

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN

----o0o----



BÁO CÁO BÀI TẬP LỚN
CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT
Đề tài: Tìm hiểu về các thuật toán tìm kiếm

GVHD: Thầy Nguyễn Đình Dương

Nhóm 3:

Hoàng Cao long - 181200401

Đào Văn Đại - 181201708

Phạm Tuấn Anh - 171202981

Hà Nội, ngày 10 tháng 11 năm 2022

TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI
KHOA CÔNG NGHỆ THÔNG TIN

----o0o----



BÁO CÁO BÀI TẬP LỚN
CẤU TRÚC DỮ LIỆU VÀ GIẢI THUẬT
Đề tài: Tìm hiểu về các thuật toán tìm kiếm

GVHD: Thầy Nguyễn Đình Dương

Nhóm 3:

Hoàng Cao long - 181200401

Đào Văn Đại - 181201708

Phạm Tuấn Anh - 171202981

Hà Nội, ngày 10 tháng 11 năm 2022

Lời cảm ơn

Chúng em xin gửi lời cảm ơn chân thành đến khoa **Công nghệ thông tin, Trường Đại Học Giao Thông Vận Tải** đã tạo điều kiện thuận lợi cho chúng em học tập và hoàn thành đề tài này. Đặc biệt, chúng em xin bày tỏ lòng biết ơn sâu sắc đến **thầy Nguyễn Đình Dương** đã dày công truyền đạt kiến thức và hướng dẫn chúng em trong quá trình làm bài.

Chúng em đã cố gắng vận dụng những kiến thức đã học được trong học kỳ qua để hoàn thành bài báo cáo. Nhưng do kiến thức hạn chế và không có nhiều kinh nghiệm thực tiễn nên khó tránh khỏi những thiếu sót trong quá trình nghiên cứu và trình bày. Rất kính mong sự góp ý thầy để bài báo cáo của chúng em được hoàn thiện hơn.

Một lần nữa, xin trân trọng cảm ơn sự quan tâm giúp đỡ của các thầy cô đã giúp đỡ chúng em trong quá trình thực hiện bài báo cáo này.

Xin trân trọng cảm ơn!

Mở đầu

Xã hội của chúng ta thay đổi từng ngày. Đi cùng với sự thay đổi đó là sự phát triển mạnh mẽ của ngành công nghệ thông tin trong thời đại số. Công nghệ thông tin xuất hiện đã tạo ra bước phát triển vượt bậc giúp ích cho đời sống xã hội và con người, nó đã ảnh hưởng to lớn đến mọi mặt đời sống mở ra những chân trời mới, những khám phá sáng tạo mới cho con người.

Tìm kiếm là một khái niệm quen thuộc trong cuộc sống. Bạn không nhớ chìa khóa ở đâu? Mình đã học qua nó chưa? Hôm nay mình mặc gì?... tất cả vấn đề ta gặp hầu như đều quy về bài toán duy nhất đó là bài toán tìm kiếm.

Trong Tin học, với sự giúp đỡ của máy tính, rất nhiều thuật toán tìm kiếm đã ra đời với tính hiệu quả ngày càng tăng cao. Những thuật toán tìm kiếm cơ bản nhất có thể kể đến là tìm kiếm tuần tự và tìm kiếm nhị phân. Ngoài ra, áp dụng thêm những cấu trúc dữ liệu trong khi tìm kiếm có thể cho ra những thuật toán có hiệu quả cao hơn nữa.

Từ những nhu cầu thực tế đó, bài toán tìm kiếm dẫn đến chúng ta phải tạo ra thuật toán tìm kiếm để giải quyết nó. Do đó chúng em chọn đề tài “**Tìm hiểu về các thuật toán tìm kiếm**”.

Mục lục

Lời cảm ơn.....	1
Mở đầu.....	2
Mục lục	3
Phần I: Thuật toán tìm kiếm tuần tự.....	4
1. Phát biểu bài toán.....	4
2. Ý tưởng thuật toán	4
3. Ví dụ minh họa	5
Phần II: Thuật toán tìm kiếm nhị phân	6
1. Định nghĩa	6
2. Quy trình	6
3. Ví dụ minh họa	7
Phần III: Thuật toán tìm kiếm trên bảng băm.....	8
1. Sơ lược về bảng băm, hàm băm	8
2. Thuật toán tìm kiếm.....	8
3. Ví dụ minh họa	8
Phần code minh họa	10
Kết luận	18
Tài liệu tham khảo	19

Phần I: Thuật toán tìm kiếm tuần tự

1. Phát biểu bài toán

- Input mảng hoặc danh sách gồm n phần tử a_1, a_2, \dots, a_n .
- k là phần tử cần tìm kiếm.
- Output vị trí của k trong mảng hoặc danh sách.
- Tìm được đối tượng có khóa tương ứng bằng k , khi đó phép tìm kiếm thành công.
- Không tìm được đối tượng nào có khóa tương ứng bằng k , khi đó phép tìm kiếm thất bại.

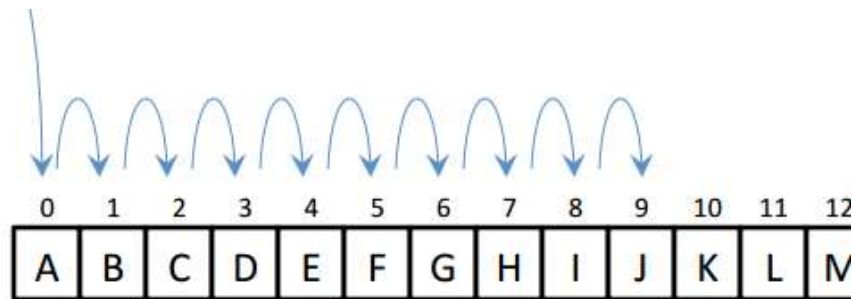
2. Ý tưởng thuật toán

B1: Duyệt từ đối tượng đầu tiên (a_1) đến đối tượng cuối cùng (a_n).

B2: Trong lúc đang duyệt nếu đối tượng cần tìm (k) = đối tượng duyệt hiện tại (a_i) thì dừng lại và trả về chỉ số duyệt (i).

B3: Nếu duyệt hết mà không tìm thấy đối tượng cần tìm (k) thì trả về -1.

Find "J"



Độ phức tạp:

- Trường hợp tốt nhất: $O(1)$.
- Trường hợp xấu nhất: $O(n)$.

Ưu điểm:

- Không yêu cầu đầu vào được sắp xếp hay không.
- Dễ dàng cài đặt.

Nhược điểm: Chỉ phù hợp với bài toán kích thước nhỏ.

3. Ví dụ minh họa

Input {5,7,1,4,2,9,8,11,25,51}.

$K=2$.

Output $a_5 = 2 = k$.

👉 Với $k(=2)$ và dãy A gồm 10 số hạng như sau:



A	5	7	1	4	2	9	8	11	25	51
i	1	2	3	4	5					

Tại vị trí $i = 5$ có $a_5 = 2 = k$

Phần II: Thuật toán tìm kiếm nhị phân

1. Định nghĩa

Trong khoa học máy tính, tìm kiếm nhị phân (binary search) còn gọi là tìm kiếm nửa khoảng (half-interval search), tìm kiếm logarit (logarithmic search), hay binary chop, là một thuật toán tìm kiếm xác định vị trí của một giá trị cần tìm trong một mảng đã được sắp xếp.

Thuật toán bắt đầu bằng việc so sánh một phần tử đứng chính giữa mảng với giá trị cần tìm. Nếu bằng nhau, vị trí của nó trong mảng sẽ được trả về. Nếu giá trị cần tìm nhỏ hơn phần tử này, quá trình tìm kiếm tiếp tục ở nửa nhỏ hơn của mảng. Nếu giá trị cần tìm lớn hơn phần tử ở giữa, quá trình tìm kiếm tiếp tục ở nửa lớn hơn của mảng. bằng cách này, ở mỗi phép lặp thuật toán có thể loại bỏ nửa mảng mà giá trị cần tìm chắc chắn không xuất hiện.

Độ phức tạp:

- Trong trường hợp tệ nhất: $O(\log n)$.
- Trong trường hợp tốt nhất: $O(1)$.
- Trong trường hợp trung bình: $O(\log n)$.

2. Quy trình

Cho một mảng A có n phần tử với các giá trị hoặc bản ghi A_0, A_1, \dots, A_{n-1} đã được sắp xếp sao cho $A_0 \leq A_1 \leq A_2 \leq \dots \leq A_{n-1}$ và giá trị cần tìm T , chương trình con sau đây sử dụng tìm kiếm nhị phân để tìm chỉ số của T trong A .

B1: Gán L với giá trị 0 và R với giá trị $n-1$.

B2: Nếu $L > R$, quá trình tìm kiếm thất bại.

B3: Gán m (vị trí của phần tử đứng giữa) với giá trị floor của $(L+R)/2$, tức là số nguyên lớn nhất nhỏ hơn hoặc bằng $(L+R)/2$.

B4: Nếu $A_m < T$, gán L với $m+1$ và quay lại B2.

B5: Nếu $A_m > T$, gán R với $m-1$ và quay lại B2.

B6: Khi $A_m = T$, quá trình tìm kiếm hoàn tất, trả về m .

Quy trình lặp lại dùng hai biến L và R để lưu giới hạn tìm kiếm.

Nếu $L < n$ và $A_L = T$ thì A_L là phần tử đứng xa nhất về bên trái có giá trị bằng T . Kể cả khi T không nằm trong mảng, L là hạng của T trong mảng hay số phần tử trong mảng nhỏ hơn T .

Nếu $L > 0$ và $A_{L-1} = T$ thì A_{L-1} là phần tử đứng xa nhất về phía bên phải có giá trị bằng T . Kể cả khi T không có trong mảng, $n-L$ sẽ là số phần tử trong mảng lớn hơn T .

3. Ví dụ minh họa

Binary Search

	0	1	2	3	4	5	6	7	8	9
Search 23	2	5	8	12	16	23	38	56	72	91
	L=0	1	2	3	M=4	5	6	7	8	H=9
23 > 16 take 2 nd half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5	6	M=7	8	H=9
23 < 56 take 1 st half	2	5	8	12	16	23	38	56	72	91
	0	1	2	3	4	L=5, M=5	H=6	7	8	9
Found 23, Return 5	2	5	8	12	16	23	38	56	72	91



Cho dãy số được sắp xếp tăng dần: 2,5,8,12,16,23,38,56,72,91. Tìm số 23.

B1: Dãy số được đánh chỉ mục từ 0 → 9, $(0+9)/2 = 4$, tìm được số tương ứng với chỉ mục 4 là 16.

B2: $23 < 16$, nên ta bỏ mảng bên trái lấy mảng bên phải, từ đó được dãy số mới 23,38,56,72,91

B3: Tiếp tục lặp lại 2 bước trên ta tìm sẽ tìm được số cần tìm.

Phần III: Thuật toán tìm kiếm trên bảng băm

1. Sơ lược về bảng băm, hàm băm

Trong khoa học máy tính, bảng băm là một cấu trúc dữ liệu sử dụng hàm băm để ánh xạ từ giá trị xác định, được gọi là khóa (ví dụ như tên của một người), đến giá trị tương ứng (ví dụ như số điện thoại của họ). Do đó, bảng băm là một mảng kết hợp.

Hàm băm được sử dụng để chuyển đổi từ khóa thành chỉ số (giá trị băm) trong mảng lưu trữ các giá trị tìm kiếm. Nếu tất cả các khóa đều được biết trước khi tạo bảng, có thể sử dụng một hàm băm hoàn hảo để tạo ra một bảng băm hoàn hảo không có va chạm. Nếu sử dụng hàm băm hoàn hảo tối thiểu, thì mọi vị trí trong bảng băm đều được sử dụng.

2. Thuật toán tìm kiếm

Để tìm được giá trị đang lưu trữ trong bảng băm, ta phải tìm được chỉ số trong bảng băm.

