計算機程式設計

Computer Programming

Strings

Instructor: 林英嘉
2024/11/25

W12 Slido: #3517219

GitHub repo

Outline

- Introduction to **characters** and **strings**
- **Input** characters and strings
- Array of strings
- Practical Questions

[Definition] Characters and Strings

- 字元：character
- 字串：string

```
'a' // this is a character `a`
"a" // this is a string `a`
"Sweet home" // this is a string `Sweet home`
```

Be careful!

- We should use single quotes (') for a character.
- We should use double quotes (") for a string.

[Declaration & Definition] Characters

Example:

```
char a_char = 'Y';
```

- char: abbreviation from the first four letters of “character”
- a char variable takes 1 byte for storage (8 bits).
    - The value range of a char variable is -128 to 127.

資料類型比較

	大小 (Byte)*	Specifier	數值範圍
`int`	4	**%d**	-2,147,483,648 到 2,147,483,647 (範圍2的32次方)
`char`	1	**%c**	-128 到 127 或 0 到 255 （取決於是否有符號）
`float`	4	**%f**	約 1.2E-38 到 3.4E+38，精度約 6 位十進制之小數
`double`	**8**	**%lf**	約 2.2E-308 到 1.7E+308，精度約 15-16 位十進制之小數

*In a 64-bit system

[Declaration] String

C-course-materials/07-Strings/declaration.c

Strings are **arrays of characters** in which a special character—the null character—marks the end.

Example:

```c
char str_11[11] = "Sweet home";
```

Array size can be omitted (compiler will allocate memory according to your string)

index	0	1	2	3	4	5	6	7	8	9	10
string	S	w	e	e	t		h	o	m	e	\0

Null character

[Usage] Specifier and Size Comparison

Type	int	char	char	float	double
Meaning	integer 整數	character 字元	string 字串	floating-point number 浮點數	Double-precision floating-point number 倍準浮點數
大小 (Byte)*	4	1	length+1	4	8
Specifier	**%d**	**%c**	**%s**	**%f**	**%lf**

*In a 64-bit system

Print chars and strings

C-course-materials/07-Strings/print_char_str.c

```c
#include <stdio.h>
int main(void){
    char a_char = 'Y';
    printf("%c\n", a_char);

    char a_string[] = "Y";
    printf("%s\n", a_string);

    char str_11[] = "Sweet home";
    printf("%s\n", str_11);
}
```

Be careful!

- Use %c for printing a character
- Use %s for printing a string

Size comparison between chars and strings

C-course-materials/07-Strings/sizes.c

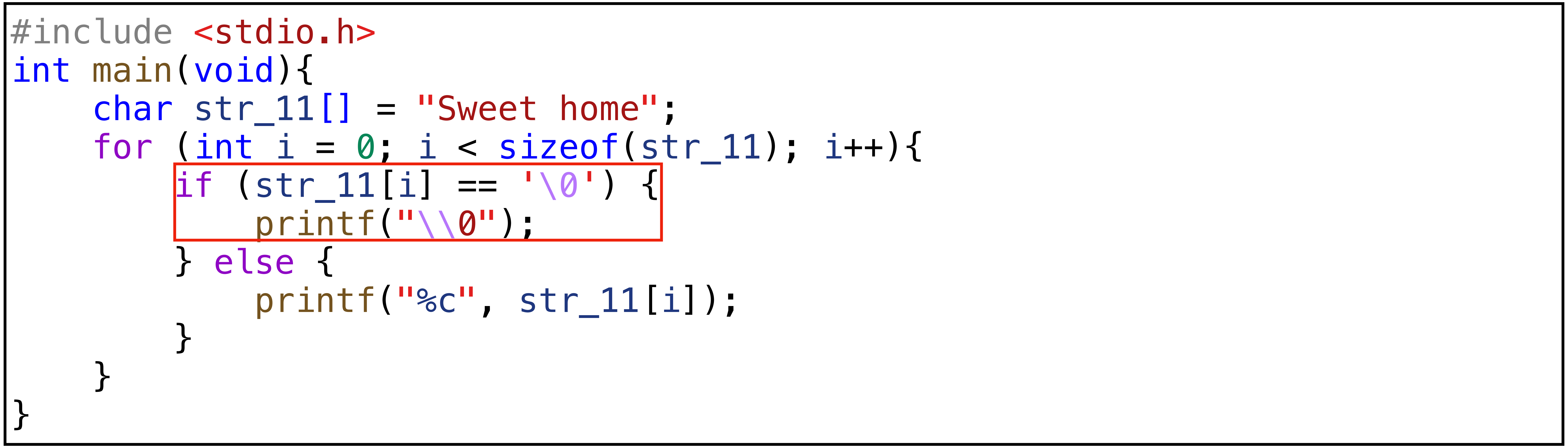
```c
#include <stdio.h>
int main(void){
    char a_char = 'Y';
    printf("%d\n", sizeof(a_char));

    char a_string[] = "Y";
    printf("%d\n", sizeof(a_string));

    char str_11[] = "Sweet home";
    printf("%d\n", sizeof(str_11));
}
```

```
1
2
11
```

