In this session we will cover the following sub-topics:

1. Identifiers
2. Variables
3. Keywords
4. Statements
5. Comments
6. Whitespaces
7. Syntax
8. Semantic

C IDENTIFIERS

1. Is a unique name that simply references to memory locations, which can hold values (data).
2. Identifiers give unique names to various objects in a program.
3. Are formed by combining letters (both upper and lowercase), digits (0–9) and underscore (_).
4. Rules for identifier naming are:
   a) The first character of an identifier must be a letter (non-digit) including underscore (_).
   b) The blank or white space character is not permitted in an identifier. Space, tab, linefeed, carriage-return, formfeed, vertical-tab, and newline characters are "white-space characters“ - they serve the same purpose as the spaces between words and lines on a printed page.
   c) Can be any length but implementation dependent.
   d) Reserved words/keywords cannot be used.

C IDENTIFIERS

Examples: variable names

Correct	Wrong
`secondName`	`2ndName /* starts with a digit */`
`_addNumber`	`%calculateSum /* contains invalid character */`
`charAndNum`	`char /* reserved word */`
`annual_rate`	`annual rate /* contains a space */`
`stage4mark`	`my\nName /* contains new line character, \n */`

C VARIABLES

- are named blocks of memory & is any valid identifier.
- Have two properties in syntax: name — a unique identifier & type — what kind of value is stored.
- It is identifier, that value may change during the program execution.
- Every variable stored in the computer's memory has a name, a value and a type.

C VARIABLES

- More examples

Correct	Wrong	Comment
`int x, y, z;`	`int 3a, 1, -p;`	
`short number_one;`	`short number+one;`	
`long TypeofCar;`	`long #number`	
`unsigned int positive_number;`	…	
`char Title;`		
`float commission, yield = 4.52;`		
`int my_data = 4;`		
`char the_initial = 'M';`		A char
`Char studentName[20] = "Anita";`		A string

C KEYWORDS/RESERVED WORDS

- Reserved words in C & are not available for re-definition.
- have special meaning in C.

auto break case char const continue default do double else enum	extern float for goto if inline (C99 beyond) int long register restrict (C99 beyond) return	short signed sizeof static struct switch typedef union unsigned void volatile while	_Alignas (C11) _Alignof (C11) _Atomic (C11) _Bool (C99 beyond) _Complex (C99 beyond) _Generic (C11) _Imaginary (C99 beyond) _Noreturn (C11) _Static_assert (C11) _Thread_local (C11)

OTHERS

- Statements are terminated with a ';'
- e.g:

```
char acharacter;
int i, j = 18, k = -20;
printf("Initially, given j = 18 and k = -20\n");

for(; count != 0; count = count - 1)
```

OTHERS

- Group of statements (compound statement) are enclosed by curly braces: { and }.
- Mark the start and the end of code block.
- Also used in initializing a list of aggregate data values such as in array and enum type.

```
int   id[7] = {1, 2, 3, 4, 5, 6, 7};
float  x[5] = {5.6, 5.7, 5.8, 5.9, 6.1};
char   vowel[6] = {'a', 'e', 'i', 'o', 'u', '\0'};

enum days {Mon, Tue, Wed, Thu, Fri,
Sat, Sun};
```

```
#include <stdio.h>

int main()
{
    int i, j = 18, k = -20;
    printf("Initially, given j = 18 and k = -20\n");
    printf("Do some operations..."
           "i = j / 12, j = k / 18 and k = k / 4\n");
    i = j / 12;
    j = k / 8;
    k = k / 4;
    printf("At the end of the operations...\n");
    printf("i = %d, j = %d and k = %d\n", i, j, k);
    return 0;
}
```

COMMENTS

- Single line of comment: `//` `comment here`
- More than single line of comment or expanded: `/*` `comment(s) here` `*/`

```
// for printf()
#include <stdio.h>
#include <string.h> // for strcpy_s() and their family

/* main() function, where program
     execution starts */
int main()
{
      /* declares variable and initializes it*/
      int i = 8;
…
```

COMMAS

- Commas separate function arguments, list of variables, aggregate values. e.g.

```
#include <stdio.h>

int main(int argc, int argv)
{
    int i = 77, j, k;
    j = i + 1;  k = j + 1;  i = k + j;
    printf("Finally, i = %d\n", i);
    printf("... and j = %d\n", j);
    printf("... and k = %d\n", k);
    return 0;
}
```

```
int   id[7] = {1, 2, 3, 4, 5, 6, 7};
float  x[5] = {5.6, 5.7, 5.8, 5.9, 6.1};
char   vowel[6] = {'a', 'e', 'i', 'o', 'u', '\0'};


enum days {Mon, Tue, Wed, Thu, Fri,
Sat, Sun};
```

WHITESPACES

- Whitespace is ignored by compiler.
- Counted if used as separators or as components of character constants or string literals.
- Space, tab, linefeed, carriage-return, formfeed, vertical-tab, and newline characters (\n).

```
#include <stdio.h>

void main(void)
{
      int MyAge = 12;
      printf("My name is Mr. C. Cplusplus\n");
… }
```

SYNTAX & SEMANTIC

- Programming language enforces a set of rules, symbols and special words used to construct a program.
- A set of rules that precisely state the *validity of the instructions* used to construct C program is called syntax or 'grammar' else syntax error will be generated.
- The correctness of the instructions used to write C program is called semantics or *correct meaning*.
- These set of rules for instructions validity and correctness are monitored by the compilers.
- Semantically one but can have many syntaxes.

SYNTAX & SEMANTIC

- e.g.

To add an integer to a variable q and store the result in q (semantic), syntaxically (correct), we can write:

```
q = q + 3; or q += 3;
```

- *Pseudocode* - an informal high-level description of the operating principle of a computer program or other algorithm.
- Uses the structural conventions of a programming language, but is intended for human reading rather than machine reading.

PSEUDOCODE & ALGORITHM

- An informal high-level description of a computer program or algorithm operating principle.
- An *algorithm* is merely the sequence of steps taken to solve a problem which are normally a sequence, selection, iteration and a *case*-type statement.
- Algorithm is a procedure for solving a problem - actions to be executed and the order in which those actions are to be executed.
- e.g. to sort ascendingly, the given unsorted integers, we can achieve this by using several different algorithms.
- Every algorithm may have different number line of code, different repetition loops, different execution speeds etc.

PSEUDOCODE & ALGORITHM

- But all the program have similar purpose: to sort the given unsorted integers in ascending order.
- Pseudocode uses programming language's structural conventions , intended for human rather than machine reading.
- helps programmers develop algorithms.

PSEUDOCODE & ALGORITHM

- e.g.

```
Set sum to zero
Set grade counter to one
While grade counter is less than or equal to ten
        Input the next grade
        Add the grade into the sum
Set the class average to the sum divided by ten
Print the class average.
```

```
IF HoursWorked > NormalMax THEN
        Display overtime message
ELSE
        Display regular time message
ENDIF
```

```
SET total to zero
REPEAT
        READ Temperature
        IF Temperature > Freezing THEN
                INCREMENT total
        END IF
UNTIL Temperature < zero
Print total
```