## Chapter 4: Introduction to C-programming Language

### 1. Compile Your Source code

Name of source code: filename.c
For example, score.c, likelihood.c, etc.

In order to run your program you need to transfer your source code to an executable program. A program that can transfer your source code to an executable file is called a compiler.

For windows user: You can use Microsoft Visual C++ compiler.

For Unix user: You can use compiler: gcc or cc to compile you source code, score.c
for example, by using the command

```
>gcc  score.c  -lm
```

This command will generate an executable file with name a.out. If you want your executable file with a specific name like Score, you can use the command

```
>gcc score.c –o Score –lm
```

Generally,

```
>Gcc source_code_name –o executable_file_name –lm
```

To run your program: write the executable file name and press *enter* key

```
>Score
```

2. Format of your source code.

Include libeary function
Declare globe variables
Declare sub-functions
Main function
Sub-functions

In source code, a line begin with // is a comment line that has no function. A paragraph begin with /* and end with */ is a comment paragraph that has no function

Every sentence ended with semicolon **;**
Every variable used in the program must be declared at the beginning of the program.

Example1.c : an example of a C source code

```
// include standard library
#include<math.h>
```

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>

// declare globe variable (not necessary)
int nn, mm;
double a,b;

// declare subfunctions
int sum(int n); /*this function calculate the sum of integers from 1 to n*/


// main function
main()
{
	int m, i;
	for(i=1;i<=10;i++)
	{
		m=sum(i);
		printf("sum from 1 to %2d = %d\n",i,m);
	}
}

//subsunction
int sum(int n)
{
	int m, i;
	m=0;
	for(i=1;i<=n;i++)
	{
		m+=i;
		// or m=m+i;
	}
	return m;
}
```

Compile

- gcc Example1.c –o Example –lm
- Example

Output in screen

```
sum from 1 to  1 = 1
sum from 1 to  2 = 3
sum from 1 to  3 = 6
sum from 1 to  4 = 10
sum from 1 to  5 = 15
sum from 1 to  6 = 21
sum from 1 to  7 = 28
sum from 1 to  8 = 36
```

sum from 1 to  9 = 45
sum from 1 to 10 = 55

*First,  include Standard library files*

math.h  contains all the math functions such as
sin(x), cos(x), asin(x), log(x), log10(x),
exp(x), pow(x,y)=x^y, sqrt(x),
ceil(x)=smallest integer not less than x
floor(x)=[x]
stdio.h  contains functions related to input and output.
stdlib.h contains utility functions
string.h contains functions related to string like string copy
ctype.h  contains functions for testing charater
for example isupper(c) return 0
if c is not a upper-case letter otherwise retur non-zero.

Second part: declare globe variables
Third part:  declare sub-functions or subroutines.
Fourth part: Main function
Fifth part: sub-function or subroutine.

**Loop:**
**Sum 1 to n, we can use the following methods:**
**1.**

```
m=0;
for(i=1;i<=n;i++)
{
        m+=i;
}
```

**2.**

```
m=0;
for(i=n;i>=1;i--)
{
  m+=i;
}
```

**3.**

```
m=0;i=1;
for(;;)
{
  m+=i;
  i++; //or i=i+1;
  if(i>n)break;
}
```

**Input and Output**

**Input and Output to screen:**

**Command:**

Input: scanf Output: printf:

Eamxple example2.c

```
#include<math.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
main()
{
  int m;
  double a;
  char ch[100],cc;
  printf("please enter a character\n");
  scanf("%c",&cc);
  printf("the character is %c\n",cc);
  printf("please enter a integer\n");
  scanf("%d",&m);
  printf("the integer is %d\n",m);
  printf("please enter a real number\n");
  scanf("%lf",&a);
  printf("the real number is %lf\n", a);
  printf("please enter a string\n");
  scanf("%s",ch);
  printf("the string is %s\n", ch);
}
```

- gcc example2.c –o ex2 –lm
- ex2

The results will be

```
please enter a character
A
the character is A
please enter a integer
2
the integer is 2
please enter a real number
2.322
the real number is 2.322000
please enter a string
zhang
the string is zhang
```

where, A in second line, 2 in 5th line, 2.322 in 8th line, and zhang in 11th line are what your input in screen.

# Input from and output to a file

For example, you already have a data file with name hapdata.txt, and you want write something to a file output.txt

First, Define a pointer ( with name fp for example) that points to a file: **FILE *Fp;**
Then, open a file

## Open a file

To read from
Fp=fopen(“hapdata.txt”,”r”)

To write to
Fp=fopen(“output.txt”,”w”) Or Fp=fopen(“output.txt”,”a”);

If “w” is used and the file output.txt already exist, this command will overwrite the existed file. If “a” is used and the file output.txt already exist, the new things will be appended to the existed material.

The standard method to open a file is as follow: (* denotes r, w, or a)

```
if((Fp=fopen(“filename”,”*”))= =NULL)
{
   printf(“can’t open file filename\n”);
   exit(1);
}
```

that is equivalent to

