

Programming For Engineers

The Six Basic Elements of C Programming

by


Wan Azhar Wan Yusoff¹, Ahmad Fakhri Ab. Nasir²
Faculty of Manufacturing Engineering
wazhar@ump.edu.my¹, afakhri@ump.edu.my²



0.0 Chapter's Information

- Expected Outcomes
 - To review six elements of basic C programming competency.
- Contents
 - 1.0 Standard Data Type
 - 2.0 Console I/O
 - 3.0 Expression
 - 4.0 Selection Structure
 - 5.0 Repetition Structure
 - 6.0 Function



- 
- Before we move to intermediate programming, review of basics C programming is necessary. The six programming elements are: standard data types, console I/O, expressions, selection structure, repetition structure and functions. We assume you know how to create, compile and link a C program under Windows and Code::Blocks IDE.



1.0 Standard Data Types

- Standard data types represent (1) logical data (TRUE or FALSE), (2) character, (3) integer and (4) real number. Issues about data types are byte size and the range of values that they represent. The following table presents the sizes of the standard data types as well as their range of values.



1.0 Standard Data Types

Standard Data Type	C-implementation	Memory Size	Minimum Value	Maximum Value
Character	char	1 byte	-128	127
	unsigned char	1 byte	0	255
Integer (unsigned)	unsigned short	2 bytes	0	65,535
	unsigned int	4 bytes	0	4,294,967,295
	unsigned long long	8 bytes	0	$2^{64} - 1$
Integer (signed)	short	2 bytes	-32,768	32,767
	int	4 bytes	-2,147,483,648	2,147,483,647
	long long	8 bytes	-2^{63}	$2^{63} - 1$
Real number	float	4 bytes	1.17549×10^{-38}	3.40282×10^{38}
	double	8 bytes	2.22507×10^{-308}	1.79769×10^{308}
	long double	12 bytes	-	-



1.0 Standard Data Types

- As programmers, we should be aware of our choice of data type when we want to represent any data. For example, if we want to represent the number of books in a library we probably should use unsigned int since there shouldn't be any negative number of books. Furthermore, the maximum number for unsigned int is 4,294,967,295. This number should be large enough to represent the number of books in any library. If we use unsigned char, certainly the maximum number is not large enough to represent the number of books (from 0 to 255 only). On the other hand, if we use unsigned long long, the maximum number is too large for our application ($2^{64} - 1$). Since unsigned int is only 4 bytes in size and unsigned long long is 8 bytes in size, we have save 4 bytes of memory size by choosing the unsigned int.



1.0 Standard Data Types

- Similarly, we have to decide which data types to use to represent temperature, voltage, number of stars, our names, our ID number or even the state of ON-OFF. Good choice of data type not only represents the true information but also saves the computer memory.
- C does not have specific logical data type to represent TRUE or FALSE. In a C program, any nonzero value is treated as TRUE and zero value as FALSE.
- Examples on how to declare data types (create variable and allocate memory space) are given below.



1.0 Standard Data Types

```
char huruf = 'B';  
/*reserve one byte memory space identified as huruf and stores the ASCII value of character B.*/  
  
short nilai = 255;  
/*reserve a 2 byte of memory space identified as nilai and stores the value 255.*/  
  
double temperature = 39.6754;  
/*reserve an 8 bytes of memory space identified as temperature and stores the value 39.6754*/
```

- Finally, to know the size of any variable, use the sizeof() operator which returns the integer value giving the number of bytes of the variable size. An example is given below.



1.0 Standard Data Types

```
/* This program is to determine the size of each standard data type.
```

```
Written by: WAWY, FKP, UMP
```

```
19 February 2012 */
```

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

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

```
int main()
```

```
{
```

```
    char a;
```

```
    short b;
```

```
    int c;
```

```
    longlong d;
```

```
    float e;
```

```
    double f;
```

```
    long double g;
```

```
    printf("Integer Standard Data Type\n\n");
```

```
    printf("Size of char:\t\t%d byte\n", sizeof(a));
```

```
    printf("Size of short:\t\t%d bytes\n", sizeof(b));
```

```
    printf("Size of int:\t\t%d bytes\n", sizeof(c));
```

```
    printf("Size of long long:\t\t%d bytes\n", sizeof(d));
```

```
    printf("\nReal Number Standard Data Type\n\n");
```

```
    printf("Size of float:\t\t%d bytes\n", sizeof(e));
```

```
    printf("Size of double:\t\t%d bytes\n", sizeof(f));
```

```
    printf("Size of long double:\t\t%d bytes\n", sizeof(g));
```

