Universitas Padjadjaran

ARRAY/LARIK

Mira Suryani, S.Pd., M.Kom
S-1 Teknik Informatika
Jatinangor, 8 November 2018

Tujuan

- Mahasiswa dapat menjelaskan pengertian array dan bisa menerangkan operasi dasar menggunakan array dengan benar
- Mahasiswa dapat : Mengoperasikan dan membuat program dari semua algoritma array dengan benar.

Pokok Bahasan

- Array 1 dimensi
- Array 2 dimensi (Matriks)
- Sorting
- Searching

Array / Larik

- Array adalah suatu tipe data bentukan yang menyimpan sekumpulan elemen data yang bertipe sama, dan memiliki indeks.
- Indeks array harus tipe data yang menyatakan keterurutan, misalnya *integer* atau karakter
- Penyimpanan di memori secara kontinyu.

Array 1 Dimensi

- Setiap elemen larik ditulis dengan notasi :

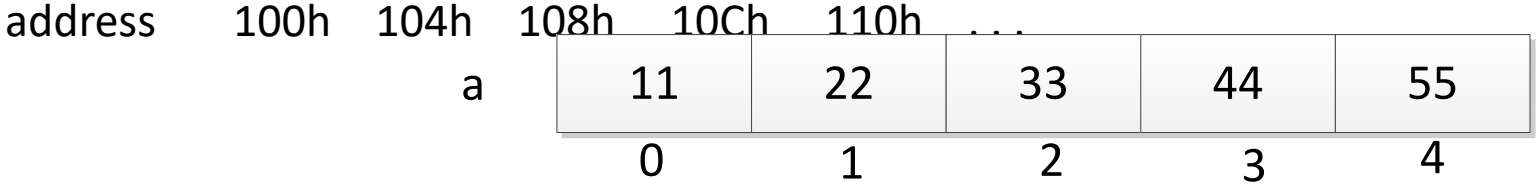
  n-1

  ∑ A[i] = A[0], A[1], A[2], A[3], A[4], …. , A[n-1]

  i=0

  Angka di dalam kurung siku menyatakan indeks array yang dimulai dari 0 sampai dengan (n-1) yang diinginkan.

Gambaran array of integer



Deklarasi

Pendeklarasian array dengan cara :

<tipe> variabel[indeks];

Contoh : int a[5];

```
main() {
   int data[5];
   int n;
   cout<<"Banyak data : "; cin>>n;

   for (int i=0; i<n; i++) {                    // isi data array
        cout<<"Data  : "; cin>>data[i];
   }

   for(int i=0; i<n; i++) {                     // print array
       cout<<"Data"<<i+1<<" = "<<data[i]<<endl;
   }
}
```

Pendeklarasian Array menggunakan Alias (typedef)

- Untuk bisa membuat program modular dengan melewatkan (passing) array baik by value atau by referensi maka disarankan untuk membuat cara sebagai berikut:
  - Definisikan suatu tipe alias dari data bertipe array
  - Variabel array kemudian mengacu ke tipe alias tersebut

- Contoh :

```
typedef int  larik[10];
larik  A;
```

A merupakan sebuah variabel yang bertipe array of integer dengan ukuran maksimal 10 elemen dengan index 0 .. 9

Contoh :

```
typedef int larik[10];
main() {
    larik data;
    int n;
    cout<<"Banyak data : "; cin>>n;

    for (int i=0;i<n;i++) {              // isi
         cout<<"Data  : "; cin>>data[i];
    }

    for(int i=0;i<n;i++) {               // print
        cout<<"Data "<<i+1<<" = "<<data[i] << endl;
    }
}
```

Inisialisasi Array

Dalam C++, sebuah array dapat diinisialisasi dengan cara sebagai berikut:

```
int a[5] = {11, 22, 33, 44, 55};
```

Perintah ini equivalent dengan:

```
int a[]  = {11, 22, 33, 44, 55};
```

dimana ukuran array akan menyesuaikan dari daftar nilai inisialisasi.

```
main() {

    int data[]  = {11, 22, 33, 44, 55};
    for(int i=0;i<n;i++) {
       cout<<"Data "<<i+1<<" = "<<data[i] << "  ";
    }
}
```

Koreksi Inisialisasi

```
main() {
   int data[]  = {11, 22, 33, 44, 55};

   for(int i = 0; i < sizeof(*data); i++) {
     cout<<"Data "<<i+1<<" = "<< data[i] << "\n"  ;
   }
}
```

Cara menampilkan data dari array dengan cara lain?

Passing Array ke Fungsi

1. Cara 1

Menggunakan parameter di fungsi dengan bentuk sbb :

TipeHasil namaFungsi (tipeLarik namaLarik[], ….)

Pemanggilan

Hasil = namaFungsi(namaLarik, …)

Atau

void namaFungsi(tipeLarik namaLarik[], …)

Pemanggilan

namaFungsi(namaLarik, …)

```
void banyakData(int& n);
void isiLarik(int a[], int n);
void printLarik(int a[], int n);
main() {
  int x[10];        // variabel array x sebanyak maks=10
  int n;
  banyakData(n);
  isiLarik(x,n);
  printLarik(x,n);
}
```

Passing Array cara 1

Terlihat bahwa fungsi :

```
void isiLarik(int a[], int n)
```

dipanggil dengan cara :

```
int x[10];
isiLarik(x,n);
```

- Pengiriman nilai dengan cara ini merupakan pengiriman alamat awal dari memori dimana array berada.
- Fungsi dapat mengubah isi dari array dengan mengakses secara langsung dimana elemen array disimpan.
- Artinya : **walaupun pengiriman dilakukan secara value tetapi elemen-elemen dapat diubah sehingga sebenarnya dilakukan pengiriman secara reference.**

2. Cara 2

Menggunakan parameter di fungsi bertipe array alias

typedef <tipe> tipeAlias[indeks];

TipeHasil namaFungsi (tipeAlias namaLarik, ….)

Pemanggilan

Hasil = namaFungsi(namaLarik, …)

Atau

void namaFungsi(tipeAlias namaLarik, …)

Pemanggilan

namaFungsi(namaLarik, …)

- Cara ini **lebih disarankan** untuk lebih memudahkan dalam memahami konversi dari notasi algoritmik ke pemrograman.
- Bisa dilihat dalam fungsi nanti apakah passing array by value atau by reference.

```
typedef int larik[10];
void banyakData(int& n);
void isiLarik(larik& a, int n);
void printLarik(larik a, int n);
main() {
  larik x;        // variabel array x
  int n;
  banyakData(n);
  isiLarik(x,n);
  printLarik(x,n);
}
```

Terlihat bahwa fungsi :

void isiLarik(larik& a, int n)

dipanggil dengan cara :

larik x;

isiLarik(x,n);

```
void banyakData(int& n){                // Input banyak data
   cout<<"Banyak data : "; cin>>n;
}


void isiLarik(larik& a, int n){         // Input data larik
  for (int i=0;i<n;i++) {
     cout<<"Masukkan data ke- "<<(i+1)<<" : "; cin>>a[i];
  }
}


void printLarik(larik a, int n){    //Mencetak data larik
  cout <<"Data yang sudah dimasukkan" <<endl;
  cout <<"-----------------------------"<<endl;
  for(int i=0;i<n;i++) {
     cout <<"Data ke-"<<(i+1)<<" = "<< a[i] <<endl;
  }
}
```

contoh fungsi untuk mencari nilai rata-rata dan mencari nilai tertinggi

```
void cariRata(larik a, int n, float& rata){
        float jumlah=0;                 // Ubah jadi fungsi ?
        for (int i=0;i<n;i++) {
                jumlah=jumlah+a[i];
        }
        rata=jumlah/n;
}

int maksimum (larik a, int n){    // Ubah jadi void?
        int maks = -999;                  // data sentinel
        for (int i = 0; i < n; i++) {
                if (maks < a[i]) {
                        maks = a[i];
                }
        }
        return (maks);
}
```

- Pemanggilan : ???

String / Array of character

- Pada C++ tidak ada tipe variable elemen yang spesific untuk menyimpan string. Untuk keperluan ini dapat digunakan array dengan tipe char, dimana berisi elemen dengan tipe char. Perlu di ingat bahwa tipe char digunakan untuk menyimpan 1 karakter, karena itu array dari char digunakan untuk menyimpan string.
- Contoh :
  - char nama [20];
  - Perhatikan, karakter NULL (” atau ‘\0’) selalu disertakan diakhir string untuk indikasi akhir dari string.

