I: C Programming Basics and Input and Output

1: Input

1.1: Scanf

```
scanf("%s",&name);//%s for string
scanf("%d",&number);//%d for intrigue
scanf("%c",&character);//%c for characters
scanf("%f",&floatnumber);//%f for floatnumbers
scanf("%a%b",&a,&b);//input two numbers
```

- Note that scanf cannot recognize the "space".

1.2: fgets()

```
fgets(name,sizeof(name),stdin);
```

- fgets() can be use to input the string.
- Notes that it can accept the "space", but there will be an additional "return" in the end.
- A pointer can be use to delete the final "return":

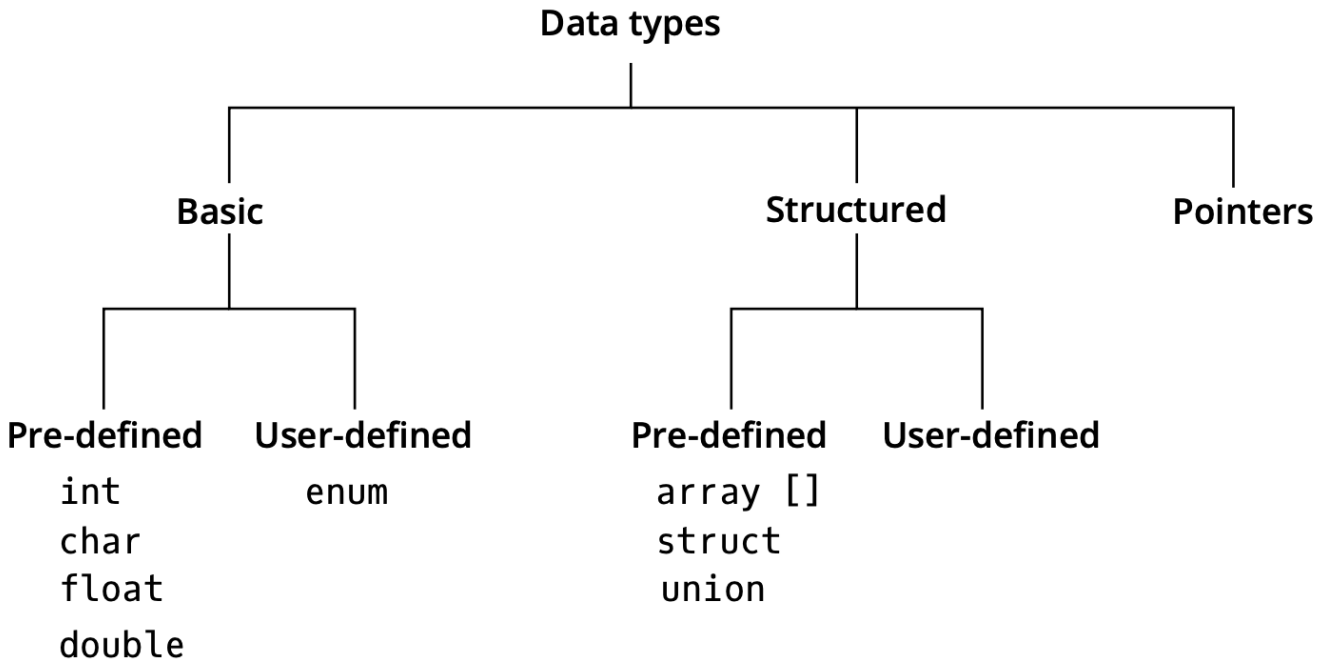
```
char *find;//a pointer of the string
fgets(name,sizeof(name),stdin);
find = strchr(name,'\n');//a function in string.h which can be used to return the addres
*find = '\0';
```

2: Output

```
printf("this is the output");
printf("%d",&a);// output intrigue
printf("%d %d",&a,&b);//output with space between them
```

II: Data Types

1: Introduction



2: Variables

- Variables are names used to refer a **location in memory** that holds a value.
- Formally, a variable is a symbol or name associated with a fixed **physical address (L- VALUE)**, which denotes a value **(R-VALUE)**.

2.1: Type: Int

- The int type differs from the intrigues in maths.
- It can be short int (16 bits) or long int (32 bits).
- The highest bit used to save the sign. (minus or plus)
- Note that the result of '/' between int will also be an int.
- Note that 'a=++i' means 'a=(++i)' while 'a=i++' means '(a=i)++'.

2.2: Type: Float and double

- Float: Single precision, 32-bits
- Double: Double precision, 64-bits

- Note that the result of '/' between float and double will be float and double.

2.3: Type: Char

- In the C language (unlike other languages) the type char does not denote a new data type, but it is equivalent to the domain of values represented in one byte (range 0-255);
- The integer value is the numeric value of the character in the code ASCII.

2.4: Type: Boolean

- In C programming value 0 means FALSE and Non-zero means TRUTH.

```
x||y;// or
x&&y;// and
!x;// not
```

2.5: Type: String (=character arrays)

- Strings (chunks of text) are represented through an array of chars - structured pre-defined data type.
- The first place of the array is '0' instead of '1'.

```
char s[10];//10 is the length of the string, s is the pointer to the beginning of the a
char a[]="hello_";
a[5]='\0';// the '\0' means the end of the string
printf("%s",a);// the output will be 'Hello' cause it ended in advance.
```

- Note that compare every letters in a word one by one can be used to do the dictionary order.

```
#include<string.h>
strcmp("a","b");//which can return a number show the '>','<','=' relation
```

2.6: Symbolic names and Constant

```
#define a 100;//define global constant
constant int N = 10;//define local constant
```

III: Conditional statement

1: If-else statement

```
if (result==0)//only one if to make the condition
{
    printf("Hello World!");
}
----------
if(result==0)//use if-else to express different outputs in different conditions
{
    printf("Yes");
}
else
printf("No");
----------
if(result==0)//nested if-else
{
    printf("Yes");
}
else
{
    if(result==1)
    {
        printf("No");
    }
    else
    printf("Unknown");
}
```

2: Switch statement

- Switch-case statement can be use to do multiple choices.

```
switch(getchar())
{
    case 'Y':printf("Yes\n");break;
    case 'N':printf("No\n");break;
    default: printf("Unknown\n");
}
switch(getchar())
{
    case 'Y'://omit this line will let 'Y' and 'y' be same.
    case 'y':printf("Yes\n");break;
    default: printf("No\n");
}
```

IV: Loop

1: while Loop

```
while(scanf("%d",&a)==1)//add the inputs continually
{
    sum+ = a;//save the sum
    if(a<0){
        break;// Terminate the loop when a<0
    }
    if(a=0){
        continue;//skip this loop can continue to next
    }
    count++;//save the counts of the numbers
}
printf("%d",sum);
----------------------
do
{
    sum+ = a;//save the sum
    if(a<0){
        break;// Terminate the loop when a<0
    }
    if(a=0){
        continue;//skip this loop can continue to next
    }
    count++;//save the counts of the numbers
}
while(count<10);//the loop can only do 10 times
```

2: For Loops

```
for(i = 0; i < count; i++)//initial,condition and increments
{
    int a;
    scanf("%d", &a);
    sum += a;
}
```

V：Functions

1: Define a function

```
void example(char str[])//non-return function
{
//the main part
}
int sum(int x,int y){//type,name,parameters
    return a+b;//the return value
}
```

2: Standard Library Function

2.1: ctype.h

```
isdigit();//check if is a decimal digit
isalpha();//check if is a letter
isupper();//check if is a uppercase letter
islower();//check if is a lowercase letter
isspace();//check if a character is white-space (space, tab, newline, etc.)
ispunct();//check if a character is a punctuation character.
tolower();//convert a letter to lowercase
toupper();//convert a letter to uppercase
```

2.2: string.h

```
strlen();//return the length of string
strcmp();//compare the two strings
strstr("hello","ello");//return the position where the second string in the first one
strcpy(first,second);// same as "first" = "second"
```

2.3: math.h

```
sqrt();
pow();
sin();
log();
fabs();
ceil();//round up to the closest int
floor();//round down to the closest
```

2.4: Recursive function

- Function in C allowed to all themselves which is known as recursion.

```
int numberOfDigits(int x)
{
 // Base case
if (x < 10) return 1;
     // Recursive case
    return numberOfDigits(x / 10) + 1;
}
```

VI: Arrays and application

1: Array

```
char a[3];//a[1],a[1],a[2], note that the index from '0'
int a[2][2];//two-dimension array
printf("%d",a[1][1]);//output as a variable
int c[2]={1,2};//initial directly
int a[2]={1};//is equal to {1,0}
int c[]={1,2};//the initialism will define its size
```

2: Linear search

- How to find the position of a given value in an array

```
int linearsearch(const int a[],int size,int value)
{
    for(int i=0;i<size;i++)
    {
        if(a[i]==value){
            return i;
        }
        else
        return -1;//express the function failed
    }
}
```

3: Binary search

- Find the position of the value of a sorted array.

```
int binarySearch(const int a[],int start, int end, int value)
{
if (start > end)//the prerequisite is sorted
{
    return -1;
}
int middle = (start + end) / 2;
if (a[middle] == value)
{
    return middle;
}
if (a[middle] > value)// find the right part
{
    return binarySearch(a, start, middle - 1, value);
}
else
    return binarySearch(a, middle + 1, end, value);//find the left part
}
```

4: Sorting

- Sorting a certain size array.

```
void SwapSort(int a[],int size)
{
    for(int i=0;i<size;i++)//sort from the first digit
    {
        int k=i;//use 'k' to save i
        for(int j=i+1;i<size;j++)//check the next digit
        {
            if(a[j]<a[i])
            {
                k=j;
            }
            int tmp=a[i];//tmp to save a[i]
            a[i]=a[k];//swap a[i] and a[j]
            a[k]=tmp;
        }
    }
}
```

VII: Files and Pointers

1: Files

1.1: Input and Output of files

```
FILE *f = fopen("numbers.txt", "r");//use a pointer to open the file, 'r' for read
fscanf(f, "%d", &a);//fscanf for input
if (f == NULL) //NULL means the file does not exit
{
    printf("Error: cannot open file\n");
    return 1;
    }
fclose(f);//fclose for closing the file
FILE *out = fopen("sum.txt" , "w");
//'w' for write
fprintf(out, "%d\n", sum);//fprintf for output
```

1.2: fgets()

```
FILE *f = fopen("modules.txt", "r");
fgets(s, size, f);//note that fgets() will end the input at '\n' and can only read size-
while(fgets(s, MAX, f))// fgets() can be used to calculate the line numbers
{
if (strlen(s) == MAX - 1 && s[MAX - 2] != '\n')
{
    printf("Error: buffer overflow\n");
    fclose(f);
    return 1;
}
    count++;
}
```

2: Pointers

2.1: Operation of Pointers

```
int x=5;
int *p;//define a pointer
p=&x;//a pointer without a '*' can only access an address(with '&')
*p=42;//access the pointer directly
x--;//now the value of x is 41
//note that pointers can only contain same data type!
int *q;
q=p+1;//if the address of p is 100, then q address will be 104(+1 means plus 4 bytes)
```

2.2: Relationship between array and pointers

```
int vet[10];//define an array
printf("%p %p",vet,&vet[0]);//will print two same address
int *pi;
pi=vet;//is equal to
pi=&vet[0];
*(pi+3)=28;//the fourth of the array is defined as 28
pi++;//change that pi takes the second place of the array
//Note that vet is a constant pointer
```