← ‘\0’ is taken into consideration.

Check \0 in a string

C-course-materials/07-Strings/check_end.c

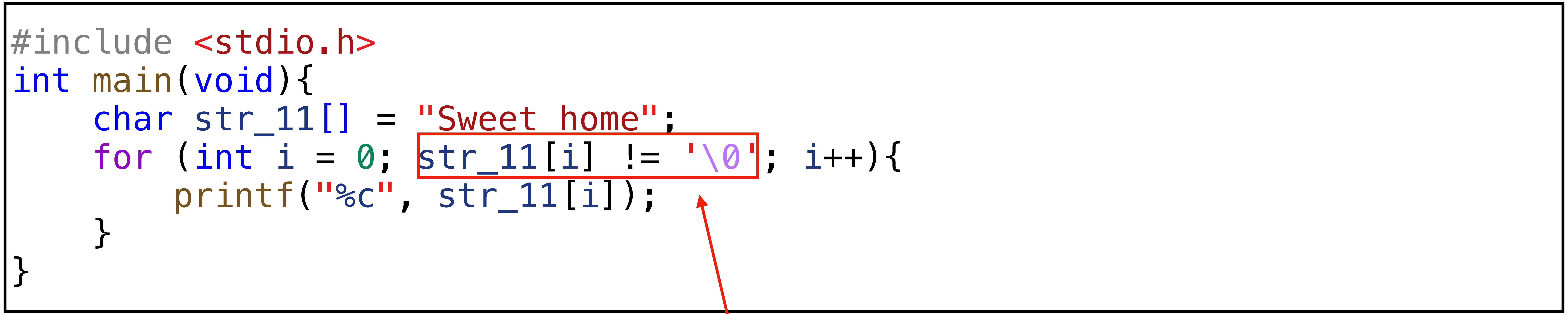


```c
#include <stdio.h>
int main(void){
    char str_11[] = "Sweet home";
    for (int i = 0; i < sizeof(str_11); i++){
        if (str_11[i] == '\0') {
            printf("\\0");
        } else {
            printf("%c", str_11[i]);
        }
    }
}
```

‘\0’ cannot be printed! But we can still detect it.

Use \0 as the end of a loop

C-course-materials/07-Strings/end_in_for.c



```c
#include <stdio.h>
int main(void){
    char str_11[] = "Sweet home";
    for (int i = 0; str_11[i] != '\0'; i++){
        printf("%c", str_11[i]);
    }
}
```

‘\0’ can be used as the stop criterion for a loop.

[Important Notes] About the null character \0

- ‘\0’ is a character that cannot be printed.
- ‘\0’ will be automatically added at the end if the length permitted.

This will automatically add \0:

```
char str_11[] = "Sweet home";
char str_11[11] = "Sweet home";
```

This will not automatically add \0:

```
char str_11[10] = "Sweet home";
```

[Declaration] Characters and Strings with an Initializer

C-course-materials/07-Strings/declaration_initializer.c

- You can also use **an initializer** to declare a `char` variable.

String example:

```
char str_5[5] = {'H', 'e', 'l', 'l', 'o'};
```

Array size can be omitted (compiler will allocate memory according to your string)

Character example:

```
char a_char = {'Y'};
```

[Important Notes] About the null character \0

- ‘\0’ is a character that cannot be printed.
- ‘\0’ will be automatically added at the end if the length permitted.

This will automatically add \0:

```
char str_5[6] = {'H', 'e', 'l', 'l', 'o'};
```

This will not automatically add \0:

```
char str_5[5] = {'H', 'e', 'l', 'l', ‘o'};
char str_5[] = {'H', 'e', 'l', 'l', 'o'};
```

We can also manually add \0:

```
char str_5[6] = {'H', 'e', 'l', 'l', 'o', '\0'};
```

[Usage] You can’t do these

C-course-materials/07-Strings/declaration.c

```
char a_name = "Y";
```

Reason: a string must be stored in an array.

```
char a_name[] = 'S';
char a_char[] = {'Y'};
```

Reason: if a char variable is declared with an array, it should not be a character. It is a string.

When length exceeds the initializer

C-course-materials/07-Strings/shorter_initializer.c

```c
#include <stdio.h>
int main(void){
    char a_string[10] = "Hello";

}
```

When length exceeds the initializer

C-course-materials/07-Strings/shorter_initializer.c

Once the assigned length exceeds the number of elements in an initializer, ‘\0’ will be appended to fill the remaining space until the array reaches the length.

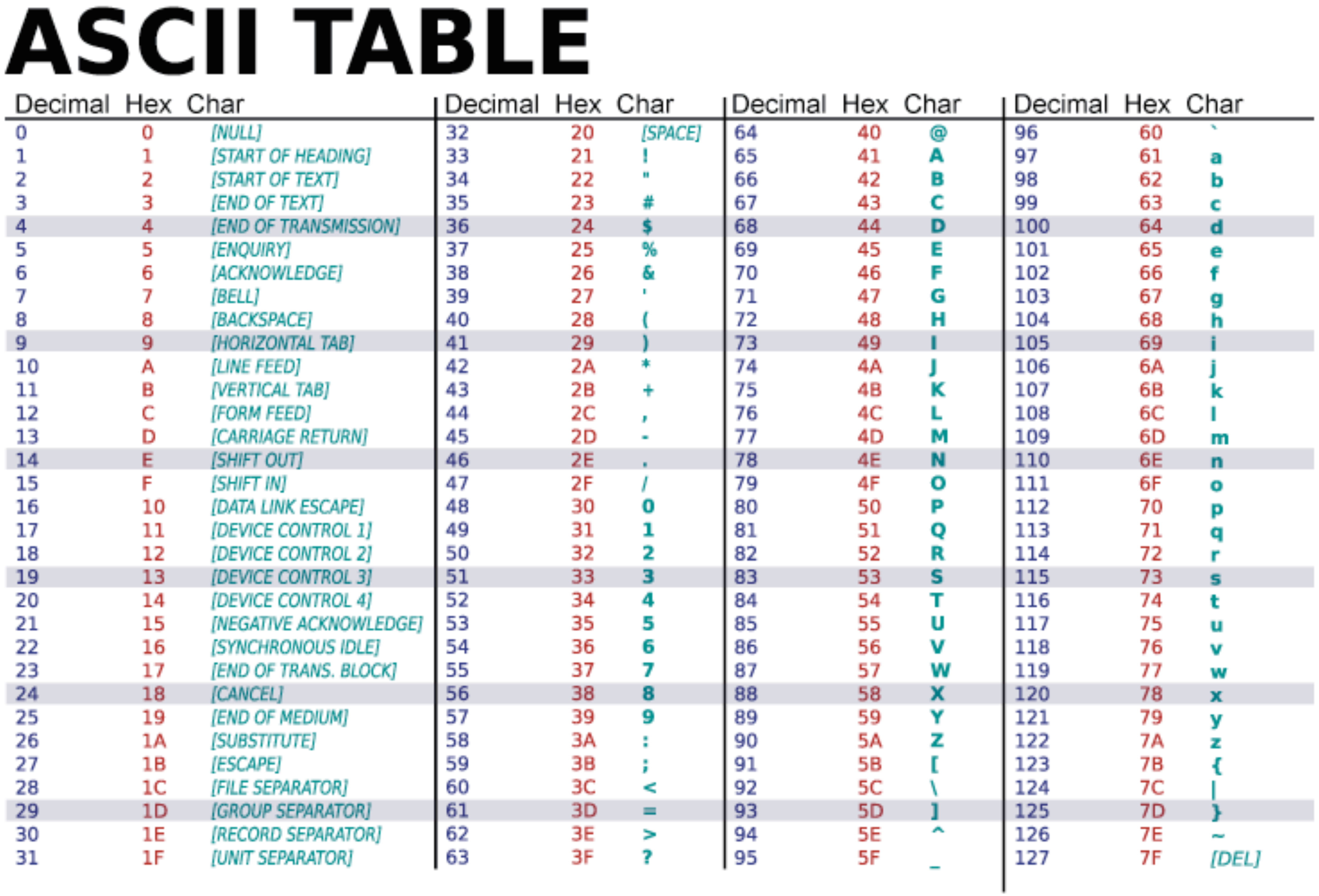
```c
#include <stdio.h>
int main(void){
    char a_string[10] = "Hello";
    for (int i = 0; i < sizeof(a_string); i++){
        if (a_string[i] == '\0') {
            printf("\\0");
        } else {
            printf("%c", a_string[i]);
        }
    }
}
```