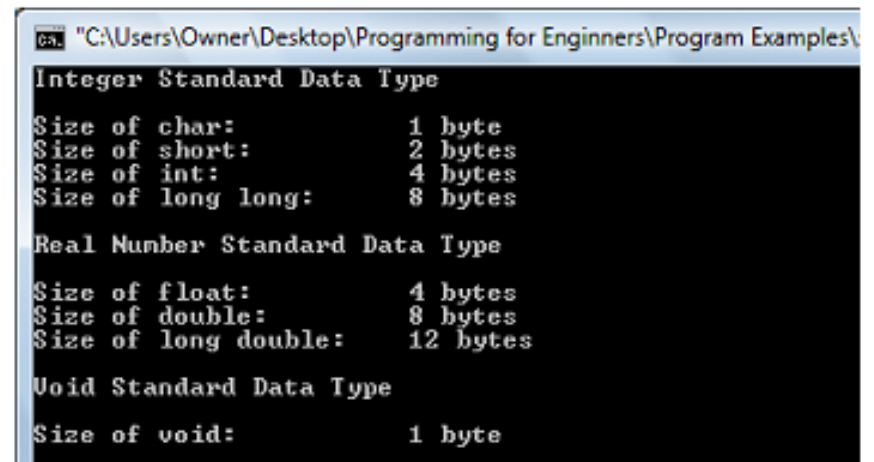
```
    printf("\nVoid Standard Data Type\n\n");
```

```
    printf("Size of void:\t\t%d byte\n", sizeof(void));
```

```
    return 0;
```

```
}
```

- In my PC, the console output is shown below.



```
cmd: "C:\Users\Owner\Desktop\Programming for Engineers\Program Examples\  
Integer Standard Data Type  
Size of char: 1 byte  
Size of short: 2 bytes  
Size of int: 4 bytes  
Size of long long: 8 bytes  
Real Number Standard Data Type  
Size of float: 4 bytes  
Size of double: 8 bytes  
Size of long double: 12 bytes  
Void Standard Data Type  
Size of void: 1 byte
```



2.0 Console Input/Output

- The term console means the keyboard and the monitor. The keyboard is the standard input device while the monitor is the standard output device. The standard C function to input data from the keyboard is the `scanf()` function whereas the standard C function to output data to the monitor is the `printf()` function. Both functions are included in the `stdio.h` header file.
- In using `scanf()` and `printf()` functions, we must specify the format specifier. For examples, integer values are represented by `%d`, the float values are represented by `%f` and the character values are represented by `%c`.
- One important issue in using the `scanf()` function is the address operator (`&`). An example of inputting data from keyboard is given below.



2.0 Console Input/Output

```
int a;  
float b;  
scanf("%d %f", &a, &b);
```

- Notice that the `scanf()` function uses the address of operator (`&`) in its parameters. Failure to include the address operator will result in error.
- One issue related to `printf()` function is the non-printing character. For example, in order to represent the new line we use `\n`, the bell sound we use `\a` and the tab space we use `\t`.



2.0 Console Input/Output

```
int a;  
float b;  
printf("The value of a is %d\tThe value of b is f\n", a, b);
```

- There are many details of using the printf() and scanf() functions. There is no need to memorize them. Just refer to any C programming reference in case we need to use specific features of printf() and scanf() functions.



3.0 Expression

- By expression, we mean the operations on data that reduce to a single value. The common expressions are the addition, subtraction, multiplication and division.
- There are two parts of an expression: the operand and the operators. For example, the expression $(a * b)$, both variables “a” and “b” are operands while the $*$ is the operator (in this case, the multiplication operator).
- There are six types of expressions: primary, postfix, unary, binary, ternary, assignment and comma. The classification is based on how many operands and how many operators are in the expressions.



3.0 Expression

- There are evaluated based upon precedence and associativity. In general, use parenthesis to ensure that the expressions are clearly intended. As an example,

```
x = a + b * c;  
/* The expression of b*c is carried out first before adding a.  
This is because the multiplication operator * is a higher precedence than the + operator.  
The result of the right side of the assignment is stored in the variable x.*/  
x = a + (b*c);  
/* The above expression is now obvious since the parenthesis removes the ambiguity of  
which expression must be carried out first.*/
```

- Several important operators are logical operators. Examples:

```
a && b // a AND b  
a || b // a OR b  
!a    // NOT a
```



3.0 Expression

- Since every expression is evaluated to a single value, it can also represent the logical value. If the value of an expression is nonzero then it represents TRUE. Else, if the value is zero, it represents FALSE.
- One important operator is the modulus operator. It gives the remainder of a division between two integers. An example,

```
int a = 13;  
int b = 5;  
int c;  
c = a % b;  
//the value of c will be 3 which is the remainder of 13/5
```

- The following program illustrates the single values evaluated from logical expressions.