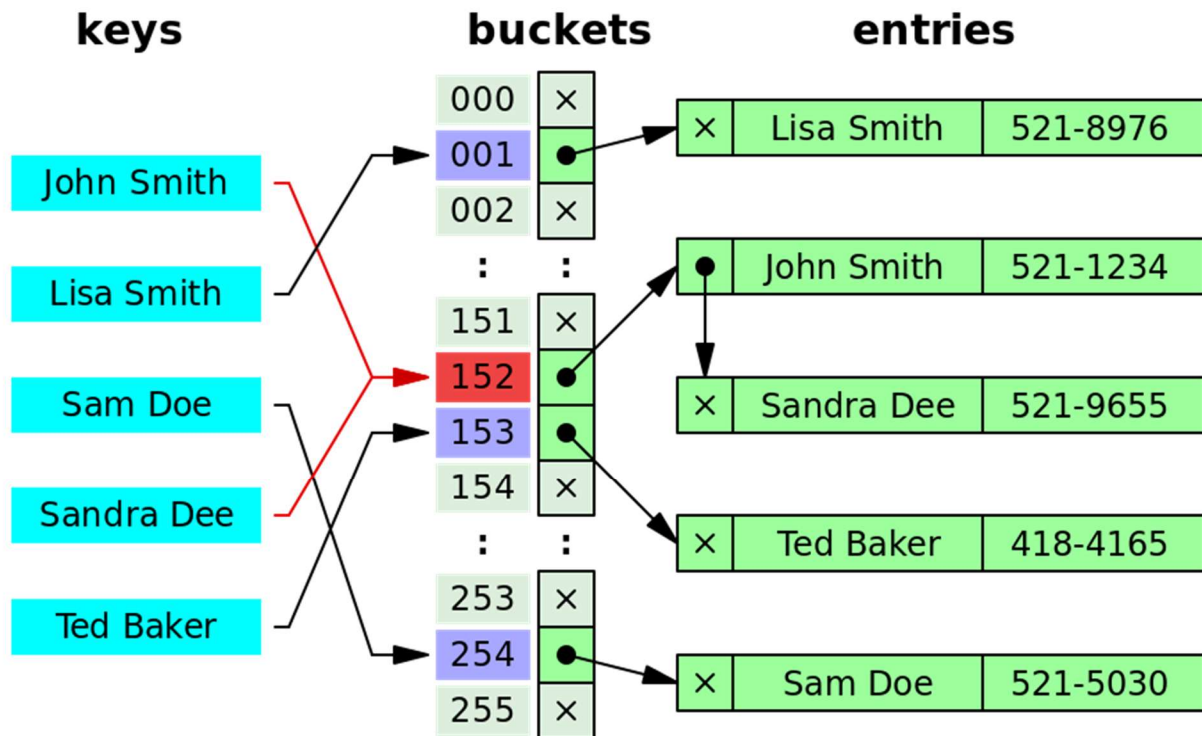
B1: Truyền vào khóa K (khóa lưu trữ giá trị), hàm băm sẽ xử lý khóa K và trả về chỉ số (Index).

B3: Duyệt các giá trị trong Index để tìm ra giá trị cần tìm.

Độ phức tạp thuật toán:

- Trong trường hợp các giá trị đều nằm trong cùng 1 Index: $O(n)$.
- Trong trường hợp chỉ có 1 giá trị duy nhất: $O(n)$.

3. Ví dụ minh họa



B1: Nhập vào key “John Smith”.

B2: Hàm băm sẽ hash key thành index 152.

B3: Trong index 152 có 2 value là số điện thoại của “John Smith” và “Sandra Dee”. Duyệt index ta sẽ tìm được key “John Smith” và số điện thoại tương ứng.

Phần code minh họa

Yêu cầu: Nhập vào danh sách sinh viên với các thuộc tính bao gồm mã sinh viên, họ tên, địa chỉ, giới tính, tuổi, điểm Cấu trúc dữ liệu và giải thuật. Trả về mã sinh viên có số điểm cần tìm được nhập từ bàn phím.

Xây dựng Class Sinh Viên với các phương thức và thuộc tính:

```
1  ✓ #include<iostream>
2  #include<string>
3  using namespace std;
4  ✓ class SinhVien
5  {
6      private:
7          string msv; // mã sinh viên
8          string hoTen; // họ tên
9          string diaChi; // địa chỉ
10         string gioiTinh; // giới tính
11         int age; // tuổi
12         double diemCTDLGT; // điểm cấu trúc dữ liệu và giải thuật
13     public:
14         void Nhap();
15         void Xuat();
16         void HoanDoi(SinhVien &SV1, SinhVien &SV2); // hoán đổi vị trí dùng để sắp xếp
17         int MaxDiem(SinhVien SV[],int SLSV); // trả về số điểm lớn nhất
18         void SapXepDiemTangDan(SinhVien SV[],int SLSV); // sắp xếp danh sách sinh viên theo thứ tự tăng dần
19         int TimKiemTuanTuDiem(SinhVien SV[], int SLSV, int diemCanTim, string msv[]); //Tìm kiếm điểm theo tk tuần tự
20         int TimKiemNhiPhanDiem(SinhVien SV[],int SLSV, int diemCanTim, string msv[]); //Tìm kiếm điểm theo tk nhị phân
21         int TimKiemBangBamDiem(SinhVien SV[], int SLSV, int diemCanTim,int *hashtable);//Tìm kiếm điểm theo Hash-table;
22     };
```

Hình 1. Khai báo lớp, thuộc tính, phương thức.

```

24 void SinhVien::Nhap() // hàm nhập thông tin sinh viên
25 {
26     cout<<"\n\n\t\t ===Nhap Thong Tin Sinh Vien===";
27     fflush(stdin);
28     cout<<"\n nhap ma sinh vien: ";
29     getline(cin,msv);
30     fflush(stdin);
31     cout<<"\n nhap ho ten: ";
32     getline(cin,hoTen);
33     fflush(stdin);
34     cout<<"\n nhap dia chi: ";
35     getline(cin,diaChi);
36     fflush(stdin);
37     cout<<"\n nhap gioi tinh: ";
38     getline(cin,gioiTinh);
39     fflush(stdin);
40     cout<<"\n nhap tuoi: ";
41     cin>>age;
42     cout<<"\n nhap diem CTDL&GT: ";
43     cin>>diemCTDLGT;
44     system("cls");
45 }

