Lectures/Week_10/week10_tutorial.c

```c
#include <stdio.h>  // Standard Input Output library
#include <stdlib.h> // Standard library for general utilities

// Main function: Entry point of the program
// Every C program must have a main function. It's where the program starts.
int main()
{
  /* 03/11's Content: C Programming Basics, Arrays
  ==============================================================
  */
  printf("03/11's Content: C Programming Basics, Arrays\n");
  printf("==============================================================\n");

  // printf is a function in <stdio.h> that prints output to the console.
  // The "\n" is a newline character that moves the cursor to the next line.
  // The text inside the double quotes is the message to be printed.
  printf("Hello, world!\n");

  // Semicolon (;) is used to terminate a statement in C.
  // It tells the compiler that the statement is complete.
  // Unlike MATLAB where semicolon (;) is optional.
  // In C, every statement must end with a semicolon.

  // Print an additional message
  printf("Welcome to learning C programming!\n");

  // Declare and initialize variables
  int a = 5;
  int b = 10;
  int sum = a + b;

  // int is a data type for integers
  // Variables a, b, and sum are declared as integers and initialized.
  // Unlike MATLAB, C requires explicit declaration of variable types.
  // In C, you must declare the type of each variable before using it.

  // int a = 5; // first declaration must include type
  // a = 6; // further assignment does not require type
  // so int a=6 here again would be an error

  // Print the sum of two numbers
  printf("The sum of %d and %d is %d\n", a, b, sum);

  int M[10] = {1, 2, 3, 40, 5, 60, 7, 80, 9, 101}; // Array of integers
  // you need to specify the size of the array when you declare it, using []
  // the initial value of an array is included inside a curly braces {}
  // the elements of the array are separated by commas
  // Note the type of M is int, so the type of each element is also int

  printf("the fourth element of M is %d\n", M[3]); // index of C starts from 0
  // KEY DIFFERENCE FROM MATLAB
  // In C, the index of an array starts from 0, not 1.

  int N[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}}; // 2D array of integers

  printf("the element at row 1 and column 3 of N is %d\n", N[0][2]);
  printf("\n\n");

  // Return 0 to indicate successful execution
  // The main function (declared as int function) must return an int value
  // back to the operating system.
  // Returning 0 typically means that the program executed successfully.

  /* 03/13's Content: Data Types, Printf, Scanf, Precision
  ==============================================================
  */
  printf("03/13's Content: Data Types, Printf, Scanf, Precision\n");
  printf("==============================================================\n");

  // Data Types
  // C has several data types, including int, float, double, char, etc.
  int x = 10, d = 100;                // integer
  float y = 3.14, e = 2.72;           // single-precision floating-point number
  double z = 3.14159, f = 1.23456; // double-precision floating-point number
  char ch = 'A', option = '1';        // character

  // Printf
  printf("x=%d\n", x);            // %d is a placeholder for an integer
  printf("y=%f\n", y);            // %f is a placeholder for a float
  printf("z=%lf\n", z);           // %lf is a placeholder for a double
  printf("ch=%c\n", ch);          // %c is a placeholder for a character
  printf("option=%c\n", option); // single quotes of anything is a character

  // Never mess up the placeholders and the variables types
  // Although it may not cause an error, it may lead to undefined results
  printf("z in integer is %d\n", z);   // %d is for a float, but z is an double
  printf("ch in integer is %d\n", ch); // %d is for an int, but ch is a
character

  // Scanf
  // scanf is a function in <stdio.h> that reads input from the console
  printf("Enter an integer for x: ");
  scanf("%d", &x); // Read an integer from the user and store it in x
                   // It would directly modify x
```

```c
    // scanf("%d", &m);        // illegal, m is not declared
    // scanf("%d", x);         // illegal, & is a must when using scanf
    // scanf("%d", &x, &y);    // illegal, num of placeholder mismatched
    // scanf("%f", &x);        // illegal, type mismatched
    printf("new x=%d\n", x); // Print the value of x, should be new now!

    printf("Enter a double for z: ");
    scanf("%lf", &z);          // Read a double from the user and store it in z
    printf("new z=%lf\n", z); // Print the value of z with 2 decimal places

    printf("Enter a char for ch: ");
    scanf(" %c", &ch); // Read a char from the user and store it in ch
    // Note the space before %c, it is used to consume the newline character
    // left in the buffer
    printf("new ch=%c\n", ch); // Print the value of ch

    // Precision
    // The precision of a number is the number of digits after the decimal point.
    // The default precision is 6 digits.

    printf("y=%.2f\n", y);  // %.2f is a flag for a float with 2 decimal places
    printf("z=%.3lf\n", z); // %.3lf is a flag for a double with 3 decimal places

    printf("f=%7.3lf\n", f);
    // %7.3lf means the total width of the number is 7
    // including the decimal point and the digits before it
    // If the number is less than 7 digits, it will be padded with spaces on left

    // left- and right+ alignment in printf
    printf("f=%-7.3lf\n", f);
    printf("f=%+7.3lf\n", f);
    // fill with 0s alignment
    printf("f=%07.3lf\n", f);

    // More on array and 2d array
    int NEW[3][4] = {{1, 2, 3, 5}, {4, 6, 7, 9}, {-11, 8, -12, 14}};
    double NEWD[2][2] = {1.1, 2.2, 3.3, 4.4}; // you can put them just in one {}

    // first [] is row index, second [] is column index
    printf("the element at row 1 and column 3 of NEW is %d\n", NEW[0][2]);

    // FIXME: printf("row 2 and column 2 of NEWD is %d\n", NEWD[2][2]);
    printf("the element at row 2 and column 2 of NEWD is %lf\n", NEWD[1][1]);

    // You can also scanf to update a value in the array
    printf("Enter a new value for NEW[0][2]: ");
    scanf("%d", &NEW[0][2]);
    printf("the element at row 1 and column 3 of NEW is %d\n", NEW[0][2]);
    printf("\n\n");

    /* 03/13's Content: Escape Sequences
    ==============================================================
    */
    printf("03/13's Content: Escape Sequences\n");
    printf("==============================================================\n");

    // Using \n to enter (go to new line)
    printf("Hello, world!\n");
    printf("This is a new line.\n");
    printf("\n");

    // How to literally print \n
    printf("Here is a backslash plus n: \n");
    printf("\\n");

    // Using \t to insert a tab
    printf("\n\n");
    printf("Column1\tColumn2\tColumn3\n");
    printf("Data1\tData2\tData3\n");
    printf("\n");

    // How to literally print \t
    printf("Here is a backslash plus t: \n");
    printf("\\t");

    // Using \\ to display a backslash
    printf("\n\n");
    printf("This is a backslash: \\");
    printf("\n");

    // Using \' to display a single quote
    printf("This is a single quote:  \' ");
    printf("\n");

    // Using \" to display a double quote
    printf("This is a double quote:  \" ");
    printf("\n");

    // Using %% to display a percent sign
    printf("This is a percent sign:  %% ");
    printf("\n");

    // Using \n\n to show a blank line
    printf("\nHere is a blank line in between:\n");
    printf("abcd\n\nabcd\n\n\n");

