>>head file

```
#include<stdio.h>
```
for basic: printf……

```
#include<math.h>
```
for sqrt()

```
#include<stdlib.h>
```
for clock(),rand(),srand(),

>> type

```
int n;
double f;

int arr[SIZE];
int matrix[SIZE][SIZE]];

char str[]={“Hello!”};

struct student { int n; double f };
typedef struct student T;
```

>>function

```
int Function(int n){}
```
回傳 int

```
double Function(int n) {}
```
回傳 double

```
void Function(int n){}
```
不回傳

>>Loop
```
for (int i =0; i <n; i++){}

int i=1;
while(i<n){i++;}
```

>>Sort

Bubblesort

```
void PrintArr(int arr[], int n);

void BubbleSort(int arr[], int n) {
     int i, j, temp;
     for (i = 0; i < n - 1; i++) {
          for (j = 0; j < n - 1 - i; j++) {
               if (arr[j] > arr[j + 1]) {
                    temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    //Swap(arr, j+1, j);

               }
               //PrintArr(arr, n);
               //getchar();
          }
     }
}
```

Insertionsort

```
void PrintArr(int arr[], int n);

void InsertionSort(int arr[], int n) {
     int i, j;
     int temp;
     for (i = 1; i < n; i++) {
          temp = arr[i];
          for (j = i - 1; j >= 0 && arr[j]>temp; j--) {
               arr[j + 1] = arr[j];
               //printf("    ");PrintArr(arr, n);
               //getchar();
          }
          arr[j + 1] = temp;
          //PrintArr(arr, n);
          //getchar();
     }
}
```

Mysort

```
int FindMaxPos(int arr[], int n);
void Swap(int arr[], int pos1, int pos2);

void MySort(int arr[], int n) {
     for (int i = 0; i < n; i++) {
          int maxPos = FindMaxPos(arr, n - i);
          /* find the position of the max number between
arr[0] and arr[n-i-1]*/
          Swap(arr, maxPos, n-i-1);
          /* swap arr[maxPos] and arr[n-i-1]*/
     }
}
```

Mergesort

```
void PrintArr(int arr[], int n);

void Merge(int arr1[], int arr2[], int arr1Size, int
arr2Size, int result[]){
    int writePos=0, readPos1 =0, readPos2 =0;
    for (writePos =0;writePos < arr1Size + arr2Size;writePos+
+){
        if(arr1[readPos1] <= arr2[readPos2]){
            result[writePos] =arr1[readPos1];
            readPos1++;
            if (readPos1 == arr1Size){
                writePos++;
                for(;writePos < arr1Size + arr2Size;writePos+
+){
                    result[writePos] = arr2[readPos2];
                    readPos2++;
                }
            }

        }else{
            result[writePos] =arr2[readPos2];
            readPos2++;
            if (readPos2 == arr2Size){
                writePos++;
                for(;writePos < arr1Size + arr2Size;writePos+
+){
                    result[writePos] = arr1[readPos1];
                    readPos1++;
         }
       }
       }
    }
}

void MergeSort(int arr[], int n){
    if(n <= 1){
        /* instead of n == 1, n <= 1 is safer and captures
more terminating conditions
         for example, what if the array originally has size 2
and then becomes size 0
         */
        return;
    }
    MergeSort(arr, n/2);
    MergeSort(arr+ n/2, n-n/2);

    static int result[ARRSIZE];
    //if a variable is static, this variable is created when
the process
    //starts, this variable will be reused in the future
    Merge(arr, arr+n/2, n/2, n-n/2, result);

    for (int i = 0; i < n; i++) {
        arr[i] = result[i];
    }
}
```