```

```

void SinhVien::Xuat() // Xuất thông tin sinh viên
{
    cout<<"\n\n\t\t=== Xuat Thong Tin Sinh Vien ===";
    cout<<"\n MSV: "<<msv;
    cout<<"\n Ho Ten: "<<hoTen;
    cout<<"\n Dia Chi: "<<diaChi;
    cout<<"\n Gioi Tinh: "<<gioiTinh;
    cout<<"\n Tui: "<<age;
    cout<<"\n Diem CTDL&GT: "<<diemCTDLGT;
}

```

Hình 2. Phương thức nhập và xuất thông tin sinh viên.

```

58 void SinhVien::HoanDoi(SinhVien &SV1, SinhVien &SV2) // Hoan doi thong tin sinh vien
59 {
60     SinhVien Temp;
61     Temp = SV1;
62     SV1 = SV2;
63     SV2 = Temp;
64 }
65 // sắp xếp sinh viên theo thứ tự tăng dần
66 // với các tham số là mảng các sinh viên và số lượng sinh viên được nhập vào
67 void SinhVien::SapXepDiemTangDan(SinhVien *SV,int SLSV)
68 {
69     for(int i = 0 ; i<SLSV - 1 ;i++)
70     {
71         for(int j = SLSV - 1 ;j >=i ; j--)
72             // sinh viên có điểm CTDLGT lớn hơn sẽ hoán đổi vị trí với sinh viên có điểm bé hơn
73             if(SV[i].diemCTDLGT>SV[j].diemCTDLGT)
74             {
75                 HoanDoi(SV[i],SV[j]);
76             }
77     }
78 }

```

Hình 3. Phương thức hoán đổi và sắp xếp (dùng cho tìm kiếm nhị phân vì cần phải sắp xếp theo thứ tự).

```

// tham số SLSV là số lượng sinh viên
int SinhVien::TimKiemTuanTuDiem(SinhVien *SV ,int SLSV, int diemCanTim, string msv[])
{
    /* biến dem trả về số lần xuất hiện, biến j để tạo bước nhảy để thêm các mã sinh viên vào
    mảng msv. Nếu có số điểm bằng với điểm cần tìm thì thêm mã của sinh viên có điểm vào mảng msv
    */
    int dem = 0, j = 0;
    for(int i = 0; i<=SLSV; i++)
    {
        if(SV[i].diemCTDLGT==diemCanTim)
        {
            dem++;
            msv[j]=SV[i].msv;
            j++;
        }
    }
    return dem;
}

```

Hình 4. Phương thức tìm kiếm tuần tự tìm trả về số lần xuất hiện và tìm ra sinh viên có mã sinh viên có số điểm bằng với số điểm cần tìm.

```

98 // Tham số gồm mảng đối tượng Sinh Viên, SVSV (số lượng sinh viên), điểm cần tìm và mảng
99 // mã sv để lưu các mã của sinh viên có điểm bằng với điểm cần tìm.
100 int SinhVien::TimKiemNhiPhanDiem(SinhVien SV[],int SLSV, int diemCanTim, string msv[])
101 {
102     int dem = 0,j=0;
103     int left = 0;
104     int right = SLSV - 1;
105     while (left <= right)
106     {
107         int mid = (left + right)/2;
108         if(SV[mid].diemCTDLGT==diemCanTim)
109         {
110             dem++;
111             msv[j]=SV[mid].msv;
112             j++;
113         }
114         if(SV[mid].diemCTDLGT>diemCanTim)
115         {
116             right = mid - 1;
117         }
118         else
119         {
120             left = mid +1;
121         }
122     }
123     return dem;
124 }
125

```

Hình 5. Phương thức tìm kiếm nhị phân trả về số lần xuất hiện và tìm ra mã sinh viên có số điểm bằng với số điểm cần tìm.


```

126  int SinhVien::MaxDiem(SinhVien SV[],int SLSV)
127  {
128      int Max = SV[0].diemCTDLGT;
129      for(int i = 1 ; i < SLSV ; i++)
130      {
131          if(Max < SV[i].diemCTDLGT)
132          {
133              Max = SV[i].diemCTDLGT;
134          }
135      }
136      return Max;
137  }