3.0 Expression

```
#include <stdio.h>

/* This program is to illustrates the value of various logic expressions
Written by: WAWY February 2012 */

int main()
{
    int a = 5;
    int b = 0;

    printf("|-----|\n");
    printf("\ta\tb\tExpression\t\tValue\n");
    printf("|-----|\n");
    printf("\t%d\t%d\tNOT a\t\t\t%d\n", a, b, !a);           // Not a
    printf("\t%d\t%d\tNOT b\t\t\t%d\n", a, b, !b);           // Not b
    printf("\t%d\t%d\ta OR b\t\t\t%d\n", a, b, a || b);       // a OR b
    printf("\t%d\t%d\ta AND b\t\t\t%d\n", a, b, a && b);      // a AND b
    printf("\t%d\t%d\ta EQUAL b\t\t\t%d\n", a, b, a == b);     // Is a EQUAL b?
    printf("\t%d\t%d\ta NOT EQUAL b\t\t\t%d\n", a, b, a != b); // Is a NOT EQUAL b?
    printf("\t%d\t%d\ta GREATER b\t\t\t%d\n", a, b, a > b);    // Is a GREATER THAN b?
    printf("\t%d\t%d\ta LESS b\t\t\t%d\n", a, b, a < b);       // Is a LESS THAN b?
    printf("|-----|\n");

    return 0;
}
```

```
*C:\Users\user\Desktop\BHM2013 Programming for Engineers\Program Examples\Exp
```

a	b	Expression	Value
5	0	NOT a	0
5	0	NOT b	1
5	0	a OR b	1
5	0	a AND b	0
5	0	a EQUAL b	0
5	0	a NOT EQUAL b	1
5	0	a GREATER b	1
5	0	a LESS b	0



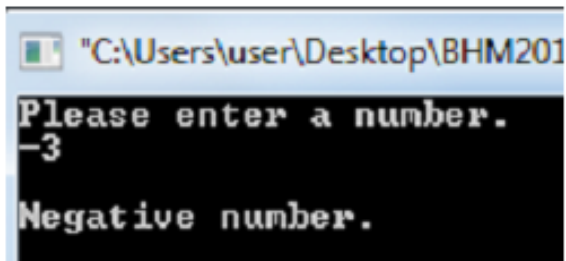
4.0 Selection Statements

- Selection statement is a statement that allows us to choose between different “action” depending on the outcome of the “condition”. If the condition is TRUE (nonzero), then do certain “action”. If the condition is FALSE (zero) than do other “action”.
- The possible actions can either be two-way or multi-way actions. In a two-way action, we only have two choices whereas in a multi-way action, we have more than two choices.
- The two way-action is constructed using the if...else structure. For example, if we want to determine whether a given number is positive (including zero) or negative, we construct the following program.



4.0 Selection Statements

```
#include <stdio.h>
/* This program is to illustrates the two-way action
Written by: WAWY February 2012 */
int main()
{
    int a;
    printf("Please enter a number.\n");
    scanf("%d",&a);
    if (a >= 0)    //This expression reduce to single value
        printf("\nPositive or zero number.\n");
    else printf("\nNegative number.\n");
    return 0;
}
```



```
"C:\Users\user\Desktop\BHM201"
Please enter a number.
-3
Negative number.
```



4.0 Selection Statements

/* This program is to illustrates the multi-way action using "switch ... case" structure

Written by: WAVY February 2012 */

```
int main()
{
    int choice;
    printf("Welcom to UMP\n");
    printf("Please enter your choice.\n");
    printf("1. Faculty of Mechanical Engineering\n");
    printf("2. Faculty of Manufacturing Engineering\n");
    printf("3. Faculty of Electrical Engineering\n");
    printf("4. Faculty of Chemical Engineering\n");
    printf("5. Faculty of Civil Engineering\n");
    scanf("%d",&choice);
    printf("\nYour choice: ");
    switch(choice)
    {
        case 1:
            printf("Faculty of Mechanical Engineering\n"); break;
        case 2:
            printf("Faculty of Manufacturing Engineering\n"); break;
        case 3:
            printf("Faculty of Electrical Engineering\n"); break;
        case 4: printf("Faculty of Chemical Engineering\n"); break;
        case 5: printf("Faculty of Civil Engineering\n"); break;
        default: printf("None\n");
    }
    return 0;
}
```