```
Fp=fopen(“filename”,”*”)
if(Fp= =NULL)
{
   printf(“can’t open file filename\n”);
   exit(1);
}
```

**Write to or read from a file**

Use the commands

fprintf(Fp, …..)
fscanf(Fp, ….)

The format is the same as printf () and scanf();

See example3.c for detail.

Dynamically Allocate Memory Storage

Using utility sub-functions

```
double *   dvector(int ,int );
int *          ivector(int,int);
double ** matrix( int, int, int,int);
int **       imatrix( int, int, int,int);
```

Free the memory storage

```
void free_ivector(int *,int,int);
void free_dvector(double *,int,int);
void free_matrix(double **,int , int , int , int );
void free_imatrix(int **, int,int,int,int);
```

For example;

```
int nn,mm;
int *hap1, **HH; //declaration of integer vector ( int *) and  integer matrix (int **)
double *x, **A; //declaration of  double vector (double *) and double matrix (double **)
nn=100;
mm=10;
hap1=ivector(1,nn);
HH=imatrix(1,nn,1,mm)
x=dvector(1,nn);
A=matrix(1,nn,1,mm);

….

free_ivector(hap1,1,nn);
free_imatrix(HH,1,nn,1,mm);
free_dvector(x,1,nn);
free_matrix(A,1,nn,1,mm);
```

An example example3.c

```
#include<math.h>
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
#include<ctype.h>
#include "subfunction1.c"
int RX,RY,RZ;
int nsample, nmarker, **gen;
 /*
   nsample: number of sampled individuals
   marker:  number of markers
   gen   :  matrix (i,j)th element is the genotype of
```

```
        ith individual at jth marker
 */
 int compHG(int *hap, int *sgen, int nmarker, int *hapcom);
 /* test if haplotype hap is compatible with genotype sgen.
    if hap is compatible with sgen, than find the haplotype hapcom
          such that hap and hapcom together is sgen
 */
  void compatible( int *sgen, int nmarker, int *numall);
   /* find all haplotypes pairs compatible with genotype sgen
      and write all these haplotype pairs in a file
   */

main()
{
        int i,j,nn,*numall;
        FILE *fp;
        nsample=200;
        nmarker=5;
        gen=imatrix(1,nsample,1,nmarker);
        numall=ivector(1,nmarker);
        for(i=1;i<=nmarker;i++)numall[i]=2;
   /* read genotype data from file hapdata.txt to matrix gen */

        fp=fopen("hapdata.txt","r");
        if(fp==NULL)
        {
                printf("can't open file hapdata.txt for reading\n");
                exit(1);
        }

        for(i=1;i<=nsample;i++)
        {
                fscanf(fp,"%d",&nn);
                for(j=1;j<=nmarker;j++)fscanf(fp,"%d",&gen[i][j]);
        }
   compatible(gen[35],nmarker,numall);
        free_imatrix(gen,1,nsample,1,nmarker);
        free_ivector(numall,1,nmarker);

}
```

```
//==========================================================
  int compHG(int *hap, int *sgen, int nmarker, int *hapcom)
  {
		int i, indicator;
		indicator=1;
		for(i=1;i<=nmarker;i++)
		{
			if(((hap[i]==0)&&(sgen[i]==2))||((hap[i]==1)&&(sgen[i]==0)))
			{
				indicator=0;
				break;
			}
		}
		if(indicator) //same as if(indicator==1)
		{
			for(i=1;i<=nmarker;i++)hapcom[i]=sgen[i]-hap[i];
		}
		return indicator;
  }

//==========================================================
  void compatible( int *sgen, int nmarker, int *numall)
  {
		int i,j,nn,sum, *hap, *hapcom,numhap;
		FILE *fp;

		hap=ivector(1,nmarker);
		hapcom=ivector(1,nmarker);
		numhap=ipower(2,nmarker);
		if((fp=fopen("haplotype.txt","w"))==NULL)
		{
			printf("can't open file haplotype.txt for writing\n");
			exit(1);
		}
		fprintf(fp,"Genotype\n");
		for(i=1;i<=nmarker;i++)
		{
			if(sgen[i]==0)fprintf(fp,"0/0 ");
			else if(sgen[i]==1)fprintf(fp,"0/1 ");
			else fprintf(fp,"1/1 ");
		}
		fprintf(fp,"\n\n");
		sum=0;
		for(i=1;i<=numhap;i++)
		{
			Arrayindex(i-1,hap,nmarker,numall);
			if(compHG(hap,sgen,nmarker,hapcom))
			{
				nn=INdex(hapcom,nmarker,numall);
				if(i-1<=nn)
				{
```

```
                    sum++;
                    fprintf(fp,"\nHaplotype Pair %d\n",sum);
                        for(j=1;j<=nmarker;j++)fprintf(fp,"%d ",hap[j]);
                        fprintf(fp,"\n");
                        for(j=1;j<=nmarker;j++)fprintf(fp,"%d ",hapcom[j]);
                        fprintf(fp,"\n");
                    }
                }
            }
            fclose(fp);
            free_ivector(hap,1,nmarker);
            free_ivector(hapcom,1,nmarker);
}
```

Note: The subroutines Arrayindex () and Index() are in the utility sun-function subfunction1.c