**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI**
**PHÂN HIỆU TẠI TP. HỒ CHÍ MINH**
**BỘ MÔN CÔNG NGHỆ THÔNG TIN**


BÁO CÁO BÀI TẬP LỚN

# ĐỀ TÀI: VIẾT CHƯƠNG TRÌNH QUẢN LÝ SINH VIÊN VÀ LÝ THUYẾT.

Giảng viên hướng dẫn: TRẦN THỊ DUNG

Sinh viên thực hiện: Phạm Quốc Thắng

Bùi Văn Thiện

Lớp: CQ.60.CNTT

Khoá:60

Tp. Hồ Chí Minh, năm 2020

**Mục lục:**

# CHƯƠNG I

## 1. Làm việc với tệp

### 1.1 Khái niệm tệp tin

+Tệp tin hay tệp dữ liệu là một tập hợp các dữ liệu có liên quan với nhau và có cùng một kiểu được nhóm lại với nhau thành một dãy. Chúng thường được chứa trong một thiết bị nhớ ngoài của mấy tính (đĩa mềm, đĩa cứng...) dưới một cái tên nào đó.

+Tệp được chứa trong bộ nhớ ngoài, điều đó có nghĩa là tệp được lưu trữ để dùng nhiều lần và tồn tại ngay cả khi chương trình kết thúc hoặc mất điện. Chính vì lý do trên, chỉ những dữ liệu nào cần lưu trữ ( như hồ sơ chẳng hạn) thì ta nên dùng đến tệp.

+Tệp là một kiểu dữ liệu có cấu trúc. Định nghĩa tệp có phần nào giống mảng ở chỗ chúng đều là tập hợp của các phần tử dữ liệu cùng kiểu, song mảng thường có số phần tử cố định, số phần tử của tệp không được xác định trong định nghĩa.

+Trong C, các thao tác tệp được thực hiện nhờ các hàm thư viện. Các hàm này được chia làm hai nhóm : nhóm 1 và nhóm 2. Các hàm cấp 1 là các hàm nhập / xuất hệ thống, chúng thực hiện việc đọc ghi như DOS. Các hàm cấp 2 làm việc với tệp thông qua một biến con trỏ tệp.

+Do các hàm cấp 2 có nhiều kiểu truy xuất và dễ dùng hơn so với các hàm cấp 1 nên trong các chương trình viết trong C, các hàm cấp 2 hay được sử dụng hơn.

+Một tệp tin dù được xây dựng bằng cách nào đi nữa cũng chỉ đơn giản là một dãy các byte ghi trên đĩa (có giá trị từ 0 đến 255). Số byte của dãy chính là độ dài của tệp.

+Có hai kiểu nhập xuất dữ liệu lên tệp : File nhị phân và File văn bản.

## 1.2. File nhị phân

+File nhị phân thường có đuôi mở rộng là .bin

+Thay vì lưu trữ dưới dạng văn bản thuần thúy, các file này được lưu dưới dạng nhị phân, chỉ bao gồm các số 0 và 1. Bạn cũng sẽ thấy các con số này nếu cố mở nó bằng 1 text editer kể trên.

+Loại file này giúp lưu trữ được dữ liệu với kích thước lớn hơn, không thể đọc bằng các text editer thông thường và thông tin lưu trữ ở loại file được bảo mật hơn so với file văn bản.

```
0
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  struct threeNum
5  {
6     int n1, n2, n3;
7  };
8
9  int main()
10 {
11    int n;
12    struct threeNum num;
13    FILE *fptr;
14
15    if ((fptr = fopen("C:\\program.bin","rb")) == NULL){
16        printf("Error! opening file");
17
18        // Program exits if the file pointer returns NULL.
19        exit(1);
20    }
21
22    for(n = 1; n < 5; ++n)
23    {
24       fread(&num, sizeof(struct threeNum), 1, fptr);
25       printf("n1: %d\tn2: %d\tn3: %d", num.n1, num.n2, num.n3);
26    }
27    fclose(fptr);
28
29    return 0;
30 }
31
```

