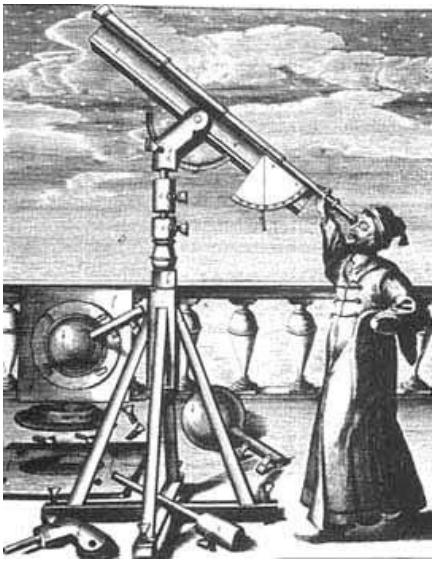
7. Pencarian



Overview

Pencarian merupakan sebuah algoritma dasar yang sering diperlukan dalam pembuatan program. Berbagai algoritma pencarian telah diciptakan dan dapat digunakan. Pemahaman tentang beberapa algoritma pencarian dasar perlu diketahui, termasuk cara penggunaannya dalam program. Di dalam bab ini akan dijelaskan penggunaan program pencarian dalam array.

Tujuan

1. Mengenal beberapa program pencarian
2. Menerapkan program pencarian untuk menyelesaikan suatu masalah

7.1 Konsep Pencarian

Pencarian adalah proses menemukan nilai (data) tertentu dari dalam sekumpulan nilai yang bertipe sama (tipe dasar maupun tipe bentukan). Dengan kata lain, algoritma pencarian adalah algoritma yang mengambil input berupa persoalan dan mengembalikan penyelesaian berupa penemuan nilai yang dicari dalam persoalan inputan.

Proses pencarian seringkali diperlukan pada saat program perlu mengubah atau menghapus nilai tertentu (sebelum bisa mengubah atau menghapus, perlu mencari dulu apakah nilai tersebut ada dalam kumpulan nilai tersebut). Kasus lain yang memerlukan algoritma pencarian adalah penyisipan data ke dalam kumpulan data (perlu dimulai dengan pencarian apakah data tersebut telah ada sehingga terhindar dari duplikasi data).

7.2 *Pencarian Sekuensial*

Pencarian sekuensial (*sequential search*) adalah proses membandingkan setiap elemen larik (array) satu persatu dengan nilai yang dicari secara beruntun, mulai dari elemen pertama sampai elemen yang dicari sudah ditemukan, atau sampai seluruh elemen sudah diperiksa.

Pencarian sekuensial ini cocok untuk pencarian nilai tertentu pada sekumpulan data terurut maupun tidak. Keunggulan pencarian ini adalah dalam mencari sebuah nilai dari sekumpulan kecil data. Pencarian sekuensial termasuk pencarian yang sederhana dan cepat karena tidak memerlukan proses persiapan data (misalnya: pengurutan).

Untuk penerapan dalam program, pencarian sekuensial akan dibuat dalam suatu prosedur yang akan dipanggil dari main atau dari prosedur yang lainnya. Prosedur pencarian akan digunakan untuk mencari data pada suatu array, dimana di dalam array terdapat sekumpulan data angka. Berikut adalah penerapan pencarian sekuensial dalam bahasa pemrograman C :

1. Pencarian Sekuensial yang mengembalikan Boolean (ketemu atau tidak ketemu)

```
1  #include <stdio.h>
2  #define TRUE  1
3  #define FALSE 0
4  typedef int bool;

5  void seqSearch(int L[10],int N,int X,bool
   *ketemu);
```

```
6  void main()
7  {
8  int arr[10]= {6,7,3,8,2,5,4,1,8,10};
9  bool pos;
10 seqSearch(arr,10,11,&pos);
11 if (pos)
12 printf("Ketemu");
13 else
14 printf("Tidak Ketemu");
15 }

16 void seqSearch(int L[10],int N,int X,bool
   *ketemu)
17 {
18 int k;
19 k=0;
20 while ((k<N) && (L[k] != X))
21 {
22 k = k+1;
23 }
24 if ((L[k] == X) && (k<10))
25 *ketemu=TRUE;
26 else
27 *ketemu=FALSE;
28 }
```

Pada contoh diatas, telah diinisialisasikan sebuah array dengan 10 elemen, kemudian dilakukan pencarian angka 8. Program akan menampilkan "Ketemu" jika angka yang dicari terdapat di dalam array dan menampilkan "Tidak Ketemu" jika angka yang dicari tidak terdapat di dalam array.