```

```

int SinhVien::TimKiemBangBamDiem(SinhVien SV[], int SLSV, int diemCanTim,int *hashtable)
{
    int dem = 0;
    for(int i = 0; i<SV[0].MaxDiem(SV,SLSV);i++)
    {
        if(i=SV[i].diemCTDLGT)
        {
            hashtable[i]=SV[i].diemCTDLGT;
        }
    }
    for(int i = 0; i<SV[0].MaxDiem(SV,SLSV);i++)
    {
        if(hashtable[i]=diemCanTim)
        {
            dem++;
        }
    }
    return dem;
}

```

Hình 6. Phương thức tìm kiếm trên bảng băm.


```

int main()
{
    int SLSV, diemCanTim, dem; // tao bien so luong sinh vien, diem can tim, va bien dem (muc dich de tra ve so lan xuat hien)
    string msv[10];
    cout<<"\n\n\t\t Ban Muon Nhap Thong Tin Cho Bao Nhiêu Sinh Vien: ";
    cin>>SLSV;
    SinhVien *SV = new SinhVien[SLSV];

    // Nhap Thong Tin Sinh Vien
    for(int i = 0 ; i < SLSV; i++)
    {
        cout<<"\n\n\t\t Nhap TT cho sinh vien thu "<<i+1;
        SV[i].Nhap();
    }

    // Sap xep diem tang dan de tim kiem nhi phan
    SV[0].SapXepDiemTangDan(SV,SLSV);
    for(int i = 0 ; i < SLSV; i++)
    {
        cout<<"\n\n\t\t Xuat Thong Tin Sinh Vien Thu "<<i+1;
        SV[i].Xuat();
    }

    // Nhap so diem can tim
    cout<<"\nNhap Diem Can Tim: ";
    cin>>diemCanTim;

    // Tim kiem tuan tu
    dem=SV[0].TimKiemTuanTuDiem(SV,SLSV,diemCanTim,msv);
    cout<<"\n Theo Thuat Toan Tim Kiem Tuan Tu thi voi so diem la "<<diemCanTim<<" co xuat hien "<<dem
    <<" lan tai cac sinh vien co ma sinh vien la: ";
    for(int i = 0; i < dem;i++)
    {
        cout<<msv[i]<<endl;
    }

    // tim kiem nhi phan
    dem = SV[0].TimKiemNhiPhanDiem(SV,SLSV,diemCanTim,msv);
    cout<<"\n Thuat Toan Tim Kiem Nhi Phan thi voi so diem la "<<diemCanTim<<" co xuat hien "<<dem
    <<" lan tai cac sinh vien co ma sinh vien la: ";
    for(int i = 0; i < dem;i++)
    {
        cout<<msv[i]<<endl;
    }

    // Hash table;
    int Max = SV[0].MaxDiem(SV,SLSV);
    int *hashtable = new int[Max];
    dem = SV[0].TimKiemBangBamDiem(SV,SLSV,diemCanTim,hashtable);
    cout<<"\n Thuat Toan Tim Kiem Bang Bam thi voi so diem la "<<10<<" co xuat hien 1 lan voi msv la: 171202981";
    system("pause");
    return 0;
}

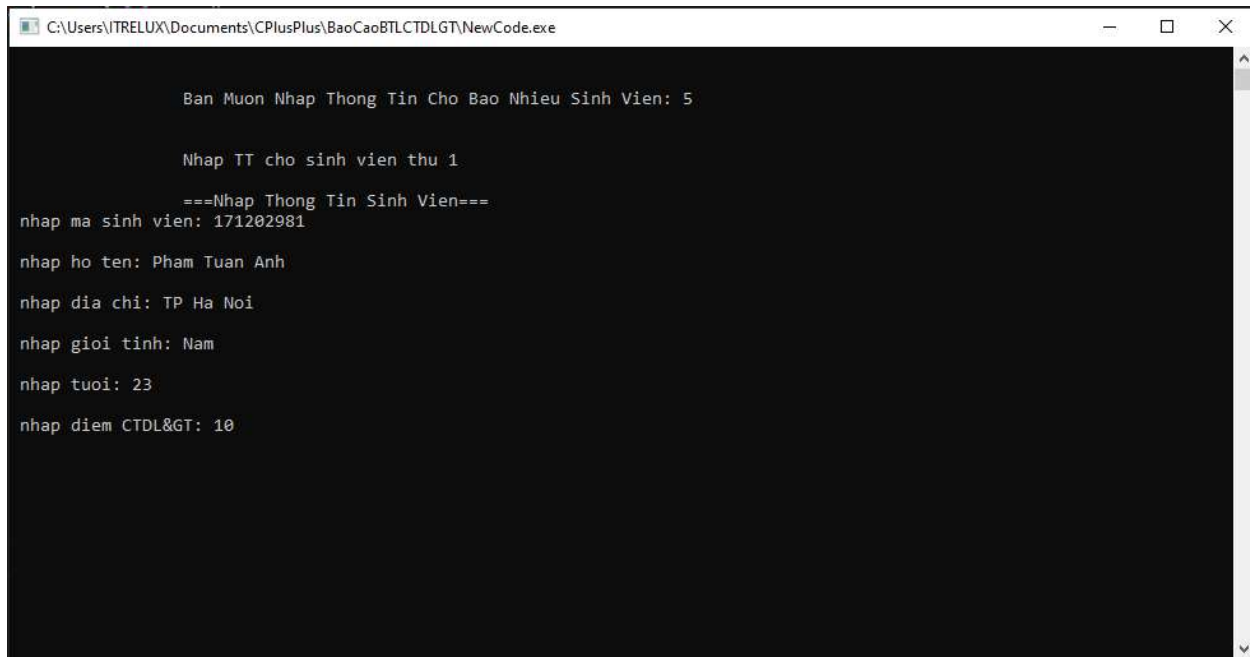
```

