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

Array in C. Agenda

- *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 -

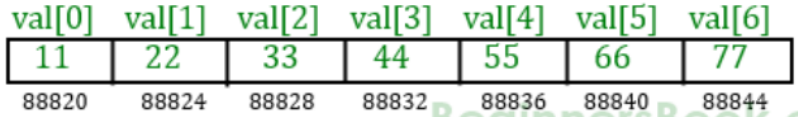
Player[0], player[1], player[2],…. Player[10]

Multidimensional Arrays

- The simplest and the most commonly used multi-dimensional 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];
```

Array in memory



All the array elements occupy contigious 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

How to declare an Array?

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]

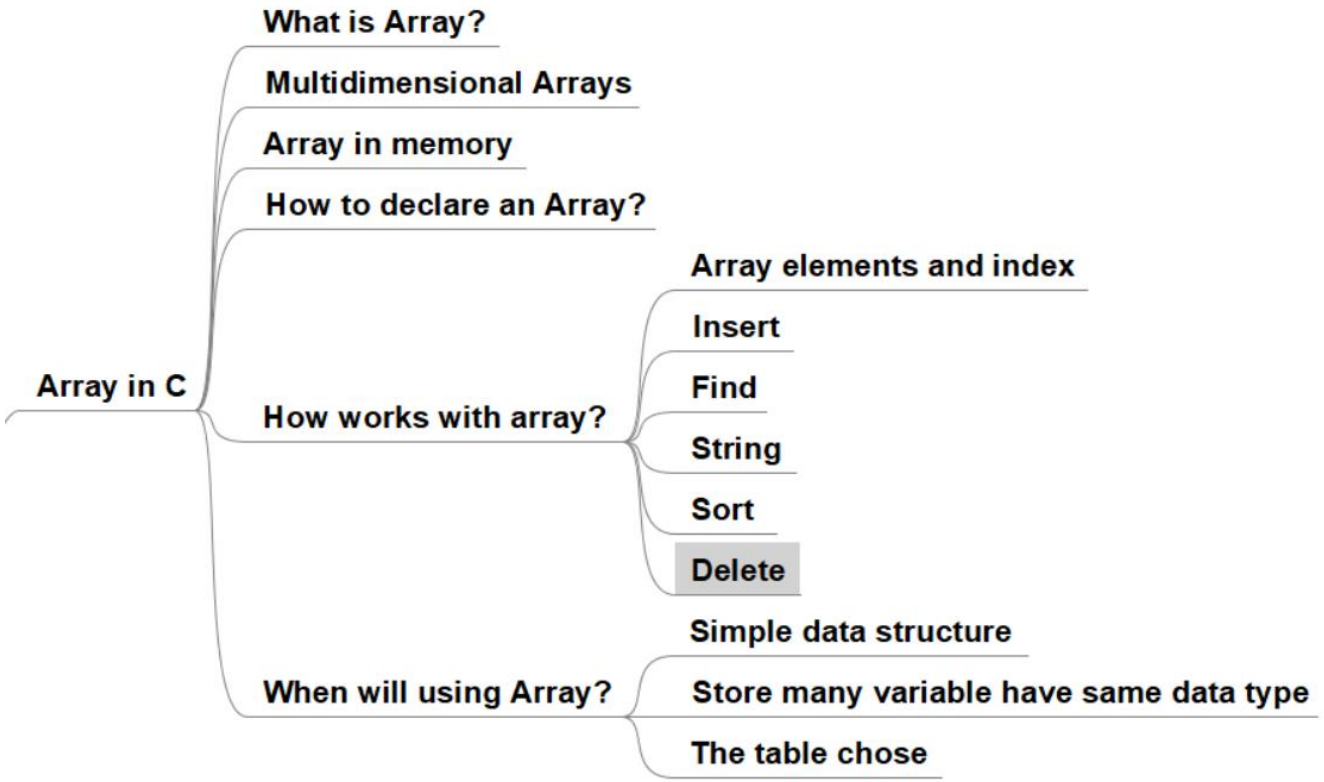
int player[11];

How works with array?

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

When will using Array?

Array in C: Summary



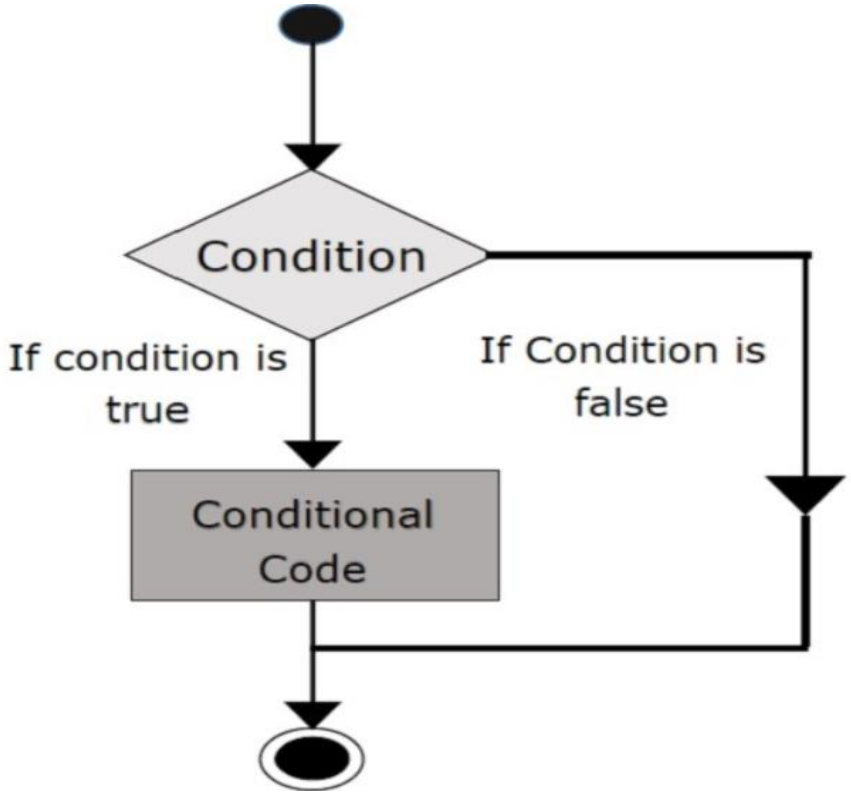
Section 3

DECISION IN C

Decision in C. Agenda

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

Introduce


Condition
If condition is true
If Condition is false
Conditional Code

How to build an expression?

- **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 (exp1)
{
    if (exp2) statement1;
    if (exp3) statement2;
    else statement3;          /*with if (exp3) */
}
else statement4;              /* with if (exp1) */
```

