



MODUL PRAKTIKUM

STRUKTUR DATA



Dosen Pengampu
M.Bahrul Ulum, S.Kom., M.Kom

Asisten Lab
Mia Kastina

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS ILMU KOMPUTER

UNIVERSITAS ESA UNGGUL

JAKARTA

DAFTAR ISI

PRAKTIKUM 1

✚ PENGENALAN STRUKTUR DATA BAHASA C++.....	1
➤ Pembelajaran Sederhana	1
➤ Pemahaman Struktur Data.....	1
➤ Program Dev C++	1
➤ Pelaksanaan Praktikum	2
➤ Latihan.....	4

PRAKTIKUM 2

✚ ARRAY	5
➤ Teori Singkat	5
➤ Array Dimensi Satu	5
➤ Array Dimensi Dua	5
➤ Array Dimensi Tiga.....	6
➤ Ilustrasi Array.....	6
➤ Pelaksanaan Praktikum	7
➤ Latihan.....	8

PRAKTIKUM 3

✚ SEARCHING.....	9
➤ Teori Singkat.....	9
➤ Sequential Search	9
➤ Binary Search	9
➤ Ilustrasi Searching.....	9
➤ Pelaksanaan Praktikum	10
➤ Latihan.....	13

PRAKTIKUM 4& 5

✚ SORTING	14
➤ Teori Singkat.....	14
➤ Bubble Sort	14
➤ Selection Sort	14
➤ Insertion Sort.....	15
➤ Quick Sort	15
➤ Exchange Sort.....	15
➤ Ilustrasi Sorting	16
➤ Pelaksanaan Praktikum	21
➤ Latihan.....	23

PRAKTIKUM 6& 7

✚ LINKED LIST	24
➤ Teori Singkat.....	24
➤ Proses Dasar Linked List	24
➤ Single Linked List.....	27
➤ Double Linked List	27
➤ Circular Linked List.....	27
➤ Multiple Linked List	28
➤ Pelaksanaan Praktikum	28
➤ Latihan.....	34

PRAKTIKUM 8

✚ STRUCT	35
➤ Teori Singkat	35
➤ Deklarasi Struktur	35
➤ Pengaksesan Elemen Struct	35
➤ Pelaksanaan Praktikum	36
➤ Latihan	36

PRAKTIKUM 9

✚ POINTER	37
➤ Teori Singkat	37
➤ Operasi Pointer	37
➤ Pelaksanaan Praktikum	38
➤ Latihan	39

PRAKTIKUM 10& 11

✚ STACK	40
➤ Teori Singkat	40
➤ Operasi Utama Stack	40
➤ Operasi – Operasi Stack	40
➤ Pelaksanaan Praktikum	42
➤ Latihan	43

PRAKTIKUM 12 & 13

✚ QUEUE	44
➤ Teori Singkat	44
➤ Operasi Utama Queue	44
➤ Operasi – Operasi Queue	44
➤ Pelaksanaan Praktikum	46
➤ Latihan	47

PRAKTIKUM 14

✚ BINARY TREE	48
➤ Teori Singkat	48
➤ Ilustrasi Binary Tree	48
➤ Pembentukan Tree	48
➤ Operasi Utama Tree	48
➤ Pelaksanaan Praktikum	49
➤ Latihan	50

REFERENSI	50
-----------------	----

PRAKTIKUM 1 PENGENALAN STRUKTUR DATA BAHASA C++

Pembelajaran Sederhana

Bagaimana cara membentuk sebuah program dengan konsep struktur data dari atomik sampai terstruktur yang diimplementasikan dalam kasus sederhana ?

1. Memahami masalah secara menyeluruh dan persiapan data
2. Keputusan operasi-operasi yang dilakukan terhadap data
3. Penyimpanan data-data pada memori sehingga tersimpan dan terstruktur secara logis, operasinya efisien
4. Memahami bahasa pemrograman C++ untuk jenis data yang ada

Pemahaman Struktur Data

Struktur data adalah cara penyimpanan dan pengorganisasian data-data pada memori komputer maupun file secara efektif sehingga dapat digunakan secara efisien, termasuk operasi-operasi di dalamnya.

Di dalam struktur data berhubungan dengan 2 aktivitas:

1. Mendeskripsikan kumpulan obyek data yang sah sesuai dengan tipe data yang ada
2. Menunjukkan mekanisme kerja operasi-operasinya

Program Dev C++

Untuk memulai membuat sebuah program, editor yang kita gunakan adalah Dev C++. Dev C++ adalah sebuah IDE (Integrated Development Environment) C/C++ yang sudah dilengkapi dengan TDM-GCC Compiler (bagian dari *GNU Compiler Collection / GCC*). Dev-C++ merupakan IDE gratis dan full featur yang didistribusikan dibawah lisensi *GNU General Public License* untuk pemrograman C dan C++. IDE sendiri adalah Lembar kerja terpadu untuk pengembangan program.

Fungsi IDE DevC++ :

1. Menulis Program / Source Code.
2. Mengkompilasi Program (Compile)
3. Melakukan Pengujian Program (Debugging)
4. Mengaitkan Object dan Library ke Program (Linking)
5. Menjalankan Program (Running)

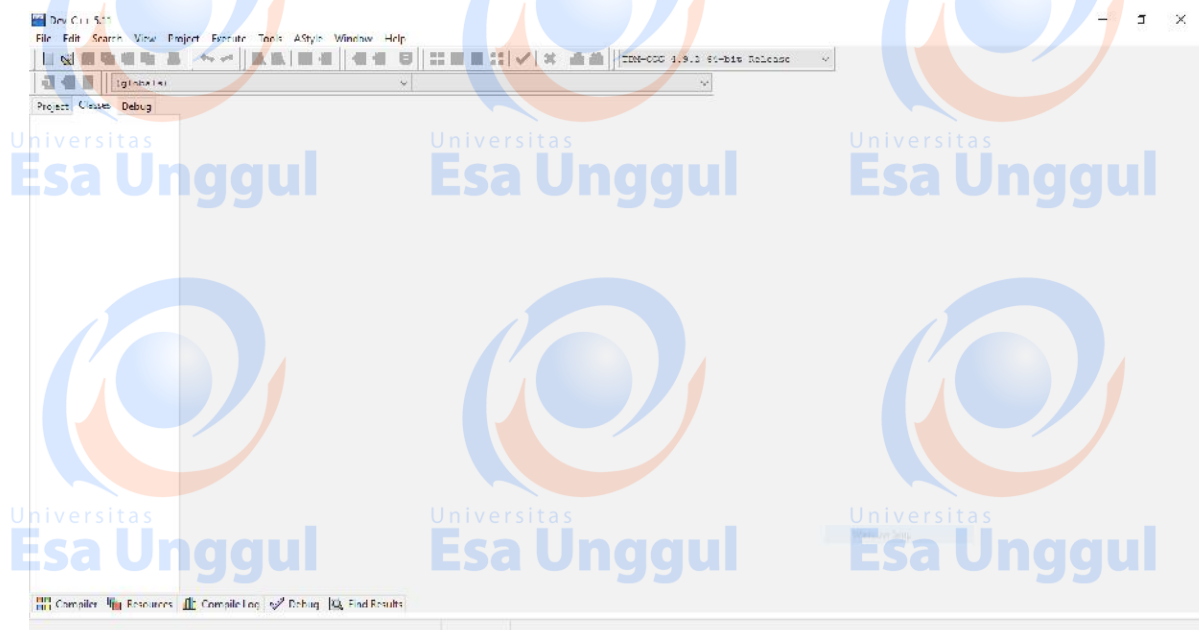
Dapat di download pada url berikut ini :

<https://sourceforge.net/projects/orwelldevcpp/>

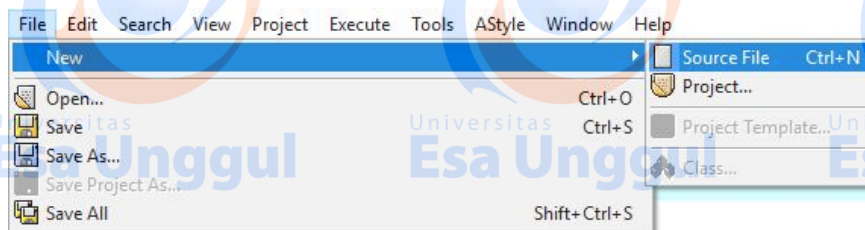
Pelaksanaan Praktikum

Langkah-langkah Praktikum

1. Buka editor **DevC++**.



2. Buatlah file baru dengan membuka menu **File > New > Source File** atau dengan **Ctrl + N**.



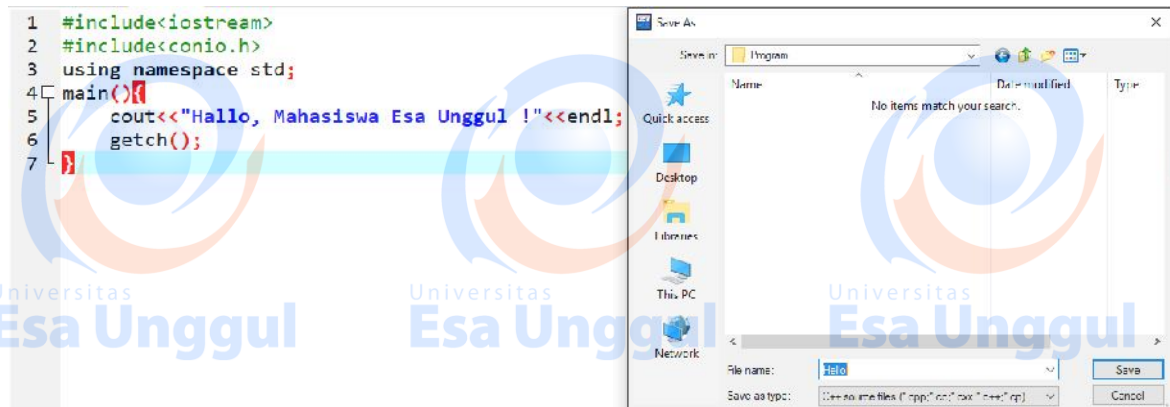
3. Tulislah Program 1.1 berikut ini.

```

1  #include<iostream>
2  #include<conio.h>
3  using namespace std;
4  main(){
5      cout<<"Hallo, Mahasiswa Esa Unggul !"<<endl;
6      getch();
7  }
```

Program 1.1 Hello.cpp

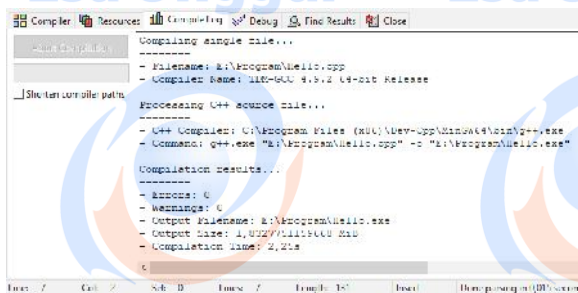
4. Simpan program yang telah dituliskan dengan membuka menu **File > Save as...**. Pilih lokasi penyimpanan dan beri nama file dengan "**Hello.cpp**".



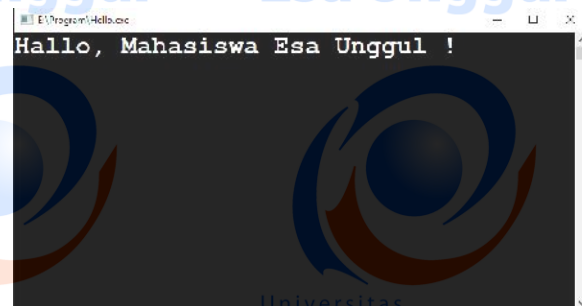
5. Lakukan kompilasi program melalui menu **Execute > Compile** atau dengan menekan **F9**, seperti Gambar 1.1.
6. Perhatikan hasil kompilasi program pada bagian “Compile Log”. Jika kompilasi sukses, maka akan ditampilkan pesan kurang lebih seperti Gambar 1.2.
7. Jalankan program (running) melalui menu **Execute > Run** atau dengan menekan **F10**. Hasil program akan ditampilkan pada window seperti pada Gambar 1.3.
8. Atau melalui menu **Compile & Run** atau menekan **F11**, untuk melakukan kompilasi dan menjalankan program secara langsung, seperti Gambar dibawah ini.



Gambar 1.1



Gambar 1.2



Gambar 1.3

Note :

Sebelum mulai melakukan coding program, diingat bahwa bahasa C atau C++ bersifat “case sensitive”, yang artinya huruf besar dan huruf kecil dibedakan. Maupun dalam code-code yang digunakan jangan sampai kurang sedikitpun. “Semangat ”

Latihan

Tuliskan dan jalankan beberapa program berikut ini dan jelaskan maksud pemrograman berikut dengan hasil running yang terjadi !

a. Program 1

```

1 #include<iostream>
2 #include<conio.h>
3 #include<windows.h>
4 using namespace std;
5 void gotoxy(int x, int y){
6     COORD coord;
7     coord.X = x;
8     coord.Y = y;
9     SetConsoleCursorPosition(GetStdHandle(STD_OUTPUT_HANDLE), coord);
10 }
11 int main(){
12     gotoxy(1,4);
13     cout<<"Hallo Mahasiswa";
14     getch();
15 }

```

b. Program 2

```

1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int main(){
5     float bil1,bil2,hasil;
6
7     bil1 = 50;
8     bil2 = 50;
9
10    hasil = bil1*bil2;
11
12    cout<<"Hasil kali dari "<<bil1<<" dan "<<bil2<<" adalah "<<hasil;
13    getch();
14 }

```

c. Program 3

```

1 #include<iostream>
2 #include<conio.h>
3 using namespace std;
4 int main(){
5     const int sks=4,nim=201581178;
6     char nama[12]="Mia Kastina",matkul[]="Struktur Data";
7     float nilai1,nilai2,nilai3;
8     nilai1=90;
9     nilai2=80;
10    nilai3=(nilai1 + nilai2)/2;
11    cout<<"Nama Mahasiswa : "<<nama<<"\n";
12    cout<<"NIM          |: "<<nim<<"\n";
13    cout<<"Mata Kuliah    : "<<matkul<<"\n";
14    cout<<"SKS           : "<<sks<<"\n";
15    cout<<"Nilai Teori    : "<<nilai1<<"\n";
16    cout<<"Nilai Praktek  : "<<nilai2<<"\n";
17    cout<<"Nilai Akhir    : "<<nilai3<<"\n";
18    getch();
19 }

```

PRAKTIKUM 2

ARRAY

Teori Singkat

Array atau Larik merupakan Struktur Data sederhana yang dapat didefinisikan sebagai pemesanan alokasi memori sementara pada komputer. Dalam implementasinya, Array merupakan kumpulan dari nilai-nilai data bertipe sama dalam urutan tertentu yang menggunakan sebuah nama yang sama. Nilai-nilai data pada suatu larik disebut dengan elemen-elemen larik. Letak urutan dari suatu larik ditunjukkan oleh suatu subscript atau index.

