FACULTY OF ENGINEERING AND TECHNOLOGY

COMPUTER SCIENCE DEPARTMENT

COMP1310
Introduction to Computer and Computing Ethics

TOP-DOWN DESIGN WITH FUNCTIONS

Functions

- A named section of a program that performs a specific task.

- Why do we use functions?
  - *Reusability: once a function is defined, it can be used over and over again.*
  - *Modularity: you can think of a program as a bunch of sub-steps, each sub-step in a separate function.*
  - *Abstraction: If you just want to use the function in your program, you don't have to know how it works inside!*

Functions – cont.

- Functions we have already seen in the previous chapter:
  - *void main()*
  - *printf(“Hello World!”);*
  - *scanf(“%d”, &age);*
  - *fopen(“input_data.txt”, “r”);*
  - *fscanf(inp,“%lf”,&miles);*
  - *fclose(inp);*

Types of Functions

- Library Functions
  - *C libraries provide many predefined functions that can be used to perform different tasks.*
  - *We need to include libraries in our code to use these functions.*

- User-defined Functions
  - *A function that you, the programmer, write.*

Defining a Function

- The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list )
{
   body of the function
}
```

- main is the most important function in a C program.

Defining a Function – cont.

- Return type:
  1. *A function with a return statement*
     - This function may return one thing only.
     - The function must end with a ‘return’ statement.
     - The return type is a data type, e.g. int, char, double… etc.

  2. *A function without a return statement*
     - This function returns nothing, i.e. ‘void’.
     - This function does not end with a ‘return’ statement.

Defining a Function

- The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list )
{
   body of the function
}
```

- main is the most important function in a C program.

Defining a Function – cont.

- Function name:
  - *This is the actual name of the function.*
  - *Naming functions follows the same rules as naming variables:*
    - Names can consist of letters, numbers, and underscores.
    - Names cannot start with numbers.
    - Names cannot include spaces.
  - *Naming conventions:*
    - Function names are verbs
    - If they return a true/false value, they start with ‘is_’, e.g., ‘is_perfect_number’.
    - Functions names start with a lower-case letter.

Defining a Function

- The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list )
{
   body of the function
}
```

- main is the most important function in a C program.

Defining a Function – cont.

- Parameter lists:
  - *A parameter is like a placeholder.*
  - *When a function is invoked, you pass a value to the parameter.*
  - *This value is referred to as argument.*
  - *The parameter list refers to the type, order, and number of the parameters of a function.*
  - *Parameters are optional; that is, a function may contain no parameters.*

Defining a Function

- The general form of a function definition in C programming language is as follows −

```
return_type function_name( parameter list )
{
   body of the function
}
```

- main is the most important function in a C program.

Defining a Function – cont.

- Body of function:
  - *Contains a collection of statements that define what the function does.*

Function declarations and calls

- The function return type, name, and parameter list together are called the function prototype.
- The function prototype must be included at the top of the program before the main function.
- Function prototypes must always end with a semicolon.

```
return_type function_name( parameter list );
```

- The function name and parameter list together are called the function signature.
- To evoke the function from within the main function or another function, we use the function signature.

```
function_name( parameter list )
```

Useful C Library Functions

These examples are from the math library, so you must add
#include <math.h>
at the beginning of your program.

Function	Signature	Argument	Return	Example	Math Equivalent
Power	pow(x, y)	double, double	double	x = 0.16, y = 0.5 pow(x, y) = 0.4	x^y
Square Root	sqrt(x)	double	double	x = 2.25 sqrt(x) = 1.5	$\sqrt[2]{x}$
Natural Logarithm	log(x)	double	double	x = 2.71828 log(x) = 1.0	$\log_e x$
Common Logarithm	log10(x)	double	double	x = 100.0 log10(x) = 2.0	$\log_{10} x$
Exponential	exp(x)	double	double	x = 1.0 exp(x) = 2.71828	e^x

Useful C Library Functions – cont.

These examples are from the math library, so you must add
#include <math.h>
at the beginning of your program.

Function	Signature	Argument	Return	Example
Sine	sin(x)	double (radians)	double	x = 1.5708 sin(x) = 1.0
Cosine	cos(x)	double (radians)	double	x = 0.0 cos(x) = 1.0
Tan	tan(x)	double (radians)	double	x = 0.0 tan(x) = 0.0

Useful C Library Functions – cont.

These examples are from the math library, so you must add
#include <math.h>
at the beginning of your program.

Function	Signature	Argument	Return	Example	Math Equivalent
Ceiling	ceil(x)	double	double	x = 45.23 ceil(x) = 46.0	$\lceil x \rceil$
Floor	floor(x)	double	double	x = 45.23 floor(x) = 45.0	$\lfloor x \rfloor$

- The floor of a number, n, denoted by $\lfloor n \rfloor$ is the largest integer variable smaller than n.
- The ceiling of a number, n, denoted by $\lceil n \rceil$ is the smallest integer variable larger than n.

Useful C Library Functions – cont.

- There are two common functions to calculate the absolute value:
- stdlib.h

Function	Signature	Argument	Return	Example	Math Equivalent
Absolute Value	abs(x)	int	int	x = -5 abs(x) = 5	$\|x\|$

- math.h

Function	Signature	Argument	Return	Example	Math Equivalent
Absolute Value	fabs(x)	double	double	x = -5.432 fabs(x) = 5.432	$\|x\|$

Example – Area of a triangle

- Write a function that calculates the area of a triangle.
- Things to keep in mind:
  - *The function's name needs to be relevant*
    `calculate_triangle_area`
  - *The function requires the triangle's base length and height as input*
    - Give each parameter a relevant name, e.g. base and height.
    - Determine the correct data type of each parameter, int vs. double.
  - *The function will return the value of the area*
    - Determine the correct data type of the return variable.
  - *What will the function do?*
    - Calculate the area of the triangle using the formula

$$area = \frac{1}{2} \times base \times height$$

Example – Area of a triangle – cont.

```c
#include <stdio.h>

double calculate_triangle_area (double, double);

void main() {
    double base, height;
    printf("Please enter the length of the base of the triangle: ");
    scanf("%lf", &base);
    printf("Please enter the height of the triangle: ");
    scanf("%lf", &height);

    double area = calculate_triangle_area(base, height);
    printf("The area of the triangle = %f", area);
}

double calculate_triangle_area (double base, double height) {
    double area;
    area = 0.5 * base * height;
    return area;
}
```

Example – Average of two numbers

- Write a function that reads two integers from the user and returns their average.
- Things to keep in mind:
  - *The function's name needs to be relevant*

    `get_average`
  - *The function requires two integers as input*
  - *The function will return the value of the average*
    - Determine the correct data type of the return variable
  - *What will the function do?*
    - Calculate the average of the two numbers

$$average = \frac{first\ number + second\ number}{2}$$

Example – Average of two numbers – cont.

```c
#include <stdio.h>

double get_average (int, int);

void main() {
    int num_1, num_2;
    printf("Please enter two integer numbers: ");
    scanf("%d%d", &num_1, &num_2);

    double average = get_average(num_1, num_2);
    printf("The average of %d and %d = %f", num_1, num_2, average);
}

double get_average (int a, int b) {
    double average;
    average = (a + b) / 2.0;
    return average;
}
```

Scope of Variables

- When a variable is declared in one section of your code, e.g. inside a function, it is only known within that scope.
- If you go to a different function that variable will not be defined.
- If you define a variable with the same name inside a different function, it is a different variable that does not relate to the first one in any way.

Scope of Variables

```
1  #include <stdio.h>
2
3  int f(int, int, int);
4
5  void main() {
6      int a;
7      a = f(3, 3, 4);
8      printf("a is %d", a);
9  }
10
11 int f(int a, int b, int c) {
12     int d;
13     d = a * b + 2 * c;
14     return (d);
15 }
```

main

int a

Function f

int a
int b
int c
int d

Output

Scope of Variables

```c
#include <stdio.h>

int f(int, int, int);

void main() {
    int a;
    a = f(3, 3, 4);
    printf("a is %d", a);
}

int f(int a, int b, int c) {
    int d;
    d = a * b + 2 * c;
    return (d);
}
```

main

int a

Function f

int a = 3
int b = 3
int c = 4
int d

Output

Scope of Variables

```c
#include <stdio.h>

int f(int, int, int);

void main() {
    int a;
    a = f(3, 3, 4);
    printf("a is %d", a);
}

int f(int a, int b, int c) {
    int d;
    d = a * b + 2 * c;
    return (d);
}
```

main

int a

Function f

int a = 3
int b = 3
int c = 4
int d = 17

Output

Scope of Variables

```c
#include <stdio.h>

int f(int, int, int);

void main() {
    int a;
    a = f(3, 3, 4);
    printf("a is %d", a);
}

int f(int a, int b, int c) {
    int d;
    d = a * b + 2 * c;
    return (d);
}
```

main

int a = 17

Function f

int a = 3
int b = 3
int c = 4
int d = 17

Output

Scope of Variables

```c
#include <stdio.h>

int f(int, int, int);

void main() {
    int a;
    a = f(3, 3, 4);
    printf("a is %d", a);
}

int f(int a, int b, int c) {
    int d;
    d = a * b + 2 * c;
    return (d);
}
```

main

int a = 17

Function f

int a = 3
int b = 3
int c = 4
int d = 17

Output

a is 17

Using C Library Functions

- To use a C library function we should:
  - *Import the proper library*
  - *Find out the function prototype*

- To calculate 6^5, we should:
  - *Import the math.h library*
  - *Find out the power function prototype*

    double pow (double, double);

Using C Library Functions – cont.

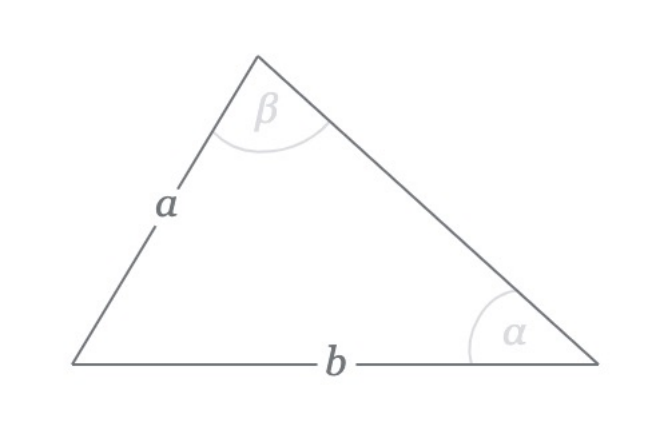
```c
#include <stdio.h>
#include <math.h>

void main() {
    double num_1, num_2;
    printf("Please enter two numbers: ");
    scanf("%lf%lf", &num_1, &num_2);

    double power = pow(num_1, num_2);
    printf("The value of %f to the power %f = %f", num_1, num_2, power);
}
```

Using C Library Functions – cont.

- Use C library functions to calculate the length of the side a of this triangle.



The function we need to use is:

$$a = b \times \frac{\sin \alpha}{\sin \beta}$$

math.h has a a sine function.

This function requires values in radians.

$$radian = \frac{degree \times \pi}{180}$$

Using C Library Functions – cont.

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

void main() {
    double alpha, beta, b;
    printf("Please enter the length of b: ");
    scanf("%lf", &b);
    printf("Please enter the angle alpha in degrees: ");
    scanf("%lf", &alpha);
    printf("Please enter the angle beta in degrees: ");
    scanf("%lf", &beta);

    double a = b * (sin(alpha * M_PI / 180) / sin(beta * M_PI / 180));
    printf("The value of a = %f", a);
}
```

Using C Library Functions – cont.

- Random Number Generator:
  - *There is a function in a C library that returns a random number.*
  - *The function is in the stdlib.h library.*
  - *The function prototype is:*

    int rand(void)
  - *The function returns a positive integer number between 0 and a constant value called RAND_MAX.*
  - *RAND_MAX is a constant defined in stdlib.h, and its value is at least 32,767. It may be different in different implementations.*
  - *Let's use this function to write a function that would roll a die three times.*

Using C Library Functions – cont.

```c
#include <stdlib.h>

int roll_die();

int main()
{
    int die_roll_1 = roll_die();
    printf("The first die roll value was %d.\n", die_roll_1);
    int die_roll_2 = roll_die();
    printf("The second die roll value was %d.\n", die_roll_2);
    int die_roll_3 = roll_die();
    printf("The third die roll value was %d.\n", die_roll_3);

    return 0;
}

int roll_die() {
    int random_number = rand();
    int die_roll = (random_number % 6) + 1;
    return die_roll;
}
```

Using C Library Functions – cont.

- Rounding using floor(x)
  - *Function floor can be used to round a number to a specific decimal place.*
  - *To round x to the tenths position (the first position to the right of the decimal point, we use the statement:*

    *y = floor (x * 10 + 0.5) / 10;*
  - *To round x to the hundredths position (the second position to the right of the decimal point, we use the statement:*

    *y = floor (x * 100 + 0.5) / 100;*

Using C Library Functions – cont.

- Rounding using floor(x)
  - *Write a program with the following functions:*
    - round_to_integer (number)
    - round_to_tenths (number)
    - round_to_hundredths (number)
    - round_to_thousandths (number)

Using C Library Functions – cont.

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

int round_to_integer ( double );
double round_to_tenths ( double );
double round_to_hundredths ( double );
double round_to_thousandths ( double );

void main() {
    printf("Please enter a number: ");
    double x;
    scanf("%lf", &x);
    printf("The number %lf rounded to the nearest integer is %d.\n", x, round_to_integer(x));
    printf("The number %lf rounded to the nearest tenths is %f.\n", x, round_to_tenths(x));
    printf("The number %lf rounded to the nearest hundredths is %f.\n", x, round_to_hundredths(x));
    printf("The number %lf rounded to the nearest thousandths is %f.\n", x, round_to_thousandths(x));
}
```

Using C Library Functions – cont.

```
19  int round_to_integer ( double x ) {
20      return floor ( x + 0.5);
21  }
22
23  double round_to_tenths ( double x ) {
24      return floor ( x * 10 + 0.5) / 10;
25  }
26
27  double round_to_hundredths ( double x ) {
28      return floor ( x * 100 + 0.5) / 100;
29  }
30
31  double round_to_thousandths ( double x ) {
32      return floor ( x * 1000 + 0.5) / 1000;
33  }
```

Example – Binary to Decimal

- Let's write a program that converts numbers from binary to decimal.
- The formula to do the conversion is as follows:
  - *Number the digits based on their location starting at 0 from the right to the left.*
  - *The value of each digit in decimal equals*

    *the value of the digit in binary * 2^position*
  - *For example:* $1\ 0\ 1\ 1 = 1 * 2^3 + 0 * 2^2 + 1 * 2^1 + 1 * 2^0$

    $= 1 * 8 + 0 * 4 + 1 * 2 + 1 * 1$

    $= 8 + 0 + 2 + 1$

    $= 11$
  - *Include a function that takes the value of a binary digit and its position and returns the equivalent decimal value to that digit.*

Example – Binary to Decimal– cont.

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

int binary_to_decimal(int, int);

void main()
{
    printf("Input a 4-digit binary number: ");
    int b;
    scanf("%d", &b);

    int sum = 0;
    int digit = b %10;
    sum += binary_to_decimal(digit, 0);

    digit = (b/10)%10;
    sum += binary_to_decimal(digit, 1);

```

Example – Binary to Decimal– cont.

```
18
19      digit = (b/100)%10;
20      sum += binary_to_decimal(digit, 2);
21
22      digit = b/1000;
23      sum += binary_to_decimal(digit, 3);
24
25      printf("This number equals %d in decimal.", sum);
26  }
27
28  int binary_to_decimal(int digit, int position) {
29      int value = digit * pow(2, position);
30      return value;
31  }
```

Exercise – Seconds since midnight

- Let's write a program that reads the time from the user as three integers representing the hour, the minutes, and the seconds, and tells the user how many seconds have passed since 12 o'clock.
- Include a function that takes three integer values representing the hour, the minutes, and the seconds, and returns the total number of seconds that have passed since 12.