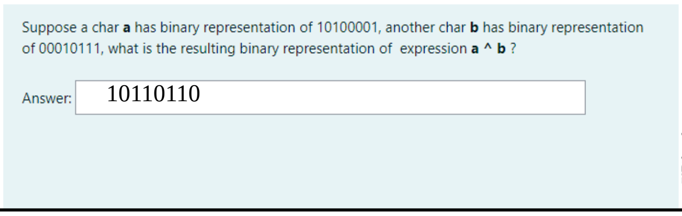
Answer: 0X6B


Given C or Java code

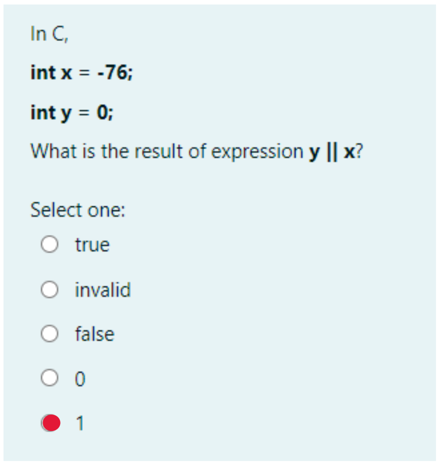
short i = 076;

Assume short is 8 bits, then what is the binary representation of i? (List all 8 bits)

Answer: 00111110


Suppose a char **a** has binary representation of 10100001, another char **b** has binary representation of 00010111, what is the resulting binary representation of expression **a ^ b** ?

Answer: 10110110

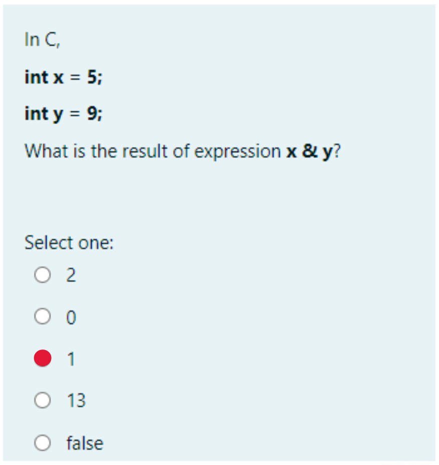

In C,

int x = -76;

int y = 0;

What is the result of expression **y || x**?

Select one:
- true
- invalid
- false
- 0
- [✓ 1]

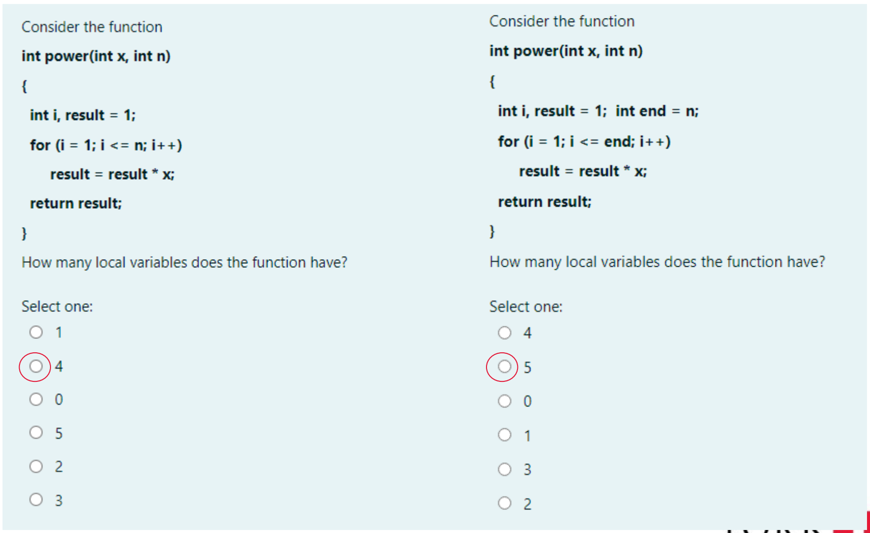

In C,

int x = 5;

int y = 9;

What is the result of expression **x & y**?

Select one:
- 2
- 0
- [✓ 1]
- 13
- false



Consider the function

```
int power(int x, int n)
{
  int i, result = 1;
  for (i = 1; i <= n; i++)
      result = result * x;
  return result;
}
```

How many local variables does the function have?

Select one:

- ○ 1
- [✓ ○ 4]
- ○ 0
- ○ 5
- ○ 2
- ○ 3

Consider the function

```
int power(int x, int n)
{
  int i, result = 1;  int end = n;
  for (i = 1; i <= end; i++)
      result = result * x;
  return result;
}
```

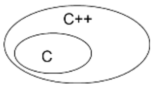
How many local variables does the function have?

Select one:

- ○ 4
- [✓ ○ 5]
- ○ 0
- ○ 1
- ○ 3
- ○ 2

Common library functions [Appendix of K+R]

Included in C++ e.g., `<cstring.h>` `<cmath.h>`



```
<stdio.h>
printf()
scanf()
getchar()
putchar()

sscanf()
sprintf()

gets()   puts()
fgets()  fputs()

fprintf()
fscanf()
```

```
<string.h>
strlen(s)
strcpy(s,s)
strcat(s,s)
strcmp(s,s)
strtok(s,s)
```

```
<math.h>
sin()  cos()
exp()
log()
pow()
sqrt()
ceil()
floor()
```

```
<stdlib.h>

int     atoi(s)
double  atof(s)
long    atol(s)
void    rand()
void    system()
void    exit()
int     abs(int)
```

```
<assert.h>
assert()
```

```
<limit.h>
```

……

```
<ctype.h>

int islower(int)
int isupper(int)
int isdigit(int)
int isxdigit(int)
int isalpha(int)

int tolower(int)
int toupper(int)
```

```
<signal.h>
```

```
<time.h>
```

String library functions

- Defined in standard library, prototype in `<string.h>`
- `unsigned int strlen(s)`
  - `# of chars before first '\0'`
  - `not counting first '\0'`

`strlen("hello");  // 5`

H	e	l	\0	o	\0		

`//strlen?`

- `char * strcpy (dest,  src)`
  - `strncpy(dest, src, n)`
  - `modify dest`

`dest = src` ✗

- `char * strcat(s1, s2)` `s1 → s1s2` `s1 + s2` ✗
  - `strncat (s1, s2, n)`
  - `modify s1` `append s2 to the end of s1`
- `int strcmp(s1, s2)` `0 if equal`
  - `strncmp(s1, s2, n)` `<0 if s1<s2, >0 if s1>s2`

lexicographical order

14

`strcpy` Compensate for the fact that cannot use = to copy strings

To get from another string (literal) <string.h>

```
char message[10];
strcpy(message, "This G")
```

0	1	2	3	4	5	6	7	8	9
.	.	.	.	.	.	.	.	.	.

0	1	2	3	4	5	6	7	8	9
T	h	i	s		G	\0	.	.	.

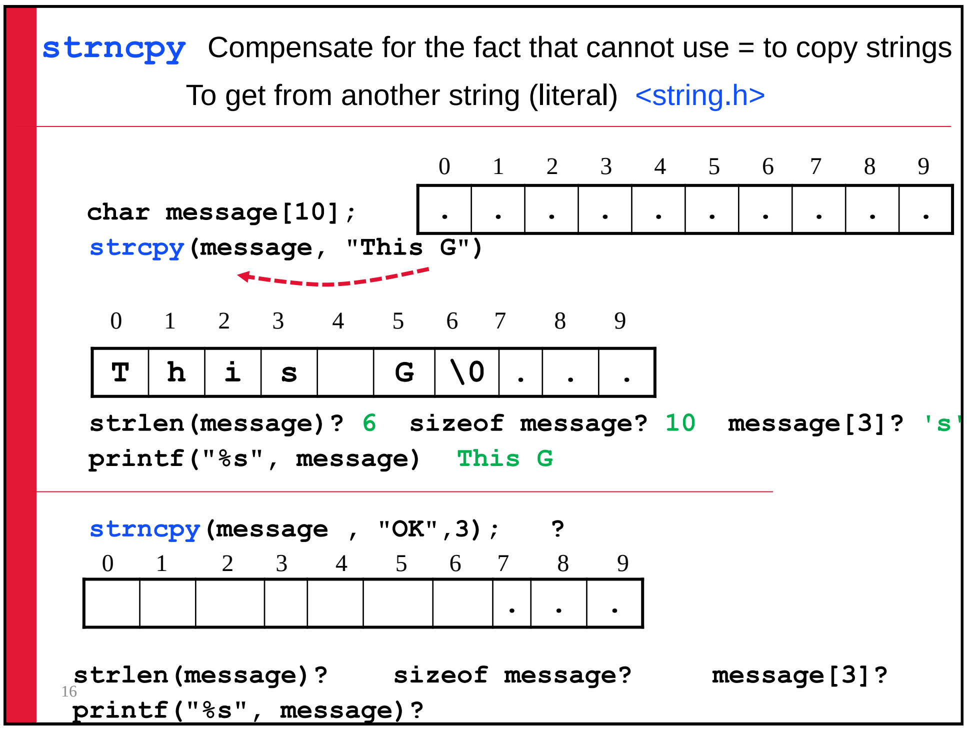
`strlen(message)? 6  sizeof message? 10  message[3]? 's'`
`printf("%s", message)  This G`

`strcpy(message , "OK");   ?`

0	1	2	3	4	5	6	7	8	9
O	K	\0	s		G	\0	.	.	.

`strlen(message)? 2  sizeof message? 10  message[3]? 's'`
`printf("%s", message)?  OK`

15



strncpy Compensate for the fact that cannot use = to copy strings

To get from another string (literal) <string.h>

```
char message[10];
```

0	1	2	3	4	5	6	7	8	9
.	.	.	.	.	.	.	.	.	.

```
strcpy(message, "This G")
```

0	1	2	3	4	5	6	7	8	9
T	h	i	s		G	\0	.	.	.

```
strlen(message)? 6   sizeof message? 10   message[3]? 's'
printf("%s", message)   This G
```

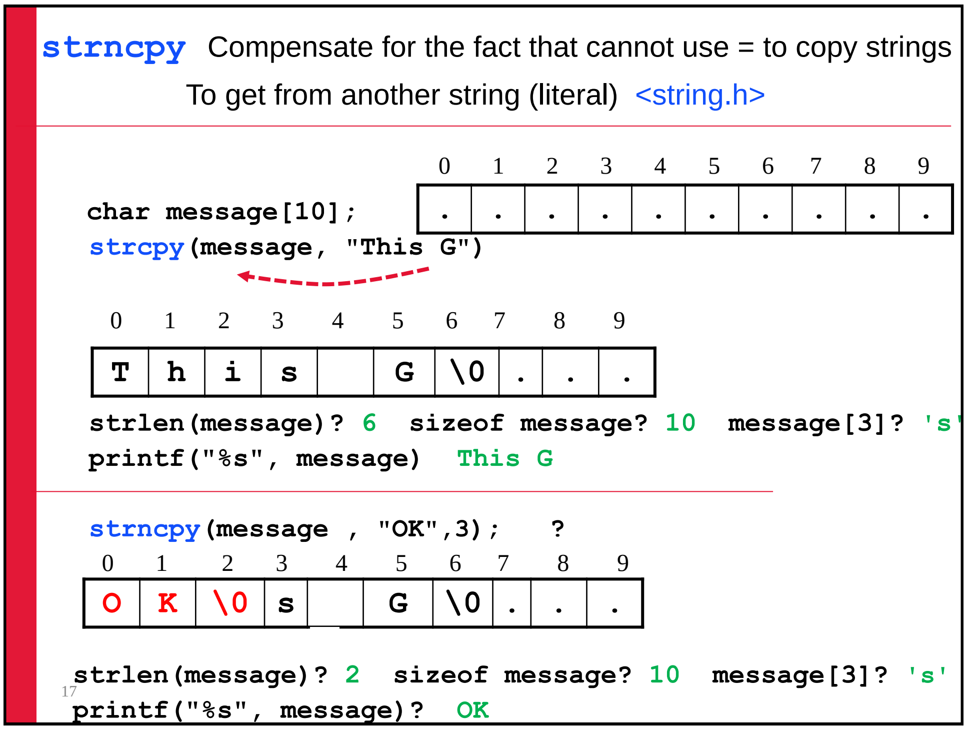
```
strncpy(message , "OK",3);    ?
```

0	1	2	3	4	5	6	7	8	9
							.	.	.

```
strlen(message)?     sizeof message?      message[3]?
printf("%s", message)?
```

16

16



strncpy Compensate for the fact that cannot use = to copy strings

To get from another string (literal) <string.h>

```
char message[10];
```

0	1	2	3	4	5	6	7	8	9
.	.	.	.	.	.	.	.	.	.

```
strcpy(message, "This G")
```

0	1	2	3	4	5	6	7	8	9
T	h	i	s		G	\0	.	.	.

```
strlen(message)? 6   sizeof message? 10   message[3]? 's'
printf("%s", message)   This G
```

```
strncpy(message , "OK",3);    ?
```

0	1	2	3	4	5	6	7	8	9
O	K	\0	s		G	\0	.	.	.

```
strlen(message)? 2   sizeof message? 10   message[3]? 's'
printf("%s", message)?   OK
```

17

17

strncpy Compensate for the fact that cannot use = to copy strings

To get from another string (literal) <string.h>

```
char message[10];
strcpy(message, "This G")
```

0	1	2	3	4	5	6	7	8	9
.	.	.	.	.	.	.	.	.	.

0	1	2	3	4	5	6	7	8	9
T	h	i	s		G	\0	.	.	.

```
strlen(message)? 6   sizeof message? 10   message[3]? 's'
printf("%s", message)   This G
```

```
strncpy(message , "OK",2);   ?
```

0	1	2	3	4	5	6	7	8	9
							.	.	.

```
strlen(message)?     sizeof message?     message[3]?
printf("%s", message)?
```

strncpy Compensate for the fact that cannot use = to copy strings

To get from another string (literal) <string.h>

```
char message[10];
strcpy(message, "This G")
```

0	1	2	3	4	5	6	7	8	9
.	.	.	.	.	.	.	.	.	.

0	1	2	3	4	5	6	7	8	9
T	h	i	s		G	\0	.	.	.

```
strlen(message)? 6   sizeof message? 10   message[3]? 's'
printf("%s", message)   This G
```

```
strncpy(message , "OK",2);   ?
```

0	1	2	3	4	5	6	7	8	9
O	K	i	s		G	\0	.	.	.

No \0 added

```
strlen(message)? 6   sizeof message? 10   message[3]? 's'
printf("%s", message)?   OKis G
```

strcat Compensate for the fact that cannot use + to glue strings

```
char message[10];
strcpy(message, "This G")
```

0	1	2	3	4	5	6	7	8	9
.	.	.	.	.	.	.	.	.	.

0	1	2	3	4	5	6	7	8	9
T	h	i	s		G	\0	.	.	.

`strlen(message)?` 6 `sizeof message?` 10 `message[3]?` 's'

`strcat(message, "OK");` ?

Append "OK" to the end of message.
'O' replaces 1st '\0'

0	1	2	3	4	5	6	7	8	9
T	h	i	s		G	O	K	\0	.

`strlen(message)?` 8 `sizeof message?` 10 `message[3]?` 's'

`printf("%s", message)?` This GOK

24

strcat Compensate for the fact that cannot use + to glue strings

Another example

0	1	2	3	4	5	6	7	8	9
O	K	\0	s		G	O	\0	.	.

`strcat(message, "Hi")`

0	1	2	3	4	5	6	7	8	9
O	K	H	i	\0	G	O	\0	.	.

Append "Hi" to the end of message.
'H' replaces 1st '\0'

`strlen(message)?` 4 `sizeof message?` 10 `message[6]?` 'o'

`printf("%s", message)?` OKHi

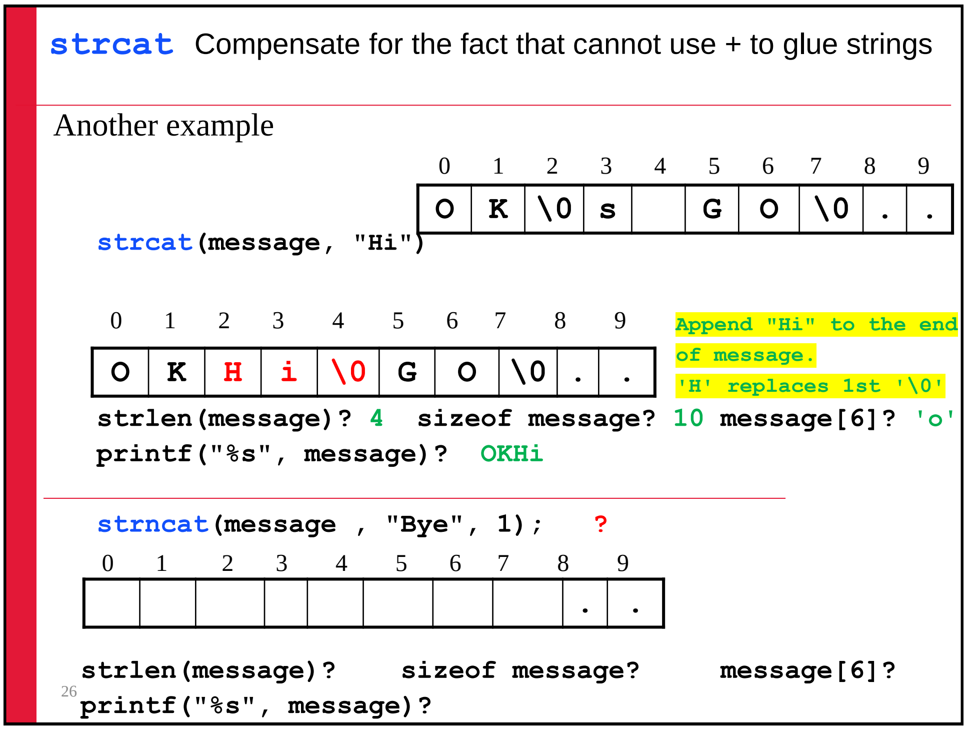
`strcat(message , "B");` ?

0	1	2	3	4	5	6	7	8	9
O	K	H	i	B	\0	O	\0	.	.

`strlen(message)?` 5 `sizeof message?` 10 `message[6]?` 'o'

`printf("%s", message)?` OKHiB

25



strcat Compensate for the fact that cannot use + to glue strings

Another example

0	1	2	3	4	5	6	7	8	9
O	K	\0	s		G	O	\0	.	.

```
strcat(message, "Hi")
```

0	1	2	3	4	5	6	7	8	9
O	K	H	i	\0	G	O	\0	.	.