switch… statement

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

switch… statement

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

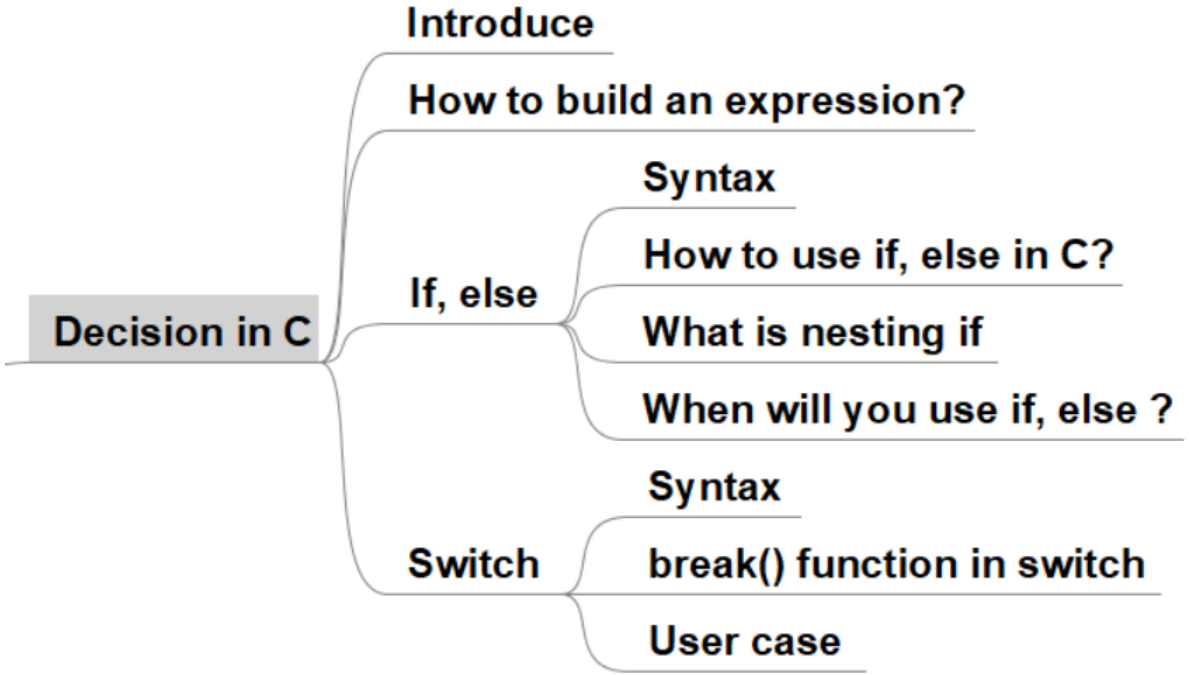
```
#include <stdio.h>
   main ()
   {
      char ch;
      clrscr ();

      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;
                        }
        }
```

Decision in C. Summary



Section 4

LOOPING IN C

Looping In C. Agenda

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

What is looping in C?

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

Syntax

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

Syntax: The while loop

```
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 ?

How to use looping?

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

Key words for Looping

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

Key words for Looping: Goto

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

Key words for Looping: Continue

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

Key words for Looping: Exit()

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

Key words for Looping: Break()

break statement

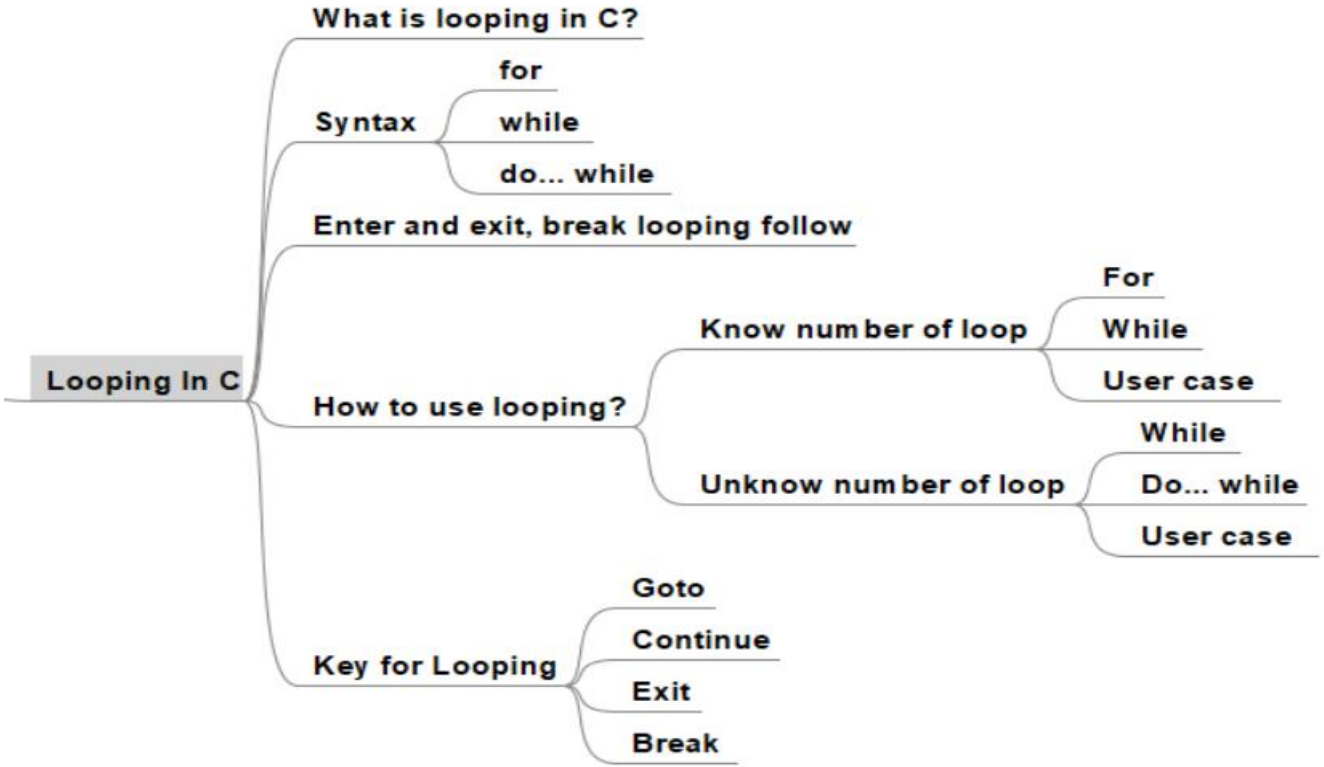
- The break statement is used to terminate a case in a switch statement
- It can also be used for abrupt termination of a loop
- When the break statement is encountered in a loop, the loop is terminated immediately and control is passed to the statement following the loop

Key words for Looping: Return

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



Lesson Summary

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

Thank you