2. Pencarian yang mengembalikan posisi angka yang dicari

```
1  #include <stdio.h>

2  void seqSearch(int L[10],int N,int X, int *idx);

3  void main()
4  {
5  int pos, arr[10]= {6,7,3,8,2,5,4,1,8,10};
6  seqSearch(arr,10,5,&pos);
7  if (pos!=-1)
```

```
8  printf("Ketemu di posisi %d",pos);
9  else
10 printf("Tidak Ketemu");
11 }

12 void seqSearch(int L[10],int N,int X, int *idx)
13 {
14 int k;
15 k=0;
16 while ((k<N) && (L[k] != X))
17 {
18 k = k+1;
19 }
20 if ((L[k] == X) && (k<N))
21 *idx=k+1;
22 else
23 *idx=-1;
24 }
```

Program diatas akan mengembalikan posisi data yang dicari. Pencarian dimulai dari elemen pertama (`k ← 0`). Kemudian pencarian dilakukan dengan perulangan yang membandingkan setiap elemen larik secara berurutan dari elemen pertama hingga terakhir dengan nilai data yang diinginkan. Perulangan berakhir jika elemen yang dibandingkan bernilai sama dengan data yang dicari (mengembalikan posisi data), atau jika elemen larik telah dibandingkan semua namun data yang dicari tidak ditemukan (mengembalikan nilai `-1`).

3. Pencarian data berdasar inputan user

```
1  #include <stdio.h>

2  void seqSearch(int L[10],int N,int X,int *idx);
3  void inputData(int arr[]);

4  void main()
5  {
6  int i,pos,cari,arr[10];
7  //panggil function inputData
8  inputData(arr);
9  //user input elemen yang dicari
10 printf("\nElemen yang dicari : ");
11 scanf("%d",&cari);
12 //menjalankan pencarian
13 seqSearch(arr,10,cari,&pos);
```

```
14 if (pos!=-1)
15 printf("Ketemu di posisi %d",pos);
16 else
17 printf("Tidak Ketemu");
18 }

19 void inputData(int arr[])
20 {
21 int i;
22 printf("\nMassukkan Elemen Array\n");
23 //user input elemen array
24 for(i=0;i<10;i++)
25 {
26 printf("Array ke %d : ",i+1);
27 scanf("%d",&arr[i]);
28 }
29 }

30 void seqSearch(int L[10],int N,int X,int *idx)
31 {
32 int k;
33 k=0;
34 while ((k<N) && (L[k] != X))
35 {
36 k = k+1;
37 }
38 if ((L[k] == X) && (k<10))
39 *idx=k+1;
40 else
41 *idx=-1;
42 }
```

Pada program diatas, akan dilakukan pencarian data yang telah diinputkan oleh user. Pertama user harus memasukkan data ke dalam larik, hal ini ditangani di dalam prosedur inputData. Kemudian user akan diminta untuk memasukkan data yang ingin dicari. Prosedur seqSearch akan dijalankan dengan parameter berupa array dan nilai yang dicari yang telah dimasukkan oleh user. Prosedur ini akan mengembalikan posisi data yang dicari jika ketemu.