Append "Hi" to the end of message.
'H' replaces 1st '\0'

```
strlen(message)? 4  sizeof message? 10 message[6]? 'o'
printf("%s", message)?  OKHi
```

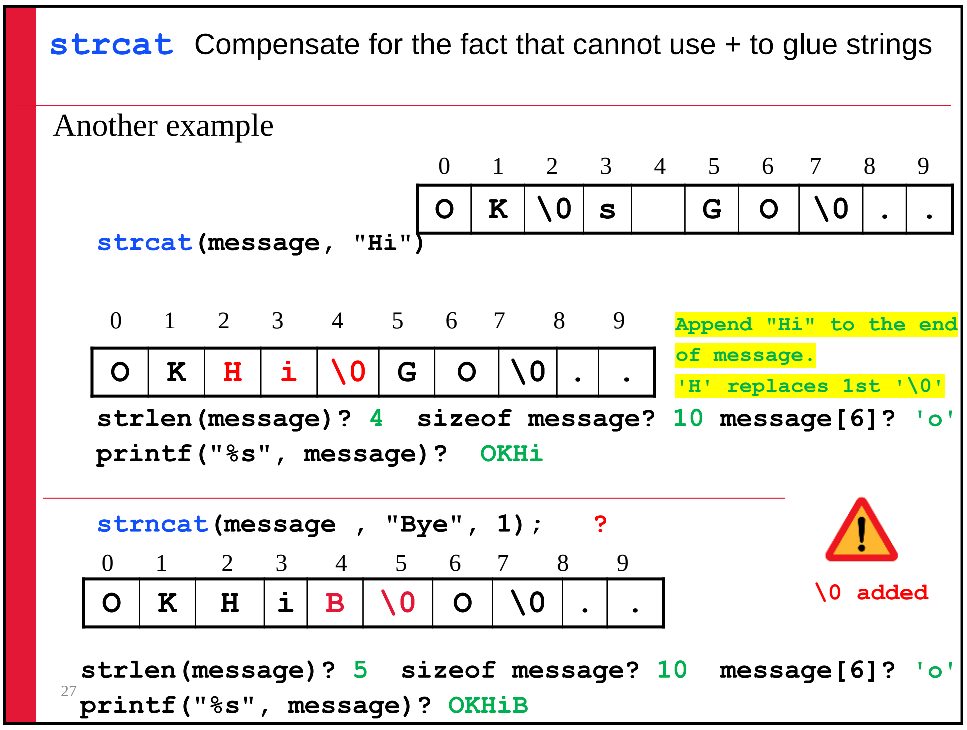
```
strncat(message , "Bye", 1);   ?
```

0	1	2	3	4	5	6	7	8	9
								.	.

```
strlen(message)?    sizeof message?     message[6]?
printf("%s", message)?
```

26

26



strcat Compensate for the fact that cannot use + to glue strings

Another example

0	1	2	3	4	5	6	7	8	9
O	K	\0	s		G	O	\0	.	.

```
strcat(message, "Hi")
```

0	1	2	3	4	5	6	7	8	9
O	K	H	i	\0	G	O	\0	.	.

Append "Hi" to the end of message.
'H' replaces 1st '\0'

```
strlen(message)? 4  sizeof message? 10 message[6]? 'o'
printf("%s", message)?  OKHi
```

```
strncat(message , "Bye", 1);   ?
```

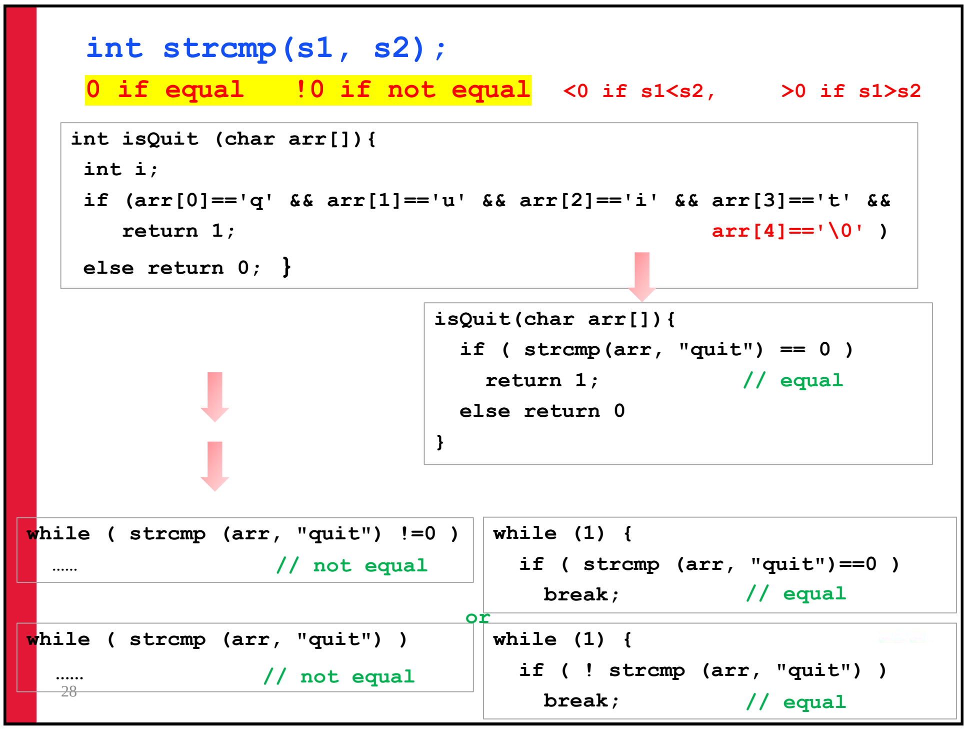
0	1	2	3	4	5	6	7	8	9
O	K	H	i	B	\0	O	\0	.	.

\0 added

```
strlen(message)? 5  sizeof message? 10  message[6]? 'o'
printf("%s", message)? OKHiB
```

27

27



int strcmp(s1, s2);

0 if equal   !0 if not equal   <0 if s1<s2,   >0 if s1>s2

```
int isQuit (char arr[]){
  int i;
  if (arr[0]=='q' && arr[1]=='u' && arr[2]=='i' && arr[3]=='t' &&
                                                  arr[4]=='\0' )
     return 1;
  else return 0; }
```

```
isQuit(char arr[]){
  if ( strcmp(arr, "quit") == 0 )
    return 1;             // equal
  else return 0
}
```

```
while ( strcmp (arr, "quit") !=0 )
  ……                  // not equal
```

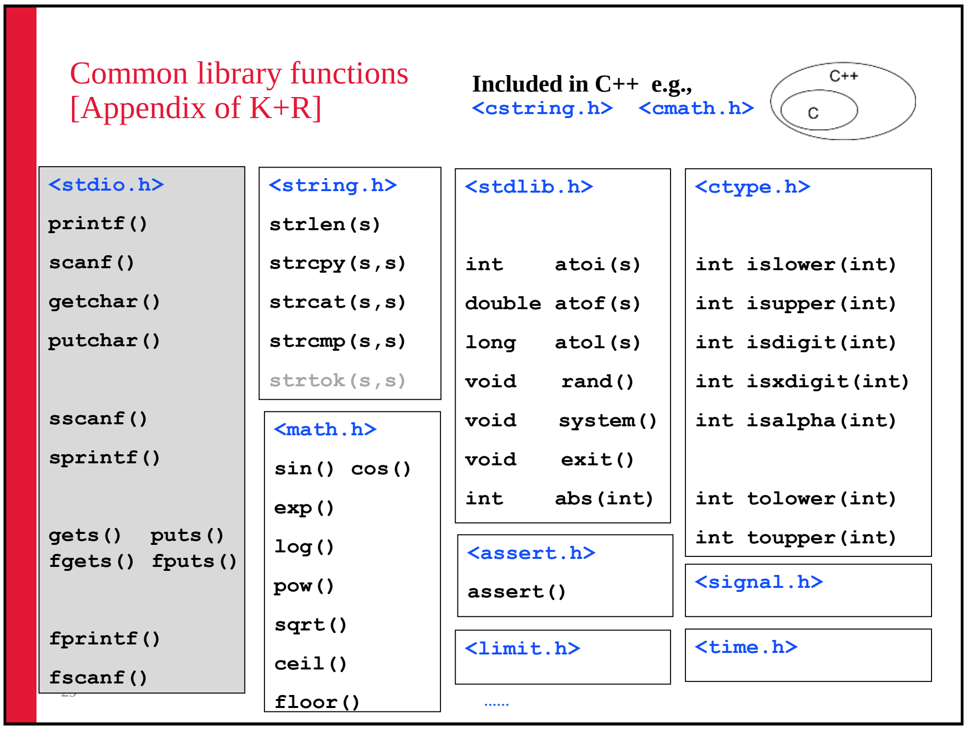
```
while (1) {
  if ( strcmp (arr, "quit")==0 )
    break;            // equal
```

or

```
while ( strcmp (arr, "quit") )
  ……                  // not equal
```

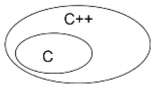
```
while (1) {
  if ( ! strcmp (arr, "quit") )
    break;            // equal
```

28



Common library functions [Appendix of K+R]

Included in C++ e.g., <cstring.h> <cmath.h>



<stdio.h>	<string.h>	<stdlib.h>	<ctype.h>
printf()	strlen(s)	int atoi(s)	int islower(int)
scanf()	strcpy(s,s)	double atof(s)	int isupper(int)
getchar()	strcat(s,s)	long atol(s)	int isdigit(int)
putchar()	strcmp(s,s)	void rand()	int isxdigit(int)
sscanf()	strtok(s,s)	void system()	int isalpha(int)
sprintf()	<math.h>	void exit()	int tolower(int)
gets() puts()	sin() cos()	int abs(int)	int toupper(int)
fgets() fputs()	exp()	<assert.h>	<signal.h>
fprintf()	log()	assert()	<time.h>
fscanf()	pow()	<limit.h>	
	sqrt()	……	
	ceil()		
	floor()		

Basic I/O functions

`<stdio.h>`

- **int printf (char *format, arg1, ….);**
  - Formats and prints arguments on standard output (screen or > outputFile)
  - **printf("This is a test %d \n", x)**

- **int scanf (char *format, arg1, ….);**
  - Formatted input from standard input (keyboard or < inputFile)
  - **scanf("%d %c", &x, &y)**

- **int sprintf (char * str, char *format, arg1,…..);**
  - Formats and prints arguments to char array (string) str
  - **sprintf(str, "This is a test %d \n", x)** // nothing print on stdout!

- **int sscanf (char * str, char *format, arg1, ….);**
  - Formatted input from char array (string) str
  - **sscanf(str, "%d %c", &x, &y)** // tokenize string str

YORK U UNIVERSITÉ UNIVERSITY

31

Char arrays: set /get in general

other ways of generating a string

- Some very import functions you might want to know

```
char message[12];  int age = 12; float rate =2.34;
```

- Defined in standard library, prototype <stdio.h>
  - `sprintf(message, "%s %d-%.1f", "Sue",age,rate);`

S	u	e		1	2	-	2	.	3	\0	

  - `sscanf(message, "%s %d-%f", name, &age, &rate);`

tokenizing the string

  - `fgets(message, 10, stdin)`
  - `fputs(message, stdout)`

`FILE * stream`

32

YORK U UNIVERSITÉ UNIVERSITY

32

```
int main()
{  char str[40];
   scanf("%s", str);
   printf("%s\n", str);
}
```

```
red 199 % a.out
hello the world!
hello
red 200 %
```

```
int main()
{  char str[40];
   scanf(" %[^\n]s", str);
   printf("%s\n", str);
}
```

hello the world!\0

```
red 199 % a.out
hello the world!
hello the world!
red 200 %
```

hello the world!\n\0

```
int main()
{  char str[40];
   fgets (str, 40, stdin);
   fputs(str, stdout);
   //or printf("%s", str);
}
```

No \n needed

Be careful the '\n'

```
red 199 % a.out
hello the world!
hello the world!
red 200 %
```

35

35

```
int main()
{  char str[40];
   fgets(str, 40, stdin);
   while (strcmp(str, "quit\n"))
   {
     fputs(str);  //  \n printed
     // printf("%s", str);

     // read again
     fgets(str, 40, stdin);
   }
}
```

```
red 199 % a.out
hello the world!
hello the world!
This is good
This is good
quit
red 200 %
```

```
int main()
{
    char str[40];
    while (1)
    {
       fgets (str, 40, stdin);
       if (! strcmp(str, "quit\n"))
          break;           // ==0
       fputs(str, stdout);
    }
}
```

No &

Be careful the '\n'

36

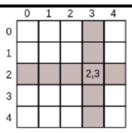
36

- Finished Ch1 – 4

- Other C materials before pointer
  - Common library functions [Appendix of K+R]
  - **2D array, table of strings**

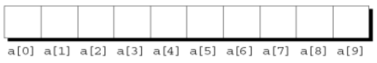
37

37



Multi-dimension array how are they stored

- `int arr1D[10];`
  - Size: type bytes * number of element
    - 4 * 10 = 40 bytes



- `int arr2D[3][2]`
  - Size: type bytes * row * column
    - 4 * 3 * 2 = 24 bytes

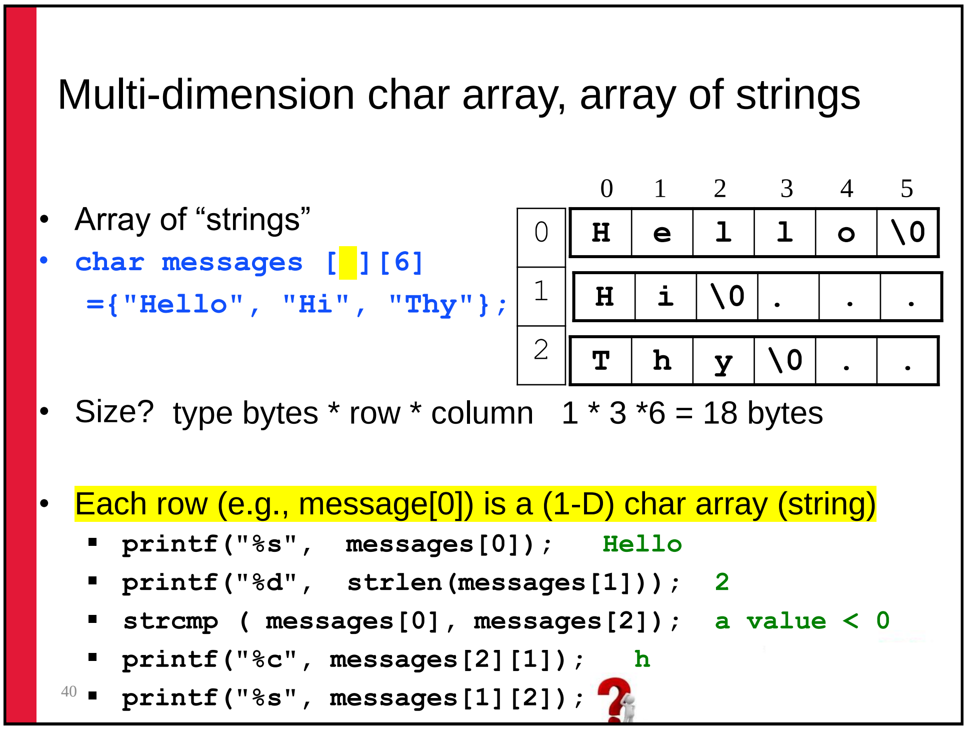
	col 0	col 1
row 0	**a[0][0]**	**a[0][1]**
row 1	**a[1][0]**	**a[1][1]**
row 2	**a[2][0]**	**a[2][1]**

