The left side of the slide features several vertical stripes in shades of blue and green. Overlaid on these stripes are several blue circles of varying sizes, arranged in a cluster that tapers towards the bottom left.

C++

EXPRESSION

TOKENS

- A token is the smallest element of a program that is meaningful to the compiler.
- Tokens can be classified as follows:
 - Keywords
 - Identifiers
 - Constants
 - Strings
 - Special Symbols
 - Operators

KEYWORDS

- Keywords are pre-defined or reserved words in a programming language.
- To perform a specific function in a program.
- Since keywords are referred names for a compiler
- They can't be used as variable names
- You cannot redefine keywords.

**auto double int struct break else long
switch case enum register typedef char
extern return union const float short
unsigned continue for signed void
default goto sizeof volatile do if static
while**

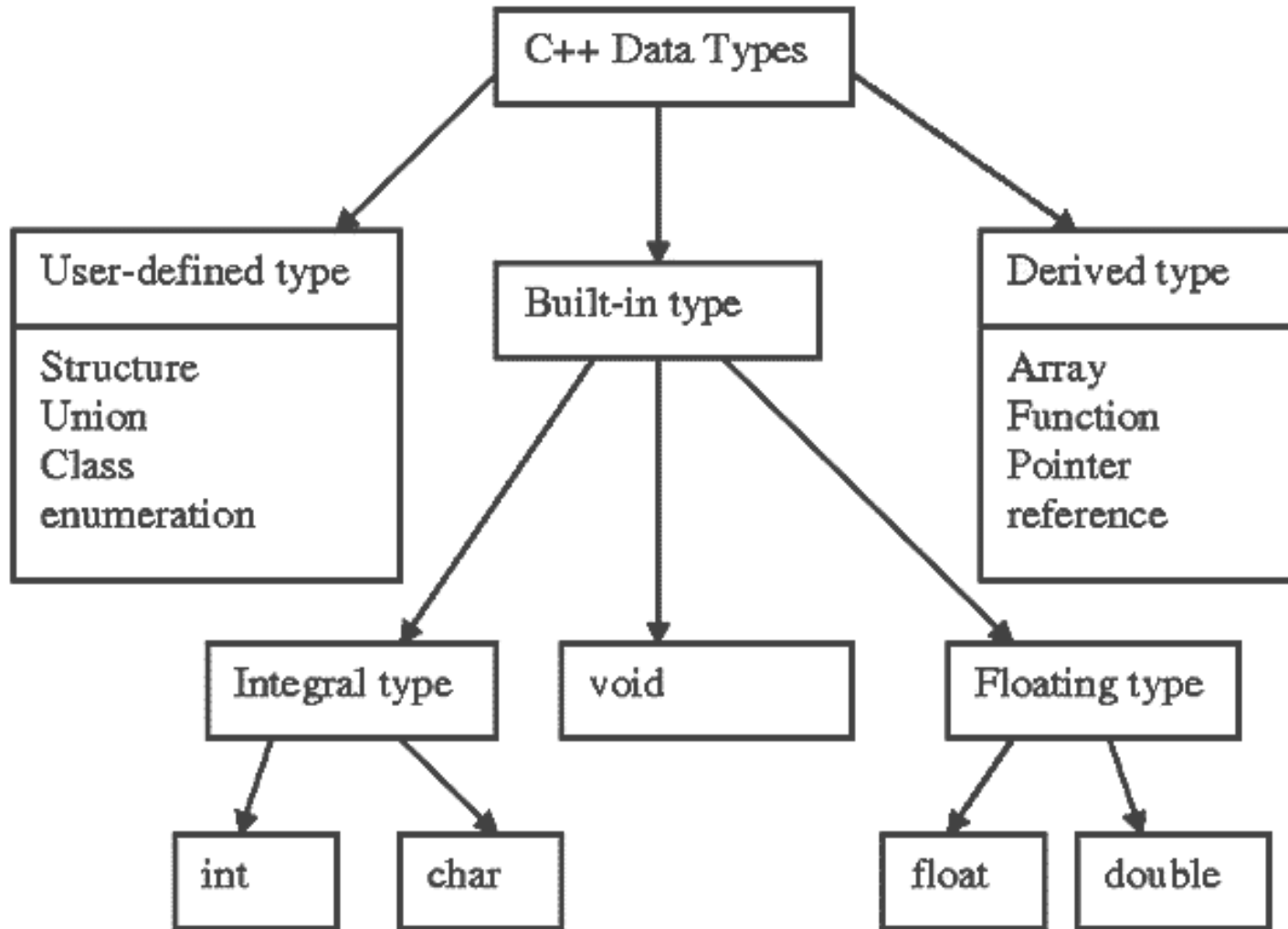
IDENTIFIERS

- ✓ Identifiers are used for identification purpose.
- ✓ Used as the general terminology for naming of variables, functions and arrays.
- ✓ consist arbitrarily long sequence of letters and digits.
- ✓ Letter or the underscore(_) can be first character.
- ✓ Identifier names must differ in spelling and case from any keywords.

CONT..

- Can be 31 characters long as only first 31 characters are significant.
- Keywords cant be used as identifiers; as they are reserved for special use.
- No spacial symbol can be used other than underscore
- Identifiers means a sequence of
 - uppercase(A,B,C,.....,Y,Z)
 - lowercase(a,b,c,.....,y,z) letters,
 - numbers(0,1 ,2,.....,9)

BASIC DATA TYPES, USER-DEFINED DATA TYPES



- Both C and C++ compilers support all the built-in data types.
- The basic data types may have several modifiers preceding them to serve the needs of various situations.
- The modifiers **signed**, **unsigned**, modifier **long** may also be applied to double.
- The void have no preceding modifiers.
- Data type representation is machine specific in C++.
- Data types in C++ is mainly divided into two types:
- **User defined data type:** These data types are defined by user itself.
- Like, defining a class in C++ or a structure.

- **Primitive Data Types:** These data types are built-in or predefined data types and can be used directly by the user to declare variables.
- Example: int, char , float, bool etc.
- Primitive data types available in C++ are:
 - Integer
 - Character
 - Boolean
 - Floating Point
 - Double Floating Point
 - Valueless or Void
 - Wide Character

○ Integer:

- Keyword used for integer data types is int.
- Requires 4 bytes of memory space
- Ranges from -2147483648 to 2147483647.

○ Character:

- Used for storing characters.
- char keyword is used
- Requires 1 byte of memory space
- Ranges from -128 to 127 or 0 to 255.

○ Boolean:

- Used for storing boolean or logical values.

- Can store either true or false.
- Keyword used for boolean data type is bool.

○ Floating Point:

- Used for storing single **precision** floating point values or decimal values.
- **float** keyword used
- Requires 4 byte of memory space.

○ Double Floating Point:

- Used for storing double precision floating point values or decimal values.
- **double** keyword is used
- Requires 8 byte of memory space.

○ void:

- Without any value.
- Represents a valueless entity.
- Used for those function which does not returns a value.

○ Wide Character:

- Also a character data type but this data type has size greater than the normal 8-bit datatype.
- Represented by **wchar_t**.
- It is generally 2 or 4 bytes long.

SYMBOLIC CONSTANT

- Two ways of creating symbolic constants

1. **Enum**

2. **Using Const**

- Enumerated constants create a set of constants with a single statement
- An **enumeration** is a user-defined type
- Consists set of named integral constants that are known as enumerators
- Enumerated data type is compiled as an integer
- Its possible values are any type of integer constant specified.
- Enum value can be modified

- If it is not specified, the integer value equivalent to the first possible value is **0** and the following ones follow a +1 progression.
- But you can give a name, a specific value by adding an initializer.
- Enum datatypes are becomes new type not int like C
- Int value cannot be directly assigned to enum type var.
- In C enum is of int type
- Syntax
 - `enum model_name { value1, value2, value3, . . }`
`object_name;`
- `enum colors_t {black, blue, green, cyan, red, purple, yellow, white};`

- Thus, in our data type **colors_t** that we defined before, **black** would be equivalent to **0**, **blue** would be equivalent to **1**, **green** to **2** and so on.

- Example

```
colors_t mycolor;  
mycolor = blue;  
if (mycolor == green)  
    mycolor = red;
```

- An enum variable takes only one value out of many possible values.

- This makes enum a good choice to work with flags

- enum days{son,mon,tue,sat};

- days today=son;

- cout<<today;

enum
Variable

values

CONSTANTS

- Is a variable, except their value never be changed during the program execution once defined.
- *Constants* are expressions with a fixed value.
- Fixed values that the program may not alter and they are called **literals**.

