MODUL PRAKTIKUM STRUKTUR DATA

Daftar Isi

Daftar Algoritma

Modul 1

Struct

Materi	:	Pengenalan dan Penggunaan Struct
Waktu	:	60 Menit
Tujuan	:	Mahasiswa memahami cara mendeklarasikan struct, mengisi struct, menampilkan isi struct, merubah isi struct dan pemanfaatannya dalam menyelesaikan masalah

Teori Dasar

Pada tahap pembelajaran bahasa C telah dipelajari mengenai tipe data dan variabe. Pada tahap awa setiap variable hanya mewakili 1 tipe data. Penggunaan struct memungkinkan 1 variabel memiliki anggota yang terdiri dari beberapa variabel dan tipe data. Struct membebaskan pemrogram untuk menyimpan data yang kompleks, data yang disimpan tidak harus bertipe data yang sama dengan data lainnya.

Pendeklarasian

Untuk mendeklarasikan sebuah struct, dapat dipilih salah satu dari dua metode berikut ini. Pendeklarasian struct yang pertama (bentuk umum).

```
struct namaStruct { tipeData namaVariabel; };
```

Contoh kode pendeklarasian struct:

```
1 struct mahasiswa {
2  char nim[25];
3  char nama[25];
4  int usia;
5 };
```

Sedangkan untuk pendeklarasian cara yang kedua, dapat digunakan fasilitas typedef untuk memberikan nama samaran (alias name) kepada struct yang ingin dideklarasikan dan digunakan nantinya.

```
typedef struct { tipeData namaVariabel; }namaStruct;
```

Contoh kode pendeklarasian struct:

```
1  typedef struct {
2   char nim[25];
3   char nama[25];
4   int usia;
5  }mahasiswa;
```

Contoh Penggunaan

Berikut ini merupakan contoh penggunaan struct :

```
1  #include <stdio.h>
2
3  struct mahasiswa {
4   char nim[25];
5   char nama[25];
6   int usia;
7  };
8
9  typedef struct {
10  char namamk[25];
11  int semester;
12  int sks;
13 }mataKuliah;
14
15 void main(){
16     struct mahasiswa mhs1 = {"2016823", "Budi Wahono",18};
17     mataKuliah mk1 = {"Struktur Data", 2, 3};
18
19     //tampilkan data Mahasiswa
20     printf("NIM : %s\n",mhs1.nim);
21     printf("Nama : %s\n",mhs1.nama);
22     printf("Usia : %d\n",mhs1.usia);
23
24     //tampilkan data Mata Kuliah
25     printf("Mata Kuliah : %s\n",mk1.namamk);
26     printf("Semester : %d\n",mk1.semester);
27     printf("SKS : %d\n",mk1.sks);
28 }
```

Pada contoh ini akan diperlihatkan struct bersarang, dalam artian didalam struct terdapat struct. Berikut ini contohnya:

```
#include <stdio.h>

struct nilai{
    char mataKuliah[25];
    int nilaiMk;
};

struct mahasiswa {
 char nim[25];
 char nama[25];
 struct nilai dataNilai;
};

void main(){

 struct mahasiswa mhs1 = {"2016823", "Budi Wahono",{"Struktur Data"
     ,90}};

  printf("NIM : %s\n",mhs1.nim);
  printf("Nama : %s\n",mhs1.nama);

  printf("Mata Kuliah : %s\n",mhs1.dataNilai.mataKuliah);
  printf("Nilai : %d\n",mhs1.dataNilai.nilaiMk);

}
```

Pada contoh ini akan dilakukan pemberian nilai melalui keyboard terhadap struct yang sudah dideklarasikan.

```
#include <stdio.h>
#include <string.h>

struct koleksi {
   char  judul[50];
   char  pengarang[50];
   char  jenis[100];
   int   buku_id;
};

void main( ) {

    struct koleksi buku1;
    struct koleksi buku2;

```

```
16      printf("Buku 1 \n");
17      printf("Judul Buku : ");
18      scanf("%[^\n]%*c", buku1.judul);
19      printf("Nama Pengarang : ");
20      scanf("%[^\n]%*c", buku1.pengarang);
21      printf("Jenis Buku : ");
22      scanf("%[^\n]%*c", buku1.jenis);
23      buku1.buku_id = 6495407;
24
25      printf("Buku 2 \n");
26      printf("Judul Buku : ");
27      scanf("%[^\n]%*c", buku2.judul);
28      printf("Nama Pengarang : ");
29      scanf("%[^\n]%*c", buku2.pengarang);
30      printf("Jenis Buku : ");
31      scanf("%[^\n]%*c", buku2.jenis);
32      buku2.buku_id = 6495700;
33
34      printf("\nData Buku \n");
35      printf( "buku 1 judul : %s\n", buku1.judul);
36      printf( "buku 1 pengarang : %s\n", buku1.pengarang);
37      printf( "buku 1 jenis : %s\n", buku1.jenis);
38      printf( "buku 1 buku_id : %d\n", buku1.buku_id);
39
40      printf( "buku 2 judul : %s\n", buku2.judul);
41      printf( "buku 2 pengarang : %s\n", buku2.pengarang);
42      printf( "buku 2 jenis : %s\n", buku2.jenis);
43      printf( "buku 2 buku_id : %d\n", buku2.buku_id);
44
45  }
```

Modul 2

Larik 1 Dimensi

Materi	:	Pengenalan dan Penggunaan Larik 1 Dimensi
Waktu	:	60 Menit
Tujuan	:	Mahasiswa memahami cara mendeklarasikan larik, mengisi larik, menampilkan isi larik, merubah isi larik dan pemanfaatannya dalam menyelesaikan masalah

Teori Dasar

- Array (larik) ialah penampung sejumlah data sejenis (homogen) yang menggunakan satu identifier (pengenal).
- Masing-masing elemen larik diakses menggunakan indeks (subscript) dari nol sampai n-1 (n menyatakan jumlah elemen larik).
- Pengolahan data larik harus per elemen. Elemen Larik dapat diakses langsung (acak), maksudnya untuk memanipulasi elemen ke-4 tidak harus melalui elemen ke-1, ke-2 dan ke-3.
- Berdasarkan banyaknya indeks larik dibagi menjadi satu dimensi dan multi dimensi (duadimensi, tiga dimensi).

Pendeklarasian

- Pendeklarasian larik 1 dimensi mempunyai pola tipeVariabel namaVariabel[ukuran]. Contohnya:
  - int panjangPadi[5];
  - float nilaiUas[5];

Pengisian Larik

Ada 2 metode dalam pengisian Larik, yaitu:

- Pengisian langsung saat deklarasi
  - int my_larik[] = {1,23,17,4,-5,100};
- Pengisian setelah deklarasi
  - int my_larik[5];
    - my_larik[0]=10;
    - my_larik[1]=20;
    - my_larik[2]=30;
    - my_larik[3]=40;
    - my_larik[4]=50;

Menampilkan Larik

Saat akan menampilkan Larik yang perlu diingat adalah indeks dari setiap Larik adalah mulai dari 0, misalkan telah di deklarasikan int my_larik[] = {1,23,17,4,-5,100}. Maka untuk menampilkan data Larik ke-2 yaitu 23 dari variabel my_larik adalah my_larik[1]. Berikut ini contoh kode lengkapnya.

Kode Menampilkan Larik 1 Dimensi

```
1  #include <stdio.h>
2  main(){
3  int my_array[6] = {1,23,17,4,-5,100};
4
5      printf("Data ke-1 =  %d\n",my_array[0]);
6      printf("Data ke-2 =  %d\n",my_array[1]);
7      printf("Data ke-3 =  %d\n",my_array[2]);
8      printf("Data ke-4 =  %d\n",my_array[3]);
9      printf("Data ke-5 =  %d\n",my_array[4]);
10     printf("Data ke-6 =  %d\n",my_array[5]);
11 }
```

Kode Menampilkan Larik 1 Dimensi Dengan Pengulangan

```c
#include <stdio.h>
main(){
    int my_array[6] = {1,23,17,4,-5,100};
    int i;
    printf("Data Larik Dari Depan\n");
    //Menggunakan Looping
    for(i=0;i<6;i++){
        printf("Data ke-%d =  %d\n",i+1,my_array[i]);
    }
    printf("\nData Larik Dari Belakang\n");
    //Menggunakan Looping Urut Terbalik
    for(i=5;i>=0;i--){
        printf("Data ke-%d =  %d\n",i+1,my_array[i]);
    }
}
```

Kode Menyalin Larik

```c
#include <stdio.h>
main(){
    int jumlah, larikA[5], larikB[5];
    // Input elemen larikA
    printf("Input elemen larikA : \n");
    for(jumlah=0; jumlah<5; jumlah++){
        printf("larikA[%i] : ", jumlah+1);
        scanf("%d", &larikA[jumlah]);
    }
    //salin larikA ke larikB
     printf("\nSalin isi larikA ke larikB\n");
    for(jumlah=0; jumlah<5; jumlah++){
        larikB[jumlah]= larikA[jumlah];
    }
    // Tampilkan larikA dan larikB
    printf("\n");
    printf("Isi Matriks A : \n");
    for(jumlah=0; jumlah<5; jumlah++){
        printf("%d ", larikA[jumlah]);
    }
    printf("\n");
    printf("Isi Matriks B : \n");
    for(jumlah=0; jumlah<5; jumlah++){
        printf("%d ", larikA[jumlah]);
    }
}
```

Latihan Larik

Cobalah kode program dibawah ini amati kode dan keluarannya!

Kode Mengelola Larik 1 Dimensi

```
#include <stdio.h>
main(){
    int index, nilai[10];
    /* input nilai mahasiswa */
    printf("Input nilai 10 mahasiswa : \n");
    for(index=0; index < 10; index++){
        printf("Mahasiswa %d : ", index+1);
        scanf("%d", &nilai[index]);
    }
    /* tampilkan nilai mahasiswa */
    printf("\nNilai mahasiswa yang telah diinput \n");
    for(index=0; index < 10; index++){
        printf("Mahasiswa %d : %d \n",index+1,nilai[index]);
    }
}
```

Kode Mengelola Larik 1 Dimensi Dengan Kondisi

```
#include <stdio.h>
main(){
    int index, nilai[10];
    /* input nilai mahasiswa */
    printf("Input nilai 10 mahasiswa : \n");
    for(index=0; index < 10; index++){
        printf("Mahasiswa %d : ", index+1);
        scanf("%d", &nilai[index]);
    }
    /* tampilkan nilai mahasiswa diatas 60*/
    printf("\nNilai mahasiswa yang telah diinput \n");
    for(index=0; index < 10; index++){
        if(nilai[index]>=60){
            printf("Mahasiswa %d : %d Status : Lulus \n",index+1,nilai
                [index]);
        }else{
            printf("Mahasiswa %d : %d Status : Tidak Lulus\n",index+1,
                nilai[index]);
        }
    }
}
```

Tugas Praktikum Larik 1 Dimensi

Kerjakan tugas praktikum berikut ini, setelah selesai kode program di tulis pada buku praktikum.

1. Buatlah program untuk mengisi larik bertipe float berukuran 5 kemudian tampilkan hasilnya

2. Misalkan sebuah larik bertipe integer berukuran 5 diketahui bernilai {70, 60, 35, 65, 45}. Buatlah sebuah program yang untuk menambahkan +5 untuk nilai elemen larik di bawah 50. Jadi larik akan bernilai {70,60,40,65,55}.

3. Misalkan sebuah larik bertipe integer berukuran 10 diketahui bernilai {60, 73, 83, 58, 68, 76, 99, 52, 74, 69}. Buatlah sebuah program yang untuk menampilkan baris bilangan genap dan baris bilangan ganjil dari elemen larik tersebut. Jadi keluarannya:

   - Baris Bilangan Genap : 60,58,68,76,52,74
   - Baris Bilangan Ganjil : 73,83,99,69

4. Buatlah sebuah program yang mengisi dan memproses larik yang berisi bilangan integer berukuran 6, kemudian hitunglah Jumlah Total dari elemen larik kemudian hitunglah rata-rata nilai dari elemen larik tersebut. Misalkan data larik : 8,5,6,4,8,6 maka keluarannya 8+5+6+4+8+6 = 37, rata-ratanya 37/6 = 6.16

Modul 3

Larik Multi Dimensi

Materi	:	Pengenalan dan Penggunaan Larik Multi Dimensi
Waktu	:	60 Menit
Tujuan	:	Mahasiswa memahami cara mendeklarasikan, mengisi, menampilkan isi, merubah isi dan pemanfaatannya Larik Multi Dimensi dalam menyelesaikan masalah

Teori Dasar

Larik multi-dimensi merupakan larik yang mempunyai ukuran lebih dari dua. Bentuk pendeklarasian larik sama saja dengan larik dimensi satu maupun larik dimensi dua.

Pendeklarasian

- Pendeklarasian larik 2 dimensi mempunyai pola tipeVariabel namaVariabel[ukuran1] [ukuran2]. Contohnya:
  - int matriksA[3][4];
  - float matrikB[4][4];
- Pendeklarasian larik 3 dimensi mempunyai pola tipeVariabel namaVariabel[ukuran1] [ukuran2] [ukuran3]. Contohnya:
  - int matriksA[3][4][2];
  - float matrikB[4][4][2];

Pengisian Larik Multidimensi

- Pengisian larik saat deklarasi

  - int larik2D[5][2]={{1,12},{2,22},{3,33},{4,44},{5,55}};

- Pengisian setelah deklarasi

  - int larik2D[5][2];
    - larik2D[0][0]=1;
    - larik2D[0][1]=2;
    - larik2D[1][0]=2;
    - larik2D[1][1]=22;
    - dst

Menampilkan Larik Multi Dimensi

Berikut ini contoh menampilkan larik multidimensi:

Kode Menampilkan Larik 2 Dimensi

```
#include <stdio.h>

main(){
    int array2D[5][2]={{1,12},{2,22},{3,33},{4,44},{5,55}};
    int i,j;

    printf("array2D[%d][%d] =  %d\n",0,0,array2D[0][0]);
    printf("array2D[%d][%d] =  %d\n",0,1,array2D[0][1]);
    printf("array2D[%d][%d] =  %d\n",1,0,array2D[1][0]);
    printf("array2D[%d][%d] =  %d\n",1,1,array2D[1][1]);
    printf("array2D[%d][%d] =  %d\n",2,0,array2D[2][0]);
    printf("array2D[%d][%d] =  %d\n",2,1,array2D[2][1]);
    printf("array2D[%d][%d] =  %d\n",3,0,array2D[3][0]);
    printf("array2D[%d][%d] =  %d\n",3,1,array2D[3][1]);
    printf("array2D[%d][%d] =  %d\n",4,0,array2D[4][0]);
    printf("array2D[%d][%d] =  %d\n",4,1,array2D[4][1]);
}
```

Kode Menampilkan Larik 2 Dimensi Dengan Perulangan

```
#include <stdio.h>

main(){
    int array2D[5][2]={{1,12},{2,22},{3,33},{4,44},{5,55}};
    int i,j;

```

```
7       for(i=0;i<5;i++){
8           for(j=0;j<2;j++){
9               printf("array2D[%d][%d] =  %d\n",i,j,array2D[i][j]);
10          }
11      }
12      printf("\n");
13      for(i=0;i<5;i++){
14          for(j=0;j<2;j++){
15              printf("%d ",array2D[i][j]);
16          }
17          printf("\n");
18      }
19      printf("\n");
20      i=0;
21      while(i<5){
22          j=0;
23          while(j<2){
24              printf("%d ",array2D[i][j]);
25              j++;
26          }
27          i++;
28          printf("\n");
29      }
30  }
```

Latihan Larik Multi Dimensi

Cobalah kode program dibawah ini amati kode dan keluarannya!

Kode Mengelola Larik 2 Dimensi

```
1   #include <stdio.h>
2
3   main(){
4       int baris, kolom, matriks[3][4];
5       // Input elemen array
6       printf("Input elemen Array : \n");
7       for(baris=0; baris<3; baris++){
8           for(kolom=0; kolom<4; kolom++){
9               printf("matriks[%i][%i] : ", baris+1, kolom+1);
10          scanf("%d", &matriks[baris][kolom]);
11          }
12          printf("\n");
13      }
14      // Tampilkan elemen Array
15      printf("Isi array : \n");
```

```
    for(baris=0; baris<3; baris++){
        for(kolom=0; kolom<4; kolom++){
            printf("%d ", matriks[baris][kolom]);
        }
    printf("\n");
    }
}
```

Kode Menyalin Larik 2 Dimensi

```
#include <stdio.h>

main(){

    int baris, kolom, matriksA[3][4], matriksB[3][4];

    // Input elemen MatriksA
    printf("Input elemen MatriksA : \n");
    for(baris=0; baris<3; baris++){
        for(kolom=0; kolom<4; kolom++){
            printf("matriksA[%i][%i] : ", baris+1, kolom+1);
        scanf("%d", &matriksA[baris][kolom]);
        }
        printf("\n");
    }

    //salin matriksA ke matriksB
    for(baris=0; baris<3; baris++){
        for(kolom=0; kolom<4; kolom++){
            matriksB[baris][kolom]=matriksA[baris][kolom];
        }
    }

    // Tampilkan matriksA dan matriksB
    printf("Isi Matriks A : \n");
    for(baris=0; baris<3; baris++){
        for(kolom=0; kolom<4; kolom++){
            printf("%d ", matriksA[baris][kolom]);
        }
    printf("\n");
    }
    printf("Isi Matriks B : \n");
    for(baris=0; baris<3; baris++){
        for(kolom=0; kolom<4; kolom++){
            printf("%d ", matriksB[baris][kolom]);
        }
    printf("\n");
```

```
38      }
39  }
```

Kode Mengelola Larik 3 Dimensi

```
1   #include <stdio.h>
2   main()
3   {
4       int i=0;
5       int angka[3][3][3];
6       angka[0][0][0]=0; angka[0][0][1]=1; angka[0][0][2]=2;
7       angka[0][1][0]=10; angka[0][1][1]=11; angka[0][1][2]=12;
8       angka[0][2][0]=20; angka[0][2][1]=21; angka[0][2][2]=22;
9
10      angka[1][0][0]=100; angka[1][0][1]=101; angka[1][0][2]=102;
11      angka[1][1][0]=110; angka[1][1][1]=111; angka[1][1][2]=112;
12      angka[1][2][0]=120; angka[1][2][1]=121; angka[1][2][2]=122;
13
14      angka[2][0][0]=200; angka[2][0][1]=201; angka[2][0][2]=202;
15      angka[2][1][0]=210; angka[2][1][1]=211; angka[2][1][2]=212;
16      angka[2][2][0]=220; angka[2][2][1]=221; angka[2][2][2]=222;
17      printf("isi dari array 3 dimensi adalah : \n");
18      for(i;i<3;i++)
19      {
20          int j=0;
21          for(j;j<3;j++)
22          {
23              int k=0;
24              for(k;k<3;k++)
25              {
26              printf("%d ", angka[i][j][k]);
27              }
28          printf("\n");
29          }
30      printf("\n\n");
31      }
32
33  }
```

Tugas Praktikum

Kerjakan tugas praktikum berikut ini, setelah selesai kode program di tulis pada buku praktikum.

