# C Programming

September 1, 2020

## Contents

# 1 Program

As mentioned in the lecture note on *Operating Systems*, the executable file is a software, or a software system. The executable file is a list of instructions to instruct the CPU to complete a task. Each instruction is coded in a form of binary machine code.

Generally speaking, the list of instructions is called a program. A program is a list of instructions. Each instruction does not need to be in a form of binary machine code. It could be in a form of English-like programming language. Programming refers to the process preparing a program. It consists of two phases – *coding* and *debugging*.

## 1.1 Types of languages

There are two types of programming languages – *compiled language* and *interpreted language*. The program developed by a complied language needs a compiler to translate the whole program to an executable file consisting of binary machine codes. C language, Visual Basic and Java are three exemplar compiled languages. Each compiled language has its own compiler.

### 1.1.1 Interpreted Language

The program developed by an interpreted language needs an interpreter to do the translation. The interpreter interprets a line of code at a time. Once the binary machine code has been generated, the binary code will then sent to instruct the CPU to do the corresponding task. To ease a user to command an operating system, each operating system must have an interpreter. The interpreter interprets the command typed by the user and then instructs the CPU to work accordingly. Here is an example.

```
C:\User\John Sum>cd Desktop

C:\User\John Sum\Desktop>dir > desktop-content
```

Once first command has been typed and the 'Enter' key has been pressed, the OS interprets the command and generates the corresponding machine codes to the CPU. Clearly, the action is to change the location of the directory, or change directory in short. Then, the OS returns the control to the user. The second command instructs the OS to generate the list of all the files and directories under the directory 'Desktop' and copy it to a text file called 'desktop-content'. Again, once the second command has been typed and the 'Enter' key has been pressed, the OS interprets the command and generates the machine codes instructing the CPU.

As a matter of fact, interpreters can be found in many systems. A browser, like Microsoft IE, Edge and Chrome, has an interpreter which is used for translating an HTML or XML file. Systems for scientific researches, like Matlab, SPSS and SAS, have interpreters to translate the codes. Moreover, Python and some other languages for AI/ML researches are also interpreted languages. To run such programs, their interpreters have to be invoked.

### 1.1.2 Compiled Language

Compiled languages are the earliest programming languages for developing programs. Instead of writing a program in a low level programming language, the assembly language, programmer would like to have a higher level programming language which could ease the task of programming. Compiled languages were thus developed in the early history of computer. Interpreted language for building sophisticated programs came later. Fortran, COBAL and C languages are three typical examples of compiled languages.

For a program which is written in interpreted language, the program is translated and executed one command at a time. In case there is an erroneous command, the interpreter responses with and error message and the execution of the erroneous command is skipped. The interpreter continues the rest of the other commands. For a program which is written in complied language, the machine code of the program will only be generated if all the commands are syntactically correct. Then, the user can run the program by invoking its executable file, i.e. the file of machine codes.

As a compiler will only translate all the commands together, it is able to do some simple optimizations for the program, such as removing redundancy and minimizing the working memory space.

## 1.2 Open Source

No matter a program is coded by a compiled language or interpreted language, the list of the English-like instructions is called the source code. Usually, the source code a commercial (and yet profitable) software like Windows OS and Office is not provided. User can only purchase the executable file of a commercial software. Its source code is not open to the public. However, some software systems their source codes are open for the developers around the world to improve the design and the functions of the systems. Unix, Linux and Android are open-source operating systems.

In the computer industry, there is a special license to protect those open source softwares. It is the **general public license** (GPL). Developer of an open source software does not have to apply for this license. Once the software has been developed, the developed can simply put a statement in the source code and claim that it is protected by GPL. For a software with GPL, it is assumed that the developers of the software are not restricted to a special group or particular firm. The development of the software is based on the effort from the developers all around the world. No single person or a firm can own this software. Thus, no one in the world can make money from selling the software.

For instance, you have developed an amazing application software which is developed on top of an open source software. The application software cannot be sold for profit, even though 99 percentage of the codes is from your effort. Nevertheless, you cannot claim that you are the owner of the application software as part of your software is from open source.

## 2 Conditional Statements

### 2.1 IF-ELSE

```
#include<stdio.h>

main() /* Main function. */
{
int SNM; /* Current no. of students in Marketing. */
int SNA; /* Current no. of students in Accounting. */
char Choice; /* Define a character variable Choice. */

SNM = 45; SNA = 35;
printf("Which course you would like to register?\n");
printf("a: Marketing, b: Accounting\n");
printf("Please enter your choice (a or b): ");
scanf("%c", &Choice);

if (Choice == 'a')
{
  SNM = SNM + 1;
  printf("You have successfully registered Marketing.\n");
  printf("You are now the %d number of student.", SNM);
}
else
{
  SNA = SNA + 1;
  printf("You have successfully registered Accounting.\n");
  printf("You are now the %d number of student.", SNA);
}
}
```

## 2.2 IF-ELSEIF

```
#include<stdio.h>

main() /* Main function. */
{
int SNM; /* Current no. of students in Marketing. */
int SNA; /* Current no. of students in Accounting. */
char Choice; /* Define a character variable Choice. */

SNM = 45; SNA = 35;
printf("Which course you would like to register?\n");
printf("a: Marketing, b: Accounting\n");
printf("Please enter your choice (a or b): ");
scanf("%c", &Choice);

if (Choice == 'a')
{
  SNM = SNM + 1;
  printf("You have successfully registered Marketing.\n");
  printf("You are now the %d number of student.", SNM);
}
elseif (Choice == 'b')
{
  SNA = SNA + 1;
  printf("You have successfully registered Accounting.\n");
  printf("You are now the %d number of student.", SNA);
}
}
```

## 2.3 While

```
#include<stdio.h>

main() /* Main function. */
{
int SNM; /* Current no. of students in Marketing. */
int SNA; /* Current no. of students in Accounting. */
char Choice = 'c'; /* Define a character variable Choice. */

SNM = 45; SNA = 35;
while((Choice != 'a')&&(Choice != 'b'))
{
  printf("Which course you would like to register?\n");
  printf("a: Marketing, b: Accounting\n");
  printf("Please enter your choice (a or b): ");
  scanf("%c", &Choice); getchar();
}

if (Choice == 'a')
{
  SNM = SNM + 1;
  printf("You have successfully registered Marketing.\n");
  printf("You are now the %d number of student.", SNM);
}
else
{
  SNA = SNA + 1;
  printf("You have successfully registered Accounting.\n");
  printf("You are now the %d number of student.", SNA);
}
}
```

### 2.4 Do-While

```
#include<stdio.h>

main() /* Main function. */
{
int SNM; /* Current no. of students in Marketing. */
int SNA; /* Current no. of students in Accounting. */
char Choice = ’a’; /* Define a character variable Choice. */

SNM = 45; SNA = 35;
do
{
  printf("Which course you would like to register?\n");
  printf("a: Marketing, b: Accounting\n");
  printf("Please enter your choice (a or b): ");
  scanf("%c", &Choice); getchar();
}while((Choice != ’a’)&&(Choice != ’b’))

if (Choice == ’a’)
{
  SNM = SNM + 1;
  printf("You have successfully registered Marketing.\n");
  printf("You are now the %d number of student.", SNM);
}
else
{
  SNA = SNA + 1;
  printf("You have successfully registered Accounting.\n");
  printf("You are now the %d number of student.", SNA);
}
}
```

## 2.5 Switch-Case

```
#include<stdio.h>

main() /* Main function. */
{
int SNM; /* Current no. of students in Marketing. */
int SNA; /* Current no. of students in Accounting. */
char Choice; /* Define a character variable Choice. */

SNM = 45; SNA = 35;
printf("Which course you would like to register?\n");
printf("a: Marketing, b: Accounting\n");
printf("Please enter your choice (a or b): ");
scanf("%c", &Choice);

switch (Choice)
{
case 'a': {
  SNM = SNM + 1;
  printf("You have successfully regitered Marketing.\n");
  printf("You are now the %d number of student.", SNM);
  break;
  }
case 'b': {
  SNA = SNA + 1;
  printf("You have successfully regitered Accounting.\n");
  printf("You are now the %d number of student.", SNA);
  break;
  }
}
}
```

# 3 For/While Loop

## 3.1 Print a line of dots

```
#include<stdio.h>

main()
{
int max, i = 0;

    printf("Total number of dots (N): ");
    scanf("%d", &max);

    printf("Print dots using FOR loop.\n");
    for(i=0; i<max; i++)
    printf("*");
    printf("\n");

    printf("Print dots using WHILE loop.\n");
    while(i<max)
    {   printf("*");
    i = i+1;  }
printf("\n");

printf("Program end!");
}
```

## 3.2 Print a triangle

```
/**********************************************************************
simpletriangle01.cpp

   Usage Example:
   C:>simpletriangle01 5
   '5' is the maximum length of the triangle.

   History: December 29, 2019. John Sum
**********************************************************************/
#include<stdio.h>
#include<stdlib.h>

main(int argc, char *argv[])
{
int length, i, j;

length = atoi(argv[1]);
for(i = 0; i <length; i++)
    {
for(j = 0; j < i+1; j++)
printf("*");
printf("\n");  }
}
```

## 3.3 Find the largest number from a list

```
/*********************************************************************
find_max_steps.cpp

   Description:
   This program demonstrates how c program inputs variables
   from the command line.

   Usage Example:
   C:>find_max_steps 45 23 12 54 78 56

   History: December 29, 2019. John Sum
*********************************************************************/
#include<stdio.h>
#include<stdlib.h>

main(int argc, char *argv[])
{
    int A[256]; /* Define integer array. */
    int i,j; /* Define indices. */
    int tmp; /* Define dummy variable for sorting. */
    int SNUM;

    /* Step 1: Read input from the command line. */
    SNUM = argc - 1;
    for(i = 0; i<SNUM; i++)
        A[i] = atoi(argv[i+1]);

    printf("SNUM = %d\n", SNUM);

    /* Step 2: Push the largest number to the leftmost. */
    for(i=0; i<SNUM-1; i++)
    {
     if(A[i]>A[i+1])
        {
         tmp = A[i];
         A[i] = A[i+1];
         A[i+1] = tmp;
        }
        for(j=0; j<SNUM; j++)
            printf("%d ", A[j]);
        printf(" i = %d \n", i);
    }
    printf("\n");
    printf("The maxmum value is %d.\n", A[SNUM-1]);
}
```

# 4 Inputs to Program

## 4.1 Input string and multiple integers (Name & Age)

```
#include<stdio.h>

main() /* Main function. */
{
int YOB; /* Define integer variable YOB. */
int Age; /* Define integer variable Age */
char Name[32]; /* Define a character string called Name. */
int Gender;

printf("Please enter your name: ");
scanf("%s", Name);
printf("Your gender (Male: 1, Female: 2): ");
scanf("%d", &Gender);
printf("Enter your year of birth: ");
scanf("%d", &YOB);

Age = 2016 - YOB;
printf("%s, your year of birth is %d.\n", Name, YOB);
printf("So, your age is %d.\n", Age);
if(Gender == 1)
  printf("You are a genetleman.\n");
else
  printf("You are a lady.\n");
}
```

## 4.2 Input multiple integers (Age & Weight)

```
#include<stdio.h>

main() /* Main function. */
{
int YOB; /* Define integer variable YOB. */
int Age; /* Define integer variable Age */
int Weight; /* Define integer variable Weight. */

printf("Enter your year of birth (e.g. 1995): ");
scanf("%d", &YOB);
printf("Enter your weight in KG: ");
scanf("%d", &Weight);

Age = 2019 - YOB;
printf("Your age is %d and your weight is %d.", Age, Weight);
}
```

## 4.3 Input multiple integers for sorting

```
#include<stdio.h>

main()
{
    int A[5]; /* Define integer array. */
    int i; /* Define index. */
    int total=0; /* Define index. */

    printf("This program demonstrates how to do sorting.\n");
    printf("Please enter 5 numbers.\n");

    /*
    for(i=0; i<5; i++)
    {
    printf("Enter the %d number: ", i+1);
    scanf("%d", &A[i]);
    }
    */

    i = 0;
    while(i < 5)
    {
    printf("Enter the %d number: ", i+1);
    scanf("%d", &A[i]);
    i = i + 1;
    }
    printf("The numbers are %d %d %d %d %d.\n", A[0], A[1], A[2], A[3], A[4]);

    for(i=0; i<5; i++)
             total = total + A[i];

    printf("The total sum is %d.\n", total);
}
```

## 4.4 Input from command prompt

```
/*********************************************************************
command_argument.cpp
   Description:
   Input variables from the command line.
   argc, argv are two default variables.

   History: December 29, 2019. John Sum
*********************************************************************/
#include<stdio.h>
#include<stdlib.h>

main(int argc, char *argv[])
{
    int i; /* Define index */

    for(i=0; i<argc; i++)
     printf("No.%d argument is %s.\n", i+1, argv[i]);
}
```

# 5 Sorting Numbers

## 5.1 Bubble-Sort with sorting style option

```
#include<stdio.h>

main()
{
    int A[5]; /* Define integer array. */
    int i; /* Define index. */
    int j; /* Define index. */
    int tmp; /* Define dummy variable for sorting. */
    char Opt;

    printf("This program demonstrates how to do sorting.\n");
    printf("Select a sorting style, 'a' for ascending and 'd' for descending.\n");
    printf("Option: ");
    scanf("%c", &Opt);

    printf("\n");
    printf("Please enter five numbers.\n");

    for(i=0; i<5; i++)
    {
    printf("Enter the %d number:", i+1);
    scanf("%d", &A[i]);
    }

   printf("The numbers entered are %d %d %d %d %d.\n", A[0],A[1],A[2],A[3],A[4]);


    if (Opt == 'a')
    {
        for(i=0; i<4; i++)
           for(j=0; j<4-i; j++)
           {
                 if(A[j] > A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
           }
        printf("The sorted numbers are %d %d %d %d %d.", A[0],A[1],A[2],A[3],A[4]);
    }
    else
    {
        for(i=0; i<4; i++)
        for(j=0; j<4-i; j++)
        {
                 if(A[j] < A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
        }
        printf("The sorted numbers are %d %d %d %d %d.", A[0],A[1],A[2],A[3],A[4]);
    }
}
```

## 5.2 Bubble-Sort with simple exceptional handling for incorrect option inputs

```
#include<stdio.h>
main()
{
    int A[5], i, j, tmp;
    char OPT[8];  char Opt = 'c';

    printf("This program demonstrates how to do sorting.\n\n");
    while((Opt != 'a')&&(Opt != 'd'))
    {
    printf("Select a sorting style, 'a' for ascending and 'd' for descending.\n");
    printf("Option : ");
    scanf("%s", OPT);
    Opt = OPT[0];
    }
    printf("\n");
    printf("Please enter five numbers.\n");

    for(i=0; i<5; i++)
    {
    printf("Enter the %d number: ", i+1);
    scanf("%d", &A[i]);
    }

    if(Opt == 'a')
        for(i=0; i<4; i++)
        for(j=0; j<4-i; j++)
                 if(A[j] > A[j+1])
                 {  tmp = A[j];
                    A[j] = A[j+1];
                    A[j+1] = tmp;   }
        printf("The sorted numbers are");
        for(i=0; i<5; i++)
            printf(" %d", A[i]);
        printf(".");
    else
        for(i=0; i<4; i++)
        for(j=0; j<4-i; j++)
                 if(A[j] < A[j+1])
                 {  tmp = A[j];
                    A[j] = A[j+1];
                    A[j+1] = tmp;   }
        printf("The sorted numbers are");
        for(i=0; i<5; i++)
            printf(" %d", A[i]);
        printf(".");
}
```

### 5.3 Bubble-Sort with advanced exceptional handling on incorrect option inputs

```
#include<stdio.h>

main()
{
    int A[5]; /* Define integer array. */
    int i,j; /* Define indices. */
    int tmp; /* Define dummy variable for sorting. */
    char OPT[8], Opt = 'c';
    int SNUM = 0;
    int OPT_ERR = 0, SNUM_ERR = 0;

    /* Step 1: Input Sorting Style with Error Handling */
    printf("This program demonstrates how to do sorting.\n\n");

    while((Opt != 'a')&&(Opt != 'd'))
    {
    if (OPT_ERR == 1)
      printf("ERROR MESSAGE: Wrong input! You can only enter either 'a' or 'd'.\n");

    printf("Select a sorting style, 'a' for ascending and 'd' for descending.\n");
    printf("Option : ");
    scanf("%s", OPT);
    Opt = OPT[0];

    if ((Opt != 'a')&&(Opt != 'd'))
            OPT_ERR = 1;
    }

    /* Step 2: Enter how many numbers to be sorted. */
    printf("\n");
    while(SNUM < 2)
    {
    if (SNUM_ERR == 1)
    {
      printf("ERROR MESSAGE: Wrong input!\n");
      printf("The number must be larger or equal to 2.\n");
    }

    printf("Please enter how many numbers to be sorted : ");
    scanf("%d", &SNUM);

    if (SNUM < 2)
            SNUM_ERR = 1;
    }

    /* Step 3: Enter the numbers. */
    printf("\n");
    printf("Please enter the numbers.\n");

    for(i=0; i<SNUM; i++)
    {
    printf("Enter the %d number: ", i+1);
    scanf("%d", &A[i]);
    }

    /* Step 4: Sorting the numbers */
    if(Opt == 'a')
```

```
        for(i=0; i<SNUM-1; i++)
        for(j=0; j<SNUM-i-1; j++)
        {
                 if(A[j] > A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
        }
    else
        for(i=0; i<SNUM-1; i++)
        for(j=0; j<SNUM-i-1; j++)
        {
                 if(A[j] < A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
        }


    /* Step 5: Show the results */
    printf("The numbers in ascending order are ");
    for(i=0; i<SNUM; i++)
        printf("%d ", A[i]);
    printf(".\n");
}
```

## 5.4 Bubble-Sort with inputs from command prompt

```
/*******************************************************************
bubblesort05.cpp

   Description:
   This program demonstrates how to read 'numbers' and 'sorting style'
   from the command line. Then, do the sorting and output the results.

   Note: Error handling has not been considered here.

   History:
   December 12, 2014. John Sum -- No input error handling.
   December 19, 2014. Input_Count is removed.
   February 21, 2017. Add 'if((Opt == 'a')||(Opt == 'd'))' statement.
*******************************************************************/

#include<stdio.h>
#include<stdlib.h>

main(int argc, char *argv[])
{
    int A[256]; /* Define integer array. */
    int i,j; /* Define indices. */
    int tmp; /* Define dummy variable for sorting. */
    char Opt = 'c';
    int SNUM;

    /* Step 1: Read input from the command line. */
    SNUM = argc - 2;
    Opt = argv[1][0];
    if((Opt == 'a')||(Opt == 'd'))
    {
    for(i = 0; i<SNUM; i++)
        A[i] = atoi(argv[i+2]);

    /* Step 2: Sorting the numbers */
    if(Opt == 'a')
        for(i=0; i<SNUM-1; i++)
        for(j=0; j<SNUM-i-1; j++)
        {
                 if(A[j] > A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
        }

    if(Opt == 'd')
        for(i=0; i<SNUM-1; i++)
        for(j=0; j<SNUM-i-1; j++)
        {
                 if(A[j] < A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
        }


    /* Step 3: Show the results */
    printf("The sorted numbers are");
    for(i=0; i<SNUM; i++)
        printf(" %d", A[i]);
    }

    else
        printf("Something wrong in your inputs.");
}
```

## 5.5 Winner & Loser

The filename of the following program is ”winnerloser.cpp”.

```
/*********************************************************************
winnerloser.cpp
*********************************************************************/
#include<stdio.h>
#include<stdlib.h>

main(int argc, char *argv[])
{
    int A[256]; /* Define integer array. */
    int i,j; /* Define indices. */
    int tmp; /* Define dummy variable for sorting. */
    char Opt = 'c';
    int SNUM, POS;

    /* Step 1: Read input from the command line. */
    SNUM = argc - 3;
    Opt = argv[1][0];
    POS = atoi(argv[2]);

    if (POS <= SNUM)
    {
      if((Opt == 'w')||(Opt == 'l'))
      {
        for(i = 0; i<SNUM; i++)
            A[i] = atoi(argv[i+3]);

        /* Step 2: Sorting the numbers */
        if(Opt == 'l')
        {
          for(i=0; i<SNUM-1; i++)
          for(j=0; j<SNUM-i-1; j++)
          {
                 if(A[j] > A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
          }
          printf("The value of the %d loser is %d.", POS, A[POS-1]);
        }
        if(Opt == 'w')
        {
          for(i=0; i<SNUM-1; i++)
          for(j=0; j<SNUM-i-1; j++)
          {
                 if(A[j] < A[j+1])
                 {
                 tmp = A[j];
                 A[j] = A[j+1];
                 A[j+1] = tmp;
                 }
          }
          printf("The value of the %d winner is %d.", POS, A[POS-1]);
        }
      }
      else
          printf("Something wrong in your inputs.");
    }
    else
        printf("The position %d is larger than %d.", POS, SNUM);
}
```

# 6 Random Numbers

## 6.1 Random integers

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

main()
{
      int MAX = 10;
      int A[MAX];
      int i;

      srand(time(NULL));
      printf("%d random numbers from [1:100] are:\n", MAX);
      for(i=0; i<MAX; i++)
      {
              A[i] = rand() % 100 + 1;
              printf("%d ", A[i]);
      }
}
```

## 6.2 Random integers with distribution

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

main()
{
      int MAX = 10000, RANGE = 10;
      int A[MAX], S[RANGE];
      int i, tmp, tmp1;

      for(i=0; i<RANGE; i++)
              S[i] = 0;

      srand(time(NULL));
      printf("%d random numbers from [1:%d] are generated.\n", MAX, RANGE);
      for(i=0; i<MAX; i++)
      {
              tmp = rand() % RANGE;
              A[i] = tmp + 1;
              S[tmp] = S[tmp] + 1;
      //        printf("( %d : %d )", A[i], S[tmp]);
      }
      printf("\n");
      printf("The statistics of the random numbers are listed below.\n");
      for(i=0; i<RANGE; i++)
         printf("Proportion of %d is %d. (%d %d)\n", i+1, S[i]/MAX, S[i], MAX);
}
```

## 6.3 Uniform distributed random number in $[0, 1]$

```
/*********************************************************
random_number_02.cpp

Description: This program demonstrates the use of type
    conversion in uniform distributed random number
generation.

Note: RAND_MAX is a constant specified in the RNG.

Key command:
    rn = ((float) rand())/((float) RAND_MAX);
    rn = (float) rand()/(float) RAND_MAX;
    rn = (float) rand()/RAND_MAX;
*********************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

main()
{
      int MAX = 10;
      float rn;
      int i, tmp;

      srand(time(NULL));
      printf("%d random numbers in [0,1] are generated.\n", MAX);

      printf("Without using type conversion.\n");
      for(i=0; i<MAX; i++)
      {
               rn = rand()/RAND_MAX;
               printf("%.4f ", rn);
      }
      printf("\n");

      printf("Using type conversion.\n");

      for(i=0; i<MAX; i++)
      {
               rn = (float) rand()/RAND_MAX;
               printf("%.4f ", rn);
      }
}
```

## 6.4 Sum of two uniform distributed random variables

```
/*****************************************************************
random_number_03.cpp

Description: This program shows the distribution of z = (x + y),
    where x and y are uniform distributed random variables in
    {0, 1, 2, 3, 4, 5}. So, there are 11 possible outcomes of z.
    z = {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10}.

*****************************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

main()
{
      int MAX = 100000;
      int RANGE = 6;
      float S[2*RANGE-1];
      int i, tmp, tmp1, tmp2;

      srand(time(NULL));
      for(i=0; i<2*RANGE-1; i++)
               S[i] = 0;

      printf("%d random numbers from [1:%d] are generated.\n", MAX, RANGE);
      for(i=0; i<MAX; i++)
      {
               tmp1 = rand() % RANGE;
               tmp2 = rand() % RANGE;
               tmp = tmp1 + tmp2;
               S[tmp] = S[tmp] + 1;
      }

      printf("\n");
      printf("The statistics of the random numbers are listed below.\n");
      for(i=0; i<2*RANGE-1; i++)
      {
               printf("Proportion of %d is %0.4f. (%4.0f %d)\n", i, S[i]/MAX, S[i], MAX);
      }
}
```

## 6.5 Normal distribution (Box-Muller Algorithm)

```
/*****************************************************************
random_number_04.cpp

Description: This program demonstrates how to generate normal distributed
       random variables from uniform distributed random number generator.
       The method is based on Box-Muller algorithm.

*****************************************************************/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>
#include<math.h>

#define PI 3.14159265358979323846

float randu(void);

main()
{
int i, MAX = 20;
float mean, sigma;
float u1, u2;
float z0;

    srand(time(NULL));
printf("Input mean: ");
scanf("%f", &mean);
printf("Input sigma: ");
scanf("%f", &sigma);
for(i=0; i<MAX; i++)
{
u1 = randu();
u2 = randu();
z0 = sqrt(-2.0 * log(u1))*cos(2*PI*u2);
printf("x = %0.8f\n", z0*sigma + mean);
}
}

float randu(void)
{
float randnum;

randnum = (float) rand()/RAND_MAX;

return randnum;
}
```

## 7 Poker Shuffle Algorithm

The filename of the following program is ”poker02.cpp”.

```
/***********************************************
poker02.cpp

Note that srand(time(NULL)) command has been added
***********************************************/
#include<stdio.h>
#include<stdlib.h>
#include<time.h>

main()
{
      int MAX = 10000;
      int RANGE = 52;
      int CARD[52];
      int PLAYER[2][5];
      int HAND, NUMBER;
      int i, j;
      int tmp, tmp1, tmp2;

      /* Step 1: Card Initialization */
      for(i=0; i<RANGE; i++)
        CARD[i] = i;

      /* Step 2: Card shuffling */
      srand(time(NULL));
      printf("The card are being shuffled now.\n");
      for(i=0; i<MAX; i++)
      {
               tmp1 = rand() % RANGE;
               tmp2 = rand() % RANGE;
               tmp = CARD[tmp1];
               CARD[tmp1] = CARD[tmp2];
               CARD[tmp2] = tmp;
      }

      /* Step 3: Card Assignment */
      for(i=0; i<2; i++)
        for(j=0; j<5; j++)
          PLAYER[i][j] = CARD[i*4+j];

      printf("Here are the cards.\n");
      for(i=0; i<2; i++)
      {
          printf("Player %2d: ", i+1);
          for(j=0; j<5; j++)
            {
              tmp = PLAYER[i][j];
              HAND = tmp/13; /* Quotient */
              NUMBER = tmp % 13; /* Remainder */

              switch ( HAND )
              {
                     case 0: printf("S%2d   ", NUMBER+1); break;
                     case 1: printf("H%2d   ", NUMBER+1); break;
                     case 2: printf("C%2d   ", NUMBER+1); break;
```

```
                    case 3: printf("D%2d   ", NUMBER+1); break;
             }
           }
         printf("\n");
       }
       /* END */
}
```

# 8 Search For Minimum

## 8.1 Brute-Force-Search

```
/*********************************************************
find_min_bfs.cpp

    Description:
    This program is to demonstrate how to find the minimum
    value of a function f(x) and the location x by
    brute-force-search.

    f(x) = x^2 - x;

    Note: Error handling has not been implemented.

*********************************************************/

#include<stdio.h>
#include<stdlib.h>

main()
{
    float h = 0.001;    /* Spacing between two consecutive numbers */
    float x[1001], y[1001], ymin;
float a = 0, b = 1;
int i, index_ymin;

    printf("Finding the minimum of a function by brute-force-search.\n");

    for(i=0; i<1001; i++)
    {
     x[i] = a + i*h;
     y[i] = x[i]*x[i] - x[i];
}

ymin = y[0]; index_ymin = 0;
for(i=1; i<1000; i++)
{
if(y[i] < ymin)
{
ymin = y[i];
index_ymin = i;
}
}

    /* Output the results on screen */
    printf("The minimum of f(x) is at x=%f.\n", x[index_ymin]);
    printf("The minimum of f(x) is at x=%.4f.\n", x[index_ymin]);
    printf("The minimum of f(x) is at x=%.2f.\n", x[index_ymin]);
    printf("Its value is %.4f.", ymin);
}
```

## 8.2 Line search

```
/***********************************************************
find_min_ls.cpp

    Description:
    This program is to demonstrate how to find the minimum
    value of a function f(x) and the location x by
    line-search.

    f(x) = x^2 - x;

    Note: Error handling has not been implemented.

***********************************************************/

#include<stdio.h>
#include<stdlib.h>
#include<math.h>

main()
{
    float h = 0.01;    /* Spacing between two consecutive numbers */
    float xmin, ymin, xnew, ynew;
    float d, ERR;
    int random_num;

    printf("Finding the minimum of a function by line-search.\n");

    xmin = 0; ymin = 0;
    do
    {
        random_num = rand()%2;
        d = (random_num - 0.5)*h;
        xnew = xmin + d;
   //   printf("%.4f ", xnew);
        ynew = xnew*xnew - xnew;
        if (ynew < ymin)
        {
            ERR = fabs(ymin - ynew);
            xmin = xnew; ymin = ynew;
        }
    }while(ERR > 0.0001);

    /* Output the results on screen */
    printf("The minimum of f(x) is at x=%.4f.\n", xmin);
    printf("Its value is %.4f.", ymin);
}
```

## 8.3 Gradient descent algorithm

```
/*********************************************************
find_min.cpp

    Description:
    This program is to demonstrate how to find the minimum
    value of a function f(x) and the location x.
        f(x) = x^2 - x;
    Note: Error handling has not been implemented.
*********************************************************/
#include<stdio.h>
#include<stdlib.h>

main()
{
    float INT;      /* Define floating number for initial data. */
    float ERR = 10; /* Error for stopping criteria */
    float a;        /* Step size */
    float x, tmp;   /* variable for iteration */

    printf("Finding the minimum of a function.\n");
    printf("f(x) = x^2 - x.\n");
    printf("Gradient descent is applied.\n");
    printf("Algorithm: x(t) = x(t-1) - a f'(x(t-1))\n");
    printf("Enter the initial guess : ");
    scanf("%f", &INT);
    printf("Enter the step size (a): ");
    scanf("%f", &a);

    x = INT;
    while(ERR > 0.0000001)
    {
      tmp = x;
      x = x - a*(2*x - 1);

      /* Calculate the absolute error */
      if ((x-tmp) > 0)
        ERR = (x-tmp);
      else
        ERR = tmp-x;
    }

    /* Output the results on screen */
    printf("The minimum of f(x) is at x=%f.\n", x);
    printf("The minimum of f(x) is at x=%.4f.\n", x);
    printf("The minimum of f(x) is at x=%.2f.\n", x);
    printf("Its value is %.4f.", x*x-x);
}
```

# 9 Search for the Root of a Function

## 9.1 Bisection method

```
/*******************************************************
bisection.cpp

    Description: Finding the root of a function f(x) by using
    bisection method.
        f(x) = tanh(0.5(x-1)) - 0.1;
    Note: Error handling has not been implemented.
*******************************************************/
#include<stdio.h>
#include<math.h>

main()
{
    float ERR; /* Error for stopping criteria */
    float a; /* Step size */
    float x, y, tmp;   /* variable for iteration */
    float fx, fy, ft;
    float precision;
    int k = 0;

    printf("Finding the root of tanh(0.5(x-1)) - 0.1.\n");
    printf("Bisection method is applied.\n");
    printf("Enter the initial guess of x (< 0): ");
    scanf("%f", &x);
    printf("Enter the initial guess of y (> 2): ");
    scanf("%f", &y);
    printf("Precision: ");
    scanf("%f", &precision);

    /* Bisection Method */
    ERR = y - x;
    fx = tanh(0.5*(x-1)) - 0.1;
    fy = tanh(0.5*(y-1)) - 0.1;
    while((ERR > precision)&&(fx*fy < 0))
    {
     tmp = (x + y)/2;
     ft = tanh(0.5*(tmp-1)) - 0.1;
     if(fx*ft < 0)
        {
            y = tmp;
            fy = ft;
        }
        if(ft*fy < 0)
        {
            x = tmp;
            fx = ft;
        }
     ERR = y - x;
     k = k + 1;
     printf("Step %d: [%0.8f, %0.8f]\n", k, x, y);
    }

    /* Output the results on screen */
    printf("The root of f(x) is at %0.8f.\n", x);
}
```

## 9.2 Bisection method with function call

```
/*******************************************************
bisection01.cpp

    Description:
    Finding the root of a function f(x) by using bisection
    method.

    f(x) = tanh(0.5(x-1)) - 0.1;

    Note: Error handling has not been implemented.
    function call is implemented.
********************************************************/
#include<stdio.h>
#include<math.h>

float nlfunction(float xtmp);

main()
{
    float ERR; /* Error for stopping criteria */
    float a; /* Step size */
    float x, y, tmp;   /* variable for iteration */
    float fx, fy, ft;
    float precision;
    int k = 0;

    printf("Finding the root of tanh(0.5(x-1)) - 0.1.\n");
    printf("Bisection method is applied.\n");
    printf("Enter the initial guess of x (< 0): ");
    scanf("%f", &x);
    printf("Enter the initial guess of y (> 2): ");
    scanf("%f", &y);
    printf("Precision: ");
    scanf("%f", &precision);

    /* Bisection Method */
    ERR = y - x;
    fx = nlfunction(x);
    fy = nlfunction(y);
    while((ERR > precision)&&(fx*fy < 0))
    {
     tmp = (x + y)/2;
     ft = nlfunction(tmp);
     if(fx*ft < 0)
        {
            y = tmp;
            fy = ft;
        }
        if(ft*fy < 0)
        {
            x = tmp;
            fx = ft;
        }
     ERR = y - x;
     k = k + 1;
     printf("Step %d: [%0.8f, %0.8f]\n", k, x, y);
    }
    /* Output the results on screen */
    printf("The root of f(x) is at %0.8f.\n", x);
}

float nlfunction(float xtmp)
{
    float ftmp;

    ftmp = tanh(0.5*(xtmp-1)) - 0.1;

    return ftmp;
}
```

## 10 Distance Between Two Points

```
/********************************************************
distance.cpp

    Description:
    Distance between two points P1 and P2.

    Step 1: Input P1, P2 coordinates.
    Step 2: Calculate their distance and their relative angle.
    Step 3: Show the results.
********************************************************/
#include<stdio.h>
#include<math.h>

#define PI 3.14159265

main()
{
    float Distance; /* distance */
    float P1[2], P2[2];   /* Coordinates */
    float angle;
    float tmp1,tmp2;

    /* Step 1: Input coordinates */
    printf("Find the distance between point P1 and point P2.\n");
    printf("Input x-coordinate of P1: ");
    scanf("%f", &P1[0]);
    printf("Input y-coordinate of P1: ");
    scanf("%f", &P1[1]);
    printf("Input x-coordinate of P2: ");
    scanf("%f", &P2[0]);
    printf("Input y-coordinate of P2: ");
    scanf("%f", &P2[1]);

    /* Step 2: Calculate distance and angle. */
    tmp1 = P2[0]-P1[0];
    tmp2 = P2[1]-P1[1];
    Distance = sqrt(tmp1*tmp1+tmp2*tmp2);
    angle = atan(tmp2/tmp1)*180/PI;

    /* Step 3: Show results. */
    printf("The distance between P1 and P2 is %.8f.\n", Distance);
    printf("P2 is at the angle %.8f degree relative to P1.", angle);
}
```