Karakteristik Array :

- a. Mempunyai batasan dari pemesanan alokasi memori (bersifat statis)
- b. Mempunyai tipe data sama (bersifat homogen)
- c. Dapat diakses secara acak

Hal yang harus diketahui dalam mendeklarasikan array :

- a. Tipe data array
- b. Nama variabel array
- c. Subskrip / index array

Jenis Array (yang akan dipelajari) adalah :

- a. Array Dimensi Satu
- b. Array Dimensi Dua
- c. Array Dimensi Tiga

Array Dimensi Satu

Array dimensi satu adalah array yang terdiri dari n buah kolom atau array yang terdiri dari 1 subskrip array saja. Setiap elemen array satu dimensi dapat diakses melalui indeks yang terdapat di dalamnya.

Bentuk deklarasi :

tipe_data nama_array [jumlah_element];

Contoh :

int ANGKA [10];

atau

int ANGKA [10] = {2,3,4,5,6,1,7,3,55,3};

Array Dimensi Dua

Array dua dimensi adalah array yang tersusun dalam bentuk baris dan kolom membentuk tabel, singkatnya dalam array dua dimensi kita membuat data yang tersusun dalam tabel.

Bentuk deklarasi :

tipe_data nama_array [jumlah_element_baris] [jumlah_element_kolom];

Contoh :

Inty[2][2];

Array Dimensi Tiga

Array tiga dimensi adalah array yang memiliki banyak dimensi. Karakteristik tersebut memiliki dimensi sesuai dengan kebutuhan.

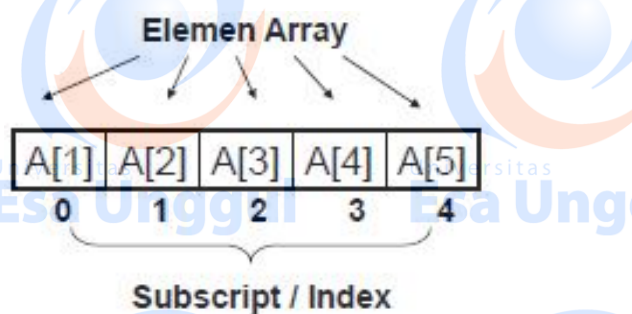
Bentuk deklarasi:

tipe_data nama_variabel[jumlah_baris][jumlah_kolom][panjang_karakter];

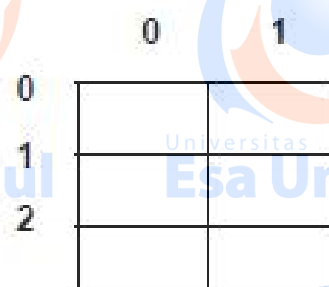
Contoh :

int x[2][2][2];

Ilustrasi Array



Gambar 2.1



Gambar 2.2



Gambar 2.3

Keterangan :

Gambar 2.1 = Array Dimensi 1

Gambar 2.2 = Array Dimensi 2

Gambar 2.3 = Array Dimensi 3

Pelaksanaan Praktikum

a. Program 1

Array Dimensi 1

```

1  #include<iostream>
2  #include<conio.h>
3  using namespace std;
4  int main()
5  {
6      int a;
7      int b[5];
8      for(a=0;a<5;a++)
9      {
10         cout<<"Masukan Angka indeks ke ["<<a<<" = ";
11         cin>>b[a];
12     }
13     for(a=0;a<5;a++)
14     {
15         cout<<"\nTampilan angka ke "<<a+1<<" = ";
16         cout<<b[a];
17     }
18     getch();
19 }

```

b. Program 2

Array Dimensi 2

```

1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4  int main(){
5      int matrik[3][4] = {{0,1,2,3},{4,5,6,7},{8,9,1,2}};
6      int i,j;
7      cout<<"Matrik 3x4\n";
8      cout<<"=====\n";
9      for(i=0;i<3;i++){
10         for(j=0;j<4;j++){
11             cout<<" "<<matrik[i][j];
12         }
13         cout<<endl;
14     }
15     getch();
16 }

```

c. Program 3 Array Dimensi 3

```

1  #include <iostream>
2  #include <conio.h>
3  using namespace std;
4  int main()
5  {
6      int i,s;
7      char h=64, nama[4][5][22]={
8          "Rifky", "Destian", "Vienie", "Andre", "Mia",
9          "Della", "Vanya", "Vio", "Indri", "Nilna",
10         "Hakim", "Bagus", "Firman", "Amin", "Awan",
11         "Ramdhan", "Gilang", "Lingga", "Rizky", "Kevin"
12     };
13     cout<<" Daftar Nama Kelompok : \n";
14     for(i=0; i<4; i++)
15     {
16         ++h;
17         cout<<" Kelompok "<<h;
18         for(s=0; s<5; s++)
19         {
20             cout<<"\n\t"<<s+1<<" . "<<nama[i][s];
21         }
22         cout<<"\n";
23     }
24     getch();
25 }

```

Latihan

Buatlah sebuah program dari pernyataan berikut ini (pilih salah satu) :

1. Suatu larik dengan nama day di deklarasikan sebagai berikut:
inthari[]={mon,tue,wed,thu,fri}
2. Suatu larik untuk menghitung rata-rata deret bilangan ganjil 1 sampai 100 yang habis dibagi 5.

PRAKTIKUM 3 SEARCHING

Teori Singkat

Searching merupakan proses fundamental dalam pengelolaan data. Proses pencarian adalah menemukan nilai (data) tertentu di dalam sekumpulan data yang bertipe sama (baik bertipe dasar atau bertipe bentukan).

Metode Searching (yang akan dipelajari) adalah :

1. Sequential Search
2. Binary Search

Sequential Search

Sequential Search atau sering disebut pencarian linier. Menggunakan prinsip data yang ada di dibandingkan satu persatu secara berurutan dengan yang dicari. Pada dasarnya, pencarian ini hanya melakukan pengulangan dari 1 sampai dengan jumlah data. Pada setiap pengulangan, di bandingkan data ke- i dengan yang dicari. Apabilasama, berarti data telah ditemukan. Sebaliknya apabila sampai akhir pengulangan, tidak ada yang sama berarti data tidak ada.

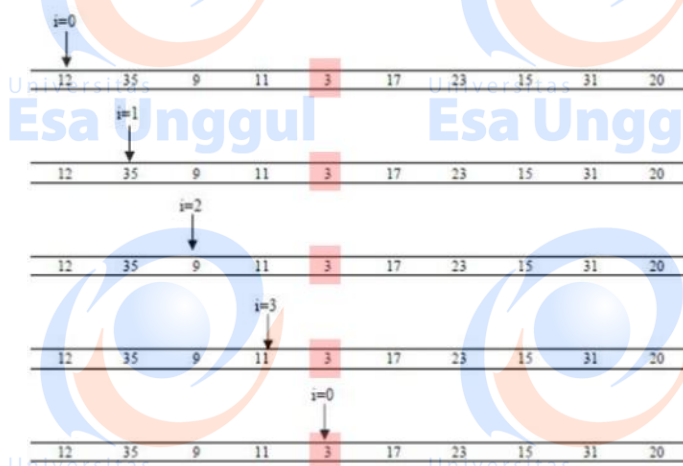
Binary Search

Binary search adalah sebuah algoritma pencarian dengan cara membagi data menjadi dua bagian setiap kali terjadi proses pencarian untuk menemukan nilai tertentu dalam sebuah larik (array) linear. Sebuah pencarian biner mencari nilai tengah (median), melakukan sebuah perbandingan untuk menentukan apakah nilai yang dicari ada sebelum atau sesudahnya, kemudian mencari setengah sisanya dengan cara yang sama.

Ilustrasi Searching

1. Sequential Search

Mencari posisi data dengan nilai 3. $key=3$;



Data[4]=3 sama dengan $key=3$ maka **data ditemukan** dan diberikan nilai pengembalian i (posisi) dan proses dihentikan.

2. Binary Search

Contoh Data:

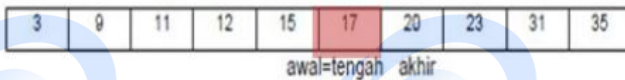
Misalnya data yang dicari 17



Karena $17 > 15$ (data tengah), maka: **awal = tengah + 1**



Karena $17 < 23$ (data tengah), maka: **akhir = tengah - 1**



Karena $17 = 17$ (data tengah), maka **KETEMU!**

Pelaksanaan Praktikum

a. Program 1

```

1  #include<iostream>
2  #include<conio.h>
3  #include<stdlib.h>
4  using namespace std;
5  int main(){
6      system("cls");
7      int data[8] = {8,27,9,20,10,1997,1996,100};
8      int cari;
9      int flag = 0;
10     cout<<"Masukkan data yang ingin dicari = ";
11     cin>>cari;
12     for(int i=0;i<8;i++){
13         if(data[i]==cari) flag=1;
14     }
15     if(flag==1)
16     {
17         cout<<"Data ada !\n";
18     }
19     else{
20         cout<<"Data tidak ada !\n";
21     }
22 }
```

b. Program 2

```
1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 int main()
5 {
6     int n,i,posisi,x,temu;
7     int a[5];
8     cout<<"Pencarian data dengan sequential search \n\n";
9     cout<<"Banyak data : ";
10    cin>>n;
11    for(i=0;i<n;i++)
12    {
13        cout<<"Masukan angka : ";
14        cin>>a[i];
15    }
16    cout<<"Data yang akan dicari : ";
17    cin>>x;
18    temu=0;
19    i=0;
20    while((temu == 0)&&(i<n))
21    {
22        if(a[i]==x)
23        {
24            temu=1;
25            posisi=i;
26        }
27        else
28            i=i+1;
29    }
30    if(temu==1)
31        cout<<"Ditemukan pada posisi "<<posisi+1<<" bernilai "<<x;
32    else
33        cout<<" Tidak ditemukan \n"<<x;
34    getch();
35 }
```

c. Program 3

```
1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 int data[10]={1,3,4,7,12,25,40,65,78,90};
5 int binary_search(int cari)
6 {
7     int a,b,c;
8     int n=10;
9     a=0;
10    b=n-1;
11    int temu=0;
12    while (a<=b && temu==0)
13    {
14        c=(a+b)/2;
15        if (data[c]==cari)
16            temu=1;
17        else
18            if(cari<data[c])
19                b=c-1;
20            else a=c+1;
21    }
22    if(temu==1) return 1; else return 0;
23 }
24 int main()
25 {
26     int cari, hasil;
27     cout<<"Pencarian data dengan binary search \n\n";
28     cout<<"Masukan data yang ingin di cari = ";
29     cin>>cari;
30     hasil = binary_search(cari);
31     if(hasil==1)
32     {
33         cout<<"Data ada!"<<endl;
34     }
35     else
36     if(hasil==0)
37     cout<<"Data tidak ada!"<<endl;
38     getch();
39 }
```

Sel: 0 Lines: 39 Length: 664 Insert Done parsing in 0,016 seconds

Latihan

Tuliskan dan jalankan dua program berikut ini dan jelaskan perbedaan dari kedua pemrograman tersebut!

a. Program 1

```

1 #include<iostream>
2 #include<conio.h>
3 #include<stdlib.h>
4 using namespace std;
5 int main()
6 {
7     system("cls");
8     int n,i,cari,arr[50];
9     cout<<"Masukkan jumlah angka : ";
10    cin>>n;
11    cout<<"Masukkan "<<n<<" angka :\n";
12    for(i=0;i<n;i++)
13    {
14        cin>>arr[i];
15    }
16    cout<<"Masukkan angka yang dicari : ";
17    cin>>cari;
18    for(i=0;i<n;i++)
19    {
20        if(arr[i]==cari)
21        {
22            cout<<cari<<" ditemukan pada lokasi "<<i+1;
23        }
24    }
25    if(i==0)
26    {
27        cout<<"Tidak ditemukan!\n"<<cari<<" tidak termasuk dalam inputan";
28    }
29    getch();
30 }

```

b. Program 2

```

1 #include<iostream>
2 #include<conio.h>
3 #include<stdlib.h>
4 using namespace std;
5 int main()
6 {
7     system("cls");
8     int n, i, arr[50],cari,atas,bawah,tengah;
9     cout<<"Masukkan jumlah angka : ";
10    cin>>n;
11    cout<<"Masukkan "<<n<<" angka :\n";
12    for (i=0; i<n; i++)
13    {
14        cin>>arr[i];
15    }
16    cout<<"Masukkan angka yang akan dicari : ";
17    cin>>cari;
18    atas = 0;
19    bawah = n-1;
20    tengah = (atas+bawah)/2;
21    while (atas <= bawah)
22    {
23        if(arr[tengah] < cari)
24        {
25            atas = tengah + 1;
26        }
27        else if(arr[tengah] == cari)
28        {
29            cout<<cari<<" ditemukan pada lokasi "<<tengah+1<<"\n";
30            break;
31        }
32        else
33        {
34            bawah = tengah - 1;
35        }
36        tengah = (atas + bawah)/2;
37    }
38    if(atas > bawah)
39    {
40        cout<<"Tidak ditemukan!\n"<<cari<<" tidak termasuk dalam inputan.";
41    }
42    getch();
43 }

```


PRAKTIKUM 4& 5

SORTING

Teori Singkat

Sorting adalah proses mengatur sekumpulan objek menurut aturan atau susunan tertentu. Urutan objek tersebut dapat menaik (ascending = dari data kecil ke data lebih besar) atau menurun (descending = dari data besar ke data lebih kecil).

Jenis Sorting(yang akan dipelajari) adalah :

1. Bubble sort (gelembung)
2. Selection sort (maksimum/minimum)
3. Insertion sort (sisip)
4. Quick sort
5. exchange sort

Bubble Sort

Bubble Sort merupakan metode sorting termudah, diberi nama “Bubble” karena proses pengurutan secara berangsur- angsur bergerak/berpindah ke posisinya yang tepat, seperti gelembung yang keluar dari sebuah gelas bersoda.

Jika pengurutan ascending :

jika elemen sekarang lebih besar dari elemen berikutnya maka kedua elemen tersebut ditukar

Jika pengurutan descending :

jika elemen sekarang lebih kecil dari elemen berikutnya, maka kedua elemen tersebut ditukar

Ketika satu proses telah selesai, maka bubble sort akan mengulangi proses, demikian seterusnya. Kapan berhentinya? Bubble sort berhenti jika seluruh array telah diperiksa dan tidak ada pertukaran lagi yang bisa dilakukan, serta tercapai perurutan yang telah diinginkan. Algoritma Bubble Sorting mudah dalam sintaks, tetapi lebih lambat dibandingkan dengan algoritma sorting yang lain

Selection Sort

Selection Sort merupakan kombinasi antara sorting dan searching. Untuk setiap proses, akan dicari elemen-elemen yang belum diurutkan yang memiliki nilai terkecil atau terbesar akan dipertukarkan ke posisi yang tepat di dalam array.

Misalnya:

Untuk putaran pertama, akan dicari data dengan nilai terkecil dan data ini akan ditempatkan di indeks terkecil (data[0]), pada putaran kedua akan dicari data kedua terkecil, dan akan ditempatkan di indeks kedua (data[1])

Selama proses, perbandingan dan pengubahan hanya dilakukan pada indeks perbandingan saja, pertukaran data secara fisik terjadi pada akhir proses.

Insertion Sort

Insertion Sort mirip dengan cara orang mengurutkan kartu, selembat demi selembat kartu diambil dan disisipkan (insert) ke tempat yang seharusnya.

- Pengurutan dimulai dari data ke-2 sampai dengan data terakhir, jika ditemukan data yang lebih kecil, maka akan ditempatkan (diinsert) diposisi yang seharusnya.
- Pada penyisipan elemen, maka elemen-elemen lain akan bergeser ke belakang

Quick Sort

Quick Sort merupakan suatu algoritma pengurutan data yang menggunakan teknik pemecahan data menjadi partisi-partisi, sehingga metode ini disebut juga dengan nama partition exchange sort. Untuk memulai irterasi pengurutan, pertama-tama sebuah elemen dipilih dari data, kemudian elemen-elemen data akan diurutkan diatur sedemikian rupa, sehingga nilai variabel Sementara berada di suatu posisi ke I yang memenuhi kondisi sebagai berikut :

1. Semua elemen di posisi ke 1 sampai dengan ke I-1 adalah lebih kecil atau sama dengan Sementara.
2. Semua elemen di posisi ke I+1 sampai dengan ke N adalah lebih besar atau sama dengan Sementara.

Exchange Sort

Exchange Sort sangat mirip dengan Bubble Sort, banyak yang mengatakan Bubble Sort sama dengan Exchange Sort.

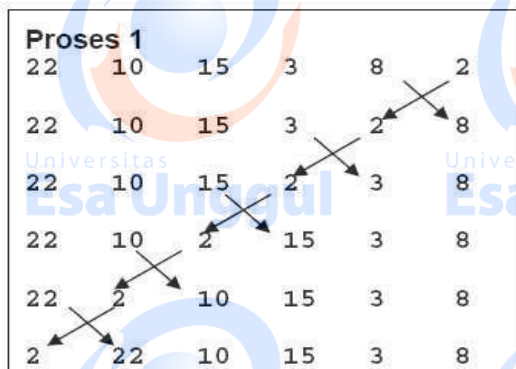
Pebedaan :

dalam hal bagaimana membandingkan antar elemen-elemennya.

- Exchange sort membandingkan suatuelemen dengan elemen-elemen lainnya dalam array tersebut, dan melakukan pertukaran elemen jika perlu. Jadi ada elemen yang selalu menjadi elemen pusat (pivot).
- Sedangkan Bubble sort akan membandingkan elemen pertama/terakhir dengan elemen sebelumnya/sesudahnya, kemudian elemen tersebut itu akan menjadi pusat (pivot) untuk dibandingkan dengan elemen sebelumnya/sesudahnya lagi, begitu seterusnya.

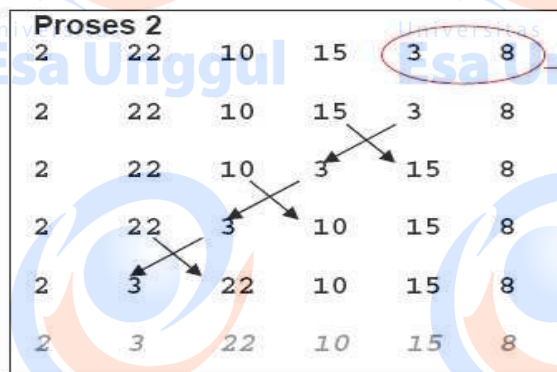
Ilustrasi Sorting

1. Bubble Sort



Gambar 1. Proses ke-1 algoritma Bubble Sorting

Pada gambar diatas, pengecekan dimulai dari data yang paling akhir, kemudiandibandingkan dengan data di depannya, jika data di depannya lebih besar maka akan ditukar.

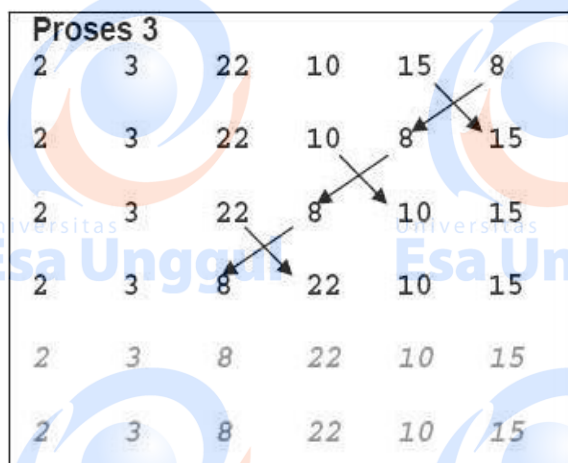


Tidak ada pertukaran, karena $3 < 8$

Pegurutan berhenti di sini!

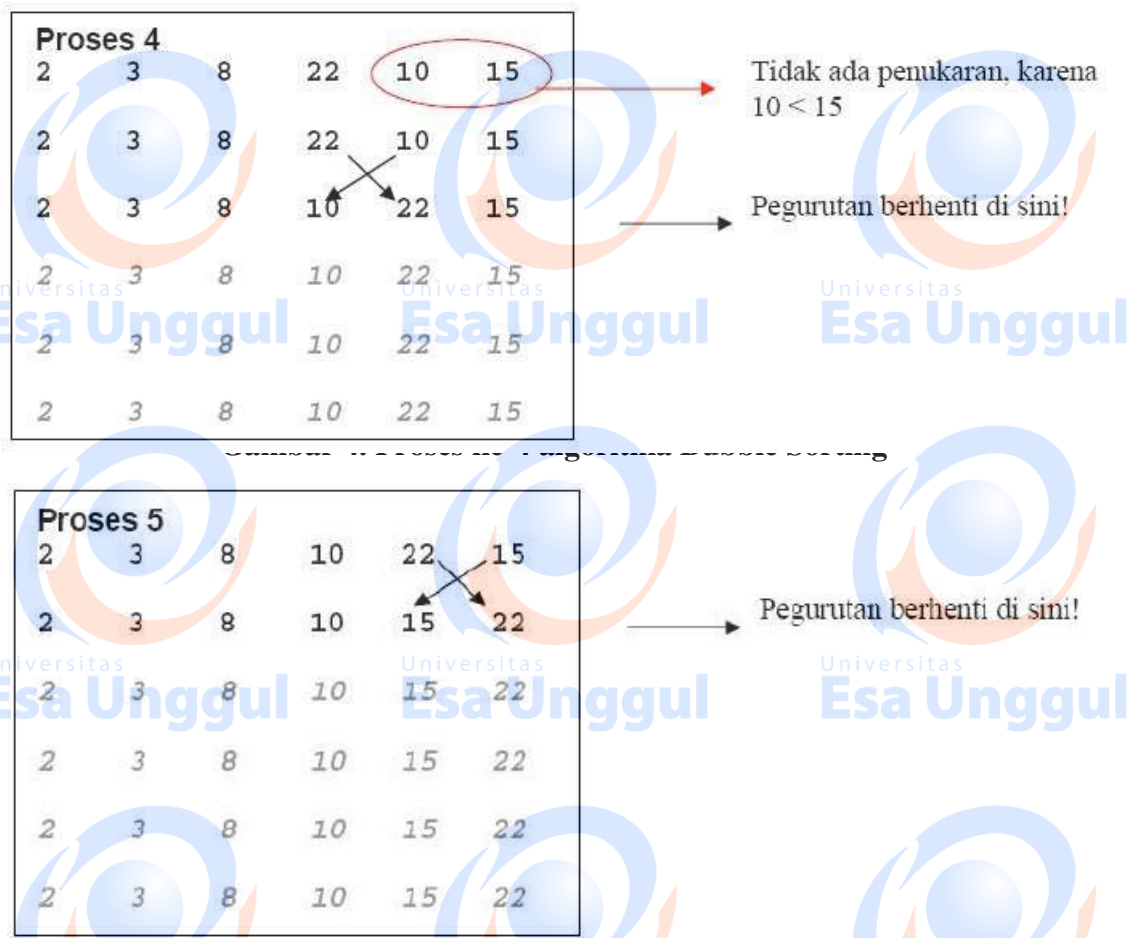
Gambar 2. Proses ke-2 algoritma Bubble Sorting