index	0	1	2	3	4	5	6	7	8	9
string	H	e	l	l	o	\0	\0	\0	\0	\0

[Introduction] ASCII

- ASCII (American Standard Code for Information Interchange)
- ASCII is a character encoding system to represent characters in computers.
- ASCII has just 128 (7 bit; **from 0 to 127**) code points, of which only **95** are **printable** characters.

[Introduction] ASCII



ASCII TABLE

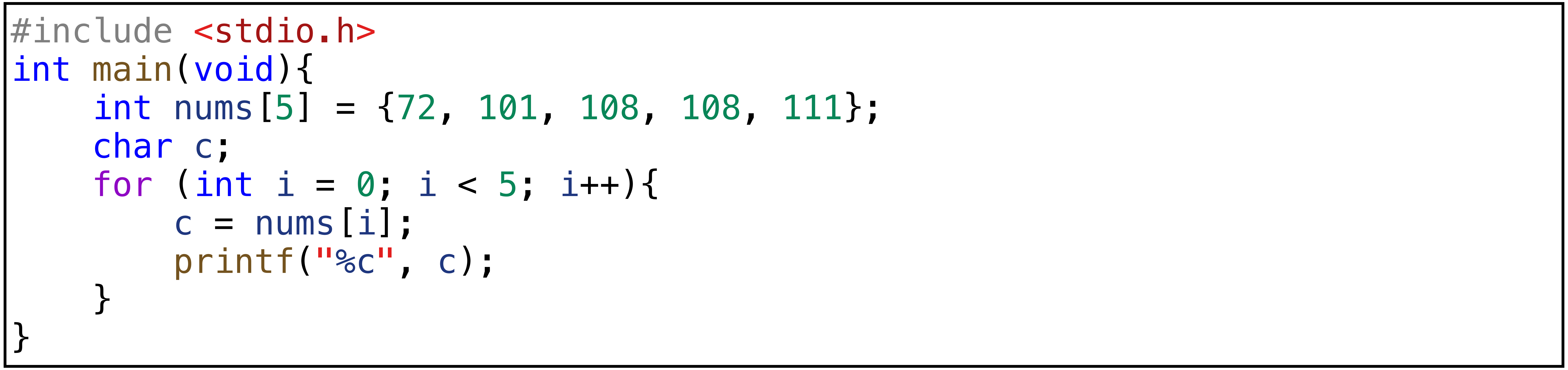
Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	[NULL]	32	20	[SPACE]	64	40	@	96	60	`
1	1	[START OF HEADING]	33	21	!	65	41	A	97	61	a
2	2	[START OF TEXT]	34	22	"	66	42	B	98	62	b
3	3	[END OF TEXT]	35	23	#	67	43	C	99	63	c
4	4	[END OF TRANSMISSION]	36	24	$	68	44	D	100	64	d
5	5	[ENQUIRY]	37	25	%	69	45	E	101	65	e
6	6	[ACKNOWLEDGE]	38	26	&	70	46	F	102	66	f
7	7	[BELL]	39	27	'	71	47	G	103	67	g
8	8	[BACKSPACE]	40	28	(	72	48	H	104	68	h
9	9	[HORIZONTAL TAB]	41	29	)	73	49	I	105	69	i
10	A	[LINE FEED]	42	2A	*	74	4A	J	106	6A	j
11	B	[VERTICAL TAB]	43	2B	+	75	4B	K	107	6B	k
12	C	[FORM FEED]	44	2C	,	76	4C	L	108	6C	l
13	D	[CARRIAGE RETURN]	45	2D	-	77	4D	M	109	6D	m
14	E	[SHIFT OUT]	46	2E	.	78	4E	N	110	6E	n
15	F	[SHIFT IN]	47	2F	/	79	4F	O	111	6F	o
16	10	[DATA LINK ESCAPE]	48	30	0	80	50	P	112	70	p
17	11	[DEVICE CONTROL 1]	49	31	1	81	51	Q	113	71	q
18	12	[DEVICE CONTROL 2]	50	32	2	82	52	R	114	72	r
19	13	[DEVICE CONTROL 3]	51	33	3	83	53	S	115	73	s
20	14	[DEVICE CONTROL 4]	52	34	4	84	54	T	116	74	t
21	15	[NEGATIVE ACKNOWLEDGE]	53	35	5	85	55	U	117	75	u
22	16	[SYNCHRONOUS IDLE]	54	36	6	86	56	V	118	76	v
23	17	[END OF TRANS. BLOCK]	55	37	7	87	57	W	119	77	w
24	18	[CANCEL]	56	38	8	88	58	X	120	78	x
25	19	[END OF MEDIUM]	57	39	9	89	59	Y	121	79	y
26	1A	[SUBSTITUTE]	58	3A	:	90	5A	Z	122	7A	z
27	1B	[ESCAPE]	59	3B	;	91	5B	[	123	7B	{
28	1C	[FILE SEPARATOR]	60	3C	<	92	5C	\	124	7C	|
29	1D	[GROUP SEPARATOR]	61	3D	=	93	5D	]	125	7D	}
30	1E	[RECORD SEPARATOR]	62	3E	>	94	5E	^	126	7E	~
31	1F	[UNIT SEPARATOR]	63	3F	?	95	5F	_	127	7F	[DEL]

Figure source: https://www.geeksforgeeks.org/ascii-table/

Decimal to char via ASCII encodings

C-course-materials/07-Strings/decimal_to_char_ascii.c

- ASCII (American Standard Code for Information Interchange)
- Characters are internally represented by their ASCII codes, which can be displayed in decimals.

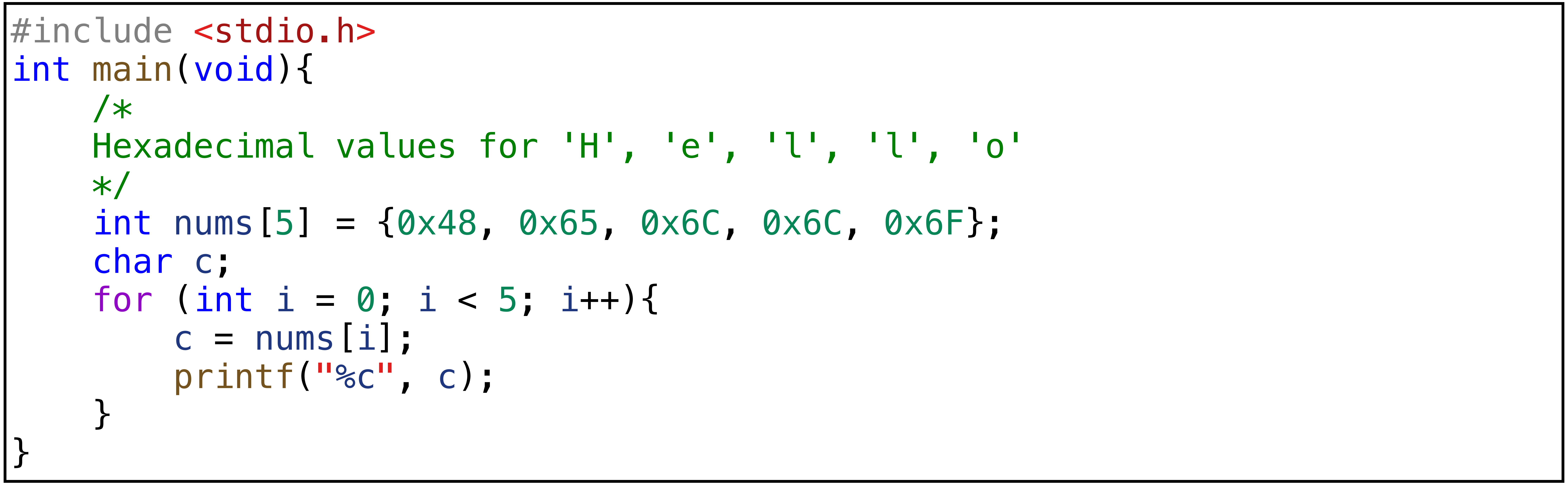


```c
#include <stdio.h>
int main(void){
    int nums[5] = {72, 101, 108, 108, 111};
    char c;
    for (int i = 0; i < 5; i++){
        c = nums[i];
        printf("%c", c);
    }
}
```

Hexadecimal to char via ASCII encodings

C-course-materials/07-Strings/hex_to_char_ascii.c

- ASCII (American Standard Code for Information Interchange)
- Characters are internally represented by their ASCII codes, which can be displayed in hexadecimals.
    - 0x is the prefix for hexadecimals in C. Hexadecimals are also integers.



```c
#include <stdio.h>
int main(void){
    /*
    Hexadecimal values for 'H', 'e', 'l', 'l', 'o'
    */
    int nums[5] = {0x48, 0x65, 0x6C, 0x6C, 0x6F};
    char c;
    for (int i = 0; i < 5; i++){
        c = nums[i];
        printf("%c", c);
    }
}
```

Print number from char via ASCII encodings

C-course-materials/07-Strings/print_num_from_char.c

- We can also print numerical values from characters.

```
#include <stdio.h>
int main(void){
    char a_string[5] = "Hello";
    for (int i = 0; i < sizeof(a_string); i++){
        printf("Decimal: %d, ", a_string[i]);
        printf("Hexadecimal: %x", a_string[i]);
        printf("\n");
    }
}
```

```
Decimal: 72, Hexadecimal: 48
Decimal: 101, Hexadecimal: 65
Decimal: 108, Hexadecimal: 6c
Decimal: 108, Hexadecimal: 6c
Decimal: 111, Hexadecimal: 6f
```

Passing a char variable to a function

C-course-materials/07-Strings/print_func.c

```c
#include <stdio.h>
void print_char(char c){
    printf("%c", c);
}
```

```c
void print_str(char str[]){
    for (int i = 0; str[i] != '\0'; i++){
        print_char(str[i]);
    }
}
```

```c
int main(void){
    char a_char = 'Y', str_5[] = "Hello";
    print_char(a_char);
    printf("\n");
    print_str(str_5);
}
```

Input characters and strings

Input a character

C-course-materials/07-Strings/scanf_char.c

```c
#include <stdio.h>
int main(void){
    char c;
    scanf("%c", &c);
    printf("%c\n", c);
    printf("Decimal: %d\n", c);
    printf("Hexadecimal: %x\n", c);
}
```

- We can use `scanf` to read a single character.
- The character can be printed as a decimal or hexadecimal number.

Input a string

C-course-materials/07-Strings/scanf_str.c

```c
#include <stdio.h>
int main(void){
    char a_string[10];
    scanf("%s", a_string);
    printf("%s!", a_string);
    return 0;
}
```

- A string, being an **array** of characters, represents the memory address of its first element.
    - Therefore, **we don’t need to use ‘&’** when using `scanf` to **input a string**.

Memory address of a string

C-course-materials/07-Strings/str_addr.c

```c
#include <stdio.h>
int main(void){
    char a_string[] = "Hello";
    printf("size: %d\n", sizeof(a_string));
    printf("a_string: %p\n", a_string);
    for (int i = 0; i < sizeof(a_string); i++){
        printf("a_string[%d]: %p\n", i, &a_string[i]);
    }
}
```

```
size: 6
a_string: 0x7fffffffdda2
a_string[0]: 0x7fffffffdda2
a_string[1]: 0x7fffffffdda3
a_string[2]: 0x7fffffffdda4
a_string[3]: 0x7fffffffdda5
a_string[4]: 0x7fffffffdda6
a_string[5]: 0x7fffffffdda7
```

← Address of \0

Input multiple strings

C-course-materials/07-Strings/scanf_multi_str.c

```c
#include <stdio.h>
int main(void){
    char a_string[10];
    while (scanf("%s", a_string) != EOF){
        printf("Now the string is: %s\n", a_string);
        for (int i = 0; i < sizeof(a_string); i++){
            if (a_string[i] == '\0') {
                break;
            }
            printf("%c ", a_string[i]);
        }
    }
    return 0;
}
```

- `scanf` will return EOF if it encounters the end of the input stream.