- Inisialisasi String
  - char mystring [] = { ‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘’ };
  - char mystring [] = { ‘H’, ‘e’, ‘l’, ‘l’, ‘o’, ‘\0’ };
  - char mystring [] = “Hello”;

Pemberian nilai pada string

1. Menggunakan fungsi strcpy.
   - strcpy (string copy) mendefinisikan cstring (string.h) library dan dapat dipanggil dengan cara :
     - strcpy (string1, string2);
     - isi dari string2 di-copy ke string1.
     - string2 dapat berupa array, pointer, atau konstanta string.
   - Contoh:
     - char myName[20];
     - strcpy (myName,"Asep Dayat");
2. menggunakan cin, dengan metode getline :
   - cin.getline(char buffer[],int length,char delimiter = '\n');

   Contoh:
     - char mybuffer [100];
     - cout << "What's your name? ";
     - cin.getline (mybuffer,100);
3. Menggunakan cin >> string
   - Ada keterbatasan hanya bisa satu kata

Konversi string ke tipe lainnya

- atoi: converts string to int type.
- atol: converts string to long type.
- atof: converts string to float type.
- Contoh :

```
main (){
        char mybuffer [100];
        float price;
        int quantity;
        cout << “Enter price: “;
        cin.getline (mybuffer,100);
        price = atof (mybuffer);
        cout << “Enter quantity: “;
        cin.getline (mybuffer,100);
        quantity = atoi (mybuffer);
        cout << “Total price: ” << price*quantity;
}
```

Fungsi untuk manipulasi string

No.	Function & Purpose
1	strcpy(s1, s2); Copies string s2 into string s1.
2	strcat(s1, s2); Concatenates string s2 onto the end of string s1.
3	strlen(s1); Returns the length of string s1.
4	strcmp(s1, s2); Returns 0 if s1 and s2 are the same; less than 0 if s1<s2; greater than 0 if s1>s2.
5	strchr(s1, ch); Returns a pointer to the first occurrence of character ch in string s1.
6	strstr(s1, s2); Returns a pointer to the first occurrence of string s2 in string s1.

Array 2 Dimensi (Matriks)

- Deklarasi Sebagai variabel
  - <tipe> variabel[indBaris][indKolom];

- Contoh :
  - int x[5][5];

0,0	0,1	0,2	0,3	0,4
1,0	1,1	1,2	1,3	1,4
2,0	2,1	2,2	2,3	2,4
3,0	3,1	3,2	3,3	3,4
4,0	4,1	4,2	4,3	4,4

Deklarasi Matriks

- Cara Deklarasi Matriks untuk membuat Program modular
  - Definisikan suatu tipe alias dari data bertipe matriks
  - Variabel matriks kemudian mengacu ke tipe alias tersebut

```
typedef int  matriks[10][10];
main() {
  matriks x;          // variabel array x bertipe matriks
  int nBaris, nKolom;
  banyakData(nBaris, nKolom);
  isiMatriks(x,nBaris, nKolom);
  cetakMatriks(x,nBaris, nKolom);
}
```

Catatan : Jika menggunakan compiler MinGW maka pendeklarasian tipe alias dan pengaksesan matriks bisa juga sbb :

typedef <tipe> tipeAlias[bykBaris,bykKolom];

tipeAlias <variabel>;

Hal ini sama seperti pengaksesan menggunakan Bahasa Pascal yaitu :

a[x , y]

```
void banyakData(int& nBaris, int& nKolom){
        cout << "Banyak baris : "; cin >> nBaris;
        cout << "Banyak kolom : "; cin >> nKolom;
}

void isiMatriks(matriks& x, int nBaris, int nKolom){
     for (int i=0; i < nBaris; i++) {
        for (int j=0; j < nKolom; j++) {
                cout<<"Data ke-["<<i+1<<","<<j+1 <<"] : ";
              cin >> x[i][j];
        }
     }
}

void cetakMatriks(matriks x, int nBaris, int nKolom){
        cout << "\nPencetakan Matriks :"<<endl;
        for (int i=0; i < nBaris; i++) {
                for (int j=0; j < nKolom; j++) {
                        cout << x[i][j] << "  ";
                }
                cout << endl;
        }
}
```

Sorting

- Sorting bisa didefinisikan sebagai suatu pengurutan data yang sebelumnya tersusun secara acak menjadi data yang tersusun secara teratur menurut aturan tertentu.
- Urutan dari data yang kecil ke yang lebih besar (menaik) disebut dengan Ascending dan yang menurun disebut dengan Descending.

