Tugas elearning

Nama :Romindo Sinaga
Kelas : TIF 16 D
Mata kuliah : Alpro 2
NPM : 16 111 340

1. Sequential Search

Sequential Search merupakan metode pencarian data dalam array dengan cara membandingkan data yang dicari dengan data yang ada di dalam array secara berurutan.

```
//ini adalah yang si sebut kepala program. dalam c++ kita juga dapat
//membuat liblary sendiri sesuai dengan yang di perlukan
#include <iostream>
using namespace std;

#include <conio.h>
#include <iomanip>

int main()//main program
{
// Deklarasi data bertipe integer variabel dataku dengan nilai indeks 10
      int dataku[10] = {7,9,2,10,15,4,5,6,13,11};
// Deklarasi data bertipe integer variabel cari data,i,flag=0
      int caridata, i, flag = 0;
// Output pada layar judul program

      cout<<"PENCARIAN DENGAN SEQUENTIAL SEARCH"<<endl;
      cout<<"----------------------------------"<<endl;
      cout<<"Data    : ";
//Baca elemen array
       for(int n=0; n<10; n++)
                  cout<<setw(4)<<dataku[n];
      cout<<endl;

      cout<<"\nMasukkan data yang ingin Anda cari : ";
      cin>>caridata;

 //cari dengan metode sequential search()
 //proses
      for(i = 0; i<10; i++)
      {
            if(dataku[i]==caridata)
            {
                  flag = 1;
//digunakan untuk keluar dari suatu blok kode, disini break digunakan untuk keluar dai perualangan for

                  break;
            }
      }

      //cetak hasil
      if(flag==1)
            cout<<"Data ditemukan pada indek ke-"<<i<<endl;
      else
            cout<<"Data tidak ditemukan"<<endl;
// Setelah pengurutan berhasil maka nilai akan dicetak  atau  ditampilkan pada baris ini.
      _getche();
      return EXIT_SUCCESS;
}
```

2.Binary Search

Metode pencarian Binary yaitu mencari data dengan melakukan mengelompokkan array menjadi bagian-bagian. Binary Search ini hanya dapat diimplementasikan pada data yang telah terurut baik ascending maupun descending dalam suatu array.

```
Algoritma binary search :
1.Data diambil dari posisi 1 sampai posisi akhir n
2.Kemudian cari posisi data tengah dengan rumus: (posisi awal +
posisi akhir) / 2
3.Kemudian data yang dicari dibandingkan dengan data yang di
tengah,apakah sama atau lebih kecil, atau lebih besar?
4.Jika lebih besar, maka proses pencarian dicari dengan posisi awal
adalah posisi tengah + 1
5.Jika lebih kecil, maka proses pencarian dicari dengan posisi akhir
adalah posisi tengah - 1
6.Jika data sama, berarti ketemu
```

```
#include <iostream>
//ini adalah yang si sebut program utama.

using namespace std;
//tidak semua kompiler menggunakan ini,

#include <conio.h>//perintah untuk membersihkan layar

#include <iomanip>//diperlukan bila melibatkan setw() yang bermanfaat
//untuk mengatur lebar dari suatu tampilan

int main()//main program mengembalikan nilai int secara default
{
      int data[10];
   // Deklarasi data bertipe integer dengan maksimal nilai indeks 10

   int cari;// Deklarasi cari bertipe integer

      cout<<"\t   'BINARY SEARCH'"<<endl;// Output pada layar judul
program

   cout<<"\nMasukkan 10 Data : ";
   // Output pada layar untuk perintah memasukkan data awal

   for(int x = 0; x<10; x++)//Baca elemen array
            cin>>data[x];
```

```
//  Menginputkan data
           cout<<"\nMasukkan data yang ingin Anda cari : ";
   // Output pada layar untuk perintah memasukkan data yang akan
dicari

   cin>>cari; //  Menginputkan data
      cout<<"\nData diurutkan : "; // Output pada layar

      for(int x = 0; x<10;x++) //Baca elemen array
            cout<<setw(3)<<data[x];
   // Output pada layar untuk menampilkan data setelah diurutkan
     cout<<endl;
      int awal, akhir, tengah, b_flag = 0;
   // Deklarasi awal,akhir,tengah bertipe integer dan

   awal = 0; //Mula-mula diambil posisi awal 0
      akhir = 10; // posisi akhir = 10

   while (b_flag == 0 && awal<=akhir)//kondisi
      {
            tengah = (awal + akhir)/2;
   //kemudian dicari posisi data tengah dengan
   //rumus (posisi awal + posisi akhir) / 2

   if(data[tengah] == cari)
   //Kemudian data yang dicari dibandingkan dengan data tengah.
            {
                  b_flag = 1; // turn b_flag on
                  break; //proses perulangan berhenti
            }
            else if(data[tengah]<cari) //Jika lebih kecil
      awal = tengah + 1;
      //proses dilakukan kembali tetapi posisi akhir dianggap
      //sama dengan posisi tengah +1.
            else
                  akhir = tengah -1;
                  //Jika lebih besar, proses dilakukan kembali tetapi
      //posisi awal dianggap sama dengan posisi tengah -1.
      }
      if(b_flag == 1) //Jika data sama, berarti ketemu
            cout<<"\nData ditemukan pada index ke-"<<tengah<<endl;
            // Output pada layar untuk menampilkan indeks data

   else //Jika data tidak sama, berarti tidak ketemu
            cout<<"\nData tidak ditemukan\n";
            // Output pada layar untuk menampilkan bahwa data tidak
ditemukan

      return 0; //memberitahu kepada sistem operasi bahwa program
telah berakhir
  }
```

3. BUBBLE SORT

Bubble sort adalah salah satu metode sorting atau mengurutkan dari data terkecil ke data terbesar ataupun dengan cara membandingkan elemen kesatu dengan elemen yang selanjutnya

```
#include <iostream>
using namespace std;
int main()
{   int a,k,c,d,g;        >>>>>  tipe data yang digunakan (yang berhubungan dengan variabel)
    k=4;
    int b[4];

// proses inputan data dalam array
    cout<<"mengurutkan nilai dari besar ke kecil"<<endl<<endl;
    for(a=0;a<k;a++)
    {
       cout<<"Masukkan nilai "<<a+1<<" : ";cin>>b[a];
    }

    for(a=0;a<k-1;a++)                  // Proses pengurutan antara elemen satu dengan yang lain dan
                                         apabila elemen satu lebih kecil daripada elemen berikutnya
                                        (mengurtkan besar ke kecil) maka proses pertukaran akan
                                        terjadi pada pada baris 21- 23.
    {

       for(d=a+1;d<k;d++)
       {
       c=a;
          if(b[c]<b[d])
          {
             c=d;
          }
       g=b[c];
       b[c]=b[a];// pertukaran data jika yang ditukar lebih kecil dari pada elemen berikutnya
       b[a]=g;
       }

    }
    cout<<"\n setelah diurutkan akan menjadi : \n";
    for(a=0;a<k;a++)
    {
       cout<<b[a]<<" \n";               //// Setelah pengurutan berhasil maka nilai akan dicetak/
                                          ditampilkan pada baris ini.
    }
}
```

4. Selection Sort

Metode pengurutan ini didasarkan pada pemilihan elemen maksimum atau minimum kemudian mempertukarkan elemen maksimum-minimum tersebut dengan elemen terujung larik (elemen ujung kiri atau elemen ujung kanan), selanjutnya elemen terujung itu kita “isolasi” dan tidak diikut sertakan pada proses selanjutnya. Karena proses utama dalam pengurutan adalah pemilihan elemen maksimum/minimum, maka metode ini disebut metode pemilihan (selection)

Contoh programnya adalah :

```
//kepala program
#include <iostream.h>
#include <conio.h>
#include <iomanip.h>
int main(){
//deklarasi array dengan 7 elemen
int A[7];
int j,k,i,temp;
int jmax,u=6;
//memasukkan nilai sebelum diurutkan
cout<<“Masukkan nilai pada elemen array :”<<endl;
for(i=0;i<7;i++)
{
cout<<“A[“<<i<<“]=”;
cin>>A[i];
}
//Proses pengurutan (Ascending)
for(j=0;j<7;j++)
{
jmax=0;
for(k=1;k<=u;k++)
if (A[k] > A[jmax])
jmax=k;
temp=A[u];
A[u]=A[jmax];
A[jmax]=temp;
u–;
}
//menampilkan nilai setelah diurutkan
cout<<“\nNilai setelah diurutkan =”<<endl;
for(i=0;i<7;i++)
cout<<A[i]<<”  “;
getch();
}
```