Lectures/Week_11/week11_tutorial.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h> // Include math library for math functions

int main()
{
  /* 03/18's Content: Integer, Floating Numbers, Math Functions
  ==============================================================
  */
  printf("03/18's Content: Integer, Floating Numbers, Math Functions\n");
  printf("==============================================================\n");

  // Declare integers
  int x = 5, y = 4, z;
  // Declare doubles or floats, they are just different in precision
  double a = 5.00, b = 4.00, c;

  // Basic math operations
  z = x + y; // Addition
  c = a * b; // Multiplication
  printf("Addition (int): z = %d\n", z);
  printf("Multiplication (double): c = %lf\n", c);

  // Division examples
  z = x / y; // Integer division yields an integer (the quotient int part)
  c = a / b; // Double division yields the full double result
  printf("Integer Division: z = %d\n", z);
  printf("Double Division: c = %lf\n", c);

  z = 5 / 2 * 3; // Take a guess what is z?
  c = 5 / 2 * 3; // Take a guess what is c?
  printf("z=%d\nc=%lf\n", z, c);
  // The result is z=6, c=6.000000
  // This is because 5/2 is evaluated as an int division, which results in 2.
  // Then, 2*3 equals 6.
  // When 6 is assigned to z, it stays as int, resulting in 6
  // When 6 is assigned to c, it is converted to a double, resulting in 6.000000

  z = 5.0 / 2 * 3; // Take a guess what is z?
  c = 5.0 / 2 * 3; // Take a guess what is c?
  printf("Another version:\nz=%d\nc=%lf\n", z, c);
  // The another-version result is z=7, c=7.500000
  // since 5.0/2 is evaluated as a double division, which results in 2.5
  // Then, 2.5*3 equals 7.5.
  // When 7 is assigned to z, it is converted to an int, resulting in 7
  // When 7.5 is assigned to c, it stays as double, resulting in 7.500000

// Remainder operation
z = x % y; // Modulus operator
printf("Remainder: z = %d\n", z);
printf("\n");
// modulus is only valid for ints: if x or y is not int, it gives an error

// Math operations
double num1 = 2.0, num2 = 3.0, result;
double angle = 3.14159265358979323846 / 4; // (pi/4 in radians)
// Above pi is not a good practice, use a constant M_PI from math.h

// Exponentiation: num1 raised to the power of num2
result = pow(num1, num2);
printf("Exponentiation: %.2lf^%.2lf = %.2lf\n", num1, num2, result);
// ^ is not a power operator in C, it is for another operation
printf("Wrong result with ^ for 3^3 is: %d\n", 3 ^ 3); // 3^3 = 0, not 27

// Natural logarithm (base e)
result = log(num1);
printf("Natural Logarithm: log(%.2lf) = %.2lf\n", num1, result);

// Logarithm base 10
result = log10(num1);
printf("Logarithm Base 10: log10(%.2lf) = %.2lf\n", num1, result);

// Logarithm base 2
result = log2(num1);
printf("Logarithm Base 2: log2(%.2lf) = %.2lf\n", num1, result);

// Raising Euler constant e to the power of num2
result = exp(num2);
printf("Exponential: e^%.2lf = %.2lf\n", num2, result);

// Rounding up (ceil) and down (floor)
double num = 2.7;
result = ceil(num);
printf("Ceiling: ceil(%.2lf) = %.2lf\n", num, result);
result = floor(num);
printf("Floor: floor(%.2lf) = %.2lf\n", num, result);
// Round to nearest integer
result = round(num);
printf("Round: round(%.2lf) = %.2lf\n", num, result);

// Square root
result = sqrt(num1);
printf("Square Root: sqrt(%.2lf) = %.2lf\n", num1, result);

```

```c
    // Absolute value
    result = fabs(-13.5);
    printf("Absolute Value: fabs(-13.5) = %.2lf\n", result);

    // Trigonometric functions
    result = cos(angle);
    printf("Cosine: cos(%.2lf radians) = %.2lf\n", angle, result);
    result = sin(angle);
    printf("Sine: sin(%.2lf radians) = %.2lf\n", angle, result);
    result = tan(angle);
    printf("Tangent: tan(%.2lf radians) = %.2lf\n", angle, result);
    printf("\n\n\n");

    /* 03/20's Content: If-else, Switch-case, For, While, Do-while
    ==============================================================
    */
    printf("03/20's Content: If-else, Switch-case, For, While, Do-while\n");
    printf("==============================================================\n");

    int grade = 79;
    printf("Your grade is %d\n", grade);

    // If-example
    if (grade >= 60) // No semicolon here
    {
      printf("You passed the course! \t: this is if example\n");
      // Always put the statement inside {}
    }

    // If-else example
    // and multiple statements in the blocks
    if (grade >= 90)
    {
      printf("You aced the class!\n");
    }
    else
    {
      printf("You can do better! \t: this is if-else example\n");
      printf("%d more pts to get A \t: this is if-else example\n", 90 - grade);
    }

    // If-elseif-elseif-else example
    char letterGrade;
    if (grade >= 90)
    {
      letterGrade = 'A';
    }
    else if (grade >= 80)
```

```c
  {
    letterGrade = 'B';
  }
  else if (grade >= 60)
  {
    letterGrade = 'C';
  }
  else
  {
    letterGrade = 'F';
  }
  printf("Grade is %c \t: this is if-elseif-else example \n", letterGrade);

  // Relationship operators
  // ==, !=, >, <, >=, <=

  // Logical operators
  // &&, ||, !

  // Want to check if the grade is between 65 and 85
  if (grade >= 65 && grade <= 85)
  {
    printf("In the range 65~85\n");
  }

  // Want to check if the letterGrade is better than C (i.e., A or B)
  if (letterGrade == 'A' || letterGrade == 'B')
  {
    printf("Grade better than C\n");
  }
  else
  {
    printf("Grade not better than C\n");
  }

  // Want to check if the letterGrade is not F
  if (letterGrade != 'F')
  {
    printf("Not F\n");
  }

  // Switch-case example
  // with uses of block {} and break;
  float currentGPA = 3.75, newGPA;
  int current_credit_hrs = 18;
  printf("Your current GPA is %.2f\n", currentGPA);
  printf("Your current credit hours are %d\n", current_credit_hrs);

```

```c
switch (letterGrade)
{
case 'A': // The following code block will be executed if letterGrade=='A'
{
  newGPA = (currentGPA * current_credit_hrs + 4.0) / (current_credit_hrs + 3);
  printf("Your new GPA is %.2f\n", newGPA);
  break; // Break is needed to exit the switch-case;
         // otherwise, it will also execute the default case if present
}
case 'B': // Always better to put the block inside {}
{
  newGPA = (currentGPA * current_credit_hrs + 3.0) / (current_credit_hrs + 3);
  printf("Your new GPA is %.2f\n", newGPA);
  break;
}
case 'C':
{
  newGPA = (currentGPA * current_credit_hrs + 2.0) / (current_credit_hrs + 3);
  printf("Your new GPA is %.2f\n", newGPA);
  break;
}
case 'F':
{
  newGPA = (currentGPA * current_credit_hrs + 0.0) / (current_credit_hrs + 3);
  printf("Your new GPA is %.2f\n", newGPA);
  break;
}
default: // Default case is optional but good to have
  printf("Not a valid letter grade\n");
} // End of switch-case

// switch-case example with multiple cases sharing the same block
switch (grade)
{
case 89:
case 79:
case 59:
{
  printf("You are either 89, 79, or 59\n");
  printf("You are very close to the next grade\n");
}
default:
  printf("Wrong, shouldn't print this and below lines\n");
  printf("but you forget the \"break;\"\n");
  printf("You are not close to the next grade\n");
}
printf("\n");

```

```c
// While loop example
// Prompt to the teacher to enter a grade for 3 students
printf("While-loop example (please enter 3 grades:)\n");
int i = 0;
while (i < 3) // i is the counter variable,
              // the condition is checked before the loop
              // i is initialized to 0, so i<3 is true
{
  printf("Enter the grade for student #%d: ", i + 1);
  scanf("%d", &grade);
  i++; // Increment of i is a must, otherwise it will be an infinite loop
       // another way to do so is use i = i+1;
       // After one iteration, i becomes 1,
       // then the condition (whether i<3) is checked again, and so on
}

// Take a guess what is the value of i after the loop
printf("The value of i after the while-loop is %d\n\n", i);

// Do-while loop example
// Prompt to the teacher to enter a grade for 3 students
// This time, we also want to sum the grades and later calculate the average

// Initialize the counter and sum variables
printf("Do-while-loop example (please enter 3 grades:)\n");
int j = 0;
int sum = 0;
do
{
  printf("Enter the grade for student #%d: ", j + 1);
  scanf("%d", &grade);
  sum = sum + grade; // Add the grade to the sum
  j++;               // Increment of j is a must
                     // otherwise it will be an infinite loop
} while (j < 3); // The condition is checked after the loop

// Take a guess what is the value of j after the loop
printf("The value of j after the do-while-loop is %d\n", j);
printf("The sum of the grades is %d\n", sum);
printf("The average of the grades is %.2f\n\n", sum / 3.0);

// Compare the while and do-while loops:
// The while loop checks the condition before the loop,
// but the do-while loop checks it after the loop.
// This means that the do-while loop will always execute
// at least once, even if the condition is false
// The while loop will not execute if the condition is false at the beginning

// Another example of do-while loop
// Prompt to the teacher to enter a grade for just one student
// but we want to make sure the grade is between 0 and 100
printf("Do-while-loop example of error checking (grade 0~100):\n");
do
{
  printf("Enter the grade for the student: ");
  scanf("%d", &grade);
} while (grade < 0 || grade > 100); // The error-check is done after the loop
                                    // if the grade is of undesired category
                                    // the loop will execute once again
// That being said, the ??? after while(???) is the undesired range.
printf("The valid grade entered is %d\n\n", grade);

// Another example of while loop
// Prompt to the teacher to enter a grade for just one student
// but we want to make sure the grade is one of A, B, C, F
// We can use a while loop to check the input

printf("While-loop example of error checking (enter A/B/C/F):\n");
char LGrade2 = '0'; // Initialize to a invalid value
while (LGrade2 != 'A' && LGrade2 != 'B' && LGrade2 != 'C' && LGrade2 != 'F')
// Remember, the condition to put in the while loop is the undesired range.
{
  printf("Enter the letter grade for the student: ");
  scanf(" %c", &LGrade2); // Note the space before %c
}
printf("The valid letter grade entered is %c\n\n", LGrade2);

// For loop example
// Prompt to the teacher to enter a grade for 3 students using array
printf("For-loop example (please enter 3 grades:)\n");
int grades[3]; // Declare an array to store the grades
int k;         // Initialize the counter variable

for (k = 0; k < 3; k++) // Use a for loop to iterate over the array
{
  printf("Enter the grade for student #%d: ", k + 1);
  scanf("%d", &grades[k]); // Store the grade in the array
}

printf("The grades entered are: ");
for (k = 0; k < 3; k++) // Print the grades using a for loop
{
  printf("%d ", grades[k]); // Print each grade
}
printf("\n\n\n\n");
```

```c

/*

Summary of loops:
(1) while loop: checks the condition before the loop:
                if false, it won't execute
                if true, it will execute the loop
                         until the condition is false
(2) do-while loop: execute the code block once
                   then check the condition:
                   if true, it execute the loop until false
                   if false, it just exits the loop
(3) for loop: a compact way to write a loop:
              for (initialization; condition; increment) {...;}
              it is a good practice to use for loop
              when the number of iterations is known
              loop variable is usually initialized within the for loop
              run the loop until the condition is false

Syntax of loops:
(1) while loop
while (condition)
{
  // code block to do stuff and update the condition
}

(2) do-while loop
do
{
  // code block to do stuff and update the condition
} while (condition);

(3) for loop
for (initialization; condition; increment)
{
  // code block to do stuff and update the condition
}

*/

/* 03/20's Content: Strings, Repeated Running, Printing to File
==============================================================
*/
printf("03/20's Content: Strings, Repeated Running, Printing to File\n");
printf("==============================================================\n");

char ch = 'A';
char input[] = "Hello"; // Just like array with 5 integers
```

```c
                           // int M[5] = {1, 4, 9, 16, 25};
                           // char another_input[5] = {'H', 'e', 'l', 'l', 'o'};
                           // people usually use char another_input[5] = "Hello";
                           // or conventionally char inputs[] = "Hello";
                           // where the size is automatically determined
  printf("The string (input) is: %s\n", input);
  printf("The string (another_input) is: %s\n", input);
  // %s is the flag or placeholder for printing string

  input[0] = 'h';
  input[1] = 'a';
  input[2] = 'p';
  input[3] = 'p';
  input[4] = 'y';
  printf("The string (input) after modification is: %s\n", input); // happy

  for (int i = 0; i < 5; i++)
  {
    input[i] = 'z';
  }
  printf("The new string is: %s\n\n", input);
  // zzzzz, DON'T FEEL SLEEPY, it is just a string

  printf("Example of char array (i.e., string)\n");
  char az[100]; // Declare a long char array to store the alphabet
  for (int i = 0; i < 26; i++)
  {
    az[i] = 'a' + i; // special rule:
                     // it is okay to add a char and a integer
                     // 'a' + 0 = 'a'
                     // 'a' + 1 = 'b'
                     // 'a' + 2 = 'c'
  }
  printf("string az[]=%s\n\n", az); // string="abcdefghijklmnopqrstuvwxyz"