Bubble Sort

- Diberi nama "Bubble" karena proses pengurutan secara berangsur-angsur bergerak/berpindah ke posisi yang tepat , seperti gelembung yang keluar dari sebuah gelas bersoda.
- Bubble sort mengurutkan data dengan cara membandingkan elemen sekarang dengan elemen berikutnya.
- jika elemen sekarang lebih besar dari elemen berikutnya maka elemen tersebut ditukar (untuk pengurutan ascending)
- algoritma ini seolah olah menggeser satu per satu elemen dari kenan ke kiri atau kiri ke kanan, tergantung jenis pengurutannya.
- Ketika suatu tahap proses telah selesai, maka bubble sort akan mengalami proses selanjutnya, demikian seterusnya.
- Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan,serta tercapai pengurutan yang telah diinginkan

Contoh : Dari data 22 10 15 3 8 2

Proses 1 :

22	10	15	3	8	2
10	22	15	3	8	2
10	15	22	3	8	2
10	15	3	22	8	2
10	15	3	8	22	2
10	15	3	8	2	22

Pengecekan dimulai dari data yang paling awal, kemudian dibandingkan dengan data di depannya,jika data didepannya lebih besar maka akan di tukar.

Proses 2:

10	15	3	8	2	22
10	15	3	8	2	22
10	3	15	8	2	22
10	3	8	15	2	22
10	3	8	2	15	22

pengecekan dilakukan sampai dengan data yang kedua terakhir karena data terakhir pasti sudah paling besar.

Contoh : Dari data 22 10 15 3 8 2

Proses 3 :

10	3	8	2	15	22
3	10	8	2	15	22
3	8	10	2	15	22
3	8	2	10	15	22

Proses 4 :

3	8	2	10	15	22
3	8	2	10	15	22
3	2	8	10	15	22

Proses 5 :