Integer Literals

- An integer literal can be a decimal, octal, or hexadecimal constant.
- A prefix specifies the base or radix: 0x or 0X for hexadecimal, 0 for octal, and nothing for decimal.
- 212 215

Character Literals

- Character literals are enclosed in single quotes.

- If the literal begins with L
 - e.g. 'x'

Floating-point Literals

- A floating-point literal has an integer part, a decimal point, a fractional part, and an exponent part.
- You can represent floating point literals either in decimal form or exponential form.
 - 3.14159

String literals

- String literals are enclosed in double quotes.
- A string contains characters that are similar to character literals: plain characters, escape sequences, and universal characters.
 - “hello”

- There are two simple ways in C++ to define constants –
 - Using **#define** preprocessor.
 - Using **const** keyword.

The #define Preprocessor

- Syntax - #define identifier value
 - #define max 10

The const Keyword

- **const** keyword used to declare constants with a specific type as follows –
- const Datatype variable = value;
- const x=2; x is int (DT is not given default int)
- const int a=22 ⇔ int const a=44 //Ok
- Constant variable should be initialised in c++

Data Type	Size	Range
short int	2	-32,768 to 32,767
unsigned short int	2	0 to 65,535
unsigned int	4	0 to 4,294,967,295
int	4	-2,147,483,648 to 2,147,483,647
long int	4	-2,147,483,648 to 2,147,483,647
unsigned long int	4	0 to 4,294,967,295
long long int	8	-(2^63) to (2^63)-1
unsigned long long int	8	0 to 18,446,744,073,709,551,615
signed char	1	-128 to 127
unsigned char	1	0 to 255
float	4	
double	8	
long double	12	
wchar_t	2 or 4	1 wide character

9/10/2022

TYPE COMPATIBILITY

- C++ is Strict with type compatibility
- C++ checks type of values assigned to any another type variable.
- Type compatibility is very close to automatic or implicit type conversion
- Cast must be used when different types value are assigned to one another
- Like int, short int, long int or char, signed char, unsigned char
- Values must be same else cast must be applied in assignment

- In assignment compatibility, if the type of variable assigned to another variable is different
- It results into loss of value of assigned variable if the size of assigned variable is large than the size of variable to which it is assigned.

- For Example

```
float n1 = 12.5;
```

```
int n2 = n1;
```

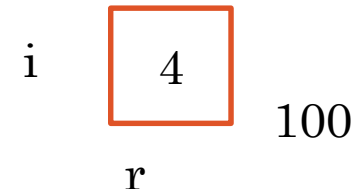
- *Int n = 3/2;*

```
cout<<n;
```

- *Cast is needed for proper result*

REFERENCE VARIABLES,

- A **reference variable** is an alias, i.e. another name for an already existing **variable**.
- Once a **reference** is initialized with a **variable**, either the **variable** name or the **reference** name may be used.
- Reference is a second label attached to that memory location.
- access the contents of the variable through either the original variable name or the reference.



- For example,
- `int i = 17;` dec. reference variables for `i` as follows.
- `int & r = i;` `r` **refer** to the **variable i**.

OPERATOR IN C++

- C++ has rich set of operators
- All C operators are valid in C++ with addition it introduces some new operators
- `::` - Scope resolution operator
- `::*` - Pointer to member Declarator
- `->*`
- `.*` }

Pointer to member operator/ Dereferencing Operator
- `New` - Memory Allocation Operator
- `Delete` –Memory release operator
- `endl` – Line feed operator
- `setw` - Field width operator

SCOPE RESOLUTION OPERATOR (::)

○ Unary Scope Resolution Operator

- Used to access a hidden global variable

<pre>int y=2 void main() { Int y=44; cout <<y; { Int y=9;</pre>	<pre> cout <<y; cout<<::y; }</pre>
---	---

○ Binary Scope Resolution Operator

- Used to associate a member function with its class (will be discussed shortly)
- Used to access a hidden class member variable (will be discussed shortly)

MEMORY MANAGEMENT OPERATORS

- There are serious drawbacks of using arrays.
- It is must for the programmer to allocate the memory of an array while declaring array
- Exact memory that is needed cannot be determined until runtime.
- We need to declare the array with maximum possible memory required but this wastes memory.
- To avoid wastage of memory.
- To dynamically allocate the memory at runtime by new and delete operators.
- It manipulate free store, which is a system-provided memory collection for variables
- Also called as free store operators.