Pada proses kedua, pengecekan dilakukan sampai dengan data ke-2 karena data pertama sudah paling kecil.



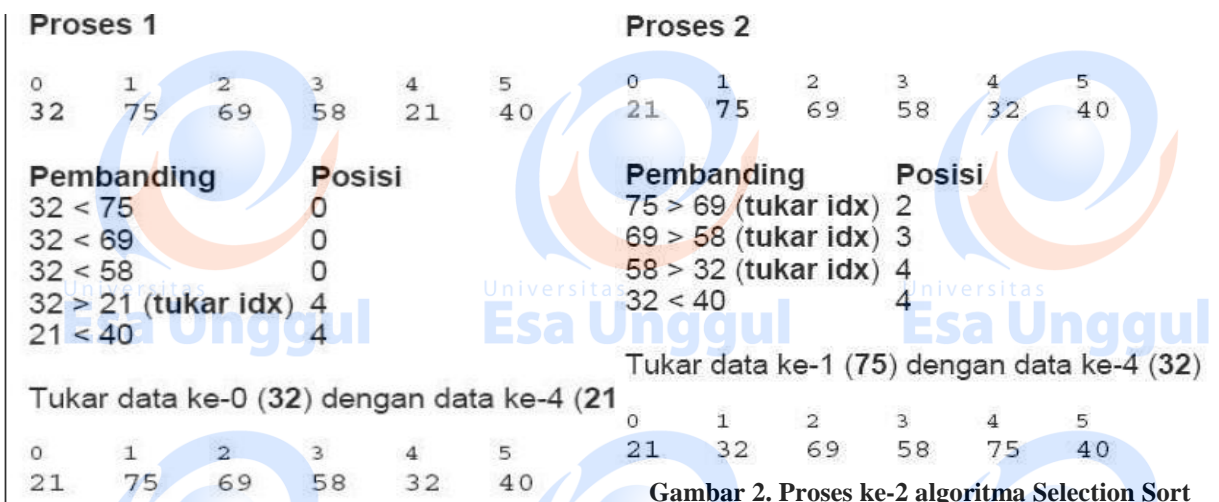
Pegurutan berhenti di sini!

Gambar 3. Proses ke-3 algoritma Bubble Sorting



Gambar 5. Proses ke-5 algoritma Bubble Sorting

2. Selection Sort



Gambar 2. Proses ke-2 algoritma Selection Sort

Proses 3

0 1 2 3 4 5
21 32 69 58 75 40

Pembanding Posisi
69 > 58 (tukar idx) 3
58 < 75 3
58 > 40 5

Tukar data ke-2 (69) dengan data ke-5 (40)

0 1 2 3 4 5
21 32 40 58 75 69

Gambar 3. Proses ke-3 algoritma Selection Sort

Proses 4

0 1 2 3 4 5
21 32 40 58 75 69

Pembanding Posisi
58 < 75 3
58 < 69 3

Tukar data ke-3 (58) dengan data ke-3 (58)

0 1 2 3 4 5
21 32 40 58 75 69

Gambar 4. Proses ke-4 algoritma Selection Sort

Proses 5

0 1 2 3 4 5
21 32 40 58 75 69

Pembanding Posisi
75 > 69 5

Tukar data ke-4 (75) dengan data ke-5 (69)

0 1 2 3 4 5
21 32 40 58 69 75

Gambar 5. Proses ke-5 algoritma Selection Sort

3. Insertion Sort

<p>Proses 1</p> <p>0 1 2 3 4 5 22 10 15 3 8 2</p> <table border="1"> <thead> <tr> <th>Temp</th> <th>Cek</th> <th>Geser</th> </tr> </thead> <tbody> <tr> <td>10</td> <td>Temp < 22?</td> <td>Data ke-0 ke posisi 1</td> </tr> </tbody> </table> <p>Temp menempati posisi ke -0</p> <p>0 1 2 3 4 5 10 22 15 3 8 2</p>	Temp	Cek	Geser	10	Temp < 22?	Data ke-0 ke posisi 1	<p>Proses 2</p> <p>0 1 2 3 4 5 10 22 15 3 8 2</p> <table border="1"> <thead> <tr> <th>Temp</th> <th>Cek</th> <th>Geser</th> </tr> </thead> <tbody> <tr> <td>15</td> <td>Temp < 22</td> <td>Data ke-1 ke posisi 2</td> </tr> <tr> <td>15</td> <td>Temp > 10</td> <td>-</td> </tr> </tbody> </table> <p>Temp menempati posisi ke-1</p> <p>0 1 2 3 4 5 10 15 22 3 8 2</p>	Temp	Cek	Geser	15	Temp < 22	Data ke-1 ke posisi 2	15	Temp > 10	-												
Temp	Cek	Geser																										
10	Temp < 22?	Data ke-0 ke posisi 1																										
Temp	Cek	Geser																										
15	Temp < 22	Data ke-1 ke posisi 2																										
15	Temp > 10	-																										
<p>Proses 3</p> <p>0 1 2 3 4 5 10 15 22 3 8 2</p> <table border="1"> <thead> <tr> <th>Temp</th> <th>Cek</th> <th>Geser</th> </tr> </thead> <tbody> <tr> <td>3</td> <td>Temp < 22</td> <td>Data ke-2 ke posisi 3</td> </tr> <tr> <td>3</td> <td>Temp < 15</td> <td>Data ke-1 ke posisi 2</td> </tr> <tr> <td>3</td> <td>Temp < 10</td> <td>Data ke-0 ke posisi 1</td> </tr> </tbody> </table> <p>Temp menempati posisi ke-0</p> <p>0 1 2 3 4 5 3 10 15 22 8 2</p>	Temp	Cek	Geser	3	Temp < 22	Data ke-2 ke posisi 3	3	Temp < 15	Data ke-1 ke posisi 2	3	Temp < 10	Data ke-0 ke posisi 1	<p>Proses 4</p> <p>0 1 2 3 4 5 3 10 15 22 8 2</p> <table border="1"> <thead> <tr> <th>Temp</th> <th>Cek</th> <th>Geser</th> </tr> </thead> <tbody> <tr> <td>8</td> <td>Temp < 22</td> <td>Data ke-3 ke posisi 4</td> </tr> <tr> <td>8</td> <td>Temp < 15</td> <td>Data ke-2 ke posisi 3</td> </tr> <tr> <td>8</td> <td>Temp < 10</td> <td>Data ke-1 ke posisi 2</td> </tr> <tr> <td>8</td> <td>Temp > 3</td> <td>-</td> </tr> </tbody> </table> <p>Temp menempati posisi ke-1</p> <p>0 1 2 3 4 5 3 8 10 15 22 2</p>	Temp	Cek	Geser	8	Temp < 22	Data ke-3 ke posisi 4	8	Temp < 15	Data ke-2 ke posisi 3	8	Temp < 10	Data ke-1 ke posisi 2	8	Temp > 3	-
Temp	Cek	Geser																										
3	Temp < 22	Data ke-2 ke posisi 3																										
3	Temp < 15	Data ke-1 ke posisi 2																										
3	Temp < 10	Data ke-0 ke posisi 1																										
Temp	Cek	Geser																										
8	Temp < 22	Data ke-3 ke posisi 4																										
8	Temp < 15	Data ke-2 ke posisi 3																										
8	Temp < 10	Data ke-1 ke posisi 2																										
8	Temp > 3	-																										

