Programming Fundamentals for Engineers
0702113

1. Introduction and C Fundamentals

YACOUB SABATIN
MUNTASER ABULAFI
OMAR QARAEEN

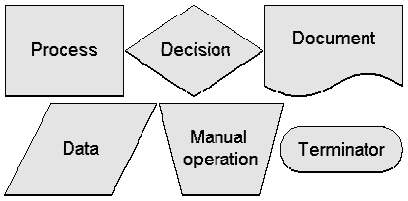
www.e4t.net/c

Designing & Documenting Programs

To understand, analyze a problem statement and develop an algorithm to solve the problem, we use program development techniques (flow-charts, pseudo-code, Algorithms, etc.):

1. **Structured Programming:** Major steps in structured programming include:
   1. Define the problem;
   2. Identify input, output, and processes;
   3. Design algorithm; AND / OR Develop flowchart, pseudo-code, etc.;

4. Code the program -- write the program;
5. Test the program --run it;
6. Debug the program (fix errors if any);
7. Document completed program (final pseudo-code, flowchart, print chart, etc.).



2. Flow Charts:

1. A flow chart is defined as a pictorial representation describing a procedure;
1. Provides people (Developers, programmers, etc.) with a common language or symbolic reference point; (Use: MS Visio)

3

3. An algorithm is a formula or set of steps for solving a particular problem;

1. A clear set of rules; and
2. Have a clear stopping point.
3. Algorithms can be expressed in any language, from natural languages like English or French or Hindi to programming languages like QBASIC, C, etc.
4. **Example**: Add Two numbers, display both numbers and the sum:

1. **START**
2. **Input number1, number2**
3. **Sum = number1 + number2**
4. **Display heading message "Number1, Number2, Sum"**
5. **Display number1, number2, Sum**
6. **End**

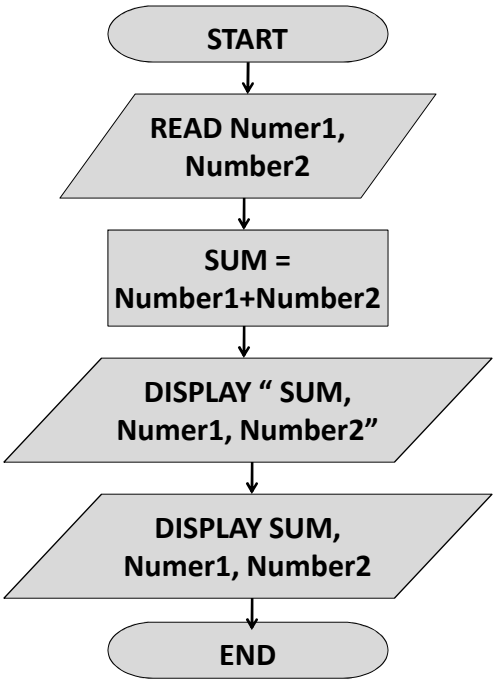
4

5. Pseudo-code:

1. An outline of a program or algorithm, written in a form that can easily be converted into real programming statements;
2. E.g. the pseudo-code for a bubble sort routine might be written:

while not at end of list
compare adjacent elements
if second is greater than first
switch them
get next two elements
if elements were switched
repeat for entire list

5



1. START
2. Input number1, number2
3. Sum=number1+number2
4. Display heading message "SUM, Number1, Number2,"
5. Display SUM, number1, number2
6. End

6

5. **Pseudo-code:** An outline of a program or algorithm, written in a form that can easily be converted into real programming statements;
   1. **Example**: The pseudo-code to add two numbers might be written:

      Start
      Input number1, number2
      Sum = number1 + number2
      Display heading message "Number1, Number2, Sum"
      Display number1, number2, Sum
      End

   2. Can not be compiled nor executed, and there are no real formatting or syntax rules.
6. **Caution:** ***Never write your program before either Algorithming, flowcharting or pseudo-coding it.***

7

Develop a C program to Compute the Volume and Dimensional weight of a box?

