Always put down:

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

Comment out /\* [program] \*/ (can be use in multiple line)

Comment out // (only one line)

Sample code:

float opp;

printf(“\n%f”,opp); // \n means new line

Sept. 12

Run code:

gcc example.c

./a.exe

(everytime)

Sample code - pointer

int x;

int\* y; //y us a pointer to int

y = &x; //assign memory to y

\*y; //process pointer

sin(\*y);

Sample code – pointer on array

array[] = {12,14,16,18}

int\* crusor = array;

int i;

for(i=0;i<4;i++)

{

printf(“\n%d”, \*cursor);

cursor ++;

}

Sample code – send the pointer around

int divide\_list(int\* inp, int divisor, int\* outp, int len)

{

int i =0;

for(i=0;i<len;i++)

{

outp[i] = inp[i]/divisor;

}

}

int main()

{

int a[]={1,2,3,5};

int b[4];

divide\_list(a,2,b,4);

}

Sept. 16

1 byte = 0 -> 255 = -127 -> 127

unsigned

char letter = 59;

char letter = ‘9’; //above mean the same

int -> %d float -> %f char -> %c string -> %s

sample code for character:

char\* words = “Hello world”;

char\* cursor = words;

while(\*cursor != ‘\0’) //print array “words” until reaches last word ‘\0’

{

printf(“%c”,\*cursor);

cursor++;

}

printf/scanf to/from console

sprintf/sscanf to/from string

viprintf/viscanf to/from another machine

fprintf/fscanf to/from files

printf always use actual variable

scanf always use the ADDRESS of the variable

scan from other file:

FILE \*input\_file;

float x;

float y;

input\_file = fopen(“data.dat”, “r”); //open the file data.dat on computer and read(r)

if(input\_file != NULL) //check if the file is open successfully

{

fscanf(input\_file, “%f %f”, &x, &y); //scan from input\_file

fclose(input\_file); //close the file, close the file as soon as when is not needed

}

input\_file = fopen(“hello.txt”, “w”);

fprintf(input\_file, “Hello from x %f and y”, x,y);

fclose(input\_file);

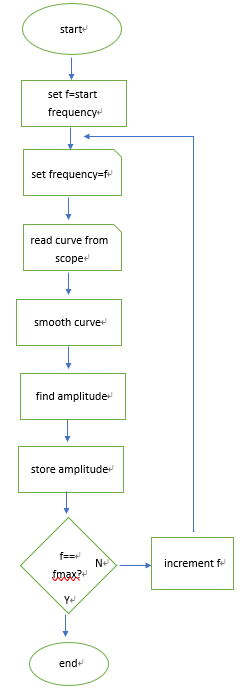
Sept. 26

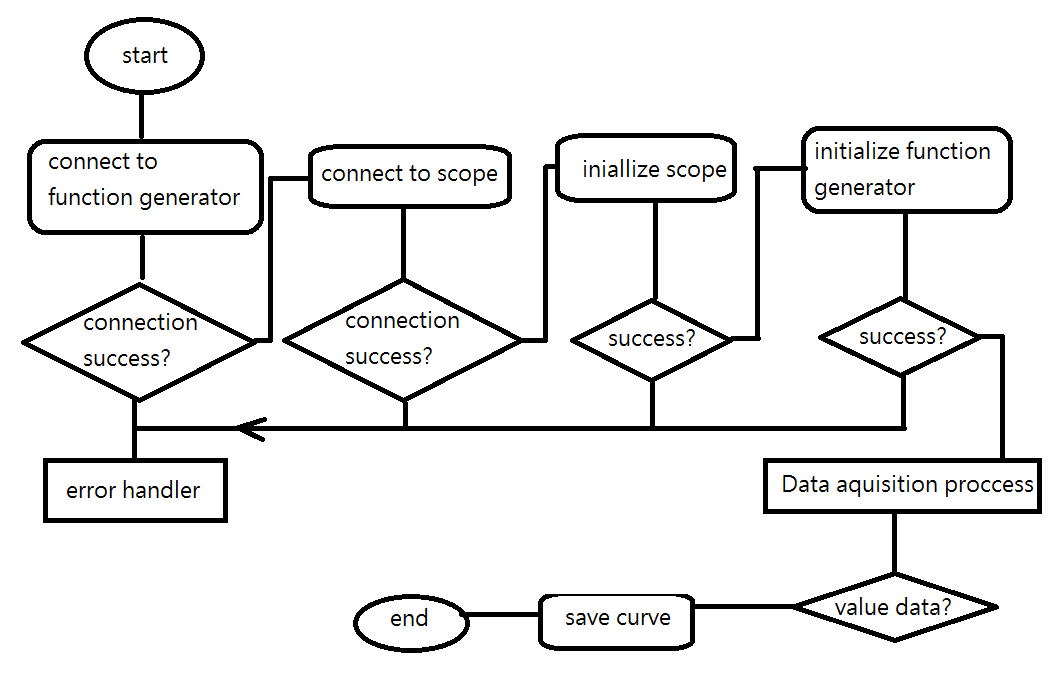
command: mv file new\_place\_new\_name (move, rename)

cp file (copy)

rm file (delete file)

Oct. 17





Nov. 14

Operators:

|  |  |  |
| --- | --- | --- |
| Or operator (|=) | | |
| A | B | Output |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

|  |  |  |
| --- | --- | --- |
| And operator (&= ~) | | |
| A | B | Output |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

|  |  |  |
| --- | --- | --- |
| xOr operator (|=) | | |
| A | B | Output |
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |