# MICROCONTROLLERS

Chapter 1

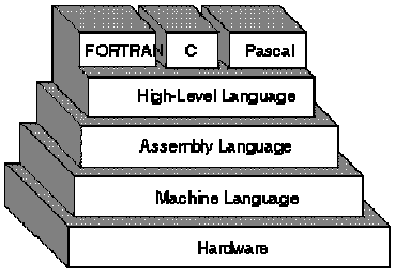
Review on C Programming

Dr. Saeed Ebadollahi

## References:

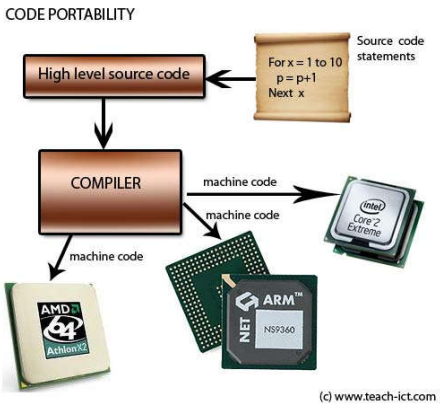

- C Primer Plus – Stephen Prata – 5th edition – Sams
- Programming in C, A complete introduction to the C programming language – Stephen G.Kochan – 3rd edition - Sams

# High and Low Level Programming Languages


# Compiler and Executive Files

Every code have to be compiled and Executed before running. •



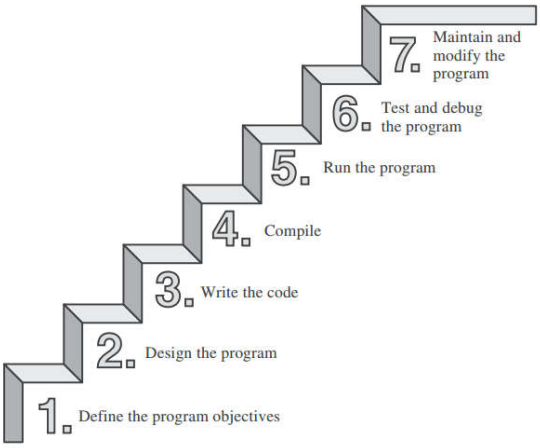
# Why C?

- During the past three decades, C has become one of the most important and popular programming languages. It has grown because people try it and like it. In the past decade, many have moved from C to the more ambitious C++ language, but C is still an important language in its own right, as well a migration path to C++. As you learn C, you will recognize its many virtues.
- But another important note about C is every microcontroller which is available in market have a compiler for C language.

# Whence C?

- Dennis Ritchie of Bell Labs created C in 1972 as he and Ken Thompson worked on designing the Unix operating system. C didn't spring full-grown from Ritchie's head, however. It came from Thompson's B language, which came from… but that's another story. The important point is that C was created as a tool for working programmers, so its chief goal is to be a useful language.
- Most languages aim to be useful, but they often have other concerns. The main goal for Pascal, for instance, was to provide a sound basis for teaching good programming principles. BASIC, on the other hand, was developed to resemble English so that it could be learned easily by students unfamiliar with computers. These are important goals, but they are not always compatible with pragmatic, workaday usefulness. C's development as a language designed for programmers, however, has made it one of the modern-day languages of choice