"C:\Users\user\Desktop\BHM2013 Programming for Engineers\Program Exam

```
Welcom to UMP
Please enter your choice.
1. Faculty of Mechanical Engineering
2. Faculty of Manufacturing Engineering
3. Faculty of Electrical Engineering
4. Faculty of Chemical Engineering
5. Faculty of Civil Engineering
4
Your choice: Faculty of Chemical Engineering
Process returned 0 (0x0)   execution time : 1.778 s
Press any key to continue.
```



5.0 Repetition Statement

- Repetition statement is used to repeat certain action while the condition is TRUE (nonzero). If the condition becomes FALSE (zero), the action stops.
- There are two kinds of looping (repetition),
 1. Pre-test loop. Check the condition first and then carry the repeat action until the condition is FALSE. In this case, we use the for or while loop.
 2. Post-test loop. Do the action first and then check until the condition is FALSE. For this case, we use the do...while loop.
- Suppose that we want user to keep guessing our number until he or she gets the correct number or quit, we can use the following program. This is a pre-test loop.



5.0 Repetition Statement

```
#include <stdio.h>
/* This program is to illustrates the pre-test looping using the while loop structure
Written by: WAWY February 2012 */
int main()
{
    int guess = 0; //Initialize to zero first
    int number = 5; //Our number
    printf("Please guess a number from 0 to 9.\n To quit enter negative number.\n");
    //Keep guessing until your guest is correct or
    //enter negative number to quit
    while ( (guess != number) && (guess >= 0) )
    {
        scanf("%d",&guess);
        printf("Your guess is %d\n",guess);
    }
    printf("Thank you for playing.\n");
    return 0;
}
```

- This is a case whereby the guess is correct.

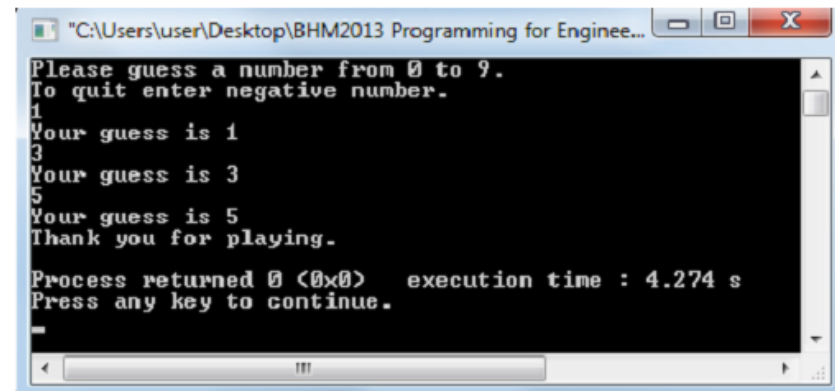
- This is a case whereby the user quit by entering the negative number!



5.0 Repetition Statement

- For the post-test loop, the same program is modified by using the do ... while loop and is given below.

```
#include <stdio.h>
/* This program is to illustrates post-test looping using the do ... while loop structure
Written by: WAWY February 2012 */
int main()
{
    int guess = 0; //Initialize to zero first
    int number = 5; //Our number
    printf("Please guess a number from 0 to 9.\nTo quit enter negative number.\n");
    //Keep guessing until your guest is correct or
    //enter negative number to quit
    do
    {
        scanf("%d",&guess);
        printf("Your guess is %d\n",guess);
    } while ( (guess != number) && (guess >= 0) );
    printf("Thank you for playing.\n");
    return 0;
}
```



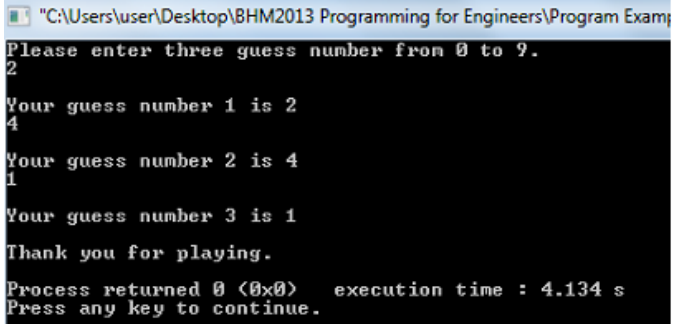
```
"C:\Users\user\Desktop\BHM2013 Programming for Enginee...
Please guess a number from 0 to 9.
To quit enter negative number.
1
Your guess is 1
3
Your guess is 3
5
Your guess is 5
Thank you for playing.
Process returned 0 (0x0) execution time : 4.274 s
Press any key to continue.
```