4. Implementasi pencarian ke data bentukan

```
1  #include <stdio.h>
2  struct dataMahasiswa
3  {
4  char nim[10];
5  char nama[20];
6  };

7  void inputDataMhs(struct dataMahasiswa mhs[]);
8  void seqSearch(struct dataMahasiswa mhs[],int
   N,char X[10],int *idx);

9  void main()
10 {
11 int pos;
12 char pil;
13 char nimCari[10];
14 struct dataMahasiswa mhs[5];
15 //panggil function input data
16 inputDataMhs(mhs);
17 while(1)
18 {
19 //user input elemen yang dicari
20 printf("\nNim yang dicari : ");
21 scanf("%s",&nimCari);
22 //menjalankan pencarian
23 seqSearch(mhs,5,nimCari,&pos);
24 if (pos!=-1)
25 {
26 printf("\nData Mahasiwa Ketemu");
27 printf("\nNim siswa  : %s",mhs[pos].nim);
28 printf("\nNama siswa : %s",mhs[pos].nama);
29 }
30 else
31 printf("\nData Mahasiwa Tidak Ketemu");

32 printf("\nCari data lagi [y/n] ? ");
33 scanf("%s",&pil);
34 if ((pil=='y')||(pil=='Y'))
35 continue;
36 else
37 break;
38 }
```

```
39 }

40 void inputDataMhs(struct dataMahasiswa mhs[])
41 {
42 int i;
43 printf("\nMasukkan Data Mahasiswa\n");
44 //user input elemen mahasiswa
45 for(i=0;i<5;i++)
46 {
47 printf("\nData Mahasiswa ke %d : \n",i+1);
48 printf("Nim : ");
49 scanf("%s",&mhs[i].nim);
50 printf("Nama : ");
51 scanf("%s",&mhs[i].nama);
52 }
53 }

54 void seqSearch(struct dataMahasiswa L[],int N,char
   X[10],int *idx)
55 {
56 int k;
57 int i;
58 k=0;
59 while ((k<N) && (strcmp(L[k].nim,X)!=0))
60 {
61 k = k+1;
62 }
63 if ((strcmp(L[k].nim,X)==0) && (k<N))
64 *idx=k;
65 else
66 *idx=-1;
67 }
```

Pada program diatas, akan dilakukan pencarian terhadap data bentukan. Data bentukan adalah dataMahasiswa dimana terdiri dari nim dan nama. Data mahasiswa tersebut akan diinputkan oleh user. Kemudian user juga diminta untuk memasukkan data yang akan dicari yaitu data nim. Jika nim mahasiswa ditemukan maka akan ditampilkan nama dan nim mahasiswa tersebut, jika tidak ada maka akan ditampilkan bahwa data mahasiswa tidak ditemukan.

Berikut adalah hasil output program pada contoh no 4 :

```
Masukkan Data Mahasiswa
```

```
Data Mahasiswa ke 1 :
Nim : 101           {inputan user}
Nama : Michael      {inputan user}

Data Mahasiswa ke 2 :
Nim : 102           {inputan user}
Nama : Linda        {inputan user}

Data Mahasiswa ke 3 :
Nim : 103           {inputan user}
Nama : Leonardo     {inputan user}

Data Mahasiswa ke 4 :
Nim : 104           {inputan user}
Nama : Jason        {inputan user}

Data Mahasiswa ke 5 :
Nim : 105           {inputan user}
Nama : Jennifer     {inputan user}

Nim yang dicari : 103           {inputan user}

Data Mahasiwa Ketemu
Nim siswa  : 103
Nama siswa : Leonardo
Cari data lagi [y/n] ? y        {inputan user}

Nim yang dicari : 106           {inputan user}

Data Mahasiwa Tidak Ketemu
Cari data lagi [y/n] ? y        {inputan user}

Nim yang dicari : 105           {inputan user}

Data Mahasiwa Ketemu
Nim siswa  : 105
Nama siswa : Jennifer
Cari data lagi [y/n] ? n        {inputan user}
```

Pencarian sequential tidak efektif jika digunakan pada data yang banyak atau data yang dicari berada pada posisi terakhir pencarian.

7.3 *Pencarian Biner*

Pencarian biner adalah proses mencari data dengan membagi data atas dua bagian secara terus menerus sampai elemen yang dicari sudah ditemukan, atau indeks kiri lebih besar dari indeks kanan.

Algoritma ini lebih efisien daripada algoritma pencarian sekuensial, tetapi pencarian ini mempunyai syarat yaitu bahwa kumpulan data yang harus dilakukan pencarian harus sudah **terurut** terlebih dahulu, baik terurut secara menaik (ascendant) atau menurun (descendant). Karena data sudah terurut, algoritma dapat menentukan apakah nilai data yang dicari berada sebelum atau sesudah elemen larik yang sedang dibandingkan pada suatu saat. Dengan cara ini, algoritma dapat lebih menghemat waktu pencarian.

Pencarian dalam data terurut bermanfaat misalnya pada penyimpanan data dengan beberapa komponen, program dapat mencari sebuah indeks terurut. Setelah menemukan indeks yang dicari, program dapat membaca data lain yang bersesuaian dengan indeks yang ditemukan tersebut.

Keterurutan data menentukan algoritma biner yang digunakan, karena algoritma biner untuk data terurut menaik sedikit berbeda dengan algoritma biner untuk data terurut menurun, tetapi intinya sama yaitu membagi data atas dua bagian.