Array of strings

[Declaration] Array of strings

Declaration

```
char strings_arr[num_of_strings][string_length];
```

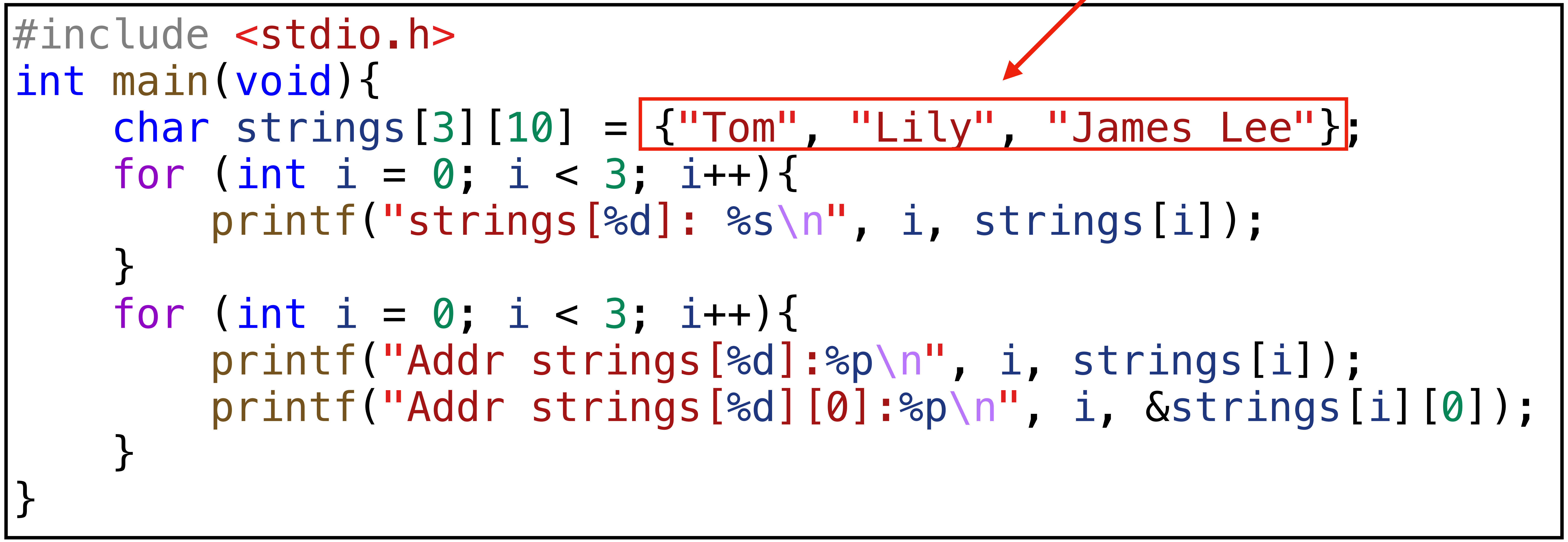
Declaration with an initializer

```
char strings_arr[num_of_strings][string_length] = {“str1”,
“str2”, ...};
```

Array of strings (Declaration with an initializer)

C-course-materials/07-Strings/arr_strings.c

- We can directly assign strings to an array with an initializer.



```c
#include <stdio.h>
int main(void){
    char strings[3][10] = {"Tom", "Lily", "James Lee"};
    for (int i = 0; i < 3; i++){
        printf("strings[%d]: %s\n", i, strings[i]);
    }
    for (int i = 0; i < 3; i++){
        printf("Addr strings[%d]:%p\n", i, strings[i]);
        printf("Addr strings[%d][0]:%p\n", i, &strings[i][0]);
    }
}
```