5.0 Repetition Statement

- If we want a specific number of looping, it is better to use the for loop structure as given by a program example below.

```
#include <stdio.h>
/* This program is to illustrates fixed looping using the for loop structure
Written by: WAWY February 2012 */
int main()
{
    int guess = 0; //Initialize to zero first
    int number = 5; //Our number int i;
    printf("Please enter three guess number from 0 to 9.\n");
    for (i=0;i<3;i++)
    {
        scanf("%d",&guess);
        printf("\nYour guess number %d is %d\n",i+1,guess);
    }
    printf("\nThank you for playing.\n");
    return 0;
}
```



```
"C:\Users\user\Desktop\BHM2013 Programming for Engineers\Program Exam
Please enter three guess number from 0 to 9.
2
Your guess number 1 is 2
4
Your guess number 2 is 4
1
Your guess number 3 is 1
Thank you for playing.
Process returned 0 (0x0)   execution time : 4.134 s
Press any key to continue.
```

- There is a special case called infinite loop whereby there is no action inside the loop that causes the condition to become FALSE. Be careful with infinite loop!



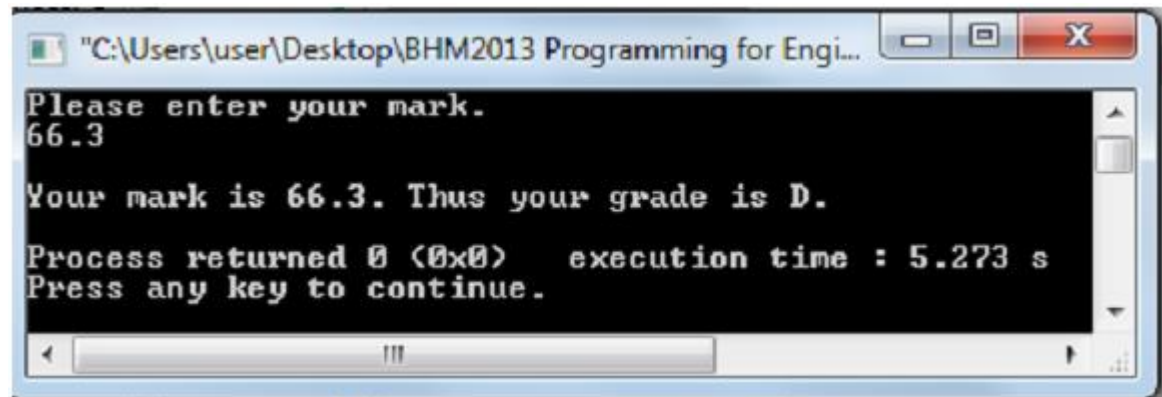
6.0 Function

- Functions are subprogram written to do a specific task. The purpose is to make program manageable and to be able to reuse the same task.
- Function has two parts: function declaration and function implementation. Function declaration consists of three parts: the return value (the output of a function), the parameters (the input of a function) and the name of the function.
- The number of inputs can be more than one but the output (the return value) must be only one.
- Functions can be user-defined or from the C Library. Functions like `printf()` and `scanf()` are from the “`stdio.h`” library.
- An example is given below.



6.0 Function

```
#include <stdio.h>
/* This program is to illustrate the creation of function.
User enter his mark, then the program will display the grade.
WAWY, EKP Page 11 of 11
Written by: WAWY February 2012 */
//Function declaration
float get_mark();
char determine_grade(double mark);
void show_result(double mark, char grade);
int main()
{
    float mark;
    char grade;
    mark = get_mark();           //get mark from user
    grade = determine_grade(mark); //determine grade
    show_result(mark, grade);     //tell the results
    return 0;
}
//Function implementations
float get_mark()
{
    float mark;
    printf("Please enter your mark.\n");
    scanf("%f", &mark);
    return mark;
}
char determine_grade(double mark)
{
    if (mark > 90) return 'A';
    else if (mark > 80) return 'B';
    else if (mark > 70) return 'C';
    else if (mark > 60) return 'D';
    else return 'F';
}
void show_result(double mark, char grade)
{
    printf("\nYour mark is %.1f. Thus your grade is %c.\n", mark, grade);
    return;
}
```



```
"C:\Users\user\Desktop\BHM2013 Programming for Engi...
Please enter your mark.
66.3
Your mark is 66.3. Thus your grade is D.
Process returned 0 (0x0) execution time : 5.273 s
Press any key to continue.
```