Hình 7: Xây dựng hàm main

Kết quả:

Với số lượng sinh viên là 5:

Nhập lần lượt thông tin

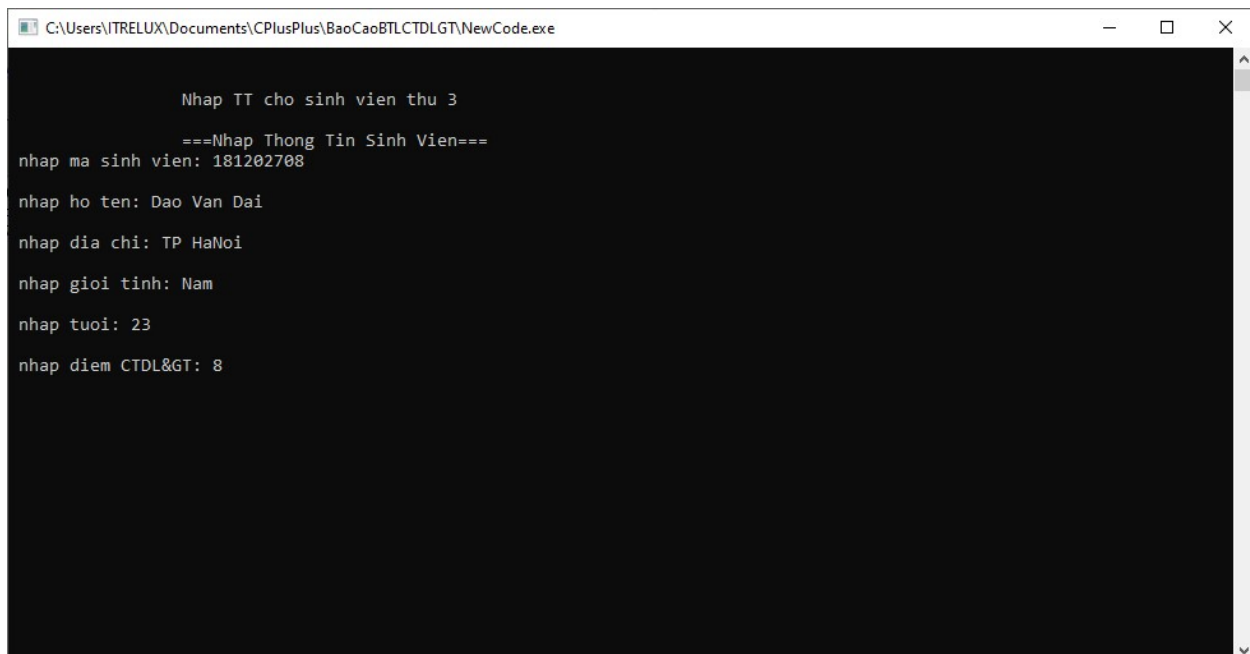


```
C:\Users\ITRELUX\Documents\CPlusPlus\BaoCaoBTLCTDLGT\NewCode.exe

Ban Muon Nhap Thong Tin Cho Bao Nhiêu Sinh Viên: 5

Nhap TT cho sinh viên thu 1

===Nhap Thong Tin Sinh Viên===
nhap ma sinh viên: 171202981
nhap họ tên: Pham Tuan Anh
nhap địa chỉ: TP Ha Noi
nhap giới tính: Nam
nhap tuổi: 23
nhap điểm CTDL&GT: 10
```



```
C:\Users\ITRELUX\Documents\CPlusPlus\BaoCaoBTLCTDLGT\NewCode.exe

Nhap TT cho sinh viên thu 3

===Nhap Thong Tin Sinh Viên===
nhap ma sinh viên: 181202708
nhap họ tên: Dao Van Dai
nhap địa chỉ: TP HaNoi
nhap giới tính: Nam
nhap tuổi: 23
nhap điểm CTDL&GT: 8
```

