

# C programming

*Arrays, Decision and Looping*



# Lesson Objectives

- Introduce
- Array in C
- Decision in C
- Looping In C

## Section 1

# ***INTRODUCE***

## Section 2

# ARRAY IN C

- *What is Array?*
- *Multidimensional Arrays*
- *Array in memory*
- *How to declare an Array?*
- *How works with array?*
- *When will using Array?*

# What is Array?

- Each member of an array is identified by unique index or subscript assigned to it
- The dimension of an array is determined by the number of indices needed to uniquely identify each element
- An index is a positive integer enclosed in [ ] placed immediately after the array name
- An index holds integer values starting with zero
- An array with 11 elements will look like -

**Player[0], player[1], player[2],.... Player[10]**

- The simplest and the most commonly used multidimensional array is the two - dimensional array
- A two-dimensional array can be thought of as an array of two single dimensional arrays
- A two-dimensional array looks like a railway time-table consisting of rows and columns
- A two-dimensional array is declared as -

**int temp[4][3];**

val[0]	val[1]	val[2]	val[3]	val[4]	val[5]	val[6]
11	22	33	44	55	66	77
88820	88824	88828	88832	88836	88840	88844

BeginnersBook.com

All the array elements occupy contiguous space in memory. There is a difference of 4 among the addresses of subsequent neighbours, this is because this array is of integer types and an integer holds 4 bytes of memory.

## Memory representation of array

s[0][0]	s[0][1]	s[1][0]	s[1][1]	s[2][0]	s[2][1]	s[3][0]	s[3][1]
1234	56	1212	33	1434	80	1312	78
65508	65510	65512	65514	65516	65518	65520	65522



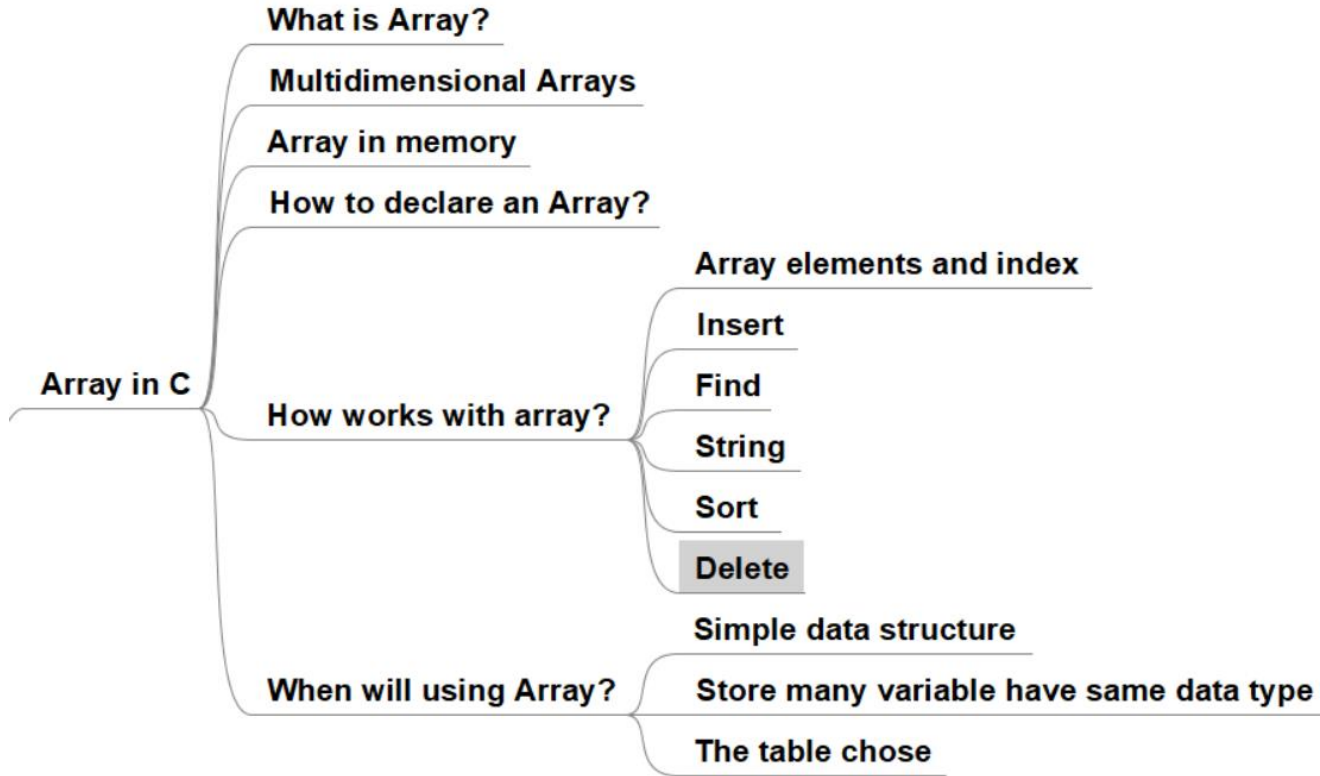
An array is defined in the same way as a variable is defined. The only change is that the array name is followed by one or more expressions, enclosed within square brackets [], specifying the array dimension.

```
Storage_Class  data_types  array_name[size]  
int player[11];
```