  // do-while loop example to prompt the user to enter a char between a and z
  printf("Do-while-loop example of error checking (small letters only):\n");
  char ch2 = '0'; // Initialize to a invalid value
  do
  {
    printf("Enter a char between a and z: ");
    scanf(" %c", &ch2); // Note the space before %c to ignore any whitespace
  } while (ch2 < 'a' || ch2 > 'z'); // The error-check is done after the loop
                                    // if the char is of undesired category
                                    // the loop will execute once again
  printf("The valid char entered is %c\n\n", ch2);

  // Nested for loop with 2d array
  // Prompt to the teacher to enter a grade for 2 students in 3 subjects

  printf("Nested for-loop of 2d array (grades for 2 students, 3 subjects:)\n");
  int grades2[2][3]; // Declare a 2d array to store the grades
  int m, n;          // Initialize the counter variables
  for (m = 0; m < 2; m++)
  {
    for (n = 0; n < 3; n++)
    {
      printf("Enter the grade for student #%d in subject #%d: ", m + 1, n + 1);
      scanf("%d", &grades2[m][n]); // Store the grade in the array
    }
  }
  printf("\n");

  // Run a program multiple times

  // Prompt to the teacher to enter a series of 5 grades and print the average
  // Use a do-while loop to check
  // if the teacher wants to continue to enter another series of grades
  printf("Do-while-loop example with repeated running:\n");
  do
  {
    printf("Please enter 5 grades, separately by space: ");
    int a, b, c, d, e;
    scanf("%d %d %d %d %d", &a, &b, &c, &d, &e); // Read 5 grades from the user
                                                 // scanf can read
                                                 // multiple inputs at once
                                                 // separated by space, no comma
    printf("The average of the grades is %.2f\n", (a + b + c + d + e) / 5.0);
    printf("Do you want to enter another series of grades? (y/n): ");
    scanf(" %c", &ch2); // Note the space before %c to ignore any whitespace
  } while (ch2 == 'y'); // The condition is checked after the loop
  // Remember, if the condition is true, it will execute the loop once again
  // if user enters 'n' or actually any other char, it will exit the loop

  // A small taste of pointer in C programming
  printf("\nPointer (DON'T WORRY IF THIS IS HARD TO UNDERSTAND NOW):\n");
  int regular_int = 5;
  printf("The regular_int is assigned value %d\n", regular_int);
  printf("The address (that this int is stored) is %p\n", &regular_int);
  // %p is the flag for printing address,
  // and & is the operator to get the address of a variable

  // Declare a pointer to an int
  int r = 90;
  int *ptr = &r; // ptr is a pointer to the address of int of 90
  printf("The pointer itself is %p\n", ptr);
  printf("The value pointed by the pointer is %d\n\n", *ptr);
  // *ptr is the value at the address of ptr

  printf("Example of writing a string to txt file:\n");
  // Similarly, declare a pointer to a file
  FILE *filePointer; // Same as int, FILE is a data type
                     // * indicates that this is a pointer to a file
                     // Syntax: FILE *any_name;

  // Open a new file for different task mode
  filePointer = fopen("out.txt", "w"); // Creates file output.txt in write mode
                                       // Syntax: assume "any_name" before
                                       // any_name = fopen("file_name", "mode");

  // Check if the file was opened successfully
  // This is especially important when in read or append mode
  // since the file may not exist
  if (filePointer == NULL)
  // == NULL is to check whether filePointer points to a valid file
  {
    printf("Error: Could not open file.\n");
    return 1; // Exit with an error code
              // remember that main() returns an int
              // return 0 means success, return 1 means error
  }

  // If you are here, it means the file was opened successfully
  double ydouble = 15.0;
  // Print to the screen
  printf("The value printed to console screen is %.2lf\n", ydouble);

  // Print to the file
  fprintf(filePointer, "The value printed to file is %.2lf\n", ydouble);
  // Syntax: supposing you use any_name before
  // fprintf(any_name, CONTENT);
  // CONTENT can be just a string, or a string with variables
  // CONTENT follows the same format as printf
  fprintf(filePointer, "Finished.");