Xuất thông tin

```

                Xuất Thông Tin Sinh Viên Thu 1
            === Xuất Thông Tin Sinh Viên ===
MSV: 11111111
Ho Ten: Nguyen Van A
Dia Chi: TP Ha Nam
Gioi Tinh: Nam
Tuoi: 22
Diem CTDL&GT: 5

                Xuất Thông Tin Sinh Viên Thu 2
            === Xuất Thông Tin Sinh Viên ===
MSV: Nguyen Van B
Ho Ten: Nguyen Van B
Dia Chi: TP Ha Dong
Gioi Tinh: Nam
Tuoi: 20
Diem CTDL&GT: 7

                Xuất Thông Tin Sinh Viên Thu 3
            === Xuất Thông Tin Sinh Viên ===
MSV: 181202708
Ho Ten: Dao Van Dai
Dia Chi: TP HaNoi
Gioi Tinh: Nam
Tuoi: 23
Diem CTDL&GT: 8

                Xuất Thông Tin Sinh Viên Thu 4
            === Xuất Thông Tin Sinh Viên ===
MSV: 181200401
Ho Ten: Hoang Cao Long
Dia Chi: TP HN
Gioi Tinh: Nam
Tuoi: 22
Diem CTDL&GT: 9

                Xuất Thông Tin Sinh Viên Thu 5
            === Xuất Thông Tin Sinh Viên ===
MSV: 171202981
Ho Ten: Pham Tuan Anh
Dia Chi: TP HaNoi
Gioi Tinh: Nam
Tuoi: 23
Diem CTDL&GT: 10
Nhap Diem Can Tim:
```

Nhập số điểm cần tìm vào: 10

```

Nhap Diem Can Tim: 10

Theo Thuật Toán Tìm Kiếm Tuần Tự thì với số điểm là 10 có xuất hiện 1 lần tại các sinh viên có mã sinh viên là: 171202981
Thuật Toán Tìm Kiếm Nhị Phân thì với số điểm là 10 có xuất hiện 1 lần tại các sinh viên có mã sinh viên là: 171202981
Thuật Toán Tìm Kiếm Bang Băm thì với số điểm là 10 có xuất hiện 1 lần với msv là: 171202981Press any key to continue . . .
```

Kết luận

Với ba thuật toán tìm kiếm kể trên ta thấy rằng:

- Thuật toán tìm kiếm nhị phân: Cần sắp xếp trước theo thứ tự để tiến hành tìm kiếm.
 - o Độ phức tạp thời gian: $O(\log [n])$ trong đó cơ số của $\log = 2$.
 - o Độ phức tạp của không gian: $O(1)$ để thực hiện lặp trong khi $O(\log [n])$ để thực hiện đệ quy vì với mỗi lần gọi đệ quy, một ngăn xếp mới được tạo ra.
- Thuật toán tìm kiếm tuần tự: Không cần xử lý đầu vào nhưng thời gian xử lý sẽ lâu hơn nhị phân trong trường hợp dữ liệu lớn.
 - o Độ phức tạp về thời gian: $O(n)$.
 - o Độ phức tạp không gian: $O(1)$.
- Thuật toán tìm kiếm bảng băm: Khá phức tạp khi cần phải chia mảng băm chính xác nhưng lại có khả năng xử lý nhanh nếu xử lý đầu vào tốt. Hỗ trợ tìm kiếm trong danh sách.
 - o Độ phức tạp thuật toán:
 - Trong trường hợp các giá trị đều nằm trong cùng 1 Index: $O(n)$.
 - Trong trường hợp chỉ có 1 giá trị duy nhất: $O(n)$.

Tài liệu tham khảo

- [1] **Website:** https://en.wikipedia.org/wiki/Binary_search_algorithm.
- [2] **Website:** https://en.wikipedia.org/wiki/Hash_table.
- [3] **Website:** <https://vnoi.info/wiki/algo/data-structures/hash-table.md>.
- [4] **Sách:** Introduction to Algorithms, 3rd Edition.