# *How works with array?*

- Array elements and index
- Insert
- Find
- String
- Sort
- Delete

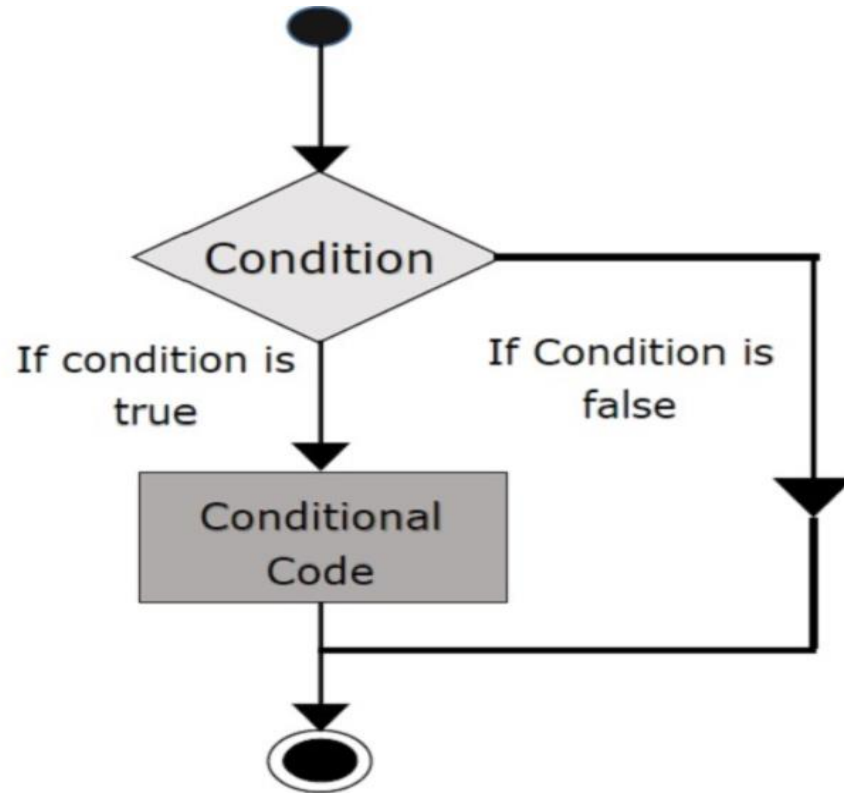
***When will using Array?***



## Section 3

# DECISION IN C

- *Introduce*
- *How to build an expression?*
- *if, else...* statement
- *switch...* statement



- **Conditional statements enable us to change the flow of the program**
- **A conditional statement evaluates to either a zero or not zero.**

**Example :**

**To find whether a number is even or odd we proceed as follows :**

- 1. Accept a number**
- 2. Find the remainder by dividing the number by 2**
- 3. If the remainder is zero, the number is "EVEN"**
- 4. Or if the remainder is not zero the number is "ODD"**



# *if, else...* statement

```
if (expression)  
    statement;
```

```
if (expression)  
    statement;  
else  
    statement;
```

# if, else... statement

```
if (expression)
    statement;
else if (expression)
    statement;
else if (expression)
    statement;
.
.
.
else
    statement;
```

```
if (expl)
{
    if (exp2) statement1;
    if (exp3) statement2;
    else statement3;          /*with if (exp3) */
}
else statement4;             /* with if (expl) */
```

# switch... statement

```
switch (expression)
{
    case constant1:
        statement sequence
        break;
    case constant2:
        statement sequence
        break;
    case constant3:
        statement sequence
        break;
    .
    .
    .
    default:
        statement sequence
}
```

## Program to check whether the entered lowercase character is vowel or 'z' or a consonant

```
#include <stdio.h>
main ()
{
    char ch;
    clrscr ();

    printf ("\nEnter a lower cased alphabet (a - z)
: ");
    scanf ("%c", &ch);
```

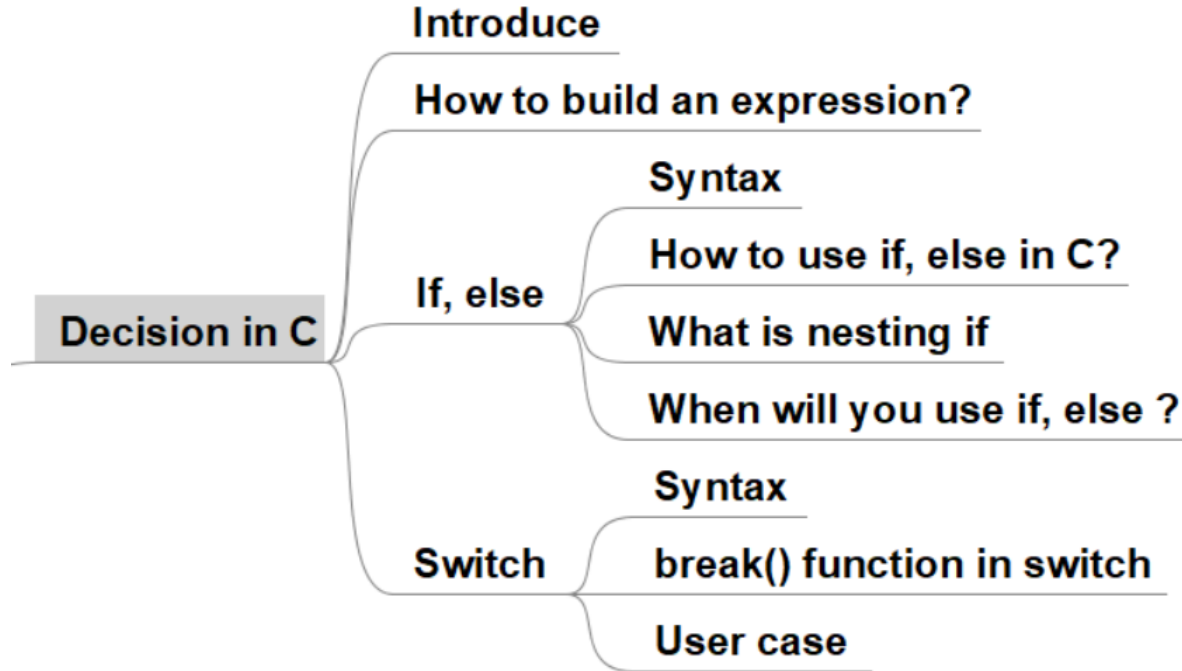
# switch... statement

```
if (ch < 'a' || ch > 'z')
    printf("\nCharacter not a lower cased alphabet");
else
    switch (ch)
    {
        case 'a' :
        case 'e' :
        case 'i' :
        case 'o' :
        case 'u' :
            printf("\nCharacter is a vowel");
            break;
        case 'z' :
            printf ("\nLast Alphabet (z) was entered");
            break;
        default :
            printf("\nCharacter is a consonant");
            break;
    }
}
```

# 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	[ENG 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]



## Section 4

# LOOPING IN C



- *What is looping in C?*
- *Syntax*
- *Enter and exit, break looping follow.*
- *How to use looping?*
- *Key words for Looping*

**Section of code in a program  
which is executed repeatedly,  
until a specific condition is satisfied**

The for loop

The while loop

The do....while loop

# Syntax: The for loop

```
for (initialize counter; conditional test; re-evaluation parameter)
{
    statement
}
```

- The initialize counter is an assignment statement that sets the loop control variable, before entering the loop
- The conditional test is a relational expression, which determines, when the loop will exit
- The evaluation parameter defines how the loop control variable changes, each time the loop is executed

```
while (condition is not zero)  
    statement ;
```

**The while loop repeats statements while a certain specified condition is not zero**

# Syntax: The **do....while** loop

```
do{  
    statement;  
} while (condition);
```

- In the **do while** loop the body of the code is executed once before the test is performed
- When the condition becomes zero in a **do while** the loop will be terminated, and the control goes to the statement that appears immediately after the **while** statement