1. Analysis:

1) Outputs: Box_Volume, Box_D_Weight;

2) Inputs:

Height, length, width;

3) Processing:

1) Box_Volume = height * length * width;

2) Box_D_Weight = (Box_volume + 165)/166;

8

2. ***Algorithm:***
    1. ***Start;***
    2. ***Input height, length, width;***
    3. ***Compute:***
        1. ***Box_Volume = height * length * width;***
        2. ***Box_D_Weight = (Box_Volume + 165)/166;***
    4. ***Output Box_Volume, Box_D_Weight;***
    5. ***End.***
3. ***DRY RUN The Algorithm;***
4. ***Code the algorithm in C:***

9

```
/*A C program to compute the Volume & Dimensional
  weight Dimensional weight of a box*/
/* STEP 1: START PART */
#include <stdio.h>
main()
{
  int height, length, width, Box_Volume,
  Box_D_Weight;  /*declaring variables*/
/* STEP 2: Reading Data*/
  printf("Enter height of box: ");
  scanf("%d", &height);
  printf("Enter length of box: ");
  scanf("%d", &length);
```

10

```
   printf("Enter width of box: ");
   scanf("%d", &width);
 / * STEP 3: Computation */
   Box_Volume = height * length * width;
   Box_D_Weight = (Box_Volume + 165) / 166;
/* step 4 OUTPUT*/
   printf ("Dimensions: %d x %d x %d \n", length,
   width, height);
   printf ("Volume (cubic inches): %d \n",
   Box_Volume); /* printing expression) */
   printf  ("Dimensional  weight  (pounds):  %d  \n",
   Box_D_Weight);
 /* STEP 5  END*/
   return 0;
}
```

11

The C Language

- Currently, the most commonly-used language
- Very portable: compilers exist for virtually every processor
- Easy-to-understand compilation
- Produces efficient code
- Fairly concise
- C is a **compiled language**

12

Benefits of learning C

- You will be able to read and write code for a large number of platforms -- everything from microcontrollers to the most advanced scientific systems can be written in C, and many modern operating systems are written in C.
- The jump to the object oriented C++ language becomes much easier. C++ is an extension of C, and it is nearly impossible to learn C++ without learning C first.

13

C History

- Developed between 1969 and 1973 along with Unix
- Due mostly to Dennis Ritchie
- Designed for systems programming
  - Operating systems
  - Utility programs
  - Compilers
  - Filters

14

- Original machine (DEC PDP-11) was very small
  - 24K bytes of memory, 12K used for operating system
- Written when computers were big, capital equipment
  - Group would get one, develop new language, OS

15

‘Hello World’ in C

```
#include <stdio.h>
main()
{
   printf("Hello, world!\n");
   return 0;
}
```

#include ➔ **C preprocessor command that lets user includes the code from another file (such as a library)**

stdio.h ➔ **standard input/output library**

#include <stdio.h> ➔ **allows you to use code from the standard input/output library that contains *printf* function**

16

- *main* ➔ Every program must have a *function* named main which contains statements enclosed in braces { and }
- *printf* ➔ A function from the standard I/O library that prints a string (f stands for formatted)
- *\n* ➔ A new line character. The \n inside the string indicates a new line
- Every *function* must return a value. *Main* is a *function* and returns a value taken by the operating system as the program terminates
- *return 0* ➔ Used to signal OS that the program terminated normally

17

Compiling & Linking

- *Programs in C must be:*

1. Preprocessor ➔ Handles all commands that begin with # (directives);
2. Compiled ➔ Compiler translates the source code (program) into machine code producing .obj file;
3. Linked – the linker combines the obj file with additional necessary code from library functions (like *printf*) to produce final executable file.

18

General C program!

```
# directives
main()
{
  statements
}
```

- **Statements** ➔ **Commands that are carried out as program runs;**
  - ***printf* is a *call* statement displaying *string* on screen*:***
    - ***printf ("hello"); /* These are comments */***
    - ***printf ("world! \n"); /* Can go any where in a C */***
- **Statements end with a *semicolon (;)***

19

Some definitions

"Hello, World!\n"

- **Character** ➔ A single letter, number, punctuation mark, etc.
- **String** ➔ Series of characters grouped together;
- Some special kinds of *characters* exist:
  - *\t* – tab character;
  - *\n* – new line character.

20

Important Basic Rules for C

- Statements of code end with a **semicolon ;**
- **Strings** are enclosed inside **double-quotes “ ”**
- **Functions** start and end with curly brackets **{ }**
- White space (Blank Space) doesn’t matter;
- C language is ***case sensitive***
- Some names are ***keywords*** (such as ***int*** and ***float***) and can’t be used as ***identifiers***

21

Question:

- Try writing a short C program to display a 5x5 square of asterisks.

22

Types & Variables

- Most programming languages perform calculations; thus **need** a way to **store data temporarily** during program execution
- These **storage locations** are called ***variables.***
- Variable names are ***identifiers*** that contain letters, digits and underscores and must begin with a letter or underscore
- **C** has various data types such as:
  - ***int*** ➔ Integer type, such as 45,
  - ***char*** ➔ Character type, such as 'a', and
  - ***float*** ➔ Floating-point type, such as 3.14159;
  - ***bool*** ➔ Boolean type; ***true*** or ***false***.

23

Declaration of Variables

- ***Variables*** ➔ Must be declared before they are used;
- **Declaration** ➔ Precedes *statements* in the main program;
- Have to declare the type of variables *(int or float)* before declarations

```
main()
{
        declaration & assignments
        statements
}
```

24

Variable Declaration Examples

```
int hours_worked;
float hourly_rate, gross_pay, net_pay;
```

Assignments Example

Variables can be given a value by means of assignments

```
height = 12;
width = 8;
length = 1;
volume = height * width * length;
```

25

Printing a Variable Value

- Use:
  - ***printf*** function to print the value of a *variable*
  - ***%d*** is a placeholder that works for ***int***
  - ***%f*** is a placeholder that works for ***float***
  - **char** (single character values, discussed later) uses ***%c***
  - **character strings** (arrays of characters, discussed later) use ***%s***
  - ***%.2f*** *displays a float value of 2 digits after the decimal points (34.21);*

```
printf ("Hours worked: %d \n", hours_worked);
printf ("Hourly Rate: $%.2f \n", hourly_rate);
```

%d and %.2 are placeholders indicating where the value of *hours_worked* & *hourly_rate* to be filled during printing

26

Example on Printing Variables

- To print out the line: ***Profit: $2267.14***

```
printf ("Profit: $%.2f \n", profit);
```

- To print:

```
printf ("Height: %d  Length: %d \n",
  height, length);
```

- *%f* by default displays 6 digits after the decimal point
- *%.nf* – forces *%f* to display **n** digits after the decimal point

27

Develop a C program to Compute the Volume and Dimensional weight (see textbook p18) of a 12"x10"x8" box?

1. ***Analysis:***
   1. ***Outputs: Box_Volume, Box_D_Weight;***
   2. ***Inputs: Known data:***
      1. ***height = 8;***
      2. ***length = 12;***
      3. ***width = 10;***
   3. ***Processing:***
      1. ***Box_Volume = height * length * width;***
      2. ***Box_D_Weight = (volume + 165) / 166;***

28

2. *Algorithm:*
   1. *Start;*
   2. *Assign:*
      1. *height = 8;*
      2. *length = 12;*
      3. *width = 10;*
   3. *Compute:*
      1. *Box_Volume = height * length * width;*
      2. *Box_D_Weight = (Box_volume + 165) / 166;*
   4. *Output Box_Volume, Box_D_Weight;*
   5. *End.*
3. *RUN The Algorithm;*
4. *Code the algorithm in C:*

29

```
/*A C program to compute the Volume &
  Dimensional weight Dimensional weight of a
  12"x10"x8" box */
/* STEP 1: START PART */
#include <stdio.h>
main()
{
  int height, length, width, Box_Volume,
  Box_D_Weight;  /*declaring variables*/
  /* STEP 2: Assingment */
  height = 8;    /* assigning values to
                    variables */
  length = 12;
  width = 10;
```

30

```
 / * STEP 3: Computation */
 Box_Volume = height * length * width;
 Box_D_Weight = (Box_volume + 165) / 166;
  //answer will be 'truncated'
  /* step 4 OUTPUT */
  printf ("Dimensions: %d x %d x %d \n",
           length, width, height);
  printf ("Volume (cubic inches): %d \n",
           Box_Volume); // printing expression)
  printf ("Dimensional weight (pounds): %d \n",
           Box_D_Weight);
  /* STEP 5  END*/
  return 0;
}
```

31

Reading a user Input

- To obtain an input from users: Use ***scanf*** function
- ***scanf*** and ***printf*** requires a ***format string*** with which we specify the format these functions will take
- **To read an:**
  - ***int*** : Use **"%d"**
    - ➔ `scanf ("%d", &hours_worked);`
  - ***float*** : Use **"%f"**
    - ➔ `scanf ("%f",&hourly_rate);`
- ***"%d"*** reads an **integer value** and **stores** it into the **integer variable** ***hours_worked;***

32

Develop a C program to Compute the Volume and Dimensional weight of a box?