Đọc file nhị phân sử dụng fread()

```c

#include <stdio.h>
#include <stdlib.h>

struct threeNum
{
   int n1, n2, n3;
};

int main()
{
   int n;
   struct threeNum num;
   FILE *fptr;

   if ((fptr = fopen("C:\\program.bin","wb")) == NULL){
       printf("Error! opening file");

       // Program exits if the file pointer returns NULL.
       exit(1);
   }

   for(n = 1; n < 5; ++n)
   {
      num.n1 = n;
      num.n2 = 5*n;
      num.n3 = 5*n + 1;
      fwrite(&num, sizeof(struct threeNum), 1, fptr);
   }
   fclose(fptr);

   return 0;
```

Ghi file nhị phân sử dụng fwrite()

## 1.3. File văn bản

+File văn bản là file thường có đuôi là .txt. Những file này bạn có thể dễ dàng tạo ra bằng cách dùng các text editer thông dụng như Notepad, Notepad++, Sublime Text,…

+Khi bạn mở các file này bằng các text editer nói trên, bạn sẽ thấy được văn bản ngay và có thể dễ dàng thao tác sửa, xóa, thêm nội dung của file này.

+Kiểu file này thuận tiện cho chúng ta trong việc sử dụng hàng ngày, nhưng nó sẽ kém bảo mật và cần nhiều bộ nhớ để lưu trữ hơn.

```
0
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main()
5  {
6     int num;
7     FILE *fptr;
8
9     if ((fptr = fopen("C:\\program.txt","r")) == NULL){
10        printf("Error! opening file");
11
12        // Program exits if the file pointer returns NULL.
13        exit(1);
14    }
15
16    fscanf(fptr,"%d", &num);
17
18    printf("Value of n=%d", num);
19    fclose(fptr);
20
21    return 0;
22 }
23
```

Đọc file sử dụng fscanf()

```c

#include <stdio.h>
#include <stdlib.h>

int main()
{
   int num;
   FILE *fptr;
   fptr = fopen("C:\\program.txt","w");

   if(fptr == NULL)
   {
      printf("Error!");
      exit(1);
   }

   printf("Enter num: ");
   scanf("%d",&num);

   fprintf(fptr,"%d",num);
   fclose(fptr);

   return 0;
}

```

Ghi file sử dụng fprintf()

# CHƯƠNG II

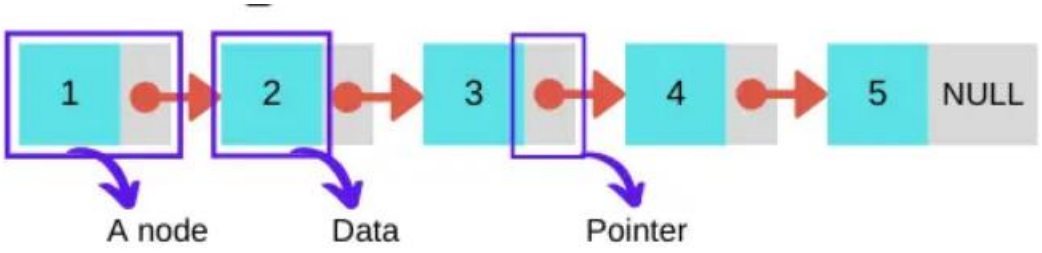
## 2. Danh sách liên kết đơn

### 2.1 Khái niệm danh sách liên kết đơn

+Danh sách liên kết đơn (Single Linked List) là một cấu trúc dữ liệu động, nó là một danh sách mà mỗi phần tử đều liên kết với phần tử đúng sau nó trong danh sách. Mỗi phần tử (được gọi là một node hay nút) trong danh sách liên kết đơn là một cấu trúc có hai thành phần:

+Thành phần dữ liệu: lưu thông tin về bản thân phần tử đó.

+phần liên kết: lưu địa chỉ phần tử đứng sau trong danh sách, nếu phần tử đó là phần tử cuối cùng thì thành phần này bằng NULL.




Minh họa danh sách liên kết đơn

## 2.2 Đặc điểm của danh sách liên kết đơn

+Do danh sách liên kết đơn là một cấu trúc dữ liệu động, được tạo nên nhờ việc cấp phát động nên nó có một số đặc điểm sau đây:

+Được cấp phát bộ nhớ khi chạy chương trình

+Có thể thay đổi kích thước qua việc thêm, xóa phần tử

+Kích thước tối đa phụ thuộc vào bộ nhớ khả dụng của RAM

+Các phần tử được lưu trữ ngẫu nhiên (không liên tiếp) trong RAM

+Và do tính liên kết của phần tử đầu và phần tử đứng sau nó trong danh sách liên kết đơn, nó có các đặc điểm sau:

+Chỉ cần nắm được phần tử đầu và cuối là có thể quản lý được danh sách

+Truy cập tới phần tử ngẫu nhiên phải duyệt từ đầu đến vị trí đó

+Chỉ có thể tìm kiếm tuyến tính một phần tử

+Cài đặt danh sách liên kết đơn

## 2.3 Tạo node

+Danh sách liên kết đơn được tạo thành từ nhiều node, do đó, chúng ta sẽ cùng đi từ node trước. Một node gồm hai thành phần là thành phần dữ liệu và thành phần liên kết. Thành phần dữ liệu có thể là kiểu dữ liệu có sẵn hoặc bạn tự định nghĩa (struct hay class…), trong bài viết này để đơn giản mình sẽ sử dụng kiểu int cho phần dữ liệu. Thành phần liên kết là địa chỉ đương nhiên sẽ là con trỏ, con trỏ này trỏ đến node tiếp theo, do đó, con trỏ này là con trỏ trỏ vào một node.

```
struct Node
{
    int data;
    Node* next;
};
```

Con trỏ vào một node

Để tạo một node mới, ta thực hiện cấp phát động cho node mới, khởi tạo giá trị ban đầu và trả về địa chỉ của node mới được cấp phát.

```
Node* CreateNode(int init_data)
{
    Node* node = new Node;
    node->data = init_data;
    node->next = NULL;      // node vừa tạo chưa thêm vào danh sách nên chưa liên kết với phần
    return node;
}
```

Tạo một node mới

## 2.4 Tạo danh sách liên kết đơn

Ta đã có được thành phần tạo nên danh sách liên kết đơn là node, tiếp theo chúng ta cần quản lý chúng bằng cách biết được phần tử đầu và cuối. Vì mỗi phần tử đều liên kết với phần tử kế vậy nên tả chỉ cần biết phần tử đầu và cuối là có thể quản lý được danh sách này. Vậy đơn giản ta cần tạo một cấu trúc lưu trữ địa chỉ phần tử đầu (head) và phần tử cuối (hay phần tử đuôi tail).

```
struct LinkedList
{
    Node* head;
    Node* tail;
};
```

Cấu trúc lưu trữ

Khi mới tạo danh sách, danh sách sẽ không có phần tử nào, do đó head và tail không trỏ vào đâu cả, ta sẽ gán chúng bằng NULL. Ta xây dựng hàm tạo danh sách như sau:

```
void CreateList(LinkedList& l)
{
    l.head = NULL;
    l.tail = NULL;
}
```

Xây dựng hàm

Bây giờ để tạo một danh sách, ta làm như sau:

```
LinkedList list;
CreateList(list); // Gán head và tail bằng NULL
```

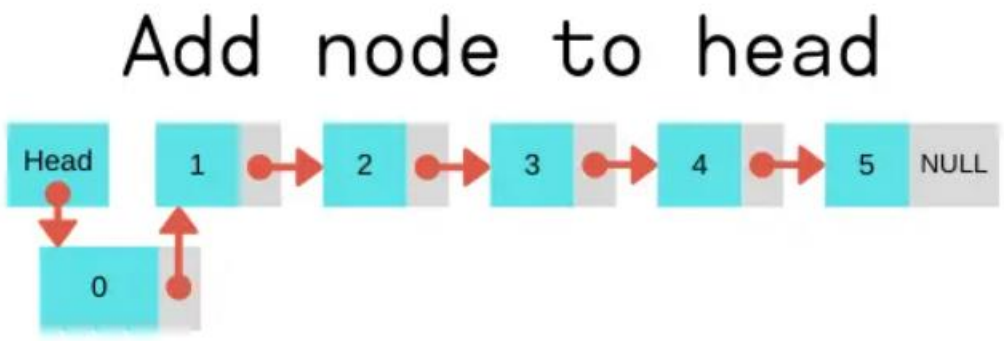
Tạo danh sách

## 2.5 Thêm phần tử vào danh sách

+Thêm vào đầu:

Để thêm node vào đầu danh sách, đầu tiên ta cần kiểm tra xem danh sách đó có rỗng hay không, nếu danh sách rỗng, ta chỉ cần gán head và tail của danh sách bằng node đó. Ngược lại nếu danh sách không rỗng, ta thực hiện trỏ thành phần liên kết vào head, sau đó gán lại head bằng node mới.

```
void AddHead(LinkedList& l, Node* node)
{
    if (l.head == NULL)
    {
        l.head = node;
        l.tail = node;
    }
    else
    {
        node->next = l.head;
        l.head = node;
    }
}
```




Thêm node vào đầu và cuối

# CHƯƠNG III

## 3. Các thuật toán sắp xếp, tìm kiếm

## 3.1 Thuật toán tìm kiếm

### 3.1.1 Tìm kiếm tuyến tính

Là kiểm tra tuần tự từng phần tử của mảng, đến khi nào giống thì thôi.

```
int linearSearch (int a[], int n, int x) {
  for (int i = 0; i < n; i++) {
    if (a[i] == x) return i; //trả về vị trí tìm thấy
    else return -1; //trả về -1 nếu ko tìm thấy
  }
}
```

Thuật toán tuyến tính

### 3.1.2 Tìm kiếm nhị phân

Điều kiện của thuật toán này là mảng đã được sắp xếp tăng dần. So sánh x với giá trị của phần tử nằm ở giữa mảng (mid=(left+right)/2). Nếu x nhỏ hơn a[mid] thì nó chỉ có thể nằm ở nửa bên trái, ngược lại x lớn hơn a[mid] thì x nằm ở nửa bên phải. Xác định x nằm ở nửa nào thì ta lặp lại thuật toán với nửa đó. Như vậy số lần kiểm tra giảm đi nhiều do ko phải mất công kiểm tra những phần tử thuộc nửa còn lại.

```
int binarySearch (int a[], int n, int x) {
  int left = 0, right = n - 1, mid;
  do {
    mid = (left + right) / 2;
    if (a[mid] == x) return mid;
    else if (a[mid] <= x) left = mid + 1;
    else right = mid - 1;
  } while (left <= right);
  return -1;
}
```

Thuật toán nhị phân

### 3.1.3 Tìm kiếm nội suy

Là cải tiến của thuật toán tìm kiếm nhị phân. Thay vì chia đôi, thuật toán này chia theo phép tính  giúp thu gọn khoảng tìm kiếm hơn. Chỉ cần thay biểu thức tính mid này vào code mẫu của thuật toán tìm kiếm nhị phân là được.

```
int interpolationSearch (int a[], int n, int x) {
  int left = 0, right = n - 1, mid;
  do {
    mid = left + (right - left) / (a[right] - a[left]) * (x - a[left]);
    if (a[mid] == x) return mid;
    else if (a[mid] <= x) left = mid + 1;
    else right = mid - 1;
  } while (left <= right);
  return -1;
}
```

Thuật toán nội suy

## 3.2 Thuật toán sắp xếp

### 3.2.1 Sắp xếp đổi chỗ (interchange sort)

Xuất phát từ đầu dãy, lần lượt so sánh phần tử đầu dãy với các phần tử còn lại, nếu thấy lớn hơn thì đổi chỗ cho nhau, mục đích là để sau khi quét một lượt, phần tử bé nhất sẽ về đầu dãy.

```
1  void interchangeSort (int a[], int n) {
2    for (int i = 0; i < n - 1; i++)
3      for (j = i + 1; j < n; j++)
4        if (a[i] > a[j]) swap(a, i, j);
5  }
6  void swap (int a[], int i, int j) {
7    int tmp = a[i];
8    a[i] = a[j];
9    a[j] = tmp;
10 }
```