***Enter and exit, break looping follow.***

***Enter and exit, break looping follow ?***

# *How to use looping?*

- **Know number of loop.**
- **Unknow number of loop.**



# *Key words for Looping*

- Goto
- Continue
- Exit()
- Break()
- Return

## goto label

- The goto statement transfers control to any other statement within the same function in a C program
- It actually violates the rules of a strictly structured programming language
- They reduce program reliability and make program difficult to maintain

## continue statement

- The continue statement causes the next iteration of the enclosing loop to begin
- When this statement is encountered, the remaining statements in the body of the loop are skipped and the control is passed on to the re-initialization step

## exit() function

- The exit() is used to break out of the program
- The use of this function causes immediate termination of the program and control rests in the hands of the operating system

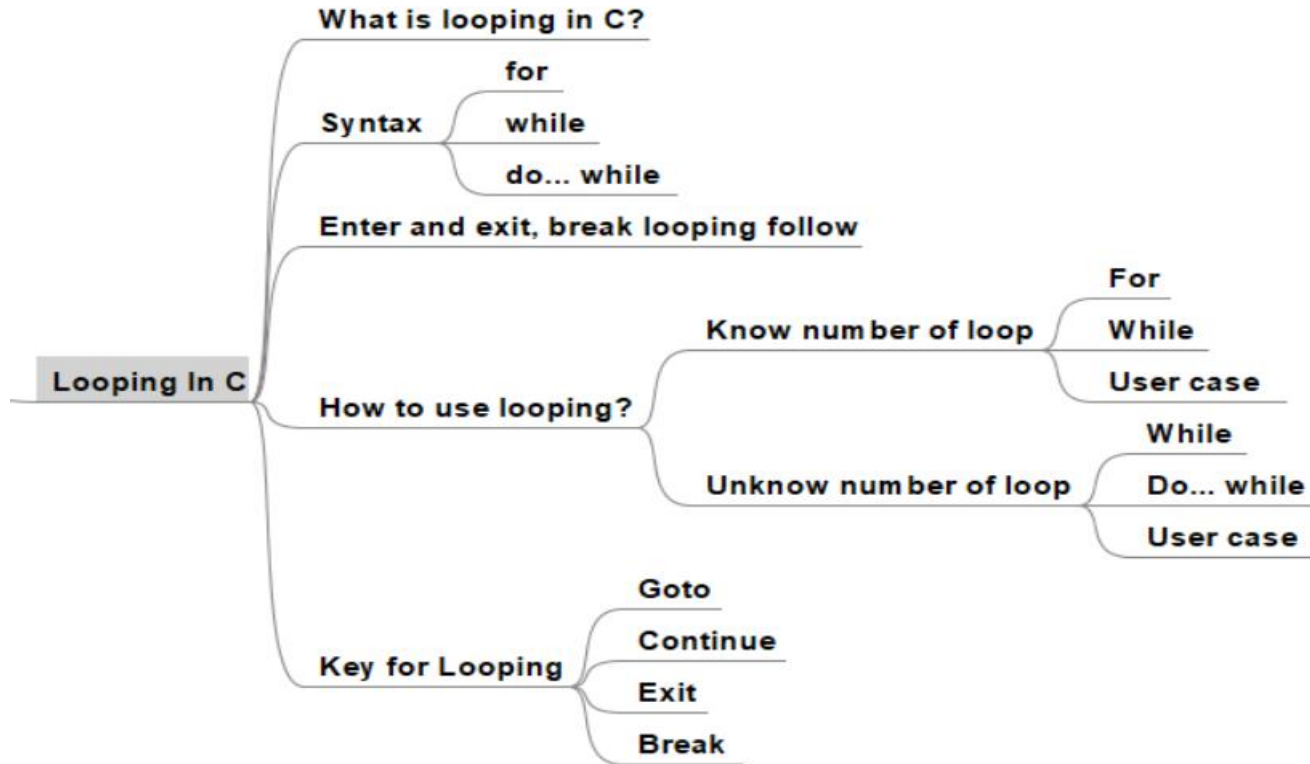
## break statement

- The break statement is used to terminate a case in a switch statement
- It can also be used for abrupt termination of a loop
- When the break statement is encountered in a loop, the loop is terminated immediately and control is passed to the statement following the loop

## return expression

- The return statement is used to return from a function
- It causes execution to return to the point at which the call to the function was made
- The return statement can have a value with it, which it returns to the program

# Looping In C. Summary



- Array in C
- Decision in C
- Looping In C



# Thank you