  // Close the file
  fclose(filePointer); // This is a must.
                       // Syntax: fclose(any_name);
  return 0;
} // end of main()

/*
Output of the program:
```

```

03/18's Content: Integer, Floating Numbers, Math Functions
=============================================================
Addition (int): z = 9
Multiplication (double): c = 20.000000
Integer Division: z = 1
Double Division: c = 1.250000
z=6
c=6.000000
Another version:
z=7
c=7.500000
Remainder: z = 1

Exponentiation: 2.00^3.00 = 8.00
Wrong result with ^ for 3^3 is: 0
Natural Logarithm: log(2.00) = 0.69
Logarithm Base 10: log10(2.00) = 0.30
Logarithm Base 2: log2(2.00) = 1.00
Exponential: e^3.00 = 20.09
Ceiling: ceil(2.70) = 3.00
Floor: floor(2.70) = 2.00
Round: round(2.70) = 3.00
Square Root: sqrt(2.00) = 1.41
Absolute Value: fabs(-13.5) = 13.50
Cosine: cos(0.79 radians) = 0.71
Sine: sin(0.79 radians) = 0.71
Tangent: tan(0.79 radians) = 1.00



03/20's Content: If-else, Switch-case, For, While, Do-while
=============================================================
Your grade is 79
You passed the course!  : this is if example
You can do better!      : this is if-else example
11 more pts to get A    : this is if-else example
Grade is C      : this is if-elseif-else example
In the range 65~85
Grade not better than C
Not F
Your current GPA is 3.75
Your current credit hours are 18
Your new GPA is 3.31
You are either 89, 79, or 59
You are very close to the next grade
Wrong, shouldn't print this and below lines
```

```
but you forget the "break;"
You are not close to the next grade

While-loop example (please enter 3 grades:)
Enter the grade for student #1: 92
Enter the grade for student #2: 95
Enter the grade for student #3: 98
The value of i after the while-loop is 3

Do-while-loop example (please enter 3 grades:)
Enter the grade for student #1: 92
Enter the grade for student #2: 95
Enter the grade for student #3: 98
The value of j after the do-while-loop is 3
The sum of the grades is 285
The average of the grades is 95.00

Do-while-loop example of error checking (grade 0~100):
Enter the grade for the student: -8
Enter the grade for the student: 104
Enter the grade for the student: 94
The valid grade entered is 94

While-loop example of error checking (enter A/B/C/F):
Enter the letter grade for the student: H
Enter the letter grade for the student: U
Enter the letter grade for the student: b
Enter the letter grade for the student: B
The valid letter grade entered is B

For-loop example (please enter 3 grades:)
Enter the grade for student #1: 98
Enter the grade for student #2: 100
Enter the grade for student #3: 72
The grades entered are: 98 100 72



03/20's Content: Strings, Repeated Running, Printing to File
==============================================================
The string (input) is: Hello
The string (another_input) is: Hello
The string (input) after modification is: happy
The new string is: zzzzz

Example of char array (i.e., string)
string az[]=abcdefghijklmnopqrstuvwxyz
```

```

Do-while-loop example of error checking (small letters only):
Enter a char between a and z: +
Enter a char between a and z: $
Enter a char between a and z: L
Enter a char between a and z: l
The valid char entered is l

Nested for-loop of 2d array (grades for 2 students, 3 subjects:)
Enter the grade for student #1 in subject #1: 83
Enter the grade for student #1 in subject #2: 84
Enter the grade for student #1 in subject #3: 100
Enter the grade for student #2 in subject #1: 89
Enter the grade for student #2 in subject #2: 78
Enter the grade for student #2 in subject #3: 98

Do-while-loop example with repeated running:
Please enter 5 grades, separately by space: 84 98 76 89 93
The average of the grades is 88.00
Do you want to enter another series of grades? (y/n): y
Please enter 5 grades, separately by space: 98 87 99 100 56
The average of the grades is 88.00
Do you want to enter another series of grades? (y/n): n

Pointer (DON'T WORRY IF THIS IS HARD TO UNDERSTAND NOW):
The regular_int is assigned value 5
The address (that this int is stored) is 0x7ffffffdb3c
The pointer itself is 0x7ffffffdb38
The value pointed by the pointer is 90

Example of writing a string to txt file:
The value printed to console screen is 15.00

*/
```