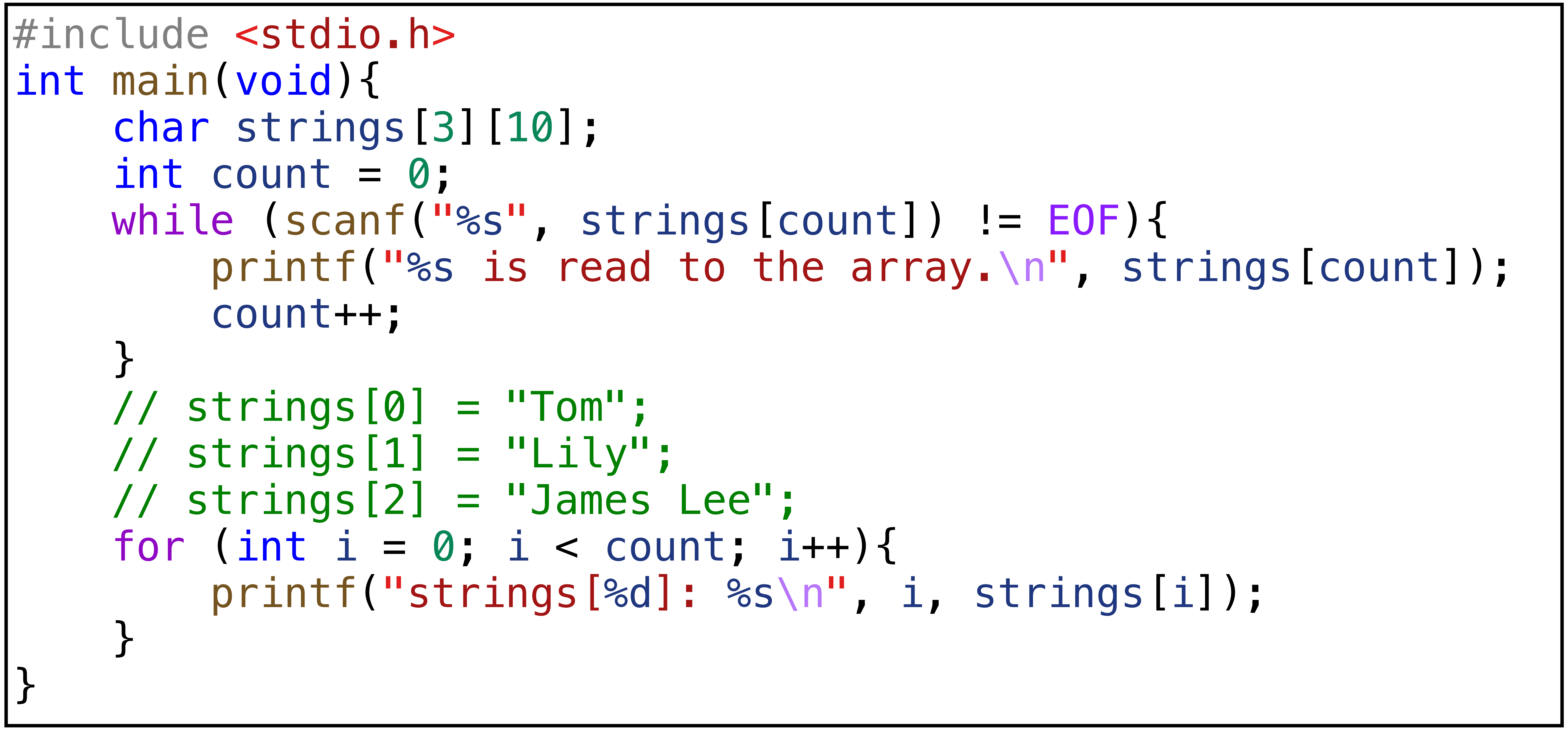
[Illustration] Array of strings

C-course-materials/07-Strings/arr_strings.c

`strings[0]`	T	o	m	\0	\0	\0	\0	\0	\0	\0
`strings[1]`	L	i	l	y	\0	\0	\0	\0	\0	\0
`strings[2]`	J	a	m	e	s		L	e	e	\0

Array of strings (value assignment with while)

C-course-materials/07-Strings/arr_strings_while.c



```c
#include <stdio.h>
int main(void){
    char strings[3][10];
    int count = 0;
    while (scanf("%s", strings[count]) != EOF){
        printf("%s is read to the array.\n", strings[count]);
        count++;
    }
    // strings[0] = "Tom";
    // strings[1] = "Lily";
    // strings[2] = "James Lee";
    for (int i = 0; i < count; i++){
        printf("strings[%d]: %s\n", i, strings[i]);
    }
}
```

Practical Questions

[Question] Check if a String is a Palindrome

Write a C program to determine whether a given string is a palindrome.

- Input:

```
racecar
```

- Output:

```
Palindrome
```

- Input:

```
hello
```

- Output:

```
Not Palindrome
```

Check if a String is a Palindrome

C-course-materials/07-Strings/check_palindrome.c

```c
#include <stdio.h>
#include <string.h>
int is_palindrome(char str[]) {
    int length = strlen(str); // strlen excludes '\0'
    for (int i = 0; i < length / 2; i++) {
        if (str[i] != str[length - i - 1]) {
            return 0;
        }
    }
    return 1;
}
```

index	0	1	2	3	4
string	H	e	l	l	o

Check if a String is a Palindrome

C-course-materials/07-Strings/check_palindrome.c

```c
int main(void) {
    char str[100];
    printf("Please enter a string: ");
    scanf("%s", str);
    if (is_palindrome(str)) {
        printf("Palindrome\n");
    } else {
        printf("Not a palindrome\n");
    }
    return 0;
}
```

[Questions] W12 Quiz

- Q1:
  - Name the code into `studentID_q1.c `
- Q2:
  - Name the code into `studentID_q2.c `
- Compress your code into `studentID_w12quiz.zip` and upload it.