Week 3 - Lecture 1

Arrays

Edited by: Heshan Du

Autumn 2024

University of Nottingham
UK | CHINA | MALAYSIA

Overview

- **One-dimensional array**
- Two-dimensional array
- String or char array

Memory Layout

Memory Address	Memory Content
0	
1	
2	
...	
5000	10
5001	0
5002	0
5003	0
...	
n-1	

Unique number

Each cell = 1 byte

int a = 10;

Compiler reserves 4 consecutive bytes and store the value 10 in it

Array Memory Layout

An array is a *continuous* block of memory to store values of the *same type*.

Memory Address	Memory Content
0	
1	
2	
...	
5000	10
5001	0
5002	0
5003	0
...	
n-1	

Declaring an Array

- **data_type** array_name[number_of_elements];

```
int arr[1000];
```

The number of elements remains fixed after declaring it.

```
#define SIZE 10
int arr[SIZE];
```

Values are stored in consecutive memory locations. arr takes 40 bytes (10 integer elements, 4 bytes each).

- Avoid useless waste of memory, declare an array with the length that is needed
- Access an array element e.g., arr[0], …, arr[9]

Index starts from zero

Array Initialisation

- int arr[4] = {10, 20, 30, 40};
- int arr[10] = {10, 20};

The values of arr[0] and arr[1] become 10 and 20, respectively, the rest of the elements are set to zero.

0	10
1	20
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

- int arr[] = {10, 20, 30, 40};

0	10
1	20
2	30
3	40

Creates an array with four items.

Assigning Values

- int arr[4] = {0};
  arr[0] = 1;

The values of arr[0], arr[1], arr[2] and arr[3] are set to zero.

- char arr[4] = {'\0'};
  arr[4] = 'a';

'\0' is null character and is used to end a string

Array out of bound, but the compiler won't tell you!!!

char and int in C

Dec	Hx	Oct	Char		Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL	(null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH	(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX	(start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX	(end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT	(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ	(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK	(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL	(bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS	(backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB	(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF	(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT	(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF	(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR	(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO	(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI	(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE	(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1	(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2	(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3	(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4	(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK	(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN	(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB	(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN	(cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM	(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB	(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC	(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS	(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS	(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS	(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US	(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

char Type

- A character in the ASCII set is represented by an integer between 0 and 127
- char ch;
  ch = 'A';
  printf("Char = %c and its ASCII code is %d\n", ch, ch);
  ch++;
  printf("Char = %c and its ASCII code is %d\n", ch, ch);

Char = A and its ASCII code is 65

First ch prints character A, the second ch prints ASCII value of A which is 65

First ch prints character B, the second ch prints ASCII value of B which is 66

Char = B and its ASCII code is 66

Array Out of Bound

C does **NOT** check if the array index you try to access is valid!

```
#include<stdio.h>
int main(void){
  int std[4];
  int i;
  std[0] = 100; //valid
  std[1] = 200; //valid
  std[2] = 300; //valid
  std[3] = 400; //valid
  std[4] = 500; //invalid(out of bounds index)
  //printing all elements
  for( i=0; i<5; i++ )
    printf("std[%d]: %d\n",i,std[i]);
  return 0;
}
```

Output:

```
std[0]: 100
std[1]: 200
std[2]: 300
std[3]: 400
std[4]: 2314
```

Array Out of Bound

0	100
1	200
2	300
3	400
4	X

```
#include<stdio.h>
int main(void){
  int std[4];
  int i;
  std[0] = 100; //valid
  std[1] = 200; //valid
  std[2] = 300; //valid
  std[3] = 400; //valid
  std[4] = 500; //invalid(out of bounds index)
  //printing all elements
  for( i=0; i<5; i++ )
    printf("std[%d]: %d\n",i,std[i]);
  return 0;
}
```

Output:

```
std[0]: 100
std[1]: 200
std[2]: 300
std[3]: 400
std[4]: 2314
```

Overview

- One-dimensional array
- **Two-dimensional array**
- String or char array

Two-Dimensional Array

- Stored as “flat” continuous memory.

```
C:\Users\z2017233\Desktop>array
Please enter the day and time you have some water: 0 1
Please enter the amount of water: 3
Please enter the day and time you have some water: 0 2
Please enter the amount of water: 4
Please enter the day and time you have some water: 0 3
Please enter the amount of water: 5
Please enter the day and time you have some water: 6 10
Please enter the amount of water: 9
Please enter the day and time you have some water: 7 10


The amount of water you drank:
0 3 4 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0
Total number of glasses: 21
```

```c
int water[7][24] = {0};

int time = 0;
int day = 0;
int sum = 0;

do
{
    printf("Please enter the day and time you have some water: ");
    scanf("%d%d", &day, &time);

    if((time < 0) || (time >= 24) || (day < 0) || (day >= 7))
    {
        break;
    }

    printf("Please enter the amount of water: ");
    scanf("%d", &water[day][time]);
    sum = sum + water[day][time];

}while((time >= 0) && (time < 24) && (day >= 0) && (day < 7));
```

