



**Secret of getting ahead = getting started!**

# UESTC HN 1005 - Introductory Programming

Lecture 3 — Pedalling along 🚴🚴

Dr Hasan T Abbas

# Questions 🙋 ?


- Ask us anything (programming-related 😎)



# Something on ChatGPT

- Compilers are very precise in their requirements
- We the humans can make sense of sentences even with a misspelled word
- a C compiler will fail to provide a translation of a syntactically incorrect program,
- No matter how small the error is
- **We got to learn to be precise in writing code**

# Today's Lecture

- Operators, expressions, and assignment 
- Make larger programs
- Interactivity: Input and Output

# The C character set 🪐

Type	Character
lowercase letters	a-z
uppercase letters	A-Z
digits	0-9
Other Characters	+ - * / = ( ) { } [ ] < > ' " ! # % _ ^ ~ \ . , : ; ?
white space characters	blank, newline, tab etc.

# More Operators +

- C has arithmetic operators
- Increment operators ( `++` , `--` )
- Assignment operator ( `=` )
- Relational operators ( `<` , `>` , `==` , `<=` , `>=` )
- Logical operators ( `&&` , `||` , `!` )

# Displaying Output

- We have been using `printf()` function to display messages on screen
- `printf()` is defined in `stdio.h` library (header file)

```
printf("<formattext>", var1, var2, ... );
```

- `<formattext>` is a **string** that indicates ...
- how many variables to expect at the end of the statement,
- the expected printing type of each variable,
- how many columns to use for printing,
- any associated fixed text



# printf() function

```
#include <stdio.h>
int main()
{
    int age = 50;
    char firstname = 'H';
    char lastname = 'A';
    float weight = 100.3;
    printf("My age is %d, my initials are %c %c, and my weight is %f.\n", age, firstname, lastname, weight);
    return 0;
}
```

## More on output format

- All this is done by special `%` characters in the `<formattext>` string
- `%<w>.<p><t>`
- `<w>` is the total width of the field (optional)
- If `w <= actual width`, output actual – no truncation
- If `w > actual width`, add zero at the left
- `<p>` is the # digits after the point (optional)
- `<t>` is the type conversion (**required**)

# printf() Rules

printf() type	Description
i or d	signed integer
u	unsigned integer
f, lf	real decimal normal format, double
e	real decimal engineering format/scientific notation
o, x	octal, hexadecimal
c, s	character, strings

# Example

```
#include <stdio.h>
int main () {
    int x = 20;
    float y = -16.7889;
    printf("Value x=%d and value y=%9.3f\n", x, y);
    printf("Value x=%i and value y=%7.1f\n", x, y);
    printf("Value x=%3d and value y=%5.1f\n", x, y);
    printf("Value x=%3d and value y=%3.1f\n", x, y);
    return 0;
}
```

# Making programs more interactive

- We have been using variables to store data in memory
- We can ask the user to insert a value
- That's the job of the `scanf()` function

```
scanf("%f",&radius);
```

- Note the `&` character

## Quiz Time 100

- Log on to menti.com: <https://www.menti.com>
- Use code 49 46 05



# A Simple Calculator

- Performs basic arithmetic on two numbers
- Addition ( `+` ), Subtraction ( `-` ), Multiplication ( `*` ) and Division ( `/` )
- Results are displayed on the screen
- Start with integer numbers
- Move to `float`
- Use `double`
- Next up ... use `scanf()` to insert the values of the variables

## Next up

- Flow Control and Conditions



# Get in touch

[Hasan.abbas@glasgow.ac.uk](mailto:Hasan.abbas@glasgow.ac.uk)