- Whenever it is required, an object can be created & destroyed.

The new operator

- The new operator returns a pointer to the beginning of the new block of memory allocated.
- It has two forms.
- a) To allocate the memory to objects of any data type
 - Syntax: `Pointer=new data_type;`
 - Example: `int *ip = new int`
- b) To allocate a block of elements for user - defined data types such as arrays, structures, classes:
 - Syntax: `Pointer = new data_type[number_of_elements];`
 - Example: `int *ip = new int[50];`

- Creating multi-dimensional arrays with new operator, Need to supply all the array sizes.
 - Syntax: `Pointer=new data_type[dim1][dim2][dim3];`
 - Example: `array_pointer = new int[3][4][3];`
 - In the above syntax, the first dimension may be a variable whose value can be supplied at run time, it is must that the remaining ones must be constants.

9/10/2022

The delete operator

- Used to release the memory that is allocated by new operator.
- It has two forms, they are:
- a) To free up the memory that is occupied in the free store:
 - Syntax: `delete pointer_variable;`

- `pointer_variable` is the pointer that points to the data object that is created with the help of `new` operator.
- Example: `delete p;`
- b) To free a dynamically allocated array:
 - Syntax: `delete [size] pointer_variable;`
 - Example: `delete [] p;`

```
int main ()  
{  
    float* fpvalue =  
        NULL;  
    fpvalue = new float;  
    *fpvalue = 10000.46;
```

```
    cout << " fpvalue : "  
    << *fpvalue << endl;  
  
    delete fpvalue;  
  
    return 0;  
}
```

MANIPULATOR OPERATOR

- Manipulators are operators used in C++ for **formatting output**.
- The data is manipulated by the programmer's choice of display.
- **Ex. endl, setw, setfill and setprecision** manipulator.
- To access manipulator need to include `<iomanip>` header file

endl -

- endl is the line feed operator in C++.
- It acts as a stream manipulator whose purpose is to feed the whole line and then point the cursor to the beginning of the next line.

- We can use `\n` (`\n` is an escape sequence) instead of `endl` for the same purpose.

setw(x)

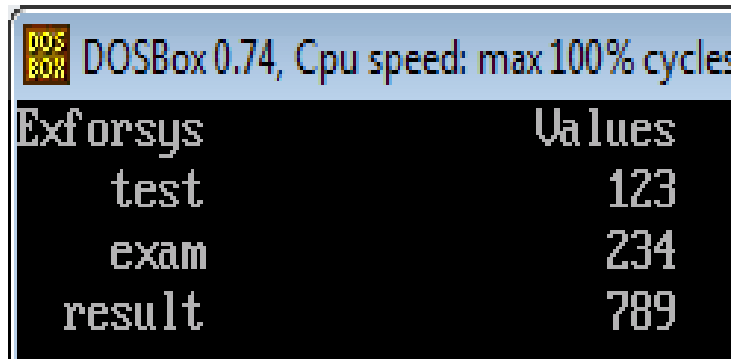
- This manipulator sets the minimum field width on output.
- Here `setw` causes the number or string that follows it to be printed within a field of `x` characters wide and `x` is the argument set in `setw` manipulator.
- The header file that must be included while using `setw` manipulator is .
- Ex. –

setfill Manipulator:

- This is used after `setw` manipulator.

```
#include <iostream>
#include <iomanip>
void main( )
{
int x1=123,x2= 234,
x3=789;
cout << setw(8) <<
"Exforsys" << setw(20)
<< "Values" << endl
<< setw(8) << "test" <<
setw(20)<< x1 << endl
```

```
<< setw(8) << "exam" <<
setw(20)<< x2 << endl
<< setw(8) << "result"
<< setw(20)<< x3 <<
endl; Output:
```



Exforsys	Values
test	123
exam	234
result	789

- If a value does not entirely fill a field, then the character specified in the setfill argument of the manipulator is used for filling the fields.
- Example:

```
#include <iostream>
#include <iomanip>
void main()
{
    cout << setw(15) <<
    setfill('*') << 99 << 97
    << endl;
}
```

Output:



9/10/2022

setprecision Manipulator:

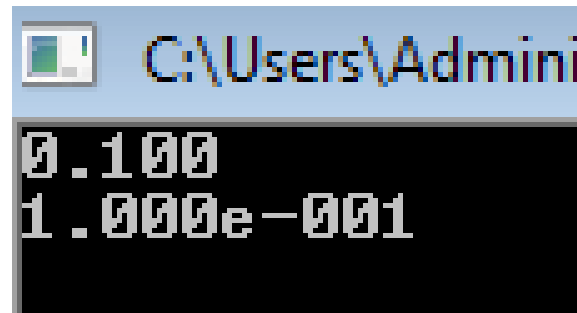
- The setprecision Manipulator is used with floating point numbers.
- It is used to set the number of digits printed to the right of the decimal point.
- This may be used in two forms:
 - 1. fixed
 - 2. scientific

- These two forms are used when the keywords `fixed` or `scientific` are appropriately used before the `setprecision` manipulator.
- The keyword `fixed` before the `setprecision` manipulator prints the floating point number in fixed notation.
- The keyword `scientific`, before the `setprecision` manipulator, prints the floating point number in scientific notation.

```
#include <iostream>
#include <iomanip>
using namespace std;
int main()
{
    float x = 0.1;
    cout << fixed <<
    setprecision(3) << x <<
    endl;
```

```
cout << scientific << x <<
endl;
return 0; }
```

Output:



IF .. ELSE, SWITCH .. CASE, STATEMENT, WHILE, FOR, BREAK, CONTINUE, GOTO STATEMENTS

9/10/2022

- To check conditions we use conditional statements
- If .. Else it has types
- If statement, if else statement, Nested if else, if

```
if(expression)
{
    //statement(s);
}else{
    //statement(s);
}
```

```
if(expression)
{
    if(expression)
    { }
    else{ }
}
```

```
if(expression)
{}else if(expression)
{}else{ }
```

- Switch case is used when multiple options are there

- Cases get execute when condition in switch get matched
- This selection statement successively tests the value of an expression against a list of integer or character constants.
- When a match is found, the statements associated with that constant are executed.
- The syntax of switch statement is as follows:

```
switch(expression)
{
    case constant1 : statement ;
                    break;
    case constant2 : statement 2;
                    break;
    case constant3 : statement 3;
                    break;
```

```
.
.
case constant n-1 : statement n-
1;
                    break;
default :    statement n;
}
```

- A switch statement can only work for equality comparisons.
- No two case labels in the same switch can have identical values.
- But, in case of nested switch statements the case constants of the inner and outer switch can contains common values.
- If character constants are used in the switch statement, they are automatically converted to their integers (i.e., their equivalent ASCII codes).
- The switch statement is more efficient than if in a situation that supports the nature of switch operation.

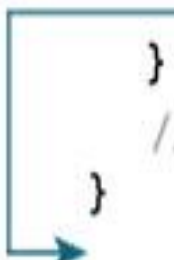
Break, Continue

- To alter the normal flow of a program.
- Sometimes, it is desirable to skip the execution of a loop for a certain test condition or terminate it immediately without checking the condition.
- For example: You want to loop through data of all aged people except people aged 65. Or, you want to find the first person aged 20.
- In scenarios like these, `continue;` or a `break;` statement is used.
- The `break;` statement terminates a loop (for, while and do..while loop) and a switch statement immediately when it appears.
- Syntax of break- `break;`

```

while (testExpression) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}

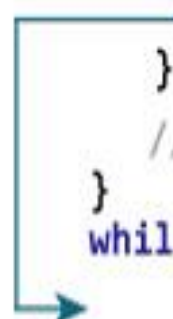
```



```

do {
    // codes
    if (condition to break) {
        break;
    }
    // codes
} while (testExpression);

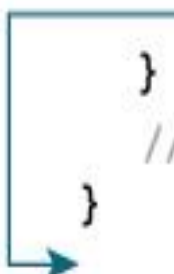
```



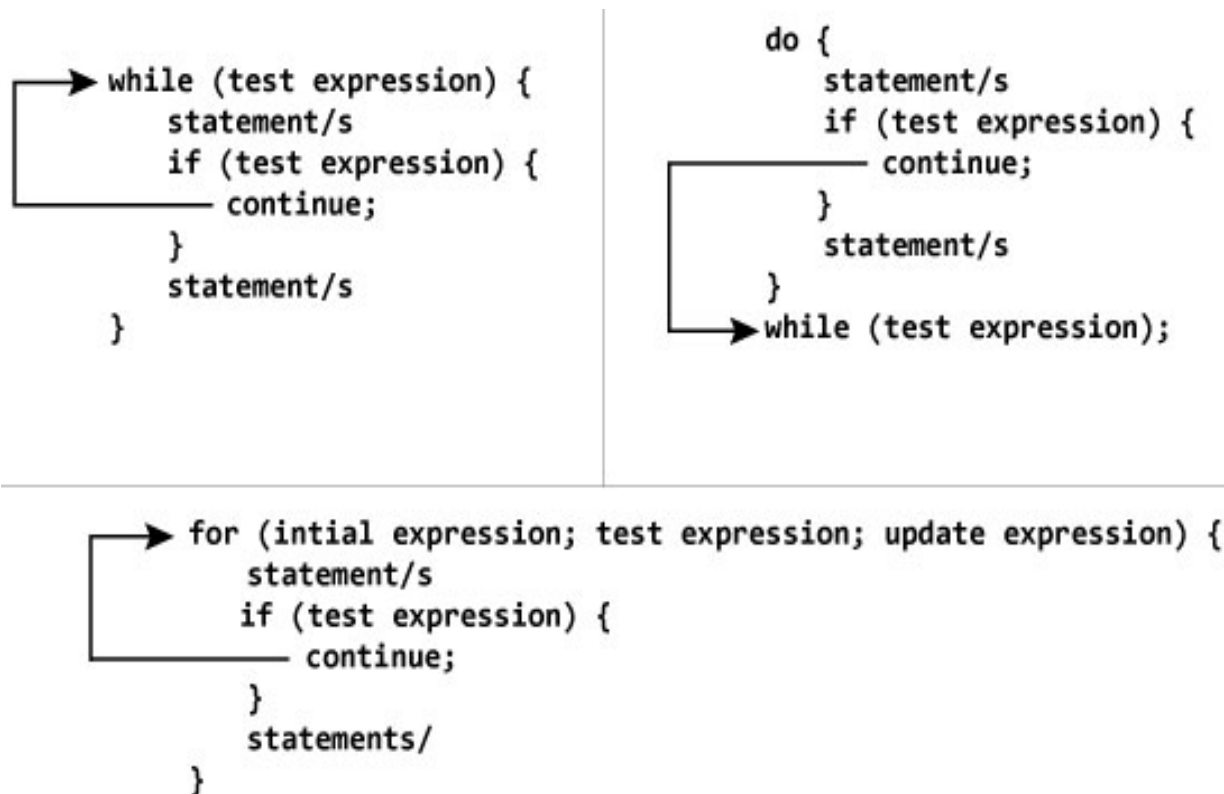
```

for (init; testExpression; update) {
    // codes
    if (condition to break) {
        break;
    }
    // codes
}

```



- It is sometimes necessary to skip a certain test condition within a loop. In such case, `continue;` statement is used in C++ programming.
- Syntax of `continue;` `continue;`
- In practice, `continue;` statement is almost always used inside a conditional statement.



While loop

- The while loop evaluates the test expression.
- If the test expression is true, codes inside the body of while loop is evaluated.
- Then, the test expression is evaluated again. This process goes on until the test expression is false.
- When the test expression is false, while loop is terminated.
- The syntax of a while loop is:

```
while (testExpression)
{
    // codes
}
```

Do...while()

- The do...while loop is a variant of the while loop with one important difference

- The body of do...while loop is executed once before the test expression is checked.

- Syntax

```
do {  
    // codes;  
}  
while (testExpression);
```

- The codes inside the body of loop is executed at least once.
- Then, only the test expression is checked.
- If the test expression is true, the body of loop is executed.
- This process continues until the test expression becomes false.
- When the test expression is false, do...while loop is terminated.

○ For loop

- Loops are used in programming to repeat a specific block until some end condition is met.

○ Syntax

```
for(initializationStatement; testExpression; updateStatement) {  
    // codes  
}
```

where, only testExpression is mandatory.

- The initialization statement is executed only once at the beginning.
- Then, the test expression is evaluated.
- If the test expression is false, for loop is terminated. But if the test expression is true, codes inside body of for loop is executed and update expression is updated.
- Again, the test expression is evaluated and this process repeats until the test expression is false.