arr

	0	1	2	3
0				
1			arr[1, 2]	
2				
3		arr[3, 1]		
4				
5				

int arr[6][4];

```
C:\Users\z2017233\Desktop>array
Please enter the day and time you have some water: 0 1
Please enter the amount of water: 3
Please enter the day and time you have some water: 0 2
Please enter the amount of water: 4
Please enter the day and time you have some water: 0 3
Please enter the amount of water: 5
Please enter the day and time you have some water: 6 10
Please enter the amount of water: 9
Please enter the day and time you have some water: 7 10


The amount of water you drank:
0 3 4 5 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
0 0 0 0 0 0 0 0 0 9 0 0 0 0 0 0 0 0 0 0 0 0 0 0
Total number of glasses: 21
```

```c
int water[7][24] = {0};

int time = 0;
int day = 0;
int sum = 0;

do
{
    printf("Please enter the day and time you have some water: ");
    scanf("%d%d", &day, &time);

    if((time < 0) || (time >= 24) || (day < 0) || (day >= 7))
    {
        break;
    }

    printf("Please enter the amount of water: ");
    scanf("%d", &water[day][time]);
    sum = sum + water[day][time];

}while((time >= 0) && (time < 24) && (day >= 0) && (day < 7));
```

Two-Dimensional Array

- **data_type** array_name [number_of_rows][number_of_columns];

```
int a[3][4];
```

	Column 0	Column 1	Column 2	Column 3
Row 0	a[0] [0]	a[0] [1]	a[0] [2]	a[0] [3]
Row 1	a[1] [0]	a[1] [1]	a[1] [2]	a[1] [3]
Row 2	a[2] [0]	a[2] [1]	a[2] [2]	a[2] [3]

Column index

Array name

Row index

- The elements are stored in row order with the elements of row 0 first, followed by the elements of row 1, and so on.

2D Array Initialisation

- int arr[3][3] = {{10, 20, 30},{40, 50, 60},{70, 80, 90}};
- int arr[3][3] = {10, 20, 30, 40, 50, 60, 70, 80, 90};
- int arr[3][3] = {{10, 20},{40, 50},{70}};

Remaining elements are set to zero.

- int arr[][3] = {10, 20, 30, 40, 50, 60};

Same as arr[2][3];

Overview

- One-dimensional array
- Two-dimensional array
- **String or char array**

Array: Char to String

A string such as "hello" is really an array of individual characters in C.

For example,

```
char array1[] = "first";
```

initializes the elements of array *array1* to the individual characters in the string literal "first".

The preceding definition is equivalent to

```
char array1[] = { 'f', 'i', 'r', 's', 't', '\0' };
```

String

ASCII code for '\0' is 0
ASCII code for 0 is 48!!

- A series of characters that end with a special character, the null character, '\0'
- e.g., "message" requires 8 bytes (7 character + null character)
- char str[8];
- char str[8] = "message";
- char str[] = "message";
- char str[] = {'m', 'e', 's', 's', 'a', 'g', 'e', '\0'};

Could get unpredicted results if no space for '\0'

Writing Strings: examples

```
char str[10];
str[0] = 'a';
printf("%s\n", str);
```

```
char str[10] = {0};
str[0] = 'a';
printf("%s\n", str);
```

```
char str[10];
str[0] = 'a';
str[1] = '\0';
printf("%s\n", str);
```

printf and '\0'

- printf prints until null character.

```c
#include <stdio.h>

int main(void)
{
    char myString3[10] = "!\0!\0!\0!\0!";
    char myString2[10] = "Hey There\0";
    char myString[10] = "!!!!!!!!!";


    printf("%s\n", myString3);
    printf("%s\n", myString2);
    printf("%s\n", myString);
```

```
C:\Users\z2017233\Desktop>array
!
Hey There
!!!!!!!!!
```

Read Strings

- scanf() reads characters until it encounters a space character i.e., space, tab or new line character
- Then appends a null character at the end of the string

Read Strings (2)

```
#include <stdio.h>
int main()
{
    char name[20];
    printf("Enter name: ");
    scanf("%s", name);
    printf("Your name is %s.", name);
    return 0;
}
```

Output:
Enter name: Dennis Ritchie
Your name is Dennis.

Read Strings(3)

fgets() function reads a line of string, puts() displays the string.

```
#include <stdio.h>
int main()
{
   char name[30];
   printf("Enter name: ");
   fgets(name, sizeof(name), stdin);  // read string
   printf("Name: ");
   puts(name);    // display string
   return 0;
}
```

Output:
Enter name: Tom Hanks
Name: Tom Hanks

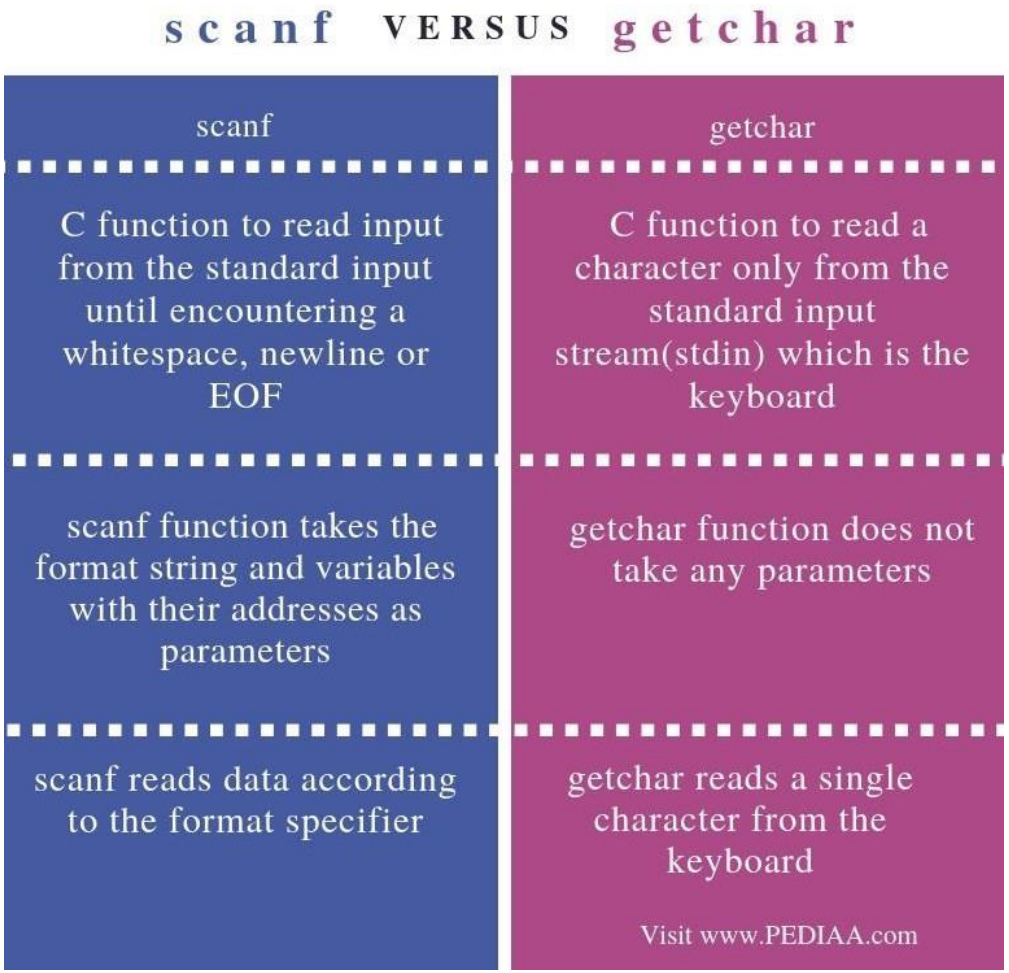
getchar() function

- … and we are back on input buffer again!!!

```
#include <stdio.h>

int main () {
  char c;

  printf("Enter character: ");
  c = getchar();

  printf("Character entered: ");
  putchar(c);

  return(0);
}
```

getchar vs. scanf

- *scanf* is a **formatted** of reading input from the keyboard.
- *getchar* reads a *single* character from the keyboard.



Source: https://pediaa.com/what-is-the-difference-between-scanf-and-getchar/

Example: calculate average

```
#include <stdio.h>
int main()
{
    int marks[10], i, n, sum = 0, average;

    printf("Enter number of elements: ");
    scanf("%d", &n);

    for(i=0; i<n; i++)
    {
        printf("Enter number%d: ",i+1);
        scanf("%d", &marks[i]);

        // adding integers entered by the user to the sum variable
        sum += marks[i];
    }

    average = sum/n;
    printf("Average = %d", average);

    return 0;
}
```

0	45
1	35
2	38
3	31
4	49
5	
6	
7	
8	
9	

Output:

Enter n: 5
Enter number1: 45
Enter number2: 35
Enter number3: 38
Enter number4: 31
Enter number5: 49
Average = 39

Summary

- One-dimensional array
- Two-dimensional array
- String or char array