1. ***Analysis:***
   1. ***Outputs: Box_Volume, Box_D_Weight;***
   2. ***Inputs:***
      1. ***Height, length, width;***
   3. ***Processing:***
      1. ***Box_Volume = height * length * width;***
      2. ***Box_D_Weight = (Box_Volume + 165) / 166;***

33

2. ***Algorithm:***
   1. ***Start;***
   2. ***Assign:***
      1. ***Input height, length, width;***
   3. ***Compute:***
      1. ***Box_Volume = height * length * width;***
      2. ***Box_D_Weight = (Box_volume + 165) / 166;***
   4. ***Output Box_Volume, Box_D_Weight;***
   5. ***End.***
3. ***RUN The Algorithm;***
4. ***Code the algorithm in C:***

34

```
/* A C program to compute the Volume &
   Dimensional weight Dimensional weight
   of a box */
/* STEP 1: START PART */
#include <stdio.h>
main()
{
  int height, length, width, Box_Volume,
  Box_D_Weight;  /* declaring variables */
  /* STEP 2: Reading Data*/
  printf("Enter height of box: ");
  scanf("%d", &height);
  printf("Enter length of box: ");
  scanf("%d", &length);
```

35

```
  printf("Enter width of box: ");
  scanf("%d", &width);
  /* STEP 3: Computation */
  Box_Volume = height * length * width;
  weight = (volume + 165) / 166;
  /* STEP 4 OUTPUT */
  printf ("Dimensions: %d x %d x %d \n",
           length, width, height);
  printf ("Volume (cubic inches): %d \n",
           Box_Volume);
  printf ("Dimensional weight (pounds): %d
           \n", Box_D_Weight);
  /* STEP 5  END*/
  return 0;
}
```

36

Question:

- Try writing a C program that takes 3 integer values, calculates and prints their summation and average.

37

Macro Definitions

- **Constants are:**
  - Values that do not change;
  - Defined in C using ***macro definitions:***
    #define label constant
    #define DAYS_PER_WEEK 7
  - ***#define*** ➔ Preprocessor directive similar to ***#include***
  - ***E*very** time compiler sees ***DAYS_PER_WEEK*** the numeric value *7* is substituted;
  - ***macro definition,*** by convention: ➔ Uses UPER-CASE letters
- **No semicolon** at end of ***#define***
- **Constants are preferable over literals.**

38

Develop a C program that Converts a Fahrenheit temperature value to a Celsius one using a macro definition?

Solution:

1) Analysis (Define: Output(s), Input(s), Processes;
2) Develop the Algorithm;
3) Dry RUN the algorithm;
4) Convert the algorithm to C code:

>>

39

```
/* A C program that Converts a Fahrenheit temperature value to a
   Celsius one */

#include <stdio.h>

#define FREEZING_PT 32.0   /* define label FREEZING_PT to
   represent constant 32 */
#define SCALE_FACTOR (5.0 / 9.0)
/* compiler would put the whole expression in the constant name
   place below, 5/9 will lead to 0 (truncation) */
int main()
{
   float fahrenheit, celsius;
   printf("Enter Fahrenheit temperature: ");
   scanf("%f", &fahrenheit);
   celsius = (fahrenheit - FREEZING_PT) *
             SCALE_FACTOR;
   printf("Celsius equivalent: %.1f\n",celsius);
   return 0;
}
```

40

Revision, examples, and more

41

printf/scanf

- You need to include the header file **`stdio.h`** to be able to use them.
- **`Scanf`** scans user input and match them with the formatted string.
- **Example (Q6 p43):**

  Books are identified by an international standard book number (ISBN) such as 0-393-30375-6 the first part represents the language (0 for English 3 for German..etc) the next group represents the publisher, the third represents the book, and the last digit is a check digit, write a C programme to break an ISBN entered by the user.

42

```
#include <stdio.h>
main()
{
   int language, publisher, book_number, check_digit;
   printf("Enter ISBN: ");
   scanf("%d-%d-%d-%d", &language, &publisher,
     &book_number, &check_digit);
   printf("Language: %d\n", language);
   printf("Publisher: %d\n", publisher);
   printf("Book number: %d\n", book_number);
   printf("Check digit: %d\n", check_digit);
   /* The four printf calls can be combined as
     follows:
   printf("Language: %d\nPublisher: %d\nBook number:
     %d\nCheck digit: %d\n",
   language, publisher, book_number, check_digit);
   */
   return 0;
}
```

43

Place holders/string formatting

- **%d** – displays integer in decimal form (base 10).
- **%e** – displays a floating point number in exponential form.

