Hospital System Visualizer

Program Design Final Project

team name

TWICE

team members

周述君 409410114

王信智 409410116

劉哲嘉 409410120

吳又成 409515049

Table of Contents

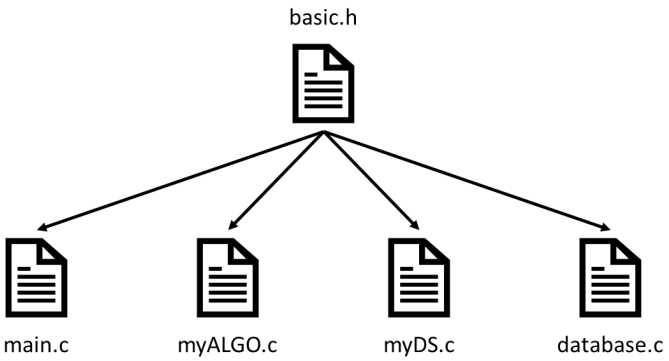
● Introduction：

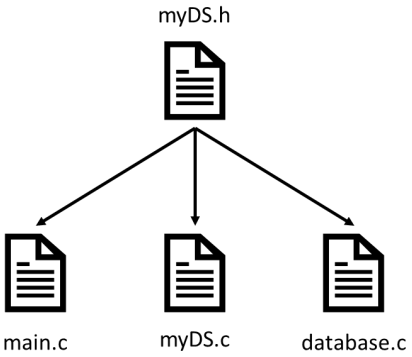
When executing the program, first, the program will show the message “[1]input by terminal [2]input by file:”, asking the user whether to input manually or input by file. If you choose 1, meaning input manually, then the program will show the message “please input number of input:” then user needs to input the number of patients and the information of patients. On the other hand, if you choose 2, the program will automatically read the “Input.txt” file.

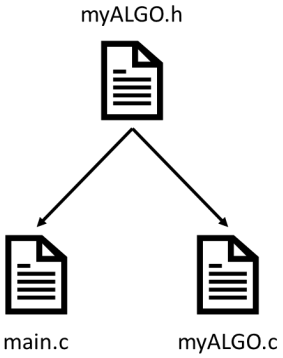
Next, the program will ask the user again to choose from age, weight and situation. The user's choice will be used as the priority order for consultation. For example, if the user chooses age, the program will sort according to the age of patients, which is from bigger to smaller. However, if patients have the same age, they will be sorted according to their situations. Then, according to this priority and the patient’s condition, the program will automatically distribute the corresponding clinic for patients.

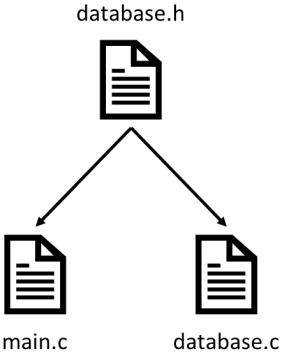
After patients are being distributed to the clinic, our program will simulate a simplified situation, all of the patients have the same consultation time, when time’s up the patient will leave the clinic. At the end of the program, the program will show “[1]database [2]clinic list:”. If you choose 1, then the program will write all of the patients’ information into the “Patient information.txt” file, otherwise, the program will show all of the clinics in the terminal if you choose 2.

- **Program Design**:
  - Header file:

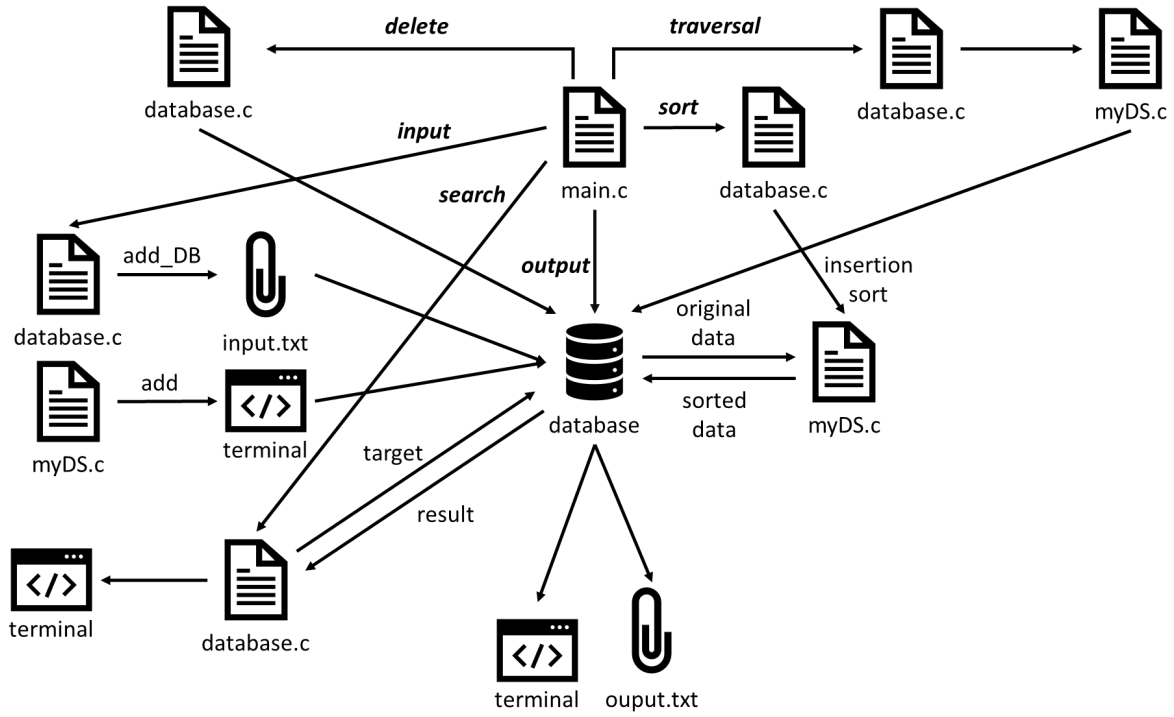








- Structure chart



- Basic Part
  - Data Type and Data Structure

```
typedef struct{
  char name[20];
  int age;
  float weight;
  int number;  //chart No.
  char situation[20];
  int situation_value;
}Inform;

typedef struct{
  Inform field;
  struct Node *next;
}Node;
```

We use “struct” to create data that store patients' information and use “linked list” to store some data in our database.

  - Operations
    - Add_DB: add the patient to the database.
    - Delete_DB: delete the patient from the database.
    - Traverse_DB: print out all the information of the patients.
    - Search_DB: search the information using the patient’s name.

```
Node *first_DB = NULL;
Node *last_DB = NULL;
int num_DB = 0;

//add the data into database
void Add_DB(Inform patient){
  Add(&first_DB, &last_DB, patient);
  num_DB++;
}

//delete the data from database
void Delete_DB(Inform patient){
  Delete(first_DB, patient.name);
  num_DB--;
}
```

```
//traverse the database
void Traverse(){
  Print_ALL(first_DB);
}

//search the database
void Search_DB(Inform patient){
  Search(first_DB, patient.name);
}

//sort the database by chart No.
void Sort_DB(char *type){
  InsertionSort(&first_DB);
}
```

- File I/O
  - File_Input_DB: input the information of the patient from the “Input.txt” file.
  - File_Output_DB: output the information of patient to “Patient information.txt” file.

```
void File_Input_DB(){
  FILE *fp = fopen("Input.txt", "r");
  Inform input;

  if(fp == NULL)
    fprintf(stderr, "Open file failed\n");
  else{
    while(fscanf(fp, "%s %d %f %s", input.name, &input.age, &input.weight,
        input.situation) != EOF){
      input.number = ++num;
      Add_DB(input);
    }
  }

  fclose(fp);
}

//output the database into a file
void File_Output_DB(){
  FILE *fp = fopen("Patient information.txt", "w");

  if(fp == NULL)
    fprintf(stderr, "Open file failed\n");
  else{
    fprintf(fp, "Chart No.\tName\t\tAge\t\tWeight\t\tSituation\n");
    Node *cur = first_DB;
    while(cur != NULL){
      fprintf(fp, "%d\t\t%s\t\t%d\t\t%.2f\t\t%s\n", cur->field.number,
          cur->field.name, cur->field.age, cur->field.weight, cur->field.situation);
      cur = cur->next;
    }
    fprintf(stderr, "Successful\n");
  }

  fclose(fp);
}
```

- Advanece Part
  - Use “queue” to distribute clinics for patients.
    - Push_room: push in the patient after he/she is being distributed to the clinic.
    - Pop_room: pop out the patient after he/she left the clinic.

```
void Push_room(HeadNode room, Inform patient){
  QueueNode *new_node = (QueueNode *)malloc(sizeof(QueueNode));
  new_node->patient = patient;
  new_node->next = NULL;
  new_node->prev = NULL;

  if(room.num == 0){
    room.first = new_node;
    room.last = new_node;
  }
  else{
    QueueNode *temp = room.last;
    temp->next = new_node;
    new_node->prev = temp;
    room.last = new_node;
  }
  room.num++;
}

void Pop_room(HeadNode room){
  QueueNode *temp = room.last;
  room.last = room.last->prev;
  room.last->next = NULL;
  free(temp);
  room.num--;
}
```

- Use “qsort” to sort the patients by age, weight or situation.
  - Cmp_age: sort the patient by their age, from big to small. However, if the patients have exactly the same age, the patient with a light situation will be shown first.
  - Cmp_weight: sort the patient by their weight, from big to small. However, if the patients have exactly the
  - same weight, the patient who is older will be shown first.
  - Cmp_situ: sort the patient by their situ, from light to emergency. However, if the patients have exactly the same situation, the patient whose weight is bigger will be shown first.
  - Situation_value:give the situation value of patients, depending on their situation. Take CAR ACCIDENT for example, its situation value is 1.

```
/*     sort compare functions     */
//AGE大到小，若一樣，病情1到5
int Cmp_age(const void *a, const void *b){
  Inform *p1 = (Inform *)a;
  Inform *p2 = (Inform *)b;
  if(p1->age == p2->age){
    return MAX_Cmpfunc(p1->situation_value, p2->situation_value);
  }
  return MIN_Cmpfunc(p1->age, p2->age);
}

//WEIGHT大到小，若一樣，AGE大到小
int Cmp_weight(const void *a, const void *b){
  Inform *p1 = (Inform *)a;
  Inform *p2 = (Inform *)b;
  if(p1->weight == p2->weight){
    return MIN_Cmpfunc(p1->age, p2->age);
  }
  return MIN_Cmpfunc(p1->weight, p2->weight);
}

//SITU小到大，若一樣，WEIGHT大到小
int Cmp_situ(const void *a, const void *b){
  Inform *p1 = (Inform *)a;
  Inform *p2 = (Inform *)b;
  if(p1->situation_value == p2->situation_value){
    return MIN_Cmpfunc(p1->weight, p2->weight);
  }
  return MAX_Cmpfunc(p1->situation_value, p2->situation_value);
}
```

```c
int Situation_value(char *situ){
  if(strcmp(situ, "PANDMIC") == 0)
    return 0;
  else if(strcmp(situ, "CARACCIDENT") == 0)
    return 1;
  else if(strcmp(situ, "PREGNANT") == 0)
    return 2;
  else if(strcmp(situ, "SCALD") == 0)
    return 3;
  else if(strcmp(situ, "INJURY") == 0)
    return 4;
  else if(strcmp(situ, "SICK") == 0)
    return 5;
```

- Demonstration

  link to the Github repository:
  https://github.com/AmosCCU/PD-II-Final-Project