3	2	8	10	15	22
2	3	8	10	15	22

Fungsi Bubble Sort

```
void swap (int& x, int& y){
     int temp = x;
     x = y;
     y = temp;
}

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

- Tentukan algoritma 2 bentuk lain? (lihat diktat)

Bubble Sort (cara 2)

Dari data 22 10 15 3 8 2

- Proses 1 :

22	10	15	3	8	2
22	10	15	3	2	8
22	10	15	2	3	8
22	10	2	15	3	8
22	2	10	15	3	8
2		22	10	15	3	8

- Pengecekan dimulai dari data yang paling akhir, kemudian dibandingkan dengan data di depannya,jika data didepannya lebih besar maka akan di tukar.

Bubble Sort (cara 2)

- Proses 2:
- 2 22 10 15 3 8
  2 22 10 15 3 8
  2 22 10 3 15 8
  2 22 3 10 15 8
  2 3 22 10 15 8

  pengecekan dilakukan sampai dengan data ke-2 karena data pertama pasti sudah paling kecil.

Bubble Sort (cara 2)

- Proses 3 :
- 2 3 22 10 15 8
  2 3 22 10 8 15
  2 3 22 8 10 15
  2 3 8 22 10 15

  Proses 4:
- 2 3 8 22 10 15
  2 3 8 22 15 10
  2 3 8 15 22 10

- Proses 5 :
- 2 3 8 15 22 10
  2 3 8 15 10 22

  Pengurutan berhenti.

Bubble Sort (cara 3)

- Proses 1 :

22	10	15	3	8	2
10	22	15	3	8	2
10	22	15	3	8	2
3	22	15	10	8	2
3	22	15	10	8	2
2	22	15	10	8	3

- Pengecekan dimulai dari data yang di posisi pertama, kemudian dibandingkan dengan data di depannya,jika data didepannya lebih besar maka akan di tukar.
- Selanjutnya pembandingan antara data yang posisi pertama dengan dengan data yang ketiga, keempat dstnya

Bubble Sort (cara 3)

- Proses 2:

2	22	15	10	8	3
2	15	22	10	8	3
2	10	22	15	8	3
2	8	22	15	10	3
2	3	22	15	10	8

- pengecekan dilakukan mulai yang kedua karena yang pertama sudah pasti yang terkecil, dengan data yang ke tiga dstnya.

Bubble Sort (cara 3)

- Proses 3:

2	3	22	15	10	8
2	3	15	22	10	8
2	3	10	22	15	8
2	3	8	22	15	10

- Proses 4:

2	3	8	22	15	10
2	3	8	15	22	10
2	3	8	10	22	15

- Proses 5:

2	3	8	10	22	15
2	3	8	10	15	22

Pengurutan berhenti.

Selection Sort

- Cara kerja metode ini didasarkan pada pencarian elemen dengan nilai terkecil. kemudian dilakukan penukaran dengan elemen ke-i.
- Secara singkat metode ini bisa dijelaskan sebagai berikut.
  - Pada langkah pertama, dicari data yang terkecil dari data pertama sampai terakhir. Kemudian data tersebut ditukar dari data pertama. Dengan demikian, data pertama sekarang mempunyai nilai paling kecil dibanding dengan data lain.
  - Pada langkah kedua, data terkecil kita cari mulai dari data kedua sampai data terakhir. Data terkecil yang kita peroleh kita tukar dengan data kedua.
  - Demikian seterusnya sampai seluruh data terurut.

Selection Sort

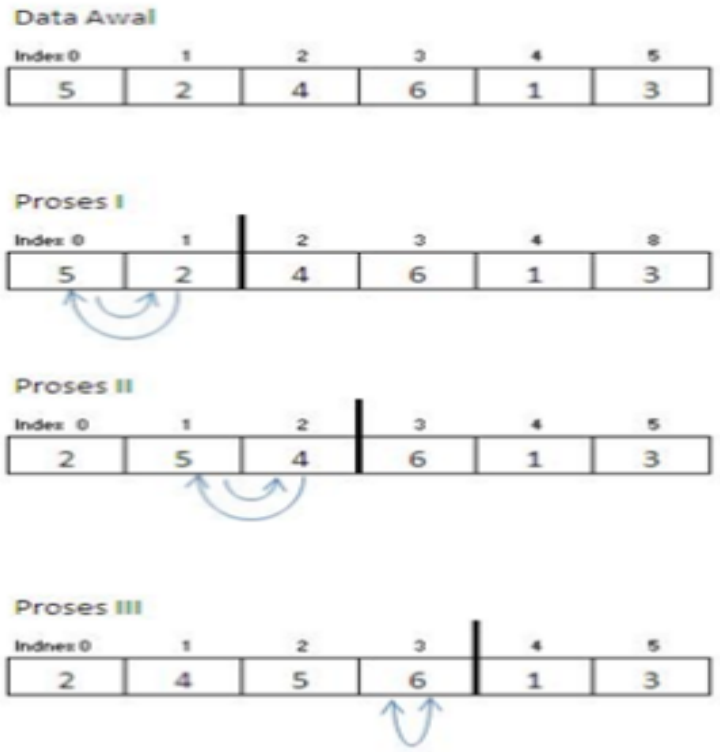
Iterasi ke-	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
awal	22	10	15	3	2	8
i = 0 , posisi = 4	2	10	15	3	22	8
i = 1, posisi = 3	2	3	15	10	22	8
i = 2, posisi = 5	2	3	8	10	22	15
i = 3, posisi = 3	2	3	8	10	22	15
i = 4, posisi = 5	2	3	8	10	15	22
akhir	2	3	8	10	15	22

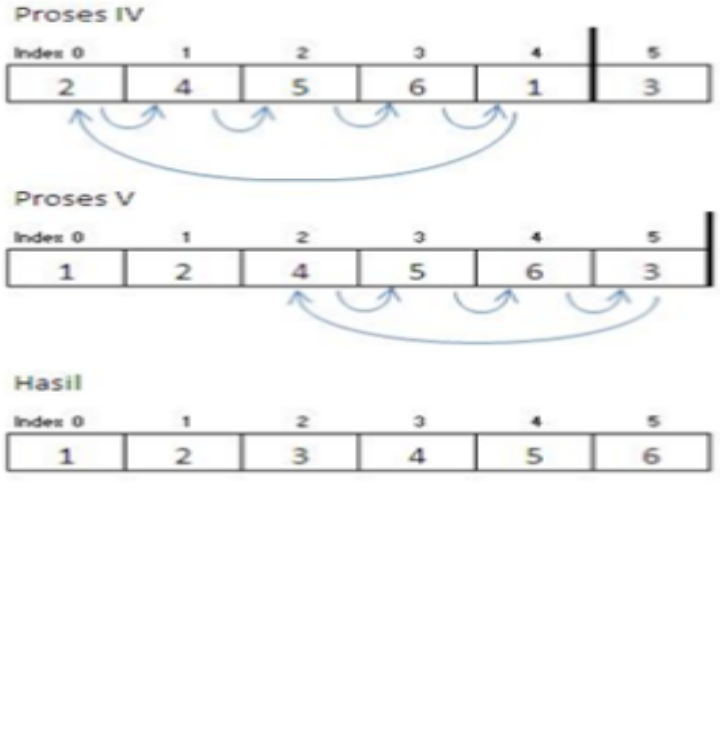
Fungsi Selection Sort

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

Insertion Sort

- Insertion sort adalah sebuah metode pengurutan data dengan menempatkan setiap elemen data pada posisinya dengan cara melakukan perbandingan dengan data – data yang ada





Fungsi Insertion Sort