Thuật toán đổi chỗ

### 3.2.2 Sắp xếp chọn (selection sort)

Tìm phần tử nhỏ nhất trước, rồi mới đổi chỗ nó cho phần tử đầu tiên, xong lặp lại với phần còn lại.

```
1  void selectionSort (int a[], int n) {
2    int min;
3    int iMin;
4    for (i = 0; i < n - 1; i++) {
5      min = a[i];
6      iMin = i;
7      for (j = i + 1; j < n; j++)
8        if (a[j] < a[iMin]) { min = a[j]; iMin = j; }
9      swap(a, iMin, i);
10   }
11 }
```

Thuật toán sắp xếp chọn

### 3.2.3 Sắp xếp nổi bọt (bubble sort)

Lần lượt đổi chỗ các cặp đôi phần tử cạnh nhau nếu chúng ngược thứ tự, mục đích cũng là sau một lượt, phần tử bé nhất sẽ về đầu dãy.

```
void bubbleSort (int a[], int n) {
  for (i = 0; i < n - 1; i++)
    for (j = n - 1; j > i; j--)
      if (a[j] < a[j - 1]) swap(a, j, j - 1);
}
```

Thuật toán sắp xếp nổi bọt

### 3.2.4 Sắp xếp hỗn hợp (cocktail shaker sort)

Lấy hình ảnh xóc bình cocktail, biến j có thể chạy 2 chiều, chiều xuôi thì đưa phần tử lớn nhất về cuối dãy, sau đó đảo lại để đưa phần tử bé nhất về đầu dãy.

```
void shakerSort (int a[], int n) {
  int left = 0, right = n - 1, k = left;
  while (left < right) {
    for (int i = left; i < right; i++)
      if (a[i] > a[i + 1]) { swap(a, i, i + 1); k = i; }
    right = k;
    for (int i = right; i > left; i--)
      if (a[i - 1] > a[i]) { swap(a, i - 1, i); k = i; }
    left = k;
  }
}
```

Thuật toán sắp xếp hỗn hợp

### 3.2.5 Sắp xếp chèn (insertion sort)

Chúng ta coi phần bên trái của biến chạy i là phần đã được sắp xếp theo đúng thứ tự (sẽ chuyển dần các phần tử từ mớ hỗn độn ở phần bên phải sang). Tìm vị trí thích hợp trong nửa trái đó để chèn phần tử a[i] vào cho đúng thứ tự, sau đó dịch chuyển dần các phần tử bên phải kể từ vị trí vừa tìm ra để nong lấy một lỗ trống, rồi mới chèn x (mang giá trị của a[i]) vào.

```
1  void insertionSort (int a[], int n) {
2    int pos, x;
3    for (int i = 1; i < n; i++) {
4      x = a[i]; pos = i - 1;
5      while (pos >= 0 && a[pos] > x) {
6        a[pos + 1] = a[pos];
7        pos--;
8      }
9      a[pos + 1] = x;
10   }
11 }
```

Thuật toán sắp xếp chèn

### 3.2.6 Chèn nhị phân (binary insertion sort)

Là cải tiến một chút của insertion sort, thay vì tìm vị trí để chèn một cách tuần tự, ta áp dụng ý tưởng như thuật toán tìm kiếm nhị phân, cứ chia đôi rồi chia đôi.

```
void binaryInsertionSort (int a[], int n) {
  int left, right, mid, x;
  for (int i = 1; i < n; i++) {
    x = a[i]; left = 0; right = i - 1;
    while (left <= right) {
      mid = (left + right) / 2;
      if (x < a[mid]) right = mid - 1;
      else left = mid + 1;
    }
    for (int j = i - 1; j >= left; j--) a[j + 1] = a[j];
    a[left] = x;
  }
}
```

Thuật toán chèn nhị phân

### 3.2.7 Sắp xếp nhanh (quick sort)

Chọn phần tử x ở giữa mảng. Tìm ở bên trái của x phần tử nào lớn hơn x, ở bên phải có phần tử nào nhỏ hơn x, rồi đổi chỗ 2 phần tử đó. Quét hết lượt thì lặp lại thuật toán với từng nửa. Đến khi nào độ dài các đoạn để quét là 1 thì dừng.

```
void quickSort (int a[], int left, int right) {
  int i, j, x;
  x = a[(left + right) / 2];
  i = left; j = right;
  do {
    while (a[i] < x) i++;
    while (a[j] > x) j--;
    if (i <= j) { swap(a, i, j); i++; j--; }
  } while (i <= j);
  if (left < j) quickSort(a, left, j);
  if(i < right) quickSort(a, i, right);
}
```

Thuật toán sắp xếp nhanh

# CHƯƠNG IV

## 4. Chương trình quản lý sinh viên

### 4.1 Lí do chọn đề tài

- Với sự phát triển của nền công nghệ hiện nay, nhu cầu sử dụng công nghệ trong công cuộc quản lí đã trở thành một nhu cầu cấp bách, đó là một yếu tố không thể thiếu trong việc quản lí và nâng cao chất lượng hiệu quả.

- Do đó, việc quản lí sinh viên trong nhà trường trở nên rất cần thiết, việc ứng dụng công nghệ vào việc quản lý sẽ giúp tiết kiệm thời gian hơn.