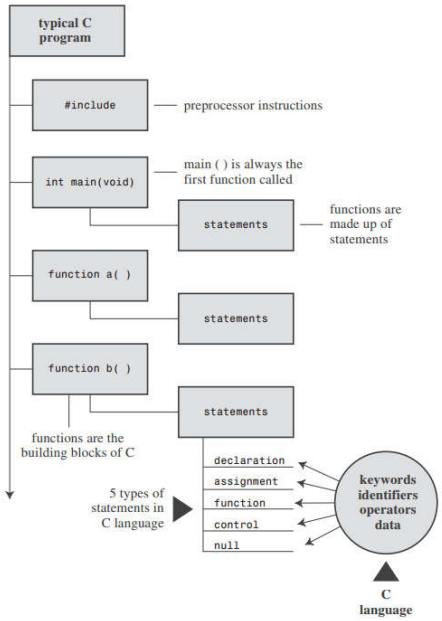
# C steps



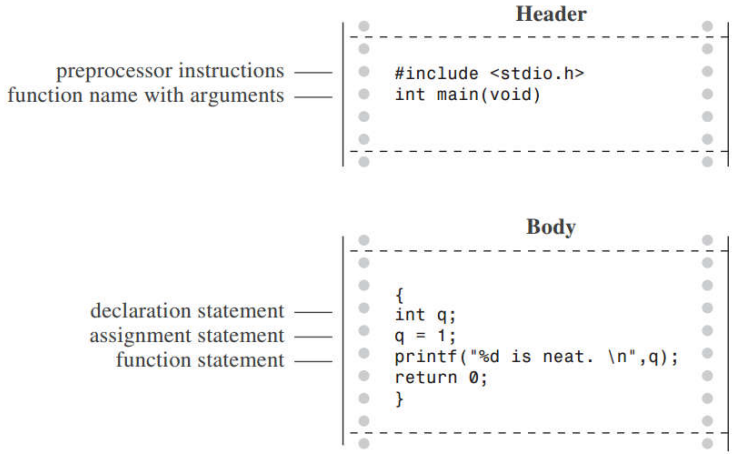

# A Simple Example of C

```
#include <stdio.h>
int main(void)                /* a simple program            */
{
    int num;                  /* define a variable called num */
    num = 1;                  /* assign a value to num        */

    printf("I am a simple "); /* use the printf() function    */
    printf("computer.\n");
    printf("My favorite number is %d because it is first.\n",num);

    return 0;
}
```

# Anatomy of a C program




# Program Structure



# Deceleration

```
int num;
```

- This line from the program is termed a *declaration statement*. The declaration statement is one of C's most important features. This particular example declares two things. First, somewhere in the function, you have a *variable* called num. Second, the int proclaims num as an integer—that is, a number without a decimal point or fractional part. (int is an example of a *data type*.)
- The compiler uses this information to arrange for suitable storage space in memory for the num variable

# Default Types

| Original K&R Keywords |
|---|
| int |
| long |
| short |
| unsigned |
| char |
| float |
| double |

# Printf

**FIGURE 4.6**
Arguments for `printf()`.

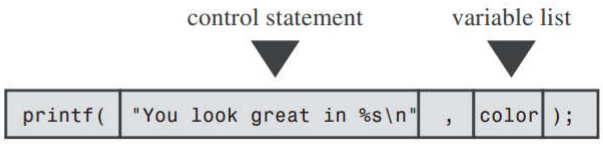
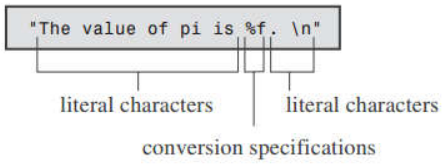



**FIGURE 4.7**
Anatomy of a control string.




# Printf Types

**TABLE 4.3** Conversion Specifiers and the Resulting Printed Output

| Conversion Specification | Output |
|---|---|
| %a | Floating-point number, hexadecimal digits and p-notation (C99). |
| %A | Floating-point number, hexadecimal digits and P-notation (C99). |
| %c | Single character. |
| %d | Signed decimal integer. |
| %e | Floating-point number, e-notation. |
| %E | Floating-point number, e-notation. |

**TABLE 4.3** Continued

| Conversion Specification | Output |
|---|---|
| %f | Floating-point number, decim |
| %g | Use %f or %e, depending on t –4 or greater than or equal to the precision. |
| %G | Use %f or %E, depending on the value. The %E style is used if the exponent is less than –4 or greater than or equal to the precision. |
| %i | Signed decimal integer (same as %d). |
| %o | Unsigned octal integer. |
| %p | A pointer. |
| %s | Character string. |
| %u | Unsigned decimal integer. |
| %x | Unsigned hexadecimal integer, using hex digits 0f. |
| %X | Unsigned hexadecimal integer, using hex digits 0F. |
| %% | Prints a percent sign. |

# Sample Code

**LISTING 4.5** The defines.c Program

```
// defines.c -- uses defined constants from limit.h and float.
#include <stdio.h>
#include <limits.h>     // integer limits
#include <float.h>      // floating-point limits
int main(void)
{
    printf("Some number limits for this system:\n");
    printf("Biggest int: %d\n", INT_MAX);
    printf("Smallest long long: %lld\n", LLONG_MIN);
    printf("One byte = %d bits on this system.\n", CHAR_BIT);
    printf("Largest double: %e\n", DBL_MAX);
    printf("Smallest normal float: %e\n", FLT_MIN);
    printf("float precision = %d digits\n", FLT_DIG);
    printf("float epsilon = %e\n", FLT_EPSILON);

    return 0;
}
```

Here is the sample output:

```
Some number limits for this system:
Biggest int: 2147483647
Smallest long long: -9223372036854775808
One byte = 8 bits on this system.
Largest double: 1.797693e+308
Smallest normal float: 1.175494e-38
float precision = 6 digits
float epsilon = 1.192093e-07
```

# Scanf

**LISTING 4.15** The input.c Program

```
// input.c -- when to use &
#include <stdio.h>
int main(void)
{
    int age;            // variable
    float assets;       // variable
    char pet[30];       // string

    printf("Enter your age, assets, and favorite pet.\n");
    scanf("%d %f", &age, &assets); // use the & here
    scanf("%s", pet);              // no & for char array
    printf("%d $%.2f %s\n", age, assets, pet);

    return 0;
}
```

# Input-Output / Example

Write a program that requests the user's first name and then the user's last name. Have it print the entered names on one line and the number of letters in each name on the following line. Align each letter count with the end of the corresponding name, as in the following:

```
Melissa Honeybee
      7        8
```

Next, have it print the same information, but with the counts aligned with the beginning of each name.

```
Melissa Honeybee
7       8
```

# Loops - while

```
/* summing.c -- sums integers entered interactively */
#include <stdio.h>
int main(void)
{
    long num;
    long sum = 0L;      /* initialize sum to zero   */
    int status;

    printf("Please enter an integer to be summed ");
    printf("(q to quit): ");
    status = scanf("%ld", &num);
    while (status == 1) /* == means "is equal to"   */
    {
        sum = sum + num;
        printf("Please enter next integer (q to quit): ");
        status = scanf("%ld", &num);
    }
    printf("Those integers sum to %ld.\n", sum);

    return 0;
}
```

# Operators

| Operator | Meaning |
|---|---|
| < | Is less than |
| <= | Is less than or equal to |
| == | Is equal to |
| >= | Is greater than or equal to |
| > | Is greater than |
| != | Is not equal to |

# Loops - for

```
// sweetie2.c -- a counting loop using for
#include <stdio.h>
int main(void)
{
    const int NUMBER = 22;
    int count;

    for (count = 1; count <= NUMBER; count++)
        printf("Be my Valentine!\n");

    return 0;
}
```

# Loops / Example1

Use nested loops to produce the following pattern:

```
$
$$
$$$
$$$$
$$$$$
```

# Loops / Example2

Write a program that requests lower and upper integer limits, calculates the sum of all the integer squares from the square of the lower limit to the square of the upper limit, and displays the answer. The program should then continue to prompt for limits and display answers until the user enters an upper limit that is equal to or less than the lower limit. A sample run should look something like this:

```
Enter lower and upper integer limits: 5 9
The sums of the squares from 25 to 81 is 255
Enter next set of limits: 3 25
The sums of the squares from 9 to 625 is 5520
Enter next set of limits: 5 5
Done
```

## Branch - if

```
// colddays.c -- finds percentage of days below freezing
#include <stdio.h>
int main(void)
{
    const int FREEZING = 0;
    float temperature;
    int cold_days = 0;
    int all_days = 0;

    printf("Enter the list of daily low temperatures.\n");
    printf("Use Celsius, and enter q to quit.\n");
    while (scanf("%f", &temperature) == 1)
    {
        all_days++;
        if (temperature < FREEZING)
            cold_days++;
    }
    if (all_days != 0)
        printf("%d days total: %.1f%% were below freezing.\n",
               all_days, 100.0 * (float) cold_days / all_days);
    if (all_days == 0)
        printf("No data entered!\n");

    return 0;
}
```

## Branch - else

```
if (expression)
    statement1
else
    statement2
```

## Branch / Example1

Write a program that reads integers until 0 is entered. After input terminates, the program should report the total number of even integers (excluding the 0) entered, the average value of the even integers, the total number of odd integers entered, and the average value of the odd integers.

## Branch / Example2

Write a program that requests the hours worked in a week and then prints the gross pay, the taxes, and the net pay. Assume the following:

a. Basic pay rate = $10.00/hr

b. Overtime (in excess of 40 hours) = time and a half

c. Tax rate: 15% of the first $300
20% of the next $150
25% of the rest

# Functions

```
/* lethead1.c */
#include <stdio.h>
#define NAME "GIGATHINK, INC."
#define ADDRESS "101 Megabuck Plaza"
#define PLACE "Megapolis, CA 94904"
#define WIDTH 40

void starbar(void);  /* prototype the function */

int main(void)
{
    starbar();
    printf("%s\n", NAME);
    printf("%s\n", ADDRESS);
    printf("%s\n", PLACE);
    starbar();       /* use the function       */

    return 0;
}

void starbar(void)   /* define the function    */
{
    int count;

    for (count = 1; count <= WIDTH; count++)
        putchar('*');
    putchar('\n');
}
```

# Functions - Output

```
****************************************
GIGATHINK, INC.
101 Megabuck Plaza
Megapolis, CA 94904
****************************************
```

# Functions / Example

Write a function that takes three arguments: a character and two integers. The character is to be printed. The first integer specifies the number of times that the character is to be printed on a line, and the second integer specifies the number of lines that are to be printed. Write a program that makes use of this function.

# Arrays

```
int main(void)
{
   float candy[365];      /* array of 365 floats */
   char code[12];         /* array of 12 chars   */
   int states[50];        /* array of 50 ints    */
   ...
}
```

# Arrays / Example

Write a function that returns the difference between the largest and smallest elements of an array-of-`double`. Test the function in a simple program.