  Ex:

  printf("The number is %15.3e\n", 123.45678); Will show “The number is 1.235e+002”

  Note the reserved 15 places reserved for the whole number, and the 3 digits shown to the right of the point.
- **%f** – for floating points.
- **%g** – displays a floating-point number in exponential format or fixed decimal format.

44

More on placeholders

- %nd ➔ n spaces reserved
- %.nd ➔ n leading zeros OR d decimal digits
- %% for % character
- %e ➔ floating point in exponential (scientific) format, ex. 3.912e+1
- %e ➔ same with capital e, ex. 3.912E+1
- %g ➔ the shortest of %e and %f
- %G ➔ same with capital (e if any)
- %d or %i ➔ signed decimal integer
- %u ➔ unsigned decimal integer
- %p ➔ pointer (address) in hex, more about pointers later
- %x ➔ unsigned hex integer, ex. 37fa
- %X ➔ same with capitals, ex. 37FA
- %o ➔ octal

45

Another example:

- Q6 p83:

  Write a program that asks the user for a 24-hour time, then displays it in 12-hour form, be careful not to display 12:00 as 0:00, Ex:

  Enter a 24-hour time: 21:11

  Equivalent 12-hour time: 9:11 PM

46

```
#include <stdio.h>
main()
{
   int hours, minutes;
   printf("Enter a 24-hour time: ");
   scanf("%d:%d", &hours, &minutes);
   /* assumption: midnight entered as 00:00, valid times
      are 00:00-23:59 */
   printf("Equivalent 12-hour time: ");
   if (hours == 0)
      printf("12:%.2d AM\n", minutes);
   else if (hours < 12)
      printf("%d:%.2d AM\n", hours, minutes);
      /* We used "%d:%.2d" instead of "%d:%2d to get 9:09
         AM for something like 9:9 instead of 9: 9 AM */
   else if (hours == 12)
      printf("%d:%.2d PM\n", hours, minutes);
   else
      printf("%d:%.2d PM\n", hours % 12, minutes);
   return 0;
}
```

47

Boolean variables

- We used to evaluate logical expressions as integers.
- Usually you can also define boolean variables using `bool` keyword.
- Ex:

```
#include <stdio.h>
int main()
{
  bool even;
  int number=99;
  if (number%2==0)
   even=true;
  else
   even=false;
  if (even==true)
   printf("The number is even\n");
  else
   printf("The number is NOT even\n");
  return 0;
}
```

- Can you think of shorter implementation?

48

Mathematical Notes:

- math.h
  - sqrt() function.

```
int x=4;
printf("The square root of %d is %f.\n", x, sqrt(x));
```

  - tan/cos/sin
    - Angles in radians.
  - Power.

49

- Example:

```
#include <stdio.h>
#include <math.h>
main()
{
  int x=4;
  printf("The square root of %d is %f.\n",
  x, sqrt(x));
  float ang=3.14/2; //Angles should be in
  radians
  printf("The sine of %f is %f.\n", ang,
  sin(ang));
  return 0;
}
```

50

Questions:

- Write a C program to calculate the 2 roots of a quadratic equation.
- Develop a concise C program to convert an area from m^2 to $inch^2$

51

- **Exercises:**

1) Code the algorithm/flowchart shown on slide 6 (sum of 2 numbers);

2) Develop a **C** program to **Calculate** the **Net Pay** of a worker. The program should **accept the number of worked hours** and **the hourly rate** from the user**?**

Hint: A *sample run of the program would be:*

```
            Simple Payroll Program:
            -------------------------------
    Enter the number of hours worked:  21
    Enter the hourly rate of pay:  12.5
    Hours Worked:              21
    Hourly Rate:          $12.50
    Gross Pay:                 $262.50
    FICA Deduction:            $20.08
    Net Pay:                   $242.42
```

52

Solved Exercise 1
Calculating Net Pay

Here is a sample run of the program:

```
Simple Payroll Program:
-------------------------------

Enter the number of hours worked:  21
Enter the hourly rate of pay:  12.5


Hours Worked:          21
Hourly Rate:           $12.50
Gross Pay:             $262.50
FICA Deduction:        $20.08
Net Pay:               $242.42
```

53