## 4.2 Chương trình

```
#include <stdio.h>
#include <conio.h>
#include <stdlib.h>
 struct SinhVien{
    char ten[50];
    char gioitinh[5];
    int tuoi;
    int MSSV;
    float diemtoan, diemly, diemhoa, diemanhvan;
    float diemtrungbinh;
};
 void nhap(SinhVien &sv);
void nhapN(SinhVien a[], int n);
void xuat(SinhVien sv);
void xuatN(SinhVien a[], int n);
void tinhdiemtrungbinh(SinhVien &sv);
void sapxep(SinhVien a[], int n);
void xeploai(SinhVien a);
void xeploaiN(SinhVien a[], int n);
void xuatFile(SinhVien a[], int n, char fileName[]);
void tinhDTB(SinhVien &sv)
{
    sv.diemtrungbinh = (sv.diemhoa + sv.diemly + sv.diemtoan + sv.diemanhvan)/4;
}
void nhap(SinhVien &sv){
printf("\nNhap MSSV: ");
scanf("%d", &sv.MSSV);
```

```
    printf("\nNhap ho va ten: ");
fflush(stdin);
gets(sv.ten);
    printf("\nNhap gioi tinh: ");
gets(sv.gioitinh);
    printf("\nNhap tuoi: ");
scanf("%d", &sv.tuoi);
    printf("\nNhap diem cac mon: ");
    printf("\nMon Toan: ");
scanf("%f", &sv.diemtoan);
    printf("\nMon Ly: ");
scanf("%f", &sv.diemly);
    printf("\nMon Hoa: ");
scanf("%f", &sv.diemhoa);
printf("\nMon Anh Van: ");
scanf("%f", &sv.diemanhvan);
    tinhDTB(sv);
}
 void nhapN(SinhVien a[], int n){
    for(int i = 0; i< n; ++i){
        printf("\nSinh vien thu %d:", i+1);
        nhap(a[i]);
    }
}

void xuat(SinhVien sv){
printf("\nMSSV: %d", sv.MSSV);
    printf("\nHo ten sinh vien: %s", sv.ten);
    printf("\nGioi tinh: %s", sv.gioitinh);
```

```
    printf("\nTuoi: %d", sv.tuoi);
    printf("\nDiem Toan: %.2f", sv.diemtoan);
    printf("\nDiem Ly: %.2f", sv.diemly);
    printf("\nDiem Hoa: %.2f", sv.diemhoa);
    printf("\nDiem Anh Van: %.2f", sv.diemanhvan);
    printf("\nDiem trung binh: %.2f\n", sv.diemtrungbinh);
}
 void xuatN(SinhVien a[], int n){
    for(int i = 0;i < n;++i){
        printf("\nThong tin sinh vien thu %d:", i+1);
        xuat(a[i]);
    }
}
 void sapxep(SinhVien a[], int n){
    SinhVien d;
    for(int i = 0;i < n;++i){
        for(int j = i+1; j < n;++j){
            if(a[i].diemtrungbinh > a[j].diemtrungbinh){
                d = a[i];
                a[i] = a[j];
                a[j] = d;
            }
        }
    }
}
void xeploai(SinhVien sv){
    if(sv.diemtrungbinh >= 8)
printf("Gioi");
    else if(sv.diemtrungbinh >= 6.5)
```

```
printf("Kha");
    else if(sv.diemtrungbinh >= 4)
printf("Trung binh");
    else
printf("Yeu");
}
 void xeploaiN(SinhVien a[], int n){
    for(int i = 0;i < n;++i){
        printf("\nXep loai cua SV thu %d la: ", i+1);
        xeploai(a[i]);
    }
}
void xuatFile(SinhVien a[], int n, char fileName[])
{
    FILE * fp;
    fp = fopen (fileName,"w");
    fprintf(fp, "%20s%5s%5s%10s%10s%10s%10s%10s\n", "Ho Ten","Gioitinh", "Tuoi", "DToan",
"DLy", "DHoa", "Danhvan", "DTB");
    for(int i = 0;i < n;i++){
        fprintf(fp, "%20s%5s%5d%10f%10f%10f%10f%10f\n", a[i].ten, a[i].gioitinh, a[i].tuoi,
a[i].diemtoan, a[i].diemly, a[i].diemhoa, a[i].diemanhvan, a[i].diemtrungbinh);
    }
    fclose (fp);
}


int main(){
    int c;
    char fileName[] = "DSSV.txt";
    int n;
    bool daNhap = false;
```

```
do{
   printf("\nNhap so luong sinh vien: "); scanf("%d", &n);
}while(n <= 0);
SinhVien a[n];
while(true){
   system("cls");
   printf("        MENU QUAN LY SINH VIEN            \n");
   printf("chon 1. Nhap thong tin sinh vien     \n");
   printf("chon 2. In thong tin sinh vien       \n");
   printf("chon 3. Sap xep sinh vien theo DTB   \n");
   printf("chon 4. Xep loai sinh vien           \n");
   printf("chon 5. Xuat file danh sach sinh vien\n");
   printf("chon 6. Thoat                        \n");
   printf("       moi ban nhap lua chon        \n");
   scanf("%d",&c);
   switch(c){
      case 1:
         printf("\nBan da chon nhap thong tin sinh vien!\n");
         nhapN(a, n);
         printf("\nBan da nhap thanh cong!");
         daNhap = true;
         printf("\nBam phim bat ky de tiep tuc!");
         getch();
         break;
      case 2:
         if(daNhap){
            printf("\nBan da chon in thong tin sinh vien!\n");
            xuatN(a, n);
           }else{
```

```
        printf("\nNhap thong tin sinh vien truoc!!!!");
    }
    printf("\nBam phim bat ky de tiep tuc!");
    getch();
    break;
case 3:
    if(daNhap){
        printf("\nBan da chon sap xep sinh vien theo DTB!\n");
        sapxep(a, n);
        xuatN(a, n);
    }else{
        printf("\nNhap thong tin sinh vien truoc!!!!");
    }
    printf("\nBam phim bat ky de tiep tuc!");
    getch();
    break;
case 4:
    if(daNhap){
        printf("\nBan da chon thoat xep loai sinh vien!\n");
        xeploaiN(a, n);
    }else{
        printf("\nNhap thong tin sinh vien truoc!!!!");
    }
    printf("\nBam phim bat ky de tiep tuc!");
    getch();
    break;
case 5:
    if(daNhap){
        printf("\nBan da chon xuat danh sach sinh vien!");
```

```
            xuatFile(a, n, fileName);
        }else{
            printf("\nNhap thong tin sinh vien truoc!!!!");
        }
        printf("\nXuat danh sach  thanh cong vao file %s!", fileName);
        printf("\nBam phim bat ky de tiep tuc!");
        getch();
        break;
    case 6:
        printf("\nBan da chon thoat chuong trinh!");
        getch();
        return 0;
    default:
        printf("\nKhong co chuc nang nay!");
        printf("\nBam phim bat ky de tiep tuc!");
        getch();
        break;
  }
 }
}
```