1. Buatlah sebuah matrik 2 dimensi 3x3 bertipe integer. Kolom 1 dan 2 setiap baris

di input dari keyboard sedangkan isi dari kolom ke 3 setiap baris adalah jumlah dari kolom 1 dan 2.

2. Buatlah sebuah program yang mengisi dan memproses larik untuk menjumlahkan dan mengurangkan 2 buah matriks berukuran 3x3.

3. Buatlah sebuah program yang mengisi dan memproses larik untuk mengalikan 2 buah matriks berukuran 3x3.

Modul 4

Pointer

Pengertian Pointer

- Pointer adalah suatu variabel yang menunjuk ke alamat memory variabel yang lainnya.
- Suatu pointer bukan berisi dengan suatu nilai data seperti halnya pada variabel biasa, variabel pointer berisi dengan suatu alamat.
- Untuk mendeklarasikan variabel pointer kita menggunakan tanda asterik / bintang (*) didepan variabel yang di deklarasikan pada tipe data tertentu.
- Tanda ini juga dapat dipakai untuk mengakses nilai dari variabel yang telah ditunjuk.
- Untuk mendapatkan alamat dari variabel pointer kita menggunakan tanda &

Deklarasi & Isi

- Deklarasi
  - int *b
- Alamat memory pointer (address of)
  - &b
- Isi pointer
  - b
- Isi dari isi pointer (value pointed by)

– *b

Contoh program dengan pointer

```
#include <stdio.h>

main(){
    int *ptr;
    int k;
    k=7;
    printf("Isi variabel k = %d",k);
    printf("\nAlamat variabel k = %d",&k);
    printf("\nAlamat variabel *ptr = %d",&ptr);
    printf("\nIsi variabel *ptr = %d",ptr);
    ptr=&k;
    printf("\nAlamat variabel *ptr = %d",&ptr);
    printf("\nIsi variabel *ptr = %d",ptr);
    printf("\nIsi dari alamat %d = %d",ptr,*ptr);
    printf("\n");
}
```

Fungsi dan Pointer

Pada bagian ini akan dibahasa mengenai penggunaan pointer pada parameter fungsi. *Parameter pass by reference* adalah pemrosesan parameter di dalam sebuah fungsi di mana yang dimasukkan didalam prosedur adalah tempat atau alamat dari variabel yang menjadi parameter sehingga dapat terjadi perubahan nilai variabel yang menjadi parameter.

Contoh program fungsi dengan *parameter pass by reference*

```
#include <stdio.h>

int hitung(int a, int *b){
    *b = 15;
    return a + *b;

}

main(){
    int y,z,hasil;
    y=10;
    z=50;
    printf("Sebelum Jalankan Fungsi\n");
    printf("y=%d\n",y);
```

```
15      printf("z=%d\n",z);
16      hasil=hitung(y,&z);
17      printf("Sebelum Jalankan Fungsi\n");
18      printf("y=%d\n",y);
19      printf("z=%d\n",z);
20      printf("hasil=%d\n",hasil);
21  }
```

Larik dan Pointer

Dalam pemrograman C, definisi larik dituliskan: type_name array_name [number_of_array], misal larik A bertipe integer dengan 10 anggota didefinisikan dengan int A[10].Apa maksudnya? Dengan penulisan itu, maka diperintahkan kepada kompiler untuk menyediakan alamat memori sebesar 10 * sizeof(int). Bila ukuran int adalah 4 byte, maka compiler akan mengalokasikan sebesar 10 * 4 byte = 40 byte memori untuk A. Maka penggambarannya dapat diilustrasikan sebagai berikut.

A[0]	0xDDDD0004
A[1]	0xDDDD0008
A[2]	0xDDDD000C
...	...
A[7]	0xDDDD0020
A[8]	0xDDDD0024
A[9]	0xDDDD0028

Tabel 4.1: Ilustrasi Memori Larik

Seperti pada pembahasan pointer dengan variabel, pointer pada larik sesungguhnya juga diperintahkan pointer agar menunjuk ke alamat yang telah dialokasikan oleh larik tersebut. Pada contoh diatas, bila didefinisikan suatu pointer

- int *P:
  - kemudian kita tunjuk ke alamat larik A
- P = &A[0];
  - maka alamat P akan menunjuk ke alamat 0xDDDD0004

Program berikut ini menampilkan alamat memory dan mengakses larik menggunakan pointer.

```c
#include <stdio.h>

main(){
    int my_array[6] = {1,23,17,4,-5,100};
    int *p_array;

     // Dapat juga ditulis  p_array = my_array;
    p_array = &my_array[0];

    printf("Alamat dari p_array=%d\n",&p_array);
    printf("Isi dari isi p_array=%d\n\n",*p_array);
    p_array++;
    printf("Isi dari p_array=%d\n",p_array);
    printf("Isi dari isi p_array=%d\n",*p_array);
}
```

Program berikut ini menampilkan larik dengan menggunakan pointer.

```c
#include <stdio.h>

main(){

    int *pArray, Array[10],i;
    for(i=0;i<10;i++){
        Array[i] = i+10; //pengisian array
    }
    // tunjuk pArray ke alamat awal array
    pArray = &Array[0]; // bisa dituliskan pArray=Array
    for(i=0;i<10;i++){
        printf("Alamat pointer= %d. Isi dari alamat %d = %d\n",&pArray
            ,pArray,*pArray++); //cetak pArray
    }
}
```

Latihan Pointer

Program berikut ini memasukan nilai dari keyboard dan perubahan nilai dari pointer.

```c
#include <stdio.h>

main(){
    int *ptr,nilai;
    //memberikan alamat memori nilai ke variabel *ptr
    ptr=&nilai;

```

```
8       printf("Isi Nilai = ");
9       scanf("%d",&nilai);
10      printf("Isi dari isi alamat ptr = %d\n\n",*ptr);
11
12      printf("Isi Nilai = ");
13      scanf("%d",&nilai);
14      printf("Isi dari isi alamat ptr = %d\n\n",*ptr);
15
16      printf("Isi Nilai = ");
17      scanf("%d",&nilai);
18      printf("Isi dari isi alamat ptr = %d\n\n",*ptr);
19  }
```

Program berikut ini mengisi dan menampilkan larik menggunakan pointer

```
1   #include <stdio.h>
2
3   main(){
4
5       int *pArray, Array[10];
6       int i;
7       // tunjuk pArray ke alamat awal array
8       pArray = &Array[0]; // bisa dituliskan pArray=Array
9       for(i=0;i<10;i++){
10          *pArray = i+100; //pengisian array melalui pArray
11          pArray++;
12      }
13      pArray = &Array[0];
14      for(i=0;i<10;i++){
15          printf("pArray = %d\n",*pArray++); //cetak pArray
16      }
17  }
```

Program berikut ini mengisi larik dari keyboard dengan pointer dan menampilkannya.

```
1   #include <stdio.h>
2
3   main(){
4       int *ptr,i,nilai,arrayA[3];
5       ptr=arrayA;
6       for(i=0;i<3;i++){
7           printf("Isi nilai[%d] = ",i);
8           scanf("%d",&nilai);
9           *ptr=nilai;
10          ptr++;
11      }
```

```
12      for(i=0;i<3;i++){
13          printf("Isi nilai[%d] = %d",i,arrayA[i]);
14          printf("\n");
15      }
16      printf("\n");
17  }
```

Program berikut ini memanipulasi larik dengan fungsi menggunakan pointer.

```
1   #include <stdio.h>
2
3   void rubah(int *b){
4        printf("\n%d",b);
5       *--b = 7;
6   }
7
8   main(){
9       int my_array[6] = {1,23,17,4,-5,100};
10      int i;
11      printf("Menampilkan Data Array\n");
12      //Menggunakan Looping
13      for(i=0;i<6;i++){
14          printf("Data ke-%d =  %d\n",i+1,my_array[i]);
15      }
16      rubah(&my_array[3]);
17      printf("Menampilkan Data Array Setelah di rubah\n");
18      for(i=0;i<6;i++){
19          printf("Data ke-%d =  %d\n",i+1,my_array[i]);
20      }
21  }
```

Praktikum

1. Modifikasilah latihan pointer ke-3 dengan ketentuan

   (a) Panjang larik menjadi 10;

   (b) Pada saat menampilkan data larik gunakanlah mekanisme pointer;

   (c) Nilai inputan yang kurang dari 50 tidak ditampilkan;.

2. Diketahui sebuah larik {60, 70, 40, 90, 50, 60, 40, 80, 90, 30}, buatlah sebuah program yang memiliki fungsi untuk menampilkan larik tersebut dengan aturan

   (a) Parameter yang dikirim berupa pointer (lihat contoh latihan ke-4);

   (b) Setiap nilai ditambahkan dengan 100;

(c) Sehingga didapatkan larik tersebut menjadi {160, 170, 140, 190, 150, 160, 140, 180, 190, 130};

Modul 5

Pointer dan Array

Materi : Penggunaan Fungsi untuk Array dan Pointer
Waktu : 60 Menit
Tujuan : Mahasiswa memahami cara membuat fungsi dan pemanfaatannya dalam menyelesaikan masalah

Konsep Alamat Memori Pada Array

Saat variabel di dekalarasikan pada kode program maka komputer akan memesan suatu tempat di memori komputer untuk menyimpan isi dari variabel tersebut. Misalkan di deklarasikan sebuah variable bertipe array dengan tipe data integer, int buku[5], int x dan int int *h, maka dapat diilustrasikan komputer akan memesan alamat memori seperti terlihat pada Tabel 5.1

variabel	alamat	isi
buku[0]	300	19
buku[1]	304	62
buku[2]	308	12
buku[3]	312	43
buku[4]	316	65
x	320	9
b	324	300

Tabel 5.1: Pengalamatan Memori Komputer

Coba perhatikan Kode Progam 5.1, pada kode program terlihat bagaimana dasar pengelolaan alamat pada array berdimensi 1.

Kode Program 5.1 Array dan Pointer

```
1  main(){
2      int buku[5] = {19,62,12,43,65};
3      int x = 9;
4      int *b;
5
6      //pemberian isi dari pointer b dengan alamat awal array buku
7      b=&buku[0];
8      printf("Isi =%d\n",b);
9      printf("Alamat buku[0]=%d\n",&buku[0]);
10
11     //atau dengan cara lain
12     b=buku;
13     printf("Isi dari b =%d\n",b);
14     printf("Alamat buku[0]=%d\n",&buku[0]);
15
16     //untuk memindahkan pointer b ke array berikutnya
17     b++;
18     printf("Isi =%d\n",b);
19     //atau dengan cara
20     b=(buku+1);
21     printf("Isi =%d\n",b);
22
23 }
```

Berikut ini cara menampilkan array dengan menggunakan mekanisme pointer, seperti terlihat pada Kode Program

Kode Program 5.2 Menampilkan Array dengan Pointer

```
1  main(){
2      int buku[5] = {19,62,12,43,65};
3      int i;
4      int *b;
5
6      printf("Dengan Menggunakan Metoda Pengalamatan b=&buku[i]\n");
7      for (i=0;i<5;i++){
8          b=&buku[i];
9          printf("%d ",*b);
10     }
11
12     printf("\n");
13     printf("Dengan Menggunakan Metoda Menggeser Pointer \n");
14     b=&buku[0];
15     for (i=0;i<5;i++){
16         printf("%d ",*b);
17         b++;
18     }
19
20     printf("\n");
21     printf("Dengan Menggunakan Metoda Pengalamatan b=buku\n");
22     for (i=0;i<5;i++){
23         b=buku+i;
24         printf("%d ",*b);
25     }
26
27 }
```

Pointer dan Array 2 Dimensi

Konsep pengalamatan pointer dengan array 2 dimensi tidak terlalu berbeda dengan 1 dimensi. Misalkan di deklarasikan sebuah array 2 dimensi : int c[2][3] dan int *k[3], maka komputer akan mengalokasikan memori seperti yang terlihat pada Tabel 5.2

blok	variabel	alamat	isi
c[0]	c[0][0]	100	1
	c[0][1]	104	3
	c[0][2]	108	5
c[1]	c[1][0]	112	7
	c[1][1]	116	9
	c[1][2]	120	11
	k	124	

Tabel 5.2: Alamat Memori Array 2 Dimensi

c[0] dan c[1] merupakan array 1 dimensi dengan panjang 3 buah integer. Berikut ini aturan pengalamatan pointer pada array 2 dimensi:

- Untuk memberikan nilai alamat awal blok array

kode	keluaran
k=c	100
k=c[0]	100

- Untuk menempatkan pointer k ke sub blok array

kode	keluaran
k=*c	100
k=*c+1	104
k=c[1]	112
k=c[1]+2	120

- Untuk menggeser pengalamatan pointer setiap blok array

kode	keluaran
k=c+1	112
k=&c[1]	112

- Untuk menggakses isi dari sub blok array

kode	keluaran
*(*c)	1
*(*c+2)	5
((c+1)+1)	9
((c+1)+1)	11

Contoh penggunaan pengalamatan ini seperti terlihat pada Kode Program

Kode Program 5.3 Pengalamatan Pointer Array 2 Dimensi

```
1  main(){
2
3      int c[2][3]={{1,3,5},{7,9,11}};
4      int (*k)[3];
5      int y,z;
6
7      //cetak alamat k
8      printf("alamat k=%d\n",&k);
9      //cetak semua alamat array c
10     for(y=0;y<2;y++){
11         for(z=0;z<3;z++){
12             printf("alamat c[%d][%d] = %d\n",y,z,&c[y][z]);
13         }
14     }
15
16     //memasukan nilai pointer k
17     printf("\nMemasukan alamat awal blok array blok c ke pointer k
           dengan cara k=&c[0] atau k=c\n");
18     k=&c[0];
19     printf("isi k=%d\n",k);
20     //geser k sebanyak 1
21     printf("Menggeser pointer k sebanyak 1 kali dengan cara k++ atau k
           +1\n");
22     k++;
23     printf("isi k=%d\n\n",k);
24
25
26     printf("Menggembalikan pointer k ke awal blok array c\n");
27     k=c;
28     printf("isi k=%d\n",k);
29     //pointer  mengakes ke blok yang lebih dalam
30     printf("Pointer k mengakes ke blok yang lebih dalam\n");
31     printf("isi k=%d\n\n",*k+1);
32
33     printf("Pengulangan untuk menampilkan alamat dari sub blok array c
           \n");
34     k=c;
35     for(z=0;z<3;z++){
36         printf("isi k=%d\n",*k+z);
37
38     }
39     printf("\nPengulangan untuk menampilkan isi dari sub blok array c\
           n");
40     k=c;
41     for(z=0;z<3;z++){
42         printf("isi k=%d\n",*(*k+z));
43
44     }
45 }
```

Cara menampilkan isi array 2 dimensi dengan mekanisme pointer

Kode Program 5.4 Menampilkan Isi Array 2 Dimensi Dengan Pointer

```
1  main(){
2
3      int c[2][3]={{1,3,5},{7,9,11}};
4      int y,z;
5
6
7      //cetak semua alamat array c
8      for(y=0;y<2;y++){
9          for(z=0;z<3;z++){
10             printf("%d ",c[y]+z);
11         }
12         printf("\n");
13     }
14     printf("\n");
15      //cetak semua alamat array c
16     for(y=0;y<2;y++){
17         for(z=0;z<3;z++){
18             printf("%d ",*(c[y]+z));
19         }
20         printf("\n");
21     }
22      printf("\n");
23      //cetak semua alamat array c
24     for(y=0;y<2;y++){
25         for(z=0;z<3;z++){
26             printf("%d ",*(*(c+y)+z));
27         }
28         printf("\n");
29     }
30
31
32 }
```

Tugas Praktikum

1. Rubahlah proses pengulangan pada Kode Program 5.2 dengan menggunakan proses pengulangan while.

2. Rubahlah proses pengulangan pada Kode Program 5.4 dengan menggunakan proses pengulangan while.

Modul 6

Pengurutan Sisip (Insertion Sort)

- Pengurutan data atau sorting merupakan hal penting untuk pengolahan data.
- Pengurutan dapat dilakukan secara menaik (ascending) atau dengan urutan menurun (descending).
  - Contoh data : 61 78 12 46 10 89
    - Urut menaik menjadi 10 12 46 61 78 89
    - Urut menurun menjadi 89 78 61 46 12 10

Insertion Sort

Metode pengurutan yang mengambil sebuah data sisip pada data yang diurutkan dan menggeser data yang lebih besar dari data sisip agar data sisip dapat di tempatkan pada tempat yang benar. Misalkan memiliki data 61 78 12 46 10 89

Langkah-langkah Pengurutan

- 61 **78** 12 46 10 89
  - Data sisip yang digunakan adalah 78, lalu dibandingkan dengan data yang ada sebelumnya jika ada sebelumnya tidak ada yang lebih besar maka tidak ada data yang harus digeser.
- 61 78 **12** 46 10 89
  - Data sisip yang digunakan adalah 12, lalu dibandingkan dengan data yang ada sebelumnya, terdapat 2 data yang lebih besar dibanding data sisip maka 2 data itu harus digeser satu tempat dan data sisip dipindah ke tempat paling depan.
- 61 78 **12** 46 10 89

  - Cek data sebelumnya apakah lebih besar data sisip? Jika ya geser 1 ke kanan

- 61 **12** 78 46 10 89

  - Cek data sebelumnya apakah lebih besar data sisip? Jika ya geser 1 ke kanan

- **12** 61 78 46 10 89

  - Jika sudah sampai di ujung index maka masukan data sisip pada index tersebut

- 12 61 78 **46** 10 89

  - Data sisip yang digunakan adalah 46, lalu dibandingkan dengan data yang ada sebelumnya, terdapat 2 data yang lebih besar dibanding data sisip dan 1 data lebih kecil dari data sisip, maka 2 data itu harus digeser satu tempat dan data sisip dipindah ke tempat sebelum data yang lebih kecil dari data sisip.

- 12 61 78 **46** 10 89

  - Cek data sebelumnya apakah lebih besar data sisip? Jika ya geser 1 ke kanan

- 12 61 **46** 78 10 89

  - Cek data sebelumnya apakah lebih besar data sisip? Jika ya geser 1 ke kanan

- 12 **46** 61 78 10 89

  - Cek data sebelumnya apakah lebih besar data sisip? Jika tidak masukan data sisip pada posisi tersebut.

- 12 46 61 78 10 89

Algoritma Pengurutan Sisip

Berikut ini merupakan algoritma pengurutan sisip:

Kode Program 6.1 Insertion Sort

```
tabInt : array [1..n] of integer
tabInt ← {n1, n2, n3, n4, n5, n6}
i ,j, dataSisip: integer
for i ←2 to n do
    dataSisip ← tabInt[i]
    j ← i-1
    while (dataSisip < tabInt[j]) and (j>0) do
          tabInt[j+1] ← tabInt[j]
          j ← j-1
    endWhile
    tabInt[j+1] ← dataSisip
endFor
```

Ascending atau Descending

- Ascending
  - while (dataSisip < tabInt[j]) and (j>0) do
- Descending
  - while (dataSisip > tabInt[j]) and (j>0) do

Latihan

Berikut ini program mengurutkan data dengan algoritm pengurutan sisip.

Mengurutkan Data Terdeklarasi

Pada Kode Program 6.2 terlihar proses pengurutkan pada data yang sudah dideklarasikan.

Kode Program 6.2 Mengurutkan Data Terdeklarasi

```
#include <stdio.h>

main(){

    int tabInt[6] = {61, 78, 12, 46, 10, 89};
    int i,j,data_sisip;

    printf("Tampilkan data belum terurut\n");
    for(i=0;i<6;i++){
        printf("%d  ", tabInt[i]);
    }
    printf("\n");

    for(i=1; i<6; i++){
        data_sisip = tabInt[i];
        j = i - 1;
        while((data_sisip > tabInt[j]) && (j >= 0)){
            /*jika data array lebih kecil dari data sisip
            maka data array digeser ke belakang*/
            tabInt[j+1] = tabInt[j];
            j = j - 1;
        }
        /*menempatkan data sisip pada array*/
        tabInt[j+1] = data_sisip;
    }

    printf("\nTampilkan data sudah terurut\n");
    for(i=0;i<6;i++){
        printf("%d  ", tabInt[i]);
    }
}
```

Mengurutkan Data Terurut Terbalik

```
#include <stdio.h>

main(){

    int tabInt[100000];
    int i,j,data_sisip,n;
    n=100000;

    //masukan data dari besar ke kecil
    for(i=0;i<n;i++){
        tabInt[i]=i;
    }

    //keluarkan data
    //printf("Data dari besar ke kecil\n");
    //for(i=0;i<n;i++){
       //printf("%d  ", tabInt[i]);
```

```
18      //}
19      //printf("\n");
20
21      //proses mengurutkan dari kecil ke besar
22      for(i=1; i<n; i++){
23          data_sisip = tabInt[i];
24          j = i - 1;
25          while((data_sisip < tabInt[j]) && (j >= 0)){
26              /*jika data array lebih kecil dari data sisip maka data
                    array digeser ke belakang*/
27              tabInt[j+1] = tabInt[j];
28              j = j - 1;
29          }
30          /*menempatkan data sisip pada array*/
31          tabInt[j+1] = data_sisip;
32      }
33
34      //keluarkan data
35      //printf("Data dari kecil ke besar\n");
36      //for(i=0;i<n;i++){
37          //printf("%d  ", tabInt[i]);
38      //}
39  }
```

Mengurutkan Data Random

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <math.h>
4   #include <time.h>
5
6   main(){
7
8       int tabInt[1000];
9       int i,j,data_sisip,random,n;
10      n=1000;
11
12      //masukan data random
13      srand(time(0));
14      for(i=0;i<n;i++){
15          random = rand() % n + 1;
16          tabInt[i]=random;
17      }
18
19      //keluarkan data
20      printf("Data Random\n");
21      for(i=0;i<n;i++){
```

```
22          printf("%d  ", tabInt[i]);
23      }
24      printf("\n\nData Sedang Diurutkan\n\n");
25
26      //proses mengurutkan dari kecil ke besar
27      for(i=1; i<n; i++){
28          data_sisip = tabInt[i];
29          j = i - 1;
30          while((data_sisip < tabInt[j]) && (j >= 0)){
31              /*jika data array lebih kecil dari data sisip maka data
                    array digeser ke belakang*/
32              tabInt[j+1] = tabInt[j];
33              j = j - 1;
34          }
35          /*menempatkan data sisip pada array*/
36          tabInt[j+1] = data_sisip;
37      }
38
39      //keluarkan data
40      printf("\nData dari kecil ke besar\n");
41      for(i=0;i<n;i++){
42          printf("%d  ", tabInt[i]);
43     }
44  }
```

Praktikum

1. Buatlah 3 buah program untuk mengurutkan 100 data, ketentuan setiap program sebagai berikut:

   (a) Program ke-1 mengurutkan data random menjadi terurut dari besar ke kecil.

   (b) Program ke-2 mengurutkan data terurut dari besar ke kecil menjadi terurut dari kecil ke besar.

   (c) Program ke-3 mengurutkan data terurut dari kecil ke besar menjadi terurut dari besar ke kecil.

2. Buatlah program untuk mengurutkan data dari kecil ke besar, kemudian hitunglah waktu yang diperlukan untuk mengurutkan:

   (a) 10 data random.

   (b) 10 data terurut dari kecil ke besar

   (c) 10 data terurut dari besar ke kecil

   (d) 100 data random.

(e) 100 data terurut dari kecil ke besar

(f) 100 data terurut dari besar ke kecil

(g) 1000 data random.

(h) 1000 data terurut dari kecil ke besar

(i) 1000 data terurut dari besar ke kecil

(j) 10000 data random.

(k) 10000 data terurut dari kecil ke besar

(l) 10000 data terurut dari besar ke kecil

(m) 25000 data random.

(n) 25000 data terurut dari kecil ke besar

(o) 25000 data terurut dari besar ke kecil

(p) 50000 data random.

(q) 50000 data terurut dari kecil ke besar

(r) 50000 data terurut dari besar ke kecil

(s) 75000 data random.

(t) 75000data terurut dari kecil ke besar

(u) 75000 data terurut dari besar ke kecil

(v) 100000 data random.

(w) 100000 data terurut dari kecil ke besar

(x) 100000 data terurut dari besar ke kecil

3. Buatlah grafik batang dari hasil pencatatan yang dikelompokan berdasarkan jumlah data yang diurutkan. Gunakan aplikasi Ms. Excel untuk membuat grafik.

Modul 7

Pengurutan Seleksi (Selection Sort)

Metode pengurutan ini mencari nilai terkecil atau terbesar dari data yang ada, kemudian ditempatkan pada index terdepan, lalu mencari lagi nilai terkecil atau terbesar kedua dari data kurang 1, setelah ketemu elemen kedua ditukar dengan nilai minimum, begitu seterusnya.

Prose Pengurutan

Misalkan data : 89 32 43 23 54 64 12 53 63 50, maka langkah-langkah pengurutannya adalah sebagai berikut:

1. Cari data terkecil pada baris bilangan untuk ditempatkan pada posisi ke-1 12 merupakan data terkecil. 89 32 43 23 54 64 12 53 63 50
2. Tukar posisi dengan bilangan yang berada di posisi ke-1 12 32 43 23 54 64 89 53 63 50
3. Cari data terkecil dari posisi bilangan ke 2 sampai ke-10 pada baris bilangan untuk ditempatkan pada posisi ke-2.
4. Maka diketahui 23 adalah bilangan terkecil ke-dua dari baris 12 32 43 23 54 64 89 53 63 50
5. Tukar posisi dengan bilangan yang berada di posisi ke-2 12 23 43 32 54 64 89 53 63 50
6. Proses terus berulang sampai posisi terakhir

Algoritma Pengurutan Seleksi

Berikut ini merupakan algoritma pengurutan sisip:

Kode Program 7.1 Pengurutan Seleksi

```
tabInt : array [1..n] of integer
tabInt ← {d1, d2, d3, d4, d5, d6,...,dn}
i,j, min,temp: integer
for i ←1 to n-1 do
    min←i
    for j← i+1 to n do
            if(tabInt[min] > tabInt[j]
                    min←j
    endFor
    temp ← tabInt[i]
    tabInt[i] ← tabInt[min]
    tabInt[min] ← temp
endFor
```

Ascending atau Descending

- Ascending
  - if(tabInt[min] > tabInt[j]
- Descending
  - if(tabInt[min] < tabInt[j]

Latihan

Berikut ini program mengurutkan data dengan algoritm pengurutan seleksi.

Mengurutkan Data Terdeklarasi

```
1  #include <stdio.h>
2
3  main(){
4
5      int tabInt[6] = {61, 78, 12, 46, 10, 89};
6      int i,j,temp,minIndeks;
7
8      //keluarkan data belum urut
```