```
void insertionSort(larik& a, int n) {
     for (int i = 1; i < n; i++) {
        for (int j = i; j >= 1; j--){
            if (a[j] < a[j-1]) {
                swap(a[j], a[j-1]);
            }
            else
                break;
        }
    }
}
```

Shell Sort

- Metode ini dikembangkan oleh Donald L. Shell pada tahun 1959.
- Dalam metode ini jarak antara dua elemen yang dibandingkan dan ditukarkan tertentu.
- Secara singkat metode ini dijelaskan sebagai berikut:
  - Pada langkah pertama, ambil elemen pertama dan bandingkan dan dengan elemen pada jarak tertentu dari elemen pertama tersebut. Kemudian elemen kedua dibandingkan dengan elemen lain dengan jarak yang sama seperti jarak yang sama seperti diatas. Demikian seterusnya sampai seluruh elemen dibandingkan.
  - Pada langkah kedua proses diulang dengan langkah yang lebih kecil,
  - pada langkah ketiga jarak tersebut diperkecil lagi seluruh proses dihentikan jika jarak sudah sama dengan satu.

Proses Shell Sort

Jarak	a[0]	a[1]	a[2]	a[3]	a[4]	a[5]
awal	22	10	15	3	2	8
Jarak = 3	22	10	15	3	2	8
	3	10	15	22	2	8
	3	2	15	22	10	8
	3	2	8	22	10	15
Jarak = 1	3	2	8	22	10	15
	2	3	8	22	10	15
	2	3	8	22	10	15
	2	3	8	22	10	15
	2	3	8	10	22	15
	2	3	8	10	15	22
Akhir	2	3	8	10	15	22

Fungsi Shell Sort

```
void shellSort(larik& a, int n) {
   int j;

  //Pengurangan jarak / gap sebanyak 2
   for (int gap = n / 2; gap > 0; gap /= 2)  {
       for (int i = gap; i < n; i++) {
          int temp = a[i];
          for (j=i; j>=gap && temp<a[j - gap]; j -= gap){
                a[j] = a[j - gap];
          }
          a[j] = temp;
       }
    }
}
```

Searching (Pencarian)

- Searching adalah metode pencarian informasi dalam suatu aplikasi dengan suatu kunci(key).
- Pencarian diperlukan untuk mencari suatu informasi khusus dari kumpulan data pada saat lokasi yang pasti dari informasi tersebut sebelumnya tidak diketahui.
- Pencarian selalu dinyatakan dengan referensi pada adanya sekelompok data yang tersimpan secara terorganisasi.
- Pencarian akan dilakukan selama data yang dicari belum ditemukan dan data masih ada, sebaliknya pencarian akan selesai jika data ditemukan atau data sudah tidak ada lagi.

Pencarian sekuensial secara Linear (linear search)

- Data yang ada di bandingkan satu persatu secara berurutan dengan yang dicari.
- Pada dasarnya, pencarian ini hanya melakukan pengulangan dari 1 sampai dengan jumlah data.
- Pada setiap perulangan , di bandingkan data ke-i dengan yang dicari.
- Apabila sama, berarti data telah ditemukan .
- Sebaliknya apabila sampai akhir pengulangan , tidak ada data yang sama berarti data tidak ditemukan.

Fungsi Linear Search

