Lectures/Week_12/short_code/1_switch_case.c

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  // Declare variables
  char letterGrade;
  printf("Enter your letter grade (A, B, C, F): ");
  scanf(" %c", &letterGrade); // Note the space before %c
  switch (letterGrade)
  {
  case 'A': // The following code block will be executed if letterGrade=='A'
  {
    printf("Your GPA is 4.0\n");
    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 {}
  {
    printf("Your GPA is 3.0\n");
    break;
  }
  case 'C':
  {
    printf("Your GPA is 2.0\n");
    break;
  }
  case 'F':
  {
    printf("Your new GPA is 0.0\n");
    break;
  }
  default: // Default case is optional but good to have
    printf("Not a valid letter grade\n");
  } // End of switch-case
}
```

Lectures/Week_12/short_code/2_error_check.c

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  // All the examples below are for error checking

  printf("Do-while-loop example of error checking (grade 0~100):\n");

  int grade; // Must declare grade before using it in scanf
  // Variable declaration must be before the do-while loop
  // if that variable is used in the loop condition
  do
  {
    printf("Enter the grade for the student: ");
    scanf("%d", &grade);
  } while (grade < 0 || grade > 100);

  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');
  printf("The valid char entered is %c\n\n", ch2);

  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);
}
/*
structure of do-while loop for error checking

MUST define var here
do
{
  printf and scanf to get the var value;
} while (var value is undesired);

```

```
example 1:

MUST define op here
do
{
  printf and scanf to get the var value;
} while (op != 'Y' && op != 'N');

example 2:

MUST define num here
do
{
  printf and scanf to get the var value;
} while (num < 10 || num > 100);
*/
```

Lectures/Week_12/short_code/3_run_again.c

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

int main()
{
  char runagain = 'Y';
  do
  {
    printf("Hello, world!\n");
    int a = 45;         // a and b can be defined inside the loop
    int b = pow(a, 2); // a's square
    printf("The square of %d is %d\n", a, b);

    printf("Do you want to run the program again? (Y/N): ");
    scanf(" %c", &runagain);
  } while (runagain == 'Y');
  return 0;
}

// Structure of do-while loop for a program to be run again

// declare char of runagain before do-while
// do
// {
//   printf and scanf to assign value to runagain;
// }
// while (runagain = 'Y');
```

Lectures/Week_12/short_code/4_char_string.c

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  // All the examples below are for chars and strings read

  // Case 1: You know how many chars to receive from the user
  // and want user to enter them one by one
  char initials[4];
  printf("Please enter your initials one by one:\n");
  for (int i = 0; i < 4; i++)
  {
    scanf(" %c", &initials[i]);
    // Note the space before %c to ignore any whitespace
  }
  printf("The work is done by %c.%c., %c.%c. for C classwork\n",
         initials[0], initials[1], initials[2], initials[3]);

  // Case 2: You don't know how many chars there are
  // and just want user to enter them all at once

  // Declare a long char array to store the filename
  char filename[100];
  printf("Please enter the name of the file all at once: ");
  scanf("%s", filename); // %s is the flag for printf/scanf string
  printf("The filename is: %s\n", filename);

  // Case 3: You don't know how many chars there are
  // but you will ask user to give you that number first
  // then ask user to enter the chars one by one

  // Declare a long char array to store
  int digit;
  printf("Please enter the number of chars you want to enter: ");
  scanf("%d", &digit); // the flag is %d

  char str[digit];
  for (int i = 0; i < digit; i++)
  {
    printf("Please enter char #%d: ", i + 1);
    scanf(" %c", &str[i]); // Note the space before %c
  }
  return 0;
}
```

Lectures/Week_12/short_code/5_printf_format.c

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  int m = 40, n = 25;
  printf("The value of m is %d\n", m);
  printf("The value of n is %d\n", n);

  // Swap the values of i and j
  int temp = m;
  m = n;
  n = temp;

  printf("After swapping, the value of n is %d\n", n);
  printf("m=%6d(R-A w=6)\n", m);  // 6 width, right aligned
  printf("m=%5d(R-A w=5)\n", m);  // 5 width, right aligned
  printf("m=%4d(R-A w=4)\n", m);  // 4 width, right aligned
  printf("m=%-4d(L-A w=4)\n", m); // 4 width, left aligned
  printf("m=%3d(R-A w=3)\n", m);  // 3 width, right aligned

  float p = m / 4.0;
  printf("p=%f(DEF-A dp=6)\n", p);             // default 6 decimal places, default aligned
  printf("p=%10f(R-A dp=6 w=10)\n", p);        // 10 width, default 6 decimal places, right aligned
  printf("p=%5.2f(R-A dp=2 w=5)\n", p);        // 5 width, 2 decimal places, right aligned
  printf("p=%7.4f(R-A dp=4 w=7)\n", p);        // 7 width, 4 decimal places, right aligned
  printf("p=%-8.3f(L-A dp=3 w=8)\n", p);       // 8 width, 3 decimal places, left aligned
  printf("p=%+8.2f(RSigned-A dp=2 w=8)\n", p); // 8 width, 2 decimal places, right aligned
  printf("p=%-10.1f(L-A dp=1 w=10)\n", p);     // 10 width, 1 decimal places, left aligned
  printf("p=%09.3f(Zero-A dp=3 w=9)\n\n", p);  // 9 width, 3 decimal places, default aligned with leading zeros
}

/**
```

```
Summary of format specifiers with width and alignment

%x.yf : x width and y decimal places, right aligned
reserve x spaces and write from right,
last decimal positions go to the right
finish y decimal places, then numbers, then spaces to make it x-width

%+x.yf : show sign, right aligned, same as above
everything is the same as above, but show sign
if negative, show - sign, if positive, show + sign

%0x.yf : leading zeros, right aligned, same as above
everything is the same as above, but leading zeros
not using spaces to make it x-width, but using 0s
if negative, show - sign, if positive, does not show

%-x.yf : left aligned,
write from left, start from numbers, then y decimal places
last with spaces to make it x-width

*/
```

Lectures/Week_12/short_code/6_sum.c

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  // Examples of finding the sum of an array with known size and elements
  int arr[5] = {2, 4, 6, 8, 10};

  // Method 1: Using a for loop
  int sum1 = 0;
  for (int i = 0; i < 5; i++) // or using i<=4
  {
    sum1 = sum1 + arr[i];
  }
  printf("Sum of the array using for loop: %d\n", sum1);

  // Method 2: Using a while loop
  int sum2 = 0, j = 0;
  while (j < 5)
  {
    sum2 = sum2 + arr[j];
    j++;
  }
  printf("Sum of the array using while loop: %d\n", sum2);

  // Method 3: Using a do-while loop
  int sum3 = 0, k = 0;
  do
  {
    sum3 = sum3 + arr[k];
    k++;
  } while (k < 5);
  printf("Sum of the array using do-while loop: %d\n", sum3);

  // Advanced example of finding the sum of an array with known size
  // but elements are from user input
  int grades[5];
  int sum4 = 0;
  printf("Enter 5 grades:\n");
  for (int i = 0; i < 5; i++)
  {
    printf("Grade %d: ", i + 1);
    scanf("%d", &grades[i]);
    sum4 = sum4 + grades[i];
  }
  printf("Sum of the grades: %d\n", sum4);
```

```c

  // Advanced example of finding the sum of a 2d array
  int arr2d[3][4] = {
      {1, 2, 3, 4},
      {5, 6, 7, 8},
      {9, 10, 11, 12}};
  int sum5 = 0;
  for (int r = 0; r < 3; r++)
  {
    for (int c = 0; c < 4; c++)
    {
      sum5 = sum5 + arr2d[r][c];
    }
  }
  printf("Sum of the 2D array: %d\n", sum5);
}
```

Lectures/Week_12/short_code/7_quiz4_recitation.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <math.h>   // Include math library for math functions, like floor
#include <string.h> // Include string library for string functions, like strcmp

int main()
{
  int a = 17, b = 10;
  int c, d;
  float s = 10.9, t;

  if (floor(s) < b)
  {
    printf("Inside block 1\n");
    t = b * s;
    d = a - b;
  }
  else if (floor(s) == b)
  {
    printf("Inside block 2\n");
    t = b / a;
    d = b - a;
  }
  else
  {
    printf("Inside block 3\n");
    t = b / s;
    d = a / b;
  }

  c = a % b;
  printf("c=%3d$\n", c);      // use $ at the end to measure the width
  printf("d=%-5d$\n", d);     // use $ at the end to measure the width
  printf("t=%-6.2f$\n\n", t); // use $ at the end to measure the width

  // ----------------------HANDOUT QUESTIONS-------------------------

  int i;
  int arr[5] = {2, 4, 6, 8, 10};
  int new_arr[7] = {5, 5, 5, 5, 5, 5, 5};

  for (i = 2; i <= 5; i = i + 3)
  {
    new_arr[i] = arr[i - 2];
  }

```

```c
  int j = 0;
  while (j < 6)
  {
    printf("%d\n", new_arr[j]);
    j = j + 1;
  }
  printf("z\nz\nz\n\n");

  // if your output exceeds the given 8 lines
  // just show the first 8 lines

  //----------------------MORE QUIZ 4 REVIEW-------------------------
  // construct an int array of 8 elements = 20, 25, 3, 25, 16, 30, 30, 79
  // how to use a for-loop to find the average of the array?
  // and printf the average value!

  int arr2[8] = {20, 25, 3, 25, 16, 30, 30, 79};
  int sum2 = 0;
  for (i = 0; i <= 7; i++) // 0:7 in matlab
  {
    sum2 = sum2 + arr2[i];
  }

  float average2 = sum2 / 8.0; // note 8.0 instead of 8
  printf("The average of the array is %.2f\n", average2);

  // first scanf two strings xxx and yyy
  // (e.g. CProgramming, C++Programming, Python, Java,
  // Matlab, Fortran, RLanguage, Julia)
  // then print: "I prefer to coding in xxx than yyy"

  char str1[20], str2[20]; // declare a long char array to store use inputs
  printf("Please enter a language you like: \n");
  scanf("%s", &str1); // read a string all at once from user
  printf("Please enter a language you don't like: \n");
  scanf("%s", &str2); // read a string all at once from user

  printf("I prefer to coding in %s than %s\n", str1, str2);

  // if you choose xxx=CProgramming, then
  // print: "In my favorite language, M[1]=25"
  // else if you choose xxx=Matlab then
  // print: "In my favorite language, M[1]=20"
  // else print: "In my favorite language, M[1]=IDK"

  // Hints: strcmp("Hello", "Hello") == 0
  // strcmp("Hello", "World") != 0
  // you can think strcmp returns the difference between the two strings
```

```
95
96    if (strcmp(str1, "CProgramming") == 0)
97    {
98      printf("In my favorite language, M[1]=25\n");
99    }
100   else if (strcmp(str1, "Matlab") == 0)
101   {
102     printf("In my favorite language, M[1]=20\n");
103   }
104   else
105   {
106     printf("In my favorite language, M[1]=IDK\n");
107   }
108 }
```

Lectures/Week_12/short_code/8_file_print.c

```c
#include <stdio.h>
#include <stdlib.h>

int main()
{
  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;
}
```