```
9       for(i=0;i<6;i++){
10          printf("%d  ", tabInt[i]);
11      }
12      printf("\n");
13
14      for(i=0; i<6; i++){
15       /*inisialisasi indeks elemen minimum*/
16       minIndeks = i ;
17      /*perulangan mencari nilai minimum sepanjang indeks i + 1 sampai
            jumlah elemen array*/
18       for(j=i+1; j<6; j++){
19          if(tabInt[minIndeks] < tabInt[j]){
20             minIndeks = j;
21          }
22       }
23       //menukar posisi elemen
24       temp = tabInt[i];
25       tabInt[i] = tabInt[minIndeks];
26       tabInt[minIndeks] = temp;
27      }
28
29      for(i=0;i<6;i++){
30          printf("%d  ", tabInt[i]);
31      }
32  }
```

Mengurutkan Data Terurut Terbalik

```
1   #include <stdio.h>
2
3   main(){
4
5       int tabInt[100];
6       int i,j,temp,minIndeks,n;
7
8       n=100;
9       //masukan data
10      for(i=0;i<n;i++){
11          tabInt[i]=n-i;
12      }
13      printf("\n");
14
15      //keluarkan data
16      for(i=0;i<n;i++){
17         printf("%d  ", tabInt[i]);
18      }
19      printf("\n");
```

```
20
21      for(i=0; i<n; i++){
22      /*inisialisasi indeks elemen minimum*/
23      minIndeks = i ;
24     /*perulangan mencari nilai minimum sepanjang indeks i + 1 sampai
           jumlah elemen array*/
25      for(j=i+1; j<n; j++){
26         if(tabInt[minIndeks] > tabInt[j]){
27            minIndeks = j;
28         }
29      }
30      //menukar posisi elemen
31      temp = tabInt[i];
32      tabInt[i] = tabInt[minIndeks];
33      tabInt[minIndeks] = temp;
34     }
35
36     //keluarkan data
37     for(i=0;i<n;i++){
38        printf("%d  ", tabInt[i]);
39     }
40 }
```

Mengurutkan Data Random

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <math.h>
4  #include <time.h>
5
6  main(){
7
8      int tabInt[10000];
9      int i,j,temp,minIndeks,n,random;
10
11     n=10000;
12     //masukan data random
13     srand(time(0));
14     for(i=0;i<n;i++){
15         random = rand() % 10000 + 1;
16         tabInt[i]=random;
17     }
18
19     //keluarkan data
20     //for(i=0;i<n;i++){
21         //printf("%d  ", tabInt[i]);
22     //}
```

```
23      //printf("\n");
24
25       for(i=0; i<n; i++){
26       /*inisialisasi indeks elemen minimum*/
27       minIndeks = i ;
28      /*perulangan mencari nilai minimum sepanjang indeks i + 1 sampai
            jumlah elemen array*/
29       for(j=i+1; j<n; j++){
30          if(tabInt[minIndeks] > tabInt[j]){
31             minIndeks = j;
32          }
33       }
34       //menukar posisi elemen
35       temp = tabInt[i];
36       tabInt[i] = tabInt[minIndeks];
37       tabInt[minIndeks] = temp;
38      }
39
40      //keluarkan data
41      //for(i=0;i<n;i++){
42        //printf("%d  ", tabInt[i]);
43     //}
44  }
```

Praktikum

1. Buatlah 3 buah program untuk mengurutkan 100 data, ketentuan setiap program sebagai berikut:

   (a) Program ke-1 mengurutkan data random menjadi terurut dari besar ke kecil.

   (b) Program ke-2 mengurutkan data terurut dari besar ke kecil menjadi terurut dari kecil ke besar.

   (c) Program ke-3 mengurutkan data terurut dari kecil ke besar menjadi terurut dari besar ke kecil.

2. Buatlah program untuk mengurutkan data dari kecil ke besar, kemudian hitunglah waktu yang diperlukan untuk mengurutkan:

   (a) 10 data random.

   (b) 10 data terurut dari kecil ke besar

   (c) 10 data terurut dari besar ke kecil

   (d) 100 data random.

(e) 100 data terurut dari kecil ke besar

(f) 100 data terurut dari besar ke kecil

(g) 1000 data random.

(h) 1000 data terurut dari kecil ke besar

(i) 1000 data terurut dari besar ke kecil

(j) 10000 data random.

(k) 10000 data terurut dari kecil ke besar

(l) 10000 data terurut dari besar ke kecil

(m) 25000 data random.

(n) 25000 data terurut dari kecil ke besar

(o) 25000 data terurut dari besar ke kecil

(p) 50000 data random.

(q) 50000 data terurut dari kecil ke besar

(r) 50000 data terurut dari besar ke kecil

(s) 75000 data random.

(t) 75000 data terurut dari kecil ke besar

(u) 75000 data terurut dari besar ke kecil

(v) 100000 data random.

(w) 100000 data terurut dari kecil ke besar

(x) 100000 data terurut dari besar ke kecil

3. Buatlah grafik batang dari hasil pencatatan yang dikelompokan berdasarkan jumlah data yang diurutkan. Gunakan aplikasi Ms. Excel untuk membuat grafik.

Modul 8

Bubble Sort

Metode pengurutan yang menukarkan 2 buah elemen secara terus menerus sampai tidak ada elemen yang layak untuk ditukar lagi.

Proses Pengurutan

- Misalkan mempunyai data larik {43,54,32,67,49}
- Langkah 1 – Bandingkan data1 dengan data2. Jika data1>data2 maka tukar posisi. Jika data1<data2 tidak dilakukan pertukaran. Data larik{43,54,32,67,49}
- Langkah 2 – Bandingkan data2dengan data3. Jika data2>data3 maka tukar posisi. Jika data2<data3 tidak dilakukan pertukaran. Data larik{43,32,54,67,49}
- Langkah 3 – Bandingkan data3 dengan data4. Jika data3>data4 maka tukar posisi. Jika data3<data4 tidak dilakukan pertukaran. Data larik{43,32,54,67,49}
- Langkah 4 – Bandingkan data4 dengan data5. Jika data4>data5 maka tukar posisi. Jika data4<data5 tidak dilakukan pertukaran. Data larik{43,32,54,49,67}
- Langkah 5 – Bandingkan data1 dengan data2. Jika data1>data2 maka tukar posisi. Jika data1<data2 tidak dilakukan pertukaran. Data larik{32,43,54,49,67}
- Langkah 6 – Bandingkan data2dengan data3. Jika data2>data3 maka tukar posisi. Jika data2<data3 tidak dilakukan pertukaran. Data larik{32,43,54,49,67}
- Langkah 7 – Bandingkan data3 dengan data4. Jika data3>data4 maka tukar posisi. Jika data3<data4 tidak dilakukan pertukaran. Data larik{32,43,49,54,67}
- Langkah 8 – Bandingkan data4 dengan data5. Jika data4>data5 maka tukar posisi. Jika data4<data5 tidak dilakukan pertukaran. Data larik{32,43,49,54,67}
- Langkah-langkah diatas diulang terus sampai tidak terjadi pertukaran data sama sekali yang dalam artian data telah terurut.

Algoritma Bubble Sort

Kode Program 8.1 Bubble Sort

```
tabInt : array [1..n] of integer
tabInt ← {d1, d2, d3, d4, d5, d6,...,dn}
i,temp,tukar: integer
repeat
   tukar ←0
   for i← i to n-1
         if(tabInt[i] > tabInt[i+1]
                  temp ← tabInt[i]
                  tabInt[i] ← tabInt[i+1]
                  tabInt[i+1] ← temp
                   tukar ←1
   endFor
until (tukar<>1)
```

- **Ascending**
  - if(tabInt[i] > tabInt[i+1]
- **Descending**
  - if(tabInt[i] < tabInt[i+1]

Latihan

Berikut ini program mengurutkan data dengan algoritma pengurutan bubble

Mengurutkan Data Terdeklarasi

```
1  #include <stdio.h>
2
3  main(){
4
5      int tabInt[6] = {61, 78, 12, 46, 10, 89};
6      int i,j,temp,tukar,n=6;
```

```
7
8       //keluarkan data belum urut
9       for(i=0;i<6;i++){
10          printf("%d  ", tabInt[i]);
11      }
12      printf("\n");
13
14       do{
15      /*inisialisasi nilai tukar sebelum ada pertukaran diset false */
16      tukar = 0;
17      /*pengulangan dan memeriksa apakah ada pertukaran */
18      for(i=0; i<(n-1); i++){
19         /*jika ada nilai yang dipertukarkan */
20         if(tabInt[i] > tabInt[i+1]){
21            /* menukar posisi elemen */
22            temp = tabInt[i];
23            tabInt[i] = tabInt[i+1];
24            tabInt[i+1] = temp;
25            tukar = 1;
26         }
27      }
28      }while(tukar == 1);
29
30      for(i=0;i<n;i++){
31          printf("%d  ", tabInt[i]);
32      }
33  }
```

Mengurutkan Data Terurut Terbalik

```
1   #include <stdio.h>
2
3   main(){
4
5       int tabInt[1000];
6       int i,j,temp,tukar,n=1000;
7
8       //masukan data
9       for(i=0;i<n;i++){
10          tabInt[i]=n-i;
11      }
12
13      //keluarkan data
14      for(i=0;i<n;i++){
15          printf("%d  ", tabInt[i]);
16      }
17      printf("\n");
```

```
18
19      do{
20      /*inisialisasi nilai tukar sebelum ada pertukaran diset false */
21      tukar = 0;
22      /*pengulangan dan memeriksa apakah ada pertukaran */
23      for(i=0; i<(n-1); i++){
24         /*jika ada nilai yang dipertukarkan */
25         if(tabInt[i] > tabInt[i+1]){
26            /* menukar posisi elemen */
27            temp = tabInt[i];
28            tabInt[i] = tabInt[i+1];
29            tabInt[i+1] = temp;
30            tukar = 1;
31         }
32      }
33      }while(tukar == 1);
34
35      //keluarkan data
36      for(i=0;i<n;i++){
37          printf("%d  ", tabInt[i]);
38      }
39  }
```

Mengurutkan Data Random

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <math.h>
4   #include <time.h>
5
6   main(){
7
8       int tabInt[100];
9       int i,j,temp,tukar,random,n=100;
10      j=n-1;
11
12      //masukan data random
13      srand(time(0));
14      for(i=0;i<n;i++){
15          random = rand() % n;
16          tabInt[i]=random;
17      }
18
19      //keluarkan data
20      for(i=0;i<n;i++){
21          printf("%d  ", tabInt[i]);
22      }
```

```
23      printf("\n\nData Sedang Diurutkan\n\n");
24
25      do{
26      /*inisialisasi nilai tukar sebelum ada pertukaran diset false */
27      tukar = 0;
28      /*pengulangan dan memeriksa apakah ada pertukaran */
29      for(i=0; i<j; i++){
30         /*jika ada nilai yang dipertukarkan */
31         if(tabInt[i] > tabInt[i+1]){
32            /* menukar posisi elemen */
33            tukar = 1;
34            temp = tabInt[i];
35            tabInt[i] = tabInt[i+1];
36            tabInt[i+1] = temp;
37
38         }
39      }
40      }while(tukar == 1);
41
42      //keluarkan data
43      for(i=0;i<n;i++){
44        printf("%d  ", tabInt[i]);
45     }
46  }
```

Praktikum

1. Buatlah 3 buah program untuk mengurutkan 100 data, ketentuan setiap program sebagai berikut:

   (a) Program ke-1 mengurutkan data random menjadi terurut dari besar ke kecil.

   (b) Program ke-2 mengurutkan data terurut dari besar ke kecil menjadi terurut dari kecil ke besar.

   (c) Program ke-3 mengurutkan data terurut dari kecil ke besar menjadi terurut dari besar ke kecil.

2. Buatlah program untuk mengurutkan data dari kecil ke besar, kemudian hitunglah waktu yang diperlukan untuk mengurutkan:

   (a) 10 data random.

   (b) 10 data terurut dari kecil ke besar

   (c) 10 data terurut dari besar ke kecil

   (d) 100 data random.

(e) 100 data terurut dari kecil ke besar

(f) 100 data terurut dari besar ke kecil

(g) 1000 data random.

(h) 1000 data terurut dari kecil ke besar

(i) 1000 data terurut dari besar ke kecil

(j) 10000 data random.

(k) 10000 data terurut dari kecil ke besar

(l) 10000 data terurut dari besar ke kecil

(m) 25000 data random.

(n) 25000 data terurut dari kecil ke besar

(o) 25000 data terurut dari besar ke kecil

(p) 50000 data random.

(q) 50000 data terurut dari kecil ke besar

(r) 50000 data terurut dari besar ke kecil

(s) 75000 data random.

(t) 75000 data terurut dari kecil ke besar

(u) 75000 data terurut dari besar ke kecil

(v) 100000 data random.

(w) 100000 data terurut dari kecil ke besar

(x) 100000 data terurut dari besar ke kecil

3. Buatlah grafik batang dari hasil pencatatan yang dikelompokan berdasarkan jumlah data yang diurutkan. Gunakan aplikasi Ms. Excel untuk membuat grafik.

Modul 9

Rekursif

Rekursif adalah proses memanggil dirinya sendiri yang biasa dilakukan oleh fungsi atau prosedur pada pemrograman prosedural. Rekursif akan terus berjalan sampai kondisi berhenti terpenuhi.

Pada fungsi rekursif terdapat blok kode:

- Basis
  - kode yang menjadi titik berhenti dari proses rekursif
- Rekursif
  - kode dimana sebuah blok program memanggil dirinya sendiri

Latihan

Berikut ini program yang menggunakan konsep rekursif.

Menampilkan Nilai Urutan Angka

```
1  #include <stdio.h>
2
3  void tulis(int i){
4      if(i>-10){
5          printf("Nilai i=%d\n",i);
6          tulis(--i);
7      }
8  }
9
10 main(){
11     int i=10;
12     tulis(i);
13 }
```

Menampilkan Deret Angka

```
1  #include <stdio.h>
2
3  void deret(int i){
4      if(i<10){
5          printf("%d ",i);
6          deret(++i);
7      }
8  }
9
10 main(){
11     int i=0;
12     deret(i);
13 }
```

Menghitung Faktorial

```
1  #include <stdio.h>
2
3  int faktorial(int i){
4      if(i==0 || i==1) return 1;
5      return i*faktorial(i-1);
6  }
7
8  main(){
9      printf("%d",faktorial(6));
10 }
```

Praktikum

1. Buatlah program dengan metode rekursif untuk menampilkan deret bilangan

   (a) 0 2 4 6 8 10 12 16 18 20

   (b) 1 3 5 7 9 11 13 15 17 19

   (c) 1 -2 3 -4 5 -6 7 -8 9 -10

2. Buatlah program untuk merima masukan sebuah nilai integer positif kemudian hitung jumlah bilangan sebelumnya.

   - Masukan : 5
   - Keluaran : 15

Modul 10

Quicksort

Quicksort mengurutkan data dengen pendekatan divide-andconquer,yaitu membagi masalah semula menjadi beberapa submasalah sejenis yang lebih kecil dan menyelesaikannya. Quicksort melakukan pengurutan dalam putaran secara rekrusif. Pengulangan secara rekursi dilakukan jika jumlah data yang akan diurutkan lebih dari satu buah.

Langkah-langka Pengurutan

Pada algorima Quicksort ada variabel yang disebut Pivot. Pivot bertugas untuk membagi larik menjadi 2 bagian kemudian membagi lagi setiap 2 bagian itu menjadi 2 bagian dan seterusnya.Perhatikan langkah-langkah pengurutannya.

Data Awal

24	17	18	15	22	26	13	21	16	28

Nilai Pivot = 24

Pointer Bergerak Dari Kanan - Pertukaran 1

16	17	18	15	22	26	13	21		28

Pointer Bergerak Dari Kiri - Pertukaran 2

16	17	18	15	22		13	21	26	28

Pointer Bergerak Dari Kanan - Pertukaran 3

16	17	18	15	22	21	13	**24**	26	28

Data Sisi Kiri

16	17	18	15	22	21	13

Nilai Pivot = 16

Pointer Bergerak dari Kanan - Pertukaran 4

13	17	18	15	22	21	

Pointer Bergerak dari Kiri - Pertukaran 5

13		18	15	22	21	17

Pointer Bergerak dari Kiri - Pertukaran 6

13	15	18		22	21	17

Pointer Bergerak dari Kiri - Pertukaran 7

13	15	**16**	18	22	21	17

Data Sisi Kiri

13	15

Nilai Pivot = 13

Pointer Bergerak Dari Kanan

13	15

Data Sisi Kanan

18	22	21	17

Nilai Pivot = 18

Pointer Bergerak Dari Kanan - Pertukaran 8

17	22	21	

Pointer Bergerak Dari Kiri - Pertukaran 9

17	**18**	21	22

Data Sisi Kanan

13	15

Nilai Pivot = 21

Pointer Bergerak Dari Kanan

21	22

Data Sisi Kanan

26	28

Nilai Pivot = 26

Pointer Bergerak Dari Kanan

26	28

Maka hasil pengurutannya adalah

13	15	**16**	17	**18**	21	22	**24**	**26**	28

Fungsi QuickSort

```
1  void quickSort (int a[], int lo, int hi){
2  //  lo adalah index bawah, hi adalah index atas
3  //  dari bagian array yang akan di urutkan
4      int i=lo, j=hi, h;
5      int pivot=a[lo];
6
7  //  pembagian
8      do{
9          while (a[i]<pivot) i++;
10         while (a[j]>pivot) j--;
11         if (i<=j)
12         {
```

