Notes | Work to do

C - Tutorial:

→ **Datatypes:** int: integer, a whole number.
float: floating point, a number with fractional par-
double: double-precision floating point value.
char: single character.

→ **Variables:** A variable is a name for an area in memory. The name of a variable must begin with either a letter or an underscore & can be composed of letters.

```
int my_var;
my_var = 42;
```

→ **Constants:** A constant stores a value that cannot be changed from its initial assignment.

```
#include <stdio.h>
int main() {
    const double PI = 3.14;
    printf ("%f", PI);
    return 0;
}
```

26 SUNDAY

→ **Input:** C supports a number of ways for taking user i/p. getchar() Returns the value of the next single character i/p.

```
#include<stdio.h>
int main(){
  char a = getchar();
```

Appointment

- Output: printf() function to generate o/p.

putchar() outputs a single character.

```
#include <stdio.h>
int main() {
char a = getchar();
 printf ("You entered:");
 putchar (a);
 return 0;
}
```

- Formatted Input:

The scanf() function is used to assign i/p to variables.

```
int x;
float num;
char text[20];
scanf ("%d %f %s", &x, &num, text);
```

- Formatted output: The printf function was introduced in your very first Hello world program.

```
printf ("The tree has %d apples.\n", 22);
/* The tree has 22 apples. */

printf ("\"Hello world!\"\n");
/* Hello world!" */
```

\n - new line.
\t - horizontal tab.

Importance | Notes | Work to do

\\ backslash
\b backspace
\' single quote.
\" double quote.

• Single-line Comments:
// - double slash comment is to comment single lines.

```
# include <stdio.h>
int main(){
int x = 42; // int for whole number
//%d is replaced by x
printf("%d", x);
return 0;
}
```

→ Operator Precedence.
The + and - are equal in precedence, as are *, /, and %. The *, /, and % are performed first in order from left to right and then + and -, also in order from left to right.

→ Type conversion:
→ Assignment Operators.
→ Increment & Decrement.
→ Conditionals.
The if statement: This is called a conditional control structure because it executes statements when an expression is true.

```
if (expression)
statements.
```

S S M T W T F S S M T W T F S S M T W T F S S M T W T F S S M Fee

Assignments | Notes | Work to do

→ If else statement.

The if statement can include an optio
- al else clause that executes statements when an expression is false.

```
if (score >= 90)
  printf (" Top 10 %. %. .\n");
  else
    print (" less then 90. \n");
  actum 0;
}
```

- Nested if :

```
if (profit > 1000)
  if (clients > 15)
    bonus = 100;
else
  bonus = 25;
```

- If else - if statement.

```
int score = 89;
if (score >= 90)
  printf ("%s", "Top 10%\n");
  else if (score >= 80)
  printf ("%", "Top 20%.\n");
else if (score > 75)
  printf ("%s", "You passed.\n");
else
  printf ("%s", "You did not pass.\n");
```

- The Switch statement:

```
switch (expression) {
case val1:
  statements
break;
 case val2:
   statements
break;
default:
   statements
}
```

- Logical Operator:

1) && operator:

```
if (n>0 && n<=100)
printf ("Range (1-100).\n");
```

2] || Operator:

```
if (n=='x' || n=='X')
printf ("Roman numeral value 10.\n");
```

3) The ! Operator.

```
if (!(n=='x' || n=='X'))
  printf ("Roman numeral is not 10.\n");
```

- while loop:

```
while (expression) {
  statements
}
```

→ The do-while loop:

```
do {
   statements
} while (expression);
```

- **Break and continue:**

```
int num = 5;
while (num > 0) {
 if (num == 3)
 break;
 printf("%d\n", num);
 num--;
}
```

```
int num = 5;
while (num > 0) {
 num--;
 if (num == 3)
   continue;
 printf("%d\n", num);
}
```

- The for loop:

```
for (initvalue; condition; increment) {
   statements;
}
```

- Functions in C

Accomplish a program solution as a series of subtasks.

- Function Parameter:

The function's parameters are used to receive values required by the function. Values are passed to these parameters as arguments through the function call.