row 0		**row 1**		**row 2**	
a[0][0]	**a[0][1]**	**a[1][0]**	**a[1][1]**	**a[2][0]**	**a[2][1]**

```
printf("%d", arr[0]);
printf("%d", arr[0][2]);
```

39

39



Multi-dimension char array, array of strings

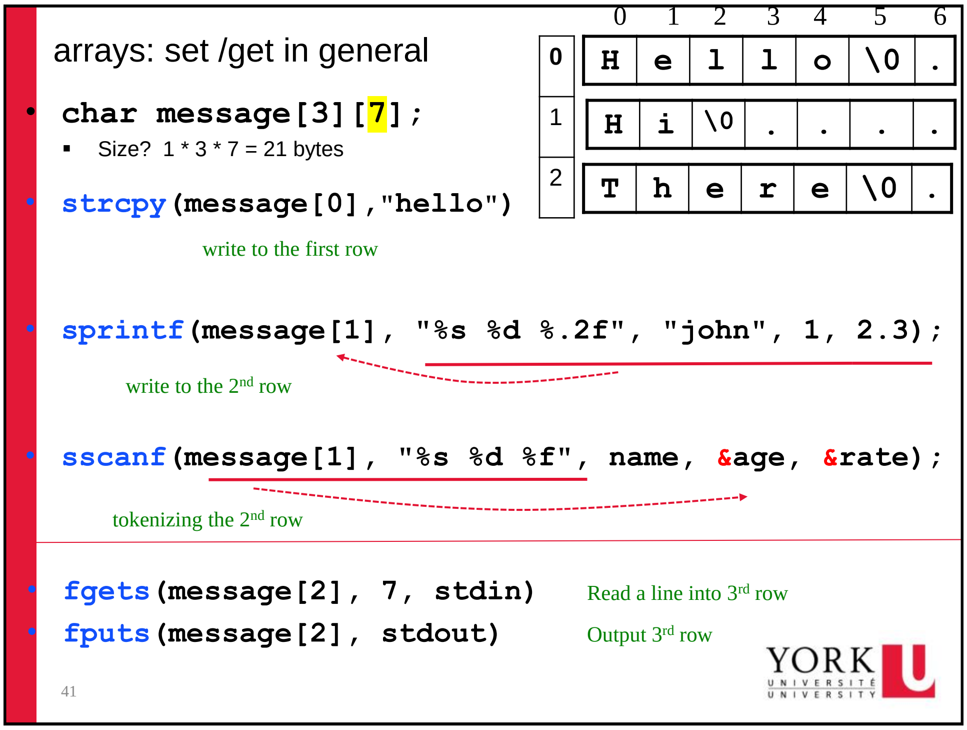
- Array of “strings”
- `char messages [ ][6]`
  `={"Hello", "Hi", "Thy"};`

	0	1	2	3	4	5
0	H	e	l	l	o	\0
1	H	i	\0	.	.	.
2	T	h	y	\0	.	.

- Size? type bytes * row * column 1 * 3 *6 = 18 bytes

- Each row (e.g., message[0]) is a (1-D) char array (string)
  - `printf("%s", messages[0]);` Hello
  - `printf("%d", strlen(messages[1]));` 2
  - `strcmp ( messages[0], messages[2]);` a value < 0
  - `printf("%c", messages[2][1]);` h
  - `printf("%s", messages[1][2]);` ?

40



arrays: set /get in general

	0	1	2	3	4	5	6
0	H	e	l	l	o	\0	.
1	H	i	\0	.	.	.	.
2	T	h	e	r	e	\0	.

- `char message[3][7];`
  - Size? 1 * 3 * 7 = 21 bytes
- `strcpy(message[0],"hello")`

  write to the first row

- `sprintf(message[1], "%s %d %.2f", "john", 1, 2.3);`

  write to the 2nd row

- `sscanf(message[1], "%s %d %f", name, &age, &rate);`

  tokenizing the 2nd row

- `fgets(message[2], 7, stdin)` Read a line into 3rd row
- `fputs(message[2], stdout)` Output 3rd row

YORK U UNIVERSITÉ UNIVERSITY

41