```
main() {
   larik x;
   int n,kunci,found,lokasi;
   cout << "Kunci Pencarian data : " ; cin >> kunci;
   linearSearch(x, n, kunci, found, lokasi);
   if (found)
           cout << "Ditemukan di posisi : " << lokasi+1 ;
   else
            cout << "Tidak ditemukan";

}

void linearSearch(larik a, int n, int kunci, int& found, int& lokasi){
          found = lokasi = 0;
          while (!found && lokasi < n) {
                    if (a[lokasi] == kunci){
                              found = 1;
                    }
                    else {
                              lokasi=lokasi+1;
                    }
          }
}
```

- Kelebihan :
  - Relatif lebih cepat dan efisien untuk data yang terbatas
  - Algoritma sederhana
- Kekurangan :
  - Kurang cepat untuk data dalam jumlah besar
  - Beban komputasi cenderung lebih besar

Binary Search

- Salah satu syarat pencarian biner (binary search) dapat dilakukan adalah data sudah dalam keadaan terurut. Dengan kata lain, apabila data belum dalam keadaan terurut , pencarian biner tidak dapat dilakukan .
- Langkah dalam pencarian biner adalah :
  - Mula-mula diambil dari posisi awal=1 dan posisi akhir = n
  - cari posisi data tengah dengan rumus posisi tengah = (posisi awal + posisi akhir) div 2
  - data yang di cari dibandingkan dengan data tengah:
    - Jika sama, data ditemukan, Proses selesai
    - Jika lebih kecil, proses dilakukan kembali tetapi posisi akhir dianggap sama dengan posisi tengah - 1,
    - Jika lebih besar , proses dilakukan kembali tetapi posisi awal dianggap sama dengan posisi tengah + 1.
  - Ulangi langkah kedua hingga data ditemukan , atau tidak ditemukan.
  - Pencarian biner ini akan berakhir jika data ditemukan posisi awal lebih besar dari pada posisi akhir. Jika posisi awal sudah lebih besar dari posisi akhir berarti data tidak diketemukan.

Fungsi Binary Search

```
void binarySearch(larik a, int n, int kunci, int& found, int& lokasi){
        int left = 0, right = n-1;
        found =  0;
        while (!found && left <= right) {
                lokasi = (left + right) / 2;  // titik tengah
                if (a[lokasi] == kunci){
                        found = 1;
                }
                else
                        if (a[lokasi] < kunci){
                                left = lokasi + 1;
                        }
                        else {
                                right = lokasi - 1;

                        }
        }
}
```

- Kelebihannya :
  - Untuk data dalam jumlah besar, waktu searching lebih cepat
  - Beban komputasi lebih kecil
- Kekuranganya :
  - Data harus sudah di-sorting lebih dulu (dalam keadaan terurut)

Latihan dan Tugas

1. Buatlah program modular Matriks berikut:
   a. Penjumlahan 2 buah matriks
   b. Perkalian 2 buah matriks
   c. Transpose suatu matriks
   d. Mencari jumlah dari setiap baris dan kolom suatu matriks. Hasil penjumlahan disimpan dalam suatu array 1 dimensi

      Contoh :

A =	1	0	4	jBaris =	5	jKolom =	8
	5	2	1		8		3
	2	1	2		5		7

Fungsi yang diperlukan antara lain : main, inputMatriks, cetakMatriks, cariJumlahBaris, cariJumlah Kolom, cetakLarik.

2. Buatlah program modular untuk mencari nilai standart deviasi dari kumpulan data bertipe integer


ANY
QUESTIONS?

Sesi Berakhir
TERIMA KASIH