Programming and Data Structure

Sujoy Ghose

Sudeshna Sarkar

Jayanta Mukhopadhyay

Dept. of Computer Science & Engineering.

Indian Institute of Technology

Kharagpur

Our First Program:
Convert Centigrade to Fahrenheit

```
#include <stdio.h>
// this program takes a centigrade value as input and converts it into fahrenheit
int main( )  {
        float C;
        float F;
        printf (“Input in centigrades\n”) ;
        scanf (“%f”, &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %f\n”, F) ;
        return 0;
}
```

A variation: intCtoF.c

```
#include <stdio.h>
// this program takes a centigrade value as input and converts it into fahrenheit
int main( )  {
        int C;
        int F;
        printf (“Input in centigrades\n”) ;
        scanf (“%d”, &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %d\n”, F) ;
        return 0;
}
```

Outputs

```
#include <stdio.h>
// this program takes a centigrade value as
// input and converts it into fahrenheit
int main( ) {
        int C;
        int F;
        printf ("Input in centigrades\n") ;
        scanf ("%d", &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %d\n", F) ;
        return 0;
}
```

Input Centigrade	4	14	27
FtoC	39.2	57.2	80.6
intFtoC			

Outputs

```
// intFtoC

#include <stdio.h>
// this program takes a centigrade value as
// input and converts it into fahrenheit
int main( ) {
        int C;
        int F;
        printf ("Input in centigrades\n") ;
        scanf ("%d", &C);
        F=C*9/5+32;
        printf (Fahrenheit value = %d\n", F) ;
        return 0;
}
```

Input Centigrade	4	14	27
FtoC	39.2	57.2	80.6
intFtoC	39	57	80

Outputs

```
// intFtoC2

#include <stdio.h>
// this program takes a centigrade value as
// input and converts it into fahrenheit
int main( ) {
        int C;
        int F;
        printf ("Input in centigrades\n") ;
        scanf ("%d", &C);
        F=C/5*9+32;
        printf (Fahrenheit value = %d\n", F) ;
        return 0;
}
```

Input Centigrade	4	14	27
FtoC	39.2	57.2	80.6
intFtoC	39	57	80
intFtoC2			

Outputs

```
// intFtoC2

#include <stdio.h>
// this program takes a centigrade value as
// input and converts it into fahrenheit
int main( ) {
        int C;
        int F;
        printf ("Input in centigrades\n") ;
        scanf ("%d", &C);
        F=C/5*9+32;
        printf (Fahrenheit value = %d\n", F) ;
        return 0;
}
```

Input Centigrade	4	14	27
FtoC	39.2	57.2	80.6
intFtoC	39	57	80
intFtoC2	0	50	77

Find the roots of a quadratic equation

$$Ax^2 + Bx + C = 0$$

Solving a Quadratic Equation

```
#include  <stdio.h>
#include  <math.h>
// solves Ax^2 + Bx + C = 0
int main(  )  {
    float A,B,C;
    float root1,root2,disc;
    printf (" Input A B C \n");
    scanf ("%f%f%f", &A, &B, &C) ;
    disc = B*B – 4*A*C;
    disc = sqrt(disc);
    root1 = (–B+disc)/(2*A);
    root2 = (–B-disc)/(2*A);
   printf ("root 1= %f, root2=%f\n",root1, root2);
    return 0;
}
```

Solving a Quadratic Equation

```
#include  <stdio.h>
#include  <math.h>
// solves Ax^2 + Bx + C = 0
int main(  )  {
    float A,B,C;
    float root1,root2,disc;
    printf (“ Input A B C \n“);
    scanf (“%f%f%f”, &A, &B, &C) ;
    disc = B*B-4*A*C;
    disc = sqrt(disc);
    root1 = (-B+disc)/(2*A);
    root2 = (-B-disc)/(2*A);
   printf ("root 1= %f, root2=%f\n“,root1, root2);
    return 0;
}
```

Solving a Quadratic Equation

```
#include  <stdio.h>
#include  <math.h>
// solves Ax^2 + Bx + C = 0
int main(  )  {
    float A,B,C;
    float root1, root2, disc;
    printf (“ Input A B C \n“);
    scanf (“%f%f%f”, &A, &B, &C) ;
    disc = B*B - 4*A*C;
    disc = sqrt(disc);
    root1 = (-B+disc)/(2*A);
    root2 = (-B-disc)/(2*A);
   printf ("root 1= %f, root2=%f\n“,root1, root2);
    return 0;
}
```

Solving a Quadratic Equation

```
#include  <stdio.h>
#include  <math.h>
// solves Ax^2 + Bx + C = 0
int main(  )  {
    float A,B,C;
    float root1, root2, disc;
    printf (“ Input A B C \n“);
    scanf (“%f%f%f”, &A, &B, &C) ;
    disc = B*B - 4*A*C;
    disc = sqrt(disc);
    root1 = (-B+disc)/(2*A);
    root2 = (-B-disc)/(2*A);
   printf ("root 1= %f, root2=%f\n“,root1, root2);
    return 0;
}
```

Solving a Quadratic Equation

```
#include  <stdio.h>
#include  <math.h>
// solves Ax^2 + Bx + C = 0
int main(  )  {
    float A,B,C;
    float root1, root2, disc;
    printf (“ Input A B C \n“);
    scanf (“%f%f%f”, &A, &B, &C) ;
    disc = B*B - 4*A*C;
    disc = sqrt(disc);
    root1 = (-B+disc)/(2*A);
    root2 = (-B-disc)/(2*A);
    printf ("root 1= %f, root2=%f\n“, root1, root2);
    return 0;
}
```

Solving a Quadratic Equation

```
#include  <stdio.h>
#include  <math.h>
// solves Ax^2 + Bx + C = 0
int main(  )  {
    float A,B,C;
    float root1, root2, disc;
    printf (“ Input A B C \n“);
    scanf (“%f%f%f”, &A, &B, &C) ;
    disc = B*B - 4*A*C;
    disc = sqrt(disc);
    root1 = (-B+disc)/(2*A);
    root2 = (-B-disc)/(2*A);
    printf ("root 1= %f, root2
        =%f\n“, root1, root2);
    return 0;
}
```

Compile and run this program.

- gcc quad.c
- ./a.out

Input A B C
1 3 2
root1=-1, root2=-2

- ./a.out

Input A B C
1 1 1
root1=nan, root2=nan

Solving a Quadratic Equation

```
#include  <stdio.h>
#include  <math.h>
// solves Ax^2 + Bx + C = 0
int main(  )  {
    float A,B,C;
    float root1, root2, disc;
    printf (“ Input A B C \n“);
    scanf (“%f%f%f”, &A, &B, &C) ;
    disc = B*B - 4*A*C;
    disc = sqrt(disc);
    root1 = (-B+disc)/(2*A);
    root2 = (-B-disc)/(2*A);
    printf ("root 1= %f, root2
        =%f\n“, root1, root2);
    return 0;
}
```

Compile and run this program.

- gcc quad.c
- ./a.out

Input A B C

1 3 2

root1=-1, root2=-2

- ./a.out

Input A B C

1 1 1

root1=nan, root2=nan

- Why has this happened?
- This happened because $B^2 - 4AC = 1 - 3 = -2$.
- The square root operation returns a special value, Not-A-Number (NAN).
- What is to be done?

new_sqrt.c

```
#include <stdio.h>
#include <math.h>
// general square roots
int main( ) {
    float r, x, y;
    printf (“Enter a real number.\n“);
    scanf (“%f”, &r) ;
    if (r<0)  {
        r = –r;
        x = sqrt(r);
       printf (“%f  i \n“, x) ;
       return 0;
    }
    x=sqrt(r);
    printf (“%f  \n“, x) ;
    return 0;
}
```

if r is negative then we take the sqrt of its negative and print it with an i.
The program then exits via return.
if r >= 0 then execution proceeds the normal way.

new2_sqrt.c

```
#include <stdio.h>
#include <math.h>
// general square root
int main( ) {
     float r, x, y;
     printf (“Enter a real number.\n“);
     scanf (“%f”, &r) ;
     if (r<0)  {
          r = -r;
          x = sqrt(r);
          printf (“%f  i \n“, x) ;
     }
     else  {
          x=sqrt(r);
          printf (“%f  \n“, x) ;
     }
     return 0;
}
```

- **Program Structure**:
  - Some **include** commands
  - **int main() {**
    code block
    }
- **Variables**: Memory registers may have names. These must be words beginning with a non-numeral.
- **Declarations**: Every variable must be declared to be of a certain type such as **int, float**. Operations must respect this type.
- **Input and Output** is enabled through **printf, scanf**. Variable contents and strings may be manipulated in-order.
- **Assignments** are done by

  var = expression;

What is C?

- Developed by Dennis Ritchie – AT&T Bell Laboratories – 1972 for use with the Unix operating system.
  (The origin of C is closely tied to the development of the Unix operating system, originally implemented in assembly language on a PDP-7)
- Widely used today
  - extends to newer system architectures
  - efficiency/performance
  - low-level access
- C features:
  - Few keywords
  - Structures, unions – compound data types
  - Pointers – memory, arrays
  - External standard library – I/O, other facilities
  - Compiles to native code
  - Macro preprocessor

The C Programming Language

- C is a fast, small, general-purpose, structured programming language.
- C can be used for applications programming as well as for systems programming (*e.g., compilers and interpreters, operating systems, database systems, microcontrollers etc.)*
- There are only 32 keywords and its strength lies in its built-in functions.
- C is highly portable, since it relegated much computer-dependent features to its library functions.
- "C is quirky, flawed, and an enormous success."–Ritchie

Versions of C

- Evolved over the years:
  - 1972 – C invented
  - 1978 – *The C Programming Language published; first specification of language*
  - 1989 – C89 standard (known as ANSI C or Standard C)
  - 1990 – ANSI C adopted by ISO, known as C901999 – C99 standard
    - mostly backward-compatible
    - not completely implemented in many compilers
  - 2007 – work on new C standard C1X (now called C11)
- In this course: ANSI/ISO C (C89/C90)

Structure of a .c file

/* Begin with comments about file contents */

Insert **#include** statements and preprocessor definitions

Function prototypes and variable declarations

Define **main()** function

{

Function body

}

Define other function

{

Function body

}

:

Comments

Comments: **/* this is a simple comment */**

- Can span multiple lines

```
/* convert.c : A C program
    than converts from fahrenheit
     to centigrade * /
```

- Completely ignored by compiler
- Can appear almost anywhere
- Alternative way to show comments

```
// This function computes the square root
// of a floating point number
```

The #include macro

- Header files: constants, functions, other declarations
- **#include <stdio.h>** – read the contents of the *header file* stdio.h
- **stdio.h**: standard I/O functions for console, files : Part of the standard C library
- other important header files: ctype.h, math.h, stdlib.h, string.h, time.h

Declaring variables

- Must declare variables before use
- Variable declaration:

  int number;

  float temperature;

- **int** - integer data type
- **float** - floating-point data type
- Many other types

Functions

- Every C program consists of one or more functions.
  - One of the functions must be called *main*.
  - The program will always begin by executing the main function.
- Each function must contain:
  - A function *heading*, which consists of the function *name*, followed by an optional list of *arguments* enclosed in parentheses.
  - A list of argument *declarations*.
  - A *compound statement*, which comprises the remainder of the function.

Contd.

- Each compound statement is enclosed within a pair of braces: ‘{‘ and ‘}’
  - The braces may contain combinations of elementary statements and other compound statements.
- Comments may appear anywhere in a program, enclosed within delimiters ‘/*’ and ‘*/’.
  - Example:

```
a = b + c;    /* ADD TWO NUMBERS */
```

Function Prototypes

- General form:

  return_type function_name(arg1,arg2,...);
- Arguments: local variables, values passed from caller
- Return value: single value returned to caller when function exits
- **void** – signifies no return value/arguments

  int rand (void);

The main () function

- main() : entry point for C program
- Simplest version: no inputs, outputs 0 when successful, and nonzero to signal some error
  int main (void) ;
- Two-argument form of main(): access command-line arguments
  int main(int argc, char **argv);
- Will be discussed later in the course

Sample C program 3

Header file includes functions for input/output

```
#include <stdio.h>

int main()
   {
      printf ("\n Our first look at a C program \n");
      return 0;
   }
```

Main function is executed when you run the program. (Later we will see how to pass its parameters)

Curly braces within which statements are executed one after another.

Statement for printing the sentence within double quotes (".."). '\n' denotes end of line.

Sample C program 4

```
#include <stdio.h>
int main()
 {
       int   a, b, c;
       a = 10;
       b = 20;
       c = a + b;
       printf ("\n The sum of %d and %d is %d\n", a,b,c);
       return 0;
 }
```

Integers variables declared before their usage.

Control character for printing value of a in decimal digits.

The sum of 10 and 20 is 30

Sample C program 5

```
#include <stdio.h>
/* FIND THE LARGEST OF THREE NUMBERS */
int main()
 {
       int   a, b, c;
       scanf ("%d %d %d", &a, &b, &c);
       if  ((a>b) && (a>c))    /* Composite condition check */
          printf ("\n Largest is %d", a);
       else  {
          if  (b>c)          /* Simple condition check */
               printf ("\n Largest is %d", b);
          else
               printf ("\n Largest is %d", c);
      }
      return 0;
 }
```

Input statement for reading three variables from the keyboard

Conditional statement

if statement

The if statement is used as:

```
prev_line
if (condn)
{
     code block1
}
else
{
    code block2
}
next_line
```

if statement

The if statement is used as:

```
prev_line
if (condn)
{
     code block1
}
else
{
    code block2
}
next_line
```

If condn evaluates to true then the sequence is:

```
prev_line
code block1
next_line;
```

if statement

The if statement is used as:

```
prev_line
if (condn)
{
     code block1
}
else
{
    code block2
}
next_line
```

If condn evaluates to true then the sequence is:

```
prev_line
code block1
next_line;
```

otherwise it is:

```
prev_line
code block2
next_line;
```

Sample C program 6

Preprocessor statement. Replace PI by 3.1415926 before compilation.

Example of a function called as per need from main programme.

```
#include <stdio.h>
#define   PI   3.1415926

/* Compute the area of a circle */
int main()
   {
       float   radius, area;
       float   myfunc (float radius);

       scanf ("%f", &radius);
       area = myfunc (radius);
       printf ("\n Area is %f \n", area);
       return 0;
   }
```

```
float   myfunc (float r)
 {
       float   a;
       a = PI * r * r;
       return (a);  /* return result */
 }
```

Function called.

main() is also a function

```
#include <stdio.h>
int main( )
{
        int   a, b, c;

        a = 10;
        b = 20;
        c = a + b;
        printf (“\n The sum of %d and %d is %d\n”,
                a,b,c);
        return 0;
   }
```

Desirable Programming Style

- Clarity
  - The program should be clearly written.
  - It should be easy to follow the program logic.
- Meaningful variable names
  - Make variable/constant names meaningful to enhance program clarity.
    - ‘area’ instead of ‘a’
    - ‘radius’ instead of ‘r’
- Program documentation
  - Insert comments in the program to make it easy to understand.
  - But never use too many comments.

Contd.

- Program indentation
  - Use proper indentation.
  - Structure of the program should be immediately visible.

Indentation Example #1 :: Bad Style

```
#include <stdio.h>
#define    PI    3.1415926
/* Compute the area of a circle */
int main ( )
{
float   radius, area;
float   myfunc (float radius);
scanf ("%f", &radius);
area = myfunc (radius);
printf ("\n Area is %f \n", area);
return 0;
}
```

```
float   myfunc (float r)
{
float   a;
a = PI * r * r;
return (a);      /* return result */
}
```

Indentation Example #1 :: Good Style

```
#include <stdio.h>
#define    PI    3.1415926
/* Compute the area of a circle */

int main()
 {
        float   radius, area;
        float   myfunc (float radius);

        scanf ("%f", &radius);
        area = myfunc (radius);
        printf ("\n Area is %f \n",
area);
        return 0;
 }
```

```
float   myfunc (float rad)
{
        float   circlearea;
        circlearea = PI * r * r;
        return (circlarea);
         /* return result */
 }
```

Indentation Example #2 :: Good Style

```
#include <stdio.h>

/* FIND THE LARGEST OF THREE NUMBERS */

int main()
{
        int   a, b, c;

        scanf (“%d %d %d”, &a, &b, &c);
       if  ((a>b) && (a>c))                /* Composite condition check */
            printf (“\n Largest is %d”, a);
        else
            if  (b>c)                          /* Simple condition check */
                printf (“\n Largest is %d”, b);
            else
                printf (“\n Largest is %d”, c);
            return 0;
 }
```

Indentation Example #2 :: Bad Style

```
#include <stdio.h>
/* FIND THE LARGEST OF THREE NUMBERS */
int main()
{
int   a, b, c;
scanf (“%d %d %d”, &a, &b, &c);
if  ((a>b) && (a>c))
printf (“\n Largest is %d”, a);
else
if  (b>c)     /* Simple condition check */
printf (“\n Largest is %d”, b);
else
printf (“\n Largest is %d”, c);
return 0;
}
```