```

```
/* 03/13's Content: HW7 Hints
==============================================================
*/
printf("03/13's Content: HW7 Hints\n");
printf("==============================================================\n");

// Part 3:
// There is no breakline between "allow the printing".
// It is the display simply due to pdf rendering
// Similarly, this is applied to consequent paragraphs
// But there should be between "do you" and "understand" and "the code"
printf("do you\nunderstand\nthe code");

/*
Part 4:
No need to show anything before scans
Expected format is:

<Text>
<more text>
A=?
B=?
C=?
<more text>
A=?
B=?
C=?
<more text>
A=?
B=?
C=?

Part 5 & 6:
Expected format is:

G=
?
?
?
?

H=
?
?
?
?
*/
```

```c

  // Can I try the code locally before submission?
  // Yes, the starter code is using the following structure
  // which we will discuss in the future classes

  // After compiling and building in Geany, you will prompt for input
  // Enter 1: you will test your Part 1 code
  // Enter 2: you will test your Part 2 code
  printf("\n\n");

  char choice;
  printf("Demo test, Enter 1 or 2: ");
  scanf(" %c", &choice); // Read a char from the user and store it in ch
  switch (choice)
  {
  case '1':
  {
    printf("You are testing part 1.\n");
    break;
  }
  case '2':
  {
    printf("You are testing part 2.\n");
    break;
  }
  }
  return 0;
}

/*
Workflow: Compiling and Running the Program

1. Write the Program:
   - Open your text editor or IDE (e.g., Geany, VS Code).
   - Write the C code as shown below and save it as `week10_tutorial.c`.
   - If you are using Geany, you can create a c file using template
   - If you are using the VS Code cloud IDE (i.e., here), compile and run are just to click the run button

2. Compile the Program:
   - Option A : if in Geany, click compile and then click build
   - Option B : if in computer terminal
   - Open the terminal, navigate to the directory where your c file stores.
   - Use the `gcc` compiler to compile the program:
     gcc -o week10_tutorial week10_tutorial.c
   - This command tells `gcc` to compile `week10_tutorial.c` and output an executable named `week10_tutorial`.

```

```
3. Run the Compiled Program:
   - Option A : if in Geany, simply click execute
   - Option B : if in computer terminal
   - In the terminal, run the compiled program by typing:
     ./week10_tutorial
   - This command executes the `week10_tutorial` program, and you should see the output in the terminal.
*/

/*
Output of the program:

03/11's Content: C Programming Basics, Arrays
==============================================================
Hello, world!
Welcome to learning C programming!
The sum of 5 and 10 is 15
the fourth element of M is 40
the element at row 1 and column 3 of N is 3


03/13's Content: Data Types, Printf, Scanf, Precision
==============================================================
x=10
y=3.140000
z=3.141590
ch=A
option=1
z in integer is 1431671456
ch in integer is 65
Enter an integer for x: 87
new x=87
Enter a double for z: 7.89
new z=7.890000
Enter a char for ch: p
new ch=p
y=3.14
z=7.890
f=  1.235
f=1.235
f= +1.235
f=001.235
the element at row 1 and column 3 of NEW is 3
the element at row 2 and column 2 of NEWD is 4.400000
Enter a new value for NEW[0][2]: 567
the element at row 1 and column 3 of NEW is 567


```

```
03/13's Content: Escape Sequences
==============================================================
Hello, world!
This is a new line.

Here is a backslash plus n:
\n

Column1 Column2 Column3
Data1   Data2   Data3

Here is a backslash plus t:
\t

This is a backslash: \
This is a single quote:  '
This is a double quote:  "
This is a percent sign:  %

Here is a blank line in between:
abcd

abcd


03/13's Content: HW7 Hints
==============================================================
do you
understand
the code

Demo test, Enter 1 or 2: 2
You are testing part 2.

*/
```