Untuk penerapan dalam program, pencarian biner akan dibuat dalam suatu prosedur yang akan dipanggil dari main atau dari prosedur yang lainnya. Prosedur pencarian akan digunakan untuk mencari data pada suatu array, dimana di dalam array terdapat sekumpulan data angka.

Berikut adalah penerapan pencarian biner dalam bahasa pemrograman C untuk data terurut **menaik** :

5. Pencarian Biner untuk data terurut menaik yang mengembalikan posisi data

```
#include <stdio.h>
#define TRUE  1
#define FALSE 0
typedef int bool;

void BinSearch(int L[7],int N,int X,int *idx);

void main()
{
      int pos,arr[7] = {0,4,6,7,12,45,67};
```

```
      BinSearch(arr,7,7,&pos);
      if(pos!=-1)
            printf("Ketemu di posisi ke-%d",pos);
      else
            printf("Tidak Ketemu");
}

void BinSearch(int L[7],int N, int X,int *idx)
{
      int i,j,k;
      bool ketemu;
      i=0;
      j=N;
      ketemu = FALSE;
      while ((!ketemu) && (i<=j))
      {
            k=(i+j)/2;
            if (L[k]==X)
                  ketemu=TRUE;
            else
            {
                  if(L[k]<X)
                        i=k+1;
                  else
                        j=k-1;
            }
      }
      if (ketemu)
            *idx = k+1;
      else
            *idx =-1;
}
```

Prosedur BinSearch pada program di atas dapat bekerja pada larik yang telah terurut menaik (*ascending*). Prosedur memerlukan parameter input berupa:

- `L` yaitu sebuah larik (array) tempat menyimpan data yang diinginkan,
- `N` yaitu jumlah elemen larik,
- `X` yaitu nilai data yang ingin dicari

serta sebuah parameter output berupa nilai `idx` (mengembalikan posisi nilai jika nilai ditemukan, dan mengembalikan -1 jika nilai tidak ditemukan).

Prosedur dimulai dengan inisialisasi pencacah (i←0) dan menyimpan jumlah elemen larik dalam variabel j. Variabel ketemu akan diberi nilai false, hal ini memberitahukan bahwa data yang dicari belum ditemukan. Perulangan diawali dengan membandingkan elemen tengah (elemen dengan indeks k) dengan nilai data yang dicari.

- Jika elemen larik yang dibandingkan bernilai sama dengan data yang dicari, maka variabel boolean ketemu bernilai true dan prosedur mengembalikan nilai indeks elemen tersebut.
- Jika elemen larik bernilai lebih kecil dari data yang dicari, maka pencarian akan bergeser ke kanan dengan cara mengubah nilai i (awal pencacah) dengan indeks di sebelah kanan nilai tengah (i←k+1).
- Jika elemen larik bernilai lebih besar dari data yang dicari, maka pencarian akan bergeser ke kiri dengan cara mengubah nilai i (awal pencacah) dengan indeks di sebelah kiri nilai tengah (j←k-1).

Selanjutnya, perulangan terus dilakukan sampai ketemu bernilai true atau nilai i>j (semua elemen larik sudah dibandingkan dengan nilai yang dicari). Jika semua elemen larik sudah dibandingkan dengan nilai yang dicari tetapi nilai tidak ditemukan, maka nilai ketemu akan tetap bernilai false sehingga akan dikembalikan nilai index= -1 (penanda bahwa nilai yang dicari tidak ditemukan dalam larik).

6. Pencarian Biner untuk data terurut menaik berdasar inputan user

```
#include <stdio.h>
#define TRUE  1
#define FALSE 0
typedef int bool;

void BinSearch(int L[10],int N,int X,int *idx);
void inputData(int arr[]);
void bubbleSort(int arr[],int N);

void main()
{
      int pos,cari,arr[10];
     //panggil fungsi input Data
      inputData(arr);
      //panggil fungsi sort
      bubbleSort(arr,10);
      //user input elemen yang dicari
      printf("\nElemen yang dicari : ");
      scanf("%d",&cari);
      //menjalankan pencarian
```

```
	BinSearch(arr,10,cari,&pos);
	if(pos!=-1)
		printf("Ketemu di posisi ke-%d",pos);
	else
		printf("Tidak Ketemu");
}
void inputData(int arr[])
{
	int i;
	printf("\nMassukkan Elemen Array\n");
	//user input elemen array
	for(i=0;i<10;i++)
	{
		printf("Array ke %d : ",i+1);
		scanf("%d",&arr[i]);
	}
}

void bubbleSort(int arr[],int N)
{
	int a,b,temp;
	//proses sortir dengan bubble sort
	for(a=0;a<=(N-2);a++)
	{
		for(b=(N-1);b>=(a+1);b--)
		{
			if (arr[b-1] > arr[b])
			{
				temp = arr[b-1];
				arr[b-1]= arr[b];
				arr[b] = temp;
			}    //endif
		} //end loop j
	}  //end loop i
	//tampil data urut
	printf("\nData terurut");
	for(a=0;a<N;a++)
	{
		printf("\nArray ke-%d : ",a+1);
		printf("%d",arr[a]);
	}
}

void BinSearch(int L[10],int N, int X,int *idx)
```

```
{
	int i,j,k;
	bool ketemu;
	i=0;
	j=N;
	ketemu = FALSE;
	while ((!ketemu) && (i<=j))
	{
		k=(i+j)/2;
		if (L[k]==X)
			ketemu=TRUE;
		else
		{
			if(L[k]<X)
				i=k+1;
			else
				j=k-1;
		}
	}
	if (ketemu)
		*idx = k+1;
	else
		*idx =-1;
}
```

Pada program diatas terdapat procedure sorting, hal ini diperlukan karena kemungkinan user akan memasukkan nilai yang tidak terurut, padahal syarat pencarian biner adalah data harus terurut terlebih dahulu.

7. Implementasi Pencarian Biner untuk data terurut menaik ke data bentukan

```
#include <stdio.h>
#define TRUE  1
#define FALSE 0
typedef int bool;
struct dataMahasiswa
{
	char nim[10];
	char nama[20];
};

void inputDataMhs(struct dataMahasiswa mhs[]);
void BinSearch(struct dataMahasiswa L[],int N, char
X[10],int *idx);
void bubbleSort(struct dataMahasiswa mhs[],int N);
```

```
void main()
{

	int pos;
	char pil;
	char nimCari[10];
	struct dataMahasiswa mhs[5];
	//panggil function input data
	inputDataMhs(mhs);
	//panggil fungsi sorting
	bubbleSort(mhs,5);
	while(1)
	{
		//user input elemen yang dicari
		printf("\nNim yang dicari : ");
		scanf("%s",&nimCari);
		//menjalankan pencarian
		BinSearch(mhs,5,nimCari,&pos);
		if (pos!=-1)
		{
			printf("\nData Mahasiwa Ketemu");
			printf("\nNim       siswa              :
%s",mhs[pos].nim);
			printf("\nNama          siswa          :
%s",mhs[pos].nama);
		}
		else
			printf("\nData      Mahasiwa      Tidak
Ketemu");
		printf("\nCari data lagi [y/n] ? ");
		scanf("%s",&pil);
		if ((pil=='y')||(pil=='Y'))
			continue;
		else
			break;
	}
}

void inputDataMhs(struct dataMahasiswa mhs[])
{
	int i;
	printf("\nMasukkan Data Mahasiswa\n");
	//user input elemen mahasiswa
```

```
	for(i=0;i<5;i++)
	{
		printf("\nData Mahasiswa ke %d : \n",i+1);
		printf("Nim : ");
		scanf("%s",mhs[i].nim);
		printf("Nama : ");
		scanf("%s",mhs[i].nama);
	}
}

void bubbleSort(struct dataMahasiswa mhs[],int N)
{
	int a,b;
	struct dataMahasiswa temp;
	//proses sortir dengan bubble sort
	for(a=0;a<=(N-2);a++)
	{
		for(b=(N-1);b>=(a+1);b--)
		{
			if	(strcmp(mhs[b-1].nim,mhs[b].nim)
>0)
			{
				temp = mhs[b-1];
				mhs[b-1]= mhs[b];
				mhs[b] = temp;
			}	//endif
		} //end loop j
	}  //end loop i
}

void BinSearch(struct dataMahasiswa L[],int N, char
X[10],int *idx)
{
	int i,j,k;
	bool ketemu;
	i=0;
	j=N;
	ketemu = FALSE;
	while ((!ketemu) && (i<=j))
	{
		k=(i+j)/2;
		if (strcmp(L[k].nim,X)==0)
			ketemu=TRUE;
		else
```

```
            {
                  if(strcmp(L[k].nim,X) < 0)
                        i=k+1;
                  else
                        j=k-1;
            }
      }
      if (ketemu)
            *idx = k;
      else
            *idx =-1;
}
```

Berikut adalah penerapan pencarian biner dalam bahasa pemrograman C untuk data terurut **menurun** :

8. Pencarian Biner untuk data terurut menurun yang mengembalikan posisi data

```
#include <stdio.h>
#define TRUE  1
#define FALSE 0
typedef int bool;

void BinSearch(int L[7],int N, int X, int *idx);

void main()
{
      int pos,arr[7] = {45,22,16,10,6,2,0};
      BinSearch(arr,7,16,&pos);
      if(pos!= -1)
            printf("Ketemu di posisi ke-%d",pos);
      else
            printf("Tidak Ketemu");
}

void BinSearch(int L[7],int N, int X, int *idx)
{
      int i,j,k;
      bool ketemu;
      i=0;
      j=N;
      ketemu = FALSE;
      while ((!ketemu) && (i<=j))
```

```
        {
                k=(i+j)/2;
                if (L[k]==X)
                        ketemu=TRUE;
                else
                {
                        if(L[k]<X)
                                j=k-1;
                        else
                                i=k+1;
                }
        }
        if (ketemu)
                *idx = k+1;
        else
                *idx =-1;
}
```

Prosedur di atas bekerja untuk pencarian data di dalam larik, dimana data terurut menurun (dari besar ke kecil). Jika nilai ditemukan dalam larik, prosedur mengembalikan nilai idx (indeks elemen larik yang nilainya sama dengan nilai yang dicari). Jika semua elemen larik sudah dibandingkan dengan nilai yang dicari namun tidak ditemukan, prosedur akan mengembalikan nilai `idx←-1` (penanda bahwa nilai yang dicari tidak ditemukan dalam larik).

Prosedur dimulai dengan inisialisasi pencacah (`i←0`) dan menyimpan jumlah elemen larik dalam variabel j. Variabel `ketemu` akan diberi nilai `false`, hal ini memberitahukan bahwa data yang dicari belum ditemukan. Perulangan diawali dengan membandingkan elemen tengah (elemen dengan indeks k) dengan nilai data yang dicari.

- Jika elemen larik yang dibandingkan bernilai sama dengan data yang dicari, maka variabel boolean ketemu bernilai `true` dan prosedur mengembalikan nilai indeks elemen tersebut.
- Jika elemen larik bernilai lebih kecil dari data yang dicari, maka pencarian akan bergeser ke kiri dengan cara mengubah nilai i (awal pencacah) dengan indeks di sebelah kanan nilai tengah (`j←k-1`).
- Jika elemen larik bernilai lebih besar dari data yang dicari, maka pencarian akan bergeser ke kanan dengan cara mengubah nilai i (awal pencacah) dengan indeks di sebelah kiri nilai tengah (`i←k+1`).

Selanjutnya, perulangan terus dilakukan sampai `ketemu` bernilai true atau nilai `i>j` (semua elemen larik sudah dibandingkan dengan nilai yang dicari). Jika semua elemen larik sudah dibandingkan dengan nilai yang dicari tetapi nilai tidak ditemukan, maka nilai ketemu akan tetap bernilai false

sehingga akan dikembalikan nilai index= -1 (penanda bahwa nilai yang dicari tidak ditemukan dalam larik).

9. Pencarian Biner untuk data terurut menurun berdasar inputan user

```
#include <stdio.h>
#define TRUE  1
#define FALSE 0
typedef int bool;

void BinSearch(int L[10],int N,int X,int *idx);
void inputData(int arr[]);
void bubbleSort(int arr[],int N);

void main()
{
      int pos,cari,arr[10];
      inputData(arr);
      //panggil fungsi sort
      bubbleSort(arr,10);
      //user input elemen yang dicari
      printf("\nElemen yang dicari : ");
      scanf("%d",&cari);
      //menjalankan pencarian
      BinSearch(arr,10,cari,&pos);
      if(pos!=-1)
            printf("Ketemu di posisi ke-%d",pos);
      else
            printf("Tidak Ketemu");
}
void inputData(int arr[])
{
      int i;
      printf("\nMassukkan Elemen Array\n");
      //user input elemen array
      for(i=0;i<10;i++)
      {
            printf("Array ke %d : ",i+1);
            scanf("%d",&arr[i]);
      }
}
void bubbleSort(int arr[],int N) //sort descending
```

```
{
      int a,b,temp;
      //proses sortir dengan bubble sort
      for(a=0;a<=(N-2);a++)
      {
            for(b=(N-1);b>=(a+1);b--)
            {
                  if (arr[b-1] < arr[b])
                  {
                        temp = arr[b-1];
                        arr[b-1]= arr[b];
                        arr[b] = temp;
                  }     //endif
            } //end loop j
      }  //end loop i
      //tampil data urut
      printf("\nData terurut");
      for(a=0;a<N;a++)
      {
            printf("\nArray ke-%d : ",a+1);
            printf("%d",arr[a]);
      }

}
//search descending
void BinSearch(int L[10],int N, int X,int *idx)
{
      int i,j,k;
      bool ketemu;
      i=0;
      j=N;
      ketemu = FALSE;
      while ((!ketemu) && (i<=j))
      {
            k=(i+j)/2;
            if (L[k]==X)
                  ketemu=TRUE;
            else
            {
                  if(L[k]<X)
                        j=k-1;
                  else
```

```
                        i=k+1;
            }
      }
      if (ketemu)
            *idx = k+1;
      else
            *idx =-1;
}
```

10. Implementasi Pencarian Biner untuk data terurut menurun ke data bentukan

```
#include <stdio.h>
#define TRUE  1
#define FALSE 0
typedef int bool;
struct dataMahasiswa
{
      char nim[10];
      char nama[20];
};

void inputDataMhs(struct dataMahasiswa mhs[]);
void BinSearch(struct dataMahasiswa L[],int N, char
X[10],int *idx);
void  bubbleSortDown(struct  dataMahasiswa  mhs[],int
N);

void main()
{

      int pos;
      char pil;
      char nimCari[10];
      struct dataMahasiswa mhs[5];
      //panggil function input data
      inputDataMhs(mhs);
      //panggil fungsi sorting
      bubbleSortDown(mhs,5);
      while(1)
      {
            //user input elemen yang dicari
            printf("\nNim yang dicari : ");
            scanf("%s",&nimCari);
            //menjalankan pencarian
            BinSearch(mhs,5,nimCari,&pos);
            if (pos!=-1)
            {
                  printf("\nData Mahasiswa Ketemu");
                  printf("\nNim      siswa            :
%s",mhs[pos].nim);
                  printf("\nNama        siswa        :
```

```
%s",mhs[pos].nama);
            }
            else
                  printf("\nData     Mahasiswa     Tidak
Ketemu");
            printf("\nCari data lagi [y/n] ? ");
            scanf("%s",&pil);
            if ((pil=='y')||(pil=='Y'))
                  continue;
            else
                  break;
      }
}

void inputDataMhs(struct dataMahasiswa mhs[])
{
      int i;
      printf("\nMasukkan Data Mahasiswa\n");
      //user input elemen mahasiswa
      for(i=0;i<5;i++)
      {
            printf("\nData    Mahasiswa    ke    %d    :
\n",i+1);
            printf("Nim : ");
            scanf("%s",mhs[i].nim);
            printf("Nama : ");
            scanf("%s",mhs[i].nama);
      }
}

void  bubbleSortDown(struct  dataMahasiswa  mhs[],int
N)
{
      int a,b;
      struct dataMahasiswa temp;
      //proses sortir dengan bubble sort
      for(a=0;a<=(N-2);a++)
      {
            for(b=(N-1);b>=(a+1);b--)
            {
                  if   (strcmp(mhs[b-1].nim,mhs[b].nim)
<0)
```

```
                {
                        temp = mhs[b-1];
                        mhs[b-1]= mhs[b];
                        mhs[b] = temp;
                }    //endif
            } //end loop j
        }  //end loop i
}

void BinSearch(struct dataMahasiswa L[],int N, char
X[10],int *idx)
{
      int i,j,k;
      bool ketemu;
      i=0;
      j=N;
      ketemu = FALSE;
      while ((!ketemu) && (i<=j))
      {
            k=(i+j)/2;
            if (strcmp(L[k].nim,X)==0)
                  ketemu=TRUE;
            else
            {
                  if(strcmp(L[k].nim,X) < 0)
                        j=k-1;
                  else
                        i=k+1;
            }
      }
      if (ketemu)
            *idx = k;
      else
            *idx =-1;
}
```

7.4 *Pencarian Lain*

Pencarian sekuensial dan pencarian biner merupakan algoritma pencarian dasar yang termasuk ke dalam kelompok pencarian daftar (*list*

search). Terdapat pula beberapa algoritma lain yang termasuk pula dalam kelompok pencarian daftar, antara lain:

- pencarian interpolasi (*interpolation search*): melakukan pencarian lebih baik daripada pencarian biner pada larik berukuran besar dengan distribusi seimbang, tapi waktu pencariannya buruk
- pencarian Grover (*Grover's search*): melakukan pencarian dalam waktu singkat, yang merupakan pengembangan dari pencarian linier biasa pada larik dengan elemen tidak berurut

Selain algoritma pencarian dalam kelompok pencarian daftar, terdapat pula beberapa kelompok algoritma lain. Beberapa di antaranya adalah sebagai berikut:

Kelompok Algoritma	Penjelasan dan Contoh Algoritma
Pencarian tanpa informasi (*uninformed search*)	Algoritma ini mencari data tanpa ada batasan tipe data persoalan.
Pencarian pohon (*tree search*)	Algoritma ini mencari data dengan bantuan struktur pohon (eksplisit maupun implisit). • *breadth-first search* (mencari level demi level) • *depth-first search* (mencari dengan meraih kedalaman pohon terlebih dahulu) • *iterative-deepening search* • *depth-limited search* • *bidirectional search* • *uniform-cost search*
Pencarian grafik (*graph search*)	Algoritma ini mencari data dengan algoritma penelurusuran grafik, misalnya • *Djikstra's algorithm* • *Kruskal's algorithm* • *nearest neighbour algorithm* • *Prim's algorithm*
Pencarian dengan informasi (*informed search*)	Algoritma ini mencari data dengan fungsi heuristik yang spesifik pada persoalan tertentu. • *best-first search* • *A**
Jenis lain	• Algoritma pencarian string • Algoritma genetik • Algoritma *minimax*

Rangkuman

1. Algoritma pencarian adalah algoritma yang mengambil input berupa persoalan dan mengembalikan penyelesaian berupa penemuan nilai yang dicari dalam persoalan inputan.
2. Dua contoh algoritma pencarian dasar adalah pencarian sekuensial dan pencarian biner.
3. Pencarian sekuensial (*sequential search*) membandingkan setiap elemen larik (array) satu persatu dengan nilai yang dicari secara beruntun, mulai dari elemen pertama sampai elemen yang dicari sudah ditemukan, atau sampai seluruh elemen sudah diperiksa.
4. Pencarian biner adalah proses mencari data dengan membagi data atas dua bagian secara terus menerus sampai elemen yang dicari sudah ditemukan.
5. Pencarian biner mempunyai prasyarat yaitu data harus terurut baik menaik atau menurun.

Latihan

1. Buatlah program kamus bahasa inggris – bahasa indonesia, minimal 10 kata. Inputan berupa kata dalam bahasa inggris dan output berupa arti kata dalam bahasa indonesianya. Gunakan proses pencarian dengan Sequential Search.
   Hint : Gunakan tipe data bentukan

```
struct kamus
{
      char inggris[20];
      char indonesia[20];
};
```

2. Pecahkan persoalan pada no.1 dengan menggunakan Binary Search
3. Buatlah program untuk pencarian data Barang pada suatu supermarket. Data barang akan diinputkan oleh user. Tiap barang mempunyai atribut : Kode Barang, Nama Barang, dan Harga Barang. Pencarian dilakukan dengan user input Kode Barang kemudian ditampilkan data Nama Barang dan Harganya.
4. Tambahkan pada soal no 3, yaitu fitur untuk update harga barang. Setelah dilakukan pencarian, user diminta untuk memasukkan harga barang yang baru, kemudian harga yang baru disimpan.
5. Tambahkan pada soal no 3, yaitu fitur untuk delete barang. Setelah dilakukan pencarian kode barang kemudian hapus data barang tersebut.
6. Gabungkan soal no 3, 4, dan 5 sehingga program tersebut akan menampilkan menu :
   MENU :
   1. Input Data
   2. Update Data
   3. Delete Data
   4. Search Data

   Menu input data digunakan untuk memasukkan data barang pada posisi yang masih kosong.
   Menu update data digunakan untuk mengubah data harga barang sesuai kodenya
   Menu delete data digunakan untuk menghapus data barang
   Menu search data digunakan untuk menampilkan data barang.