```
13              h=a[i]; a[i]=a[j]; a[j]=h;//tukar
14              i++; j--;
15          }
16      } while (i<=j);
17
18  //  pengurutan
19      if (lo<j) quickSort(a, lo, j);
20      if (i<hi) quickSort(a, i, hi);
21  }
```

Latihan Quicksort

Mengurutkan Data Terdeklarasi

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4
5   void quickSort (int a[], int lo, int hi){
6   //  lo adalah index bawah, hi adalah index atas
7   //  dari bagian array yang akan di urutkan
8       int i=lo, j=hi, h;
9       int pivot=a[lo];
10
11  //  pembagian
12      do{
13          while (a[i]<pivot) i++;
14          while (a[j]>pivot) j--;
15          if (i<=j)
16          {
17              h=a[i]; a[i]=a[j]; a[j]=h;//tukar
18              i++; j--;
19          }
20      } while (i<=j);
21
22  //  pengurutan
23      if (lo<j) quickSort(a, lo, j);
24      if (i<hi) quickSort(a, i, hi);
25  }
26
27
28  main(){
29      int tabInt[10]={24,17,18,15,22,26, 13, 21, 16, 28};
30      int i,n=10;
31
32      for(i=0;i<n;i++){
```

```
33          printf("%d ",tabInt[i]);
34      }
35       printf("\n");
36       quickSort(tabInt,0,n-1);
37
38       for(i=0;i<n;i++){
39          printf("%d ",tabInt[i]);
40       }
41  }
```

Mengurutkan Data Terurut Terbalik

```
1   #include <stdio.h>
2   #include <stdlib.h>
3
4   void quickSort (int a[], int lo, int hi){
5   //  lo adalah index bawah, hi adalah index atas
6   //  dari bagian array yang akan di urutkan
7       int i=lo, j=hi, h;
8       int pivot=a[lo];
9
10  //  pembagian
11      do{
12          while (a[i]<pivot) i++;
13          while (a[j]>pivot) j--;
14          if (i<=j)
15          {
16              h=a[i]; a[i]=a[j]; a[j]=h;//tukar
17              i++; j--;
18          }
19      } while (i<=j);
20
21  //  pengurutan
22      if (lo<j) quickSort(a, lo, j);
23      if (i<hi) quickSort(a, i, hi);
24  }
25
26
27  main(){
28      int tabInt[10];
29      int i,n=10;
30
31      for(i=0; i<n; i++){
32          tabInt[i]=100-i;//data terurut terbalik
33      }
34
35      for(i=0;i<n;i++){
```

```
36          printf("%d ",tabInt[i]);
37      }
38       printf("\n");
39       quickSort(tabInt,0,n-1);
40
41       for(i=0;i<n;i++){
42           printf("%d ",tabInt[i]);
43       }
44  }
```

Mengurutkan Data Random

```
1   #include <stdio.h>
2   #include <stdlib.h>
3   #include <math.h>
4   #include <time.h>
5
6   void quickSort (int a[], int lo, int hi){
7   //  lo adalah index bawah, hi adalah index atas
8   //  dari bagian array yang akan di urutkan
9       int i=lo, j=hi, h;
10      int pivot=a[lo];
11
12  //  pembagian
13      do{
14          while (a[i]<pivot) i++;
15          while (a[j]>pivot) j--;
16          if (i<=j)
17          {
18              h=a[i]; a[i]=a[j]; a[j]=h;//tukar
19              i++; j--;
20          }
21      } while (i<=j);
22
23  //  pengurutan
24      if (lo<j) quickSort(a, lo, j);
25      if (i<hi) quickSort(a, i, hi);
26  }
27
28
29  main(){
30      int tabInt[100000];
31      int i,n=100000;
32
33      printf("Data Random\n");
34      srand(time(0));
35      for(i=0; i<n; i++){
```

```
36          tabInt[i]=rand() % n + 1;
37      }
38
39      //for(i=0;i<n;i++){
40      //      printf("%d ",tabInt[i]);
41      // }
42      printf("\n");
43      printf("\n\nData Sedang Diurutkan\n\n");
44      quickSort(tabInt,0,n-1);
45
46      //for(i=0;i<n;i++){
47          //printf("%d ",tabInt[i]);
48      //}
49  }
```

Praktikum

1. Buatlah 3 buah program untuk mengurutkan 100 data, ketentuan setiap program sebagai berikut:

   (a) Program ke-1 mengurutkan data random menjadi terurut dari besar ke kecil.

   (b) Program ke-2 mengurutkan data terurut dari besar ke kecil menjadi terurut dari kecil ke besar.

   (c) Program ke-3 mengurutkan data terurut dari kecil ke besar menjadi terurut dari besar ke kecil.

2. Buatlah program untuk mengurutkan data dari kecil ke besar, kemudian hitunglah waktu yang diperlukan untuk mengurutkan:

   (a) 10 data random.

   (b) 10 data terurut dari kecil ke besar

   (c) 10 data terurut dari besar ke kecil

   (d) 100 data random.

   (e) 100 data terurut dari kecil ke besar

   (f) 100 data terurut dari besar ke kecil

   (g) 1000 data random.

   (h) 1000 data terurut dari kecil ke besar

   (i) 1000 data terurut dari besar ke kecil

   (j) 10000 data random.

(k) 10000 data terurut dari kecil ke besar

(l) 10000 data terurut dari besar ke kecil

(m) 25000 data random.

(n) 25000 data terurut dari kecil ke besar

(o) 25000 data terurut dari besar ke kecil

(p) 50000 data random.

(q) 50000 data terurut dari kecil ke besar

(r) 50000 data terurut dari besar ke kecil

(s) 75000 data random.

(t) 75000 data terurut dari kecil ke besar

(u) 75000 data terurut dari besar ke kecil

(v) 100000 data random.

(w) 100000 data terurut dari kecil ke besar

(x) 100000 data terurut dari besar ke kecil

3. Buatlah grafik batang dari hasil pencatatan yang dikelompokan berdasarkan jumlah data yang diurutkan. Gunakan aplikasi Ms. Excel untuk membuat grafik.

Modul 11

Pencarian

Pencarian atau searching suatu data pada sekumpulan data merupakan proses yang sangat penting. Proses pencarian dilakukan untuk mengetahui apakah data yang dicari terdapat pada sekumpulan data yang ada. Selain untuk mengetahui keberadaan data, informasi yang lain yang bisa didapat adalah letak dari data tersebut.

Jenis Pencarian

- Sequential Search
- Binary Search

Pencarian Beruntun (Sequential Search)

Sequential Search dapat dilakukan pada data yang belum terurut mau pun yang sudah terurut.Pencarian dilakukan dengan melakukan penelurusan data satu-persatu kemudian dicocokan dengan data yang dicari, jika tidak sama maka penelusuran dilanjutkan, jika sama maka penelurusan dihentikan, berarti data telah ditemukan.

Langkah-langkah Pencarian

- Misalkan terdapat data 89 23 54 26 93 43 64 76 35 58
- Ingin diketahui apakah pada sekumpulan data terdapat data 43?
- Maka dilakukan pencarian dengan mencocokan setiap nilai data dari data awal sampai data terakhir dan berhenti bila data sudah ditemukan.

Contoh Program

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 main(){
4     int tabInt[10]={24,17,18,15,22,26, 13,21, 16, 28};
5     int ketemu;
```

```
6       int cariData;
7       int i;
8       printf("Masukkan data yang dicari = ");
9       scanf("%d",&cariData);
10      i=0;
11      ketemu=0;
12      while (i<10 && ketemu!=1){
13          if (tabInt[i]==cariData){
14          ketemu=1;
15          }else{
16              i++;
17          }
18      }
19      if(ketemu==1){
20          printf("Data %d terdapat pada kumpulan data\n",cariData );
21      }else{
22          printf("Data %d tidak terdapat pada kumpulan data\n",cariData
                );
23      }
24  }
```

Pencarian Bagi Dua (Binary Search)

Pencarian bagi dua atau pencarian biner, hanya dapat dilakukan pada data yang sudah terurut. Bila data belum terurut dan akan dilakukan pencarian menggunakan metode ini maka terlebih dahulu harus diurutkan. Untuk data yang besar metode ini lebih efektif dibandingkan dengan metode pencarian beruntun (sequential search).

Proses Pencarian

- Misalkan terdapat data 99 12 23 67 56 60 34 78 29 84
- Misalkan yang di cari adalah data 84
- Langkah pertama adalah mengurutkan data
  - 12 23 29 34 56 60 67 78 84 99
  - 12 23 29 34 56 60 67 78 84 99
  - 12 23 29 34 56 60 67 78 84 99
  - 12 23 29 34 56 60 67 78 84 99
- Maka data 84 ditemukan

Contoh Program

```
#include <stdio.h>
#include <stdlib.h>
    int main(){
    int tabInt[10] = {12,23,29,34,56,60,67,78,84,99};
    int i,j,k;
    int cariData,ketemu;
    printf("Masukkan data yang dicari = ");
    scanf("%d",&cariData);
    i = 0; j = 9;ketemu = 0;
        while((ketemu == 0) && (i<=j)){
            k = (int)(i + j) / 2;
            if(tabInt[k] == cariData){
            ketemu = 1;
        }
        else{
            if(tabInt[k] > cariData){
                j = k - 1;
            }
            else{
                i = k + 1;
            }
        }
    }
    if(ketemu==1){
        printf("Data %d terdapat pada kumpulan data\n",cariData );
    }else{
        printf("Data %d tidak terdapat pada kumpulan data\n",cariData
           );
    }
}
```

Praktikum

1. Buatlah sebuah program yang yang menghasilkan data random sebanyak 100 dengan kisaran nilai antara 1-1000 kemudian lakukan pencarian berdasarkan nilai yang dimasukan dengan algoritma pencarian Sequential.

2. Buatlah sebuah program yang yang menghasilkan data random sebanyak 100 dengan kisaran nilai antara 1-1000 kemudian lakukan pencarian berdasarkan nilai yang dimasukan dengan algoritma pencarian Binary.