Proses 5					
0	1	2	3	4	5
3	8	10	15	22	2
Temp	Cek	Geser			
2	Temp<22	Data ke-4 ke posisi 5			
2	Temp<15	Data ke-3 ke posisi 4			
2	Temp<10	Data ke-2 ke posisi 3			
2	Temp<8	Data ke-1 ke posisi 2			
2	Temp<3	Data ke-0 ke posisi 1			
Temp menempati posisi ke-0					
0	1	2	3	4	5
2	3	8	10	15	22

4. Quick sort

Contoh proses pengurutan Quick sort

Data yang akan diurutkan secara ascending :

5 3 7 2

Langkah 1 :

temp	5	3	7	2
i	0	1	2	3
J	3	2	1	0

temp	2	3	7	5
i	1	2	3	4
J	2	1	0	3

temp	2	3	7	5
i	2	3	4	5
J	3	2	1	4

temp	2	3	7	5
i	3	4	5	6
J	4	3	2	5

Langkah 2 :

temp	2	3	7	5
i	0	1	2	3
J	3	2	1	0

temp	2	3	7	5
i	1	2	3	4
J	2	1	0	3

temp	2	3	7	5
i	2	3	4	5
J	3	2	1	4

Langkah 3 :

temp	2	3	7	5
i	0	1	2	3
J	3	2	1	0

temp	2	3	7	5
i	1	2	3	4
J	2	1	0	3

temp	2	3	7	5
i	2	3	4	5
J	3	2	1	4

Data yang telah terurut secara ascending :

2 3 5 7

5. Exchange Sort



Proses 1

Pivot (Pusat) →

84	69	76	86	94	91
84	69	76	86	94	91
84	69	76	86	94	91
86	69	76	84	94	91
94	69	76	84	86	91
94	69	76	84	86	91

Proses 2

Pivot (Pusat) →

94	69	76	84	86	91
94	76	69	84	86	91
94	84	69	76	86	91
94	86	69	76	84	91
94	91	69	76	84	86

Proses 3

Pivot (Pusat) →

94	91	69	76	84	86
94	91	76	69	84	86
94	91	84	69	76	86
94	91	86	69	76	84

Proses 4

Pivot (Pusat) →

94	91	86	69	76	84
94	91	86	76	69	84
94	91	86	84	69	76

Proses 5

Pivot (Pusat) →

94	91	86	84	69	76
94	91	86	84	76	69

Pelaksanaan Praktikum

a. Program

```

1  #include <iostream>
2  #include <stdlib.h>
3  #include <conio.h>
4  using namespace std;
5  int data[100],data2[100];
6  int n;
7
8  void tukar(int a,int b)
9  {
10     int t;
11     t = data[b];
12     data[b] = data[a];
13     data[a] = t;
14 }
15
16 void bubble_sort()
17 {
18     for(int i=1;i<n;i++)
19     {
20         for(int j=n-1;j>=i;j--)
21         {
22             if(data[j]<data[j-1]) tukar(j,j-1);
23         }
24     }
25     cout<<"bubble sort selesai!"<<endl;
26 }
27
28 void exchange_sort()
29 {
30     for (int i=0; i<n-1; i++)
31     {
32         for(int j = (i+1); j<n; j++)
33         {
34             if (data [i] > data[j]) tukar(i,j);
35         }
36     }
37     cout<<"exchange sort selesai!"<<endl;
38 }
39
40 void selection_sort()
41 {
42     int pos,i,j;
43     for(i=0;i<n-1;i++)
44     {
45         pos = i;
46         for(j = i+1;j<n;j++)
47         {
48             if(data[j] < data[pos]) pos = j;
49         }
50         if(pos != i) tukar(pos,i);
51     }
52     cout<<"selection sort selesai!"<<endl;
53 }
54
55 void insertion_sort()
56 {
57     int temp,i,j;
58     for(i=1;i<n;i++)
59     {
60         temp = data[i];
61         j = i -1;
62         while(data[j]>temp && j>=0)
63         {
64             data[j+1] = data[j];
65             j--;
66         }
67         data[j+1] = temp;
68     }
69     cout<<"insertion sort selesai!"<<endl;

```



```

70 }
71
72 void QuickSort(int L, int R)
73 {
74     int i, j;
75     int mid;
76     i = L;
77     j = R;
78     mid = data[(L+R) / 2];
79     do
80     {
81         while (data[i] < mid) i++;
82         while (data[j] > mid) j--;
83         if (i <= j)
84         {
85             tukar(i,j);
86             i++;
87             j--;
88         };
89     }
90     while (i < j);
91     if (L < j) QuickSort(L, j);
92     if (i < R) QuickSort(i, R);
93 }
94
95 void Input()
96 {
97     cout<<"Masukkan jumlah data = "; cin>>n;
98     for(int i=0;i<n;i++)
99     {
100         cout<<"Masukkan data ke-"<<(i+1)<<" = "; cin>>data[i];
101         data2[i] = data[i];
102     }
103 }
104
105 void Tampil()
106 {
107     cout<<"Data : "<<endl;
108     for(int i=0;i<n;i++)
109     {
110         cout<<data[i]<<" ";
111     }
112     cout<<endl;
113 }
114
115 void AcakLagi()
116 {
117     for(int i=0;i<n;i++)
118     {
119         data[i] = data2[i];
120     }
121     cout<<"Data sudah teracak!"<<endl;
122 }
123
124 int main()
125 {
126     int pil;
127     system("cls");
128     do
129     {
130         system("cls");
131         system("title Program Sorting");
132         cout<<" Menu Sorting"<<endl;
133         cout<<"*****"<<endl;
134         cout<<" 1. Input Data"<<endl;
135         cout<<" 2. Bubble Sort"<<endl;
136         cout<<" 3. Exchange Sort"<<endl;
137         cout<<" 4. Selection Sort"<<endl;
138         cout<<" 5. Insertion Sort"<<endl;

```

```

139 cout<<" 6. Quick Sort"<<endl;
140 cout<<" 7. Tampilkan Data"<<endl;
141 cout<<" 8. Acak Data"<<endl;
142 cout<<" 9. Exit"<<endl;
143 cout<<" Pilihan Anda = "; cin>>pil;
144 switch(pil)
145 {
146     case 1:Input(); break;
147     case 2:bubble_sort(); break;
148     case 3:exchange_sort(); break;
149     case 4:selection_sort(); break;
150     case 5:insertion_sort(); break;
151     case 6:QuickSort(0,n-1);
152     cout<<"quick sort selesai!"<<endl;
153     break;
154     case 7:Tampil(); break;
155     case 8:AcakLagi(); break;
156 }
157 getch();
158 }
159 while(pil!=9);
160 }
    
```

Latihan

1. Jelaskan maksud sebuah program dalam pelaksanaan praktikum diatas !
2. Buatlah sebuah program yang menunjukkan aktivitas sorting dengan menggunakan salah satu metode sorting dan jelaskan kenapa menggunakan metode tersebut !



PRAKTIKUM 6& 7 LINKED LIST

Teori Singkat

Linked List adalah suatu cara untuk menyimpan data dengan struktur sehingga programmer dapat secara otomatis menciptakan suatu tempat baru untuk menyimpan data kapan saja diperlukan. Linked list dikenal juga dengan sebutan senarai berantai adalah stuktur data yang terdiri dari urutan record data dimana setiap record memiliki field yang menyimpan alamat/referensi dari record selanjutnya (dalam urutan). Elemen data yang dihubungkan dengan link pada linked list disebut Node. Biasanya dalam suatu linked list, terdapat istilah head dan tail .

1. Head adalah elemen yang berada pada posisi pertama dalam suatu linked list
2. Tail adalah elemen yang berada pada posisi terakhir dalam suatu linked list.

Jenis Linked List (yang akan dipelajari) adalah :

1. Single Linked List
2. Double Linked List
3. Circular Linked List
4. Multiple Linked List

Proses Dasar Linked List

Ada 5 proses dasar dalam Linked List :

1. Proses Inisialisasi

- Proses awal → menyatakan Linked List belum ada.

- Algoritma :

```
First = Null;
Last = Null;
```

- Ilustrasi Proses :

\0	\0
First	Last

2. Membuat simpul baru.

- Instruksi :

```
P = (simpul *) malloc(sizeof(simpul));
```

- Algoritma :

```
void Buat_Simpul(int x)
{
    P = (simpul *) malloc(sizeof(simpul));
    if (P != NULL)
    {
        P->Info = x; }
    else
        cout<<"Simpul gagal dibuat ";
}
```

3. Membuat simpul awal.

• Algoritma :

```
void Awal()
{
    First = P;
    Last = P;
    P -> Link = NULL;
}
```

Syarat :

1. Linked List belum ada.
2. Sudah ada simpul yang akan dijadikan simpul awal.

4. Menambahkan simpul baru ke dalam Linked List (INSERT)

Syarat :

1. Linked List sudah ada.
2. Sudah ada simpul yang akan ditambahkan ke Linked List.

a. Insert Kanan/Akhir

Algoritma :

```
void Ins_Akhir()
{
    Last -> Link = P;
    Last = P;
    P -> Link = NULL;
}
```

b. Insert Kiri/Awal

Algoritma :

```
void Ins_Awal()
{
    P -> Link = First;
    First = P;
}
```

c. Insert Tengah

Algoritma :

```
void Ins_Tengah()
{
    P -> Link = Q -> Link;
    Q -> Link = P;
}
```

5. Menghapus sebuah simpul dari Linked List (DELETE)

Syarat :

1. Linked List sudah ada.

a. Delete Kanan/Akhir

Algoritma :

```
void Del_Akhir()
{
    free>Last);
    Last = Q;
    Last -> Link = NULL;
}
```

b. Delete Kiri/Awal

Algoritma :

```
void Del_Awal()
{
    Q = First;
    First = Q -> Link;
    free(Q);
}
```

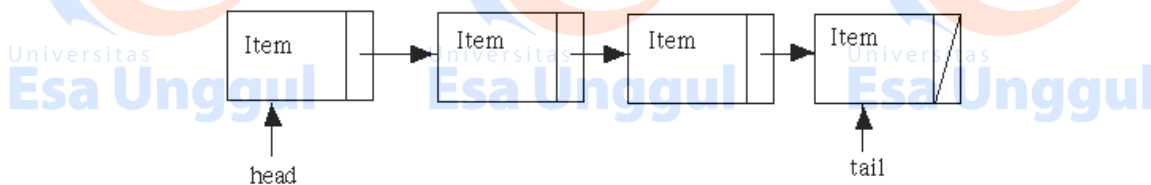
c. Delete Tengah

Algoritma :

```
void Del_Tengah()
{
    R = Q->Link ;
    Q->Link = R->Link;
    free(R);
}
```

Single Linked List

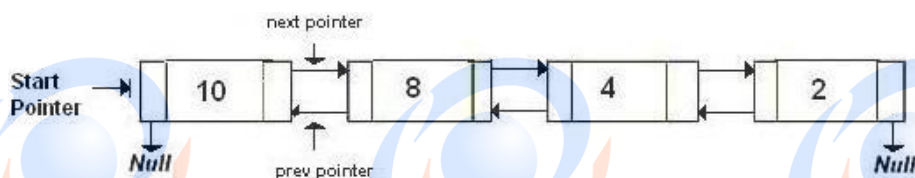
Single Linked List merupakan suatu linked list yang hanya memiliki satu variabel pointer saja. Dimana pointer tersebut menunjuk ke node selanjutnya. Biasanya field pada tail menunjuk ke NULL.



Gambar . Ilustrasi Single Linked List

Double Linked List

Double Linked List merupakan suatu linked list yang memiliki dua variabel pointer yaitu pointer yang menunjuk ke node selanjutnya dan pointer yang menunjuk ke node sebelumnya. Setiap head dan tailnya juga menunjuk ke NULL.



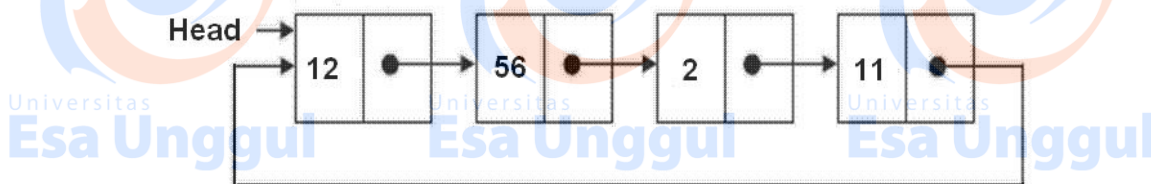
Gambar . Ilustrasi Double Linked List

Circular Linked List

Circular Linked List merupakan suatu linked list dimana tail (node terakhir) menunjuk ke head (node pertama). Jadi tidak ada pointer yang menunjuk NULL.

Ada 2 jenis CircularLinked List, yaitu :

1. Circular Single Linked List



Gambar . Ilustrasi Circular Single Linked List

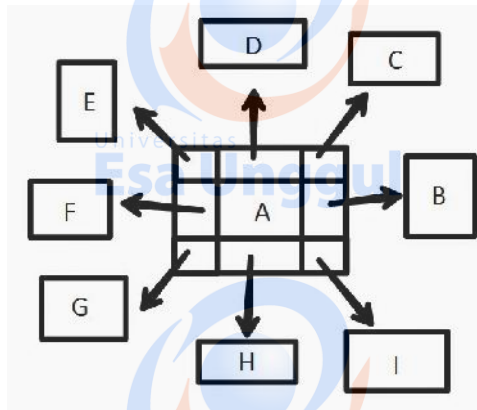
2. Circular Double Linked List



Gambar . Ilustrasi Circular Double Linked List

Multiple Linked List

Multiple Linked List merupakan