GNG1106
Fundamentals of Engineering Computation

Instructor: **Hitham Jleed**

University of Ottawa

Fall 2023 ~

In-Class Exercise:

Outline

An example function without prototype

```
#include <stdio.h>
int sum(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
int main()
{
    int x, y;
    printf("Enter 2 integers\n");
    scanf("%d%d", &x, &y);
    printf("The sum is %d\n", sum(x,
y));
    return 0;
}
```

An example function with prototype

```
#include <stdio.h>
int sum(int, int);
int main()
{
    int x, y;
    printf("Enter 2 integers\n");
    scanf("%d%d", &x, &y);
    printf("The sum is %d\n", sum(x,
y));
    return 0;
}
int sum(int a, int b)
{
    int c;
    c=a+b;
    return c;
}
```

- The syntax of a function prototype is:

  ```
  type functionName(type, type, ..., type);
  ```

  or

  ```
  type functionName(type var1Name, type var2Name, ...,
  type varNName);
  ```

  - In the example, the prototyping statement can also be written as `int sum(int a, int b);`
- Note that the prototyping statement ends with a semi-colon ";"
- The prototype:
  - declares that the function named `functionName` will be called (e.g. by main).
  - specifies the type of the value returned by the function,
  - identifies the number of parameters passed to the function and the order of their types.

- The prototype of a function must have:
  - the same name as in the function definition,
  - the same function type as in the function definition,
  - the same number of parameters as in the function definition, and
  - the same order of parameter types in the function definition.

- Function prototypes are not required in a C program but are highly encouraged, particularly for large programs.
- In order to call a function, either the function definition or its prototype must appear before the function call.
- In small program (which consists of a single file), the function prototypes are usually listed after the preprocessor directives but before the `main` function in the program file.

Function Prototyping as a Program Design Methodolgy

- The prototype specifies the input/output format of the function (via its parameter list and return type) and gives information on what it does (via its name and additional illustrative comments).
- If we consider a function as "tool", then from the program design point of view, its prototype is analogous to the "tool manual".
- The use (i.e., calling) of a well-designed function only needs to know its prototype, without needing to know its internal details.
- Function prototyping can be used to separate the design of the function and the design of higher-level logic (e.g., the `main` function) calling the function. In a large program, the two parts may be implemented in separation (i.e., by different people or at different times), with the function prototype serving as their interface.

- In a large program, the definitions of a related suite of functions (a "tool box") and their prototypes (the "manuals" for the "tool box") are often placed in separate files.
  - Usually, the file containing only the function prototypes (and possible other related global variables) is called a header file, and named as "xxx.h".
  - The functions themselves are placed in another file, say, named "xxx.c" file.
  - The `main` function is also placed in a separate file, say, named "myProgram.c".
- This allows the program files to be better organized, and the toolbox to be better reused.
- The file `stdio.h` is in fact is a header file.

Write a Program ...

Write a program that does the following. It will ask the user to enter an integer larger than 1, and the if entered integer is not larger than 1, it keeps prompting the user. After the user enters a valid integer, the program prints all the prime factors of the integer (including the repeated factors). For example, if the entered integer is 24, the program prints:
2 2 2 3

Planing

- What modules should the program have?
  - prompting the user to enter the integer until he enters a valid one.
    $\Rightarrow$ make a function
    `int getUserInput()`
  - print the prime factors
    $\Rightarrow$ make a function
    `void printPrimeFactorsFor(int x)`
- How to print prime factors? Try an example by hand, say $x = 24$.
  1. is 24 divisible by 2? Yes: print 2; divide 24 by 2 $\Rightarrow$ 12.
  2. is 12 divisible by 2? Yes: print 2; divide 12 by 2 $\Rightarrow$ 6.
  3. is 6 divisible by 2? Yes: print 2; divide 6 by 2 $\Rightarrow$ 3.
  4. is 3 divisible by 2? No.
  5. Is 3 divisible by 3? Yes: print 3; divide 3 by 3 $\Rightarrow$ 1. (Completed)

- Idea of implementing printPrimeFactors:
  - Keep dividing x by prime factors, where we test each prime factor in increasing order while also reducing x.
  - When x is reduced to 1, we complete.
- What is the state?
  - the current value of x: variable `x`
  - the current tested factor: variable `factor`
  - has x been reduced in this iteration? a true/false-valued (0/1-valued) variable `isXNew`

- Before loop:
  - `factor=2;`
  - `isXNew=1;`
- While-loop, under condition `(x!=1)`
  - `if (!isXNew)`
    `factor++;`
  - if (x is divisible by factor)
    - `print factor;`
    - `x=x/factor;`
    - `isXNew=1;`
  - Else
    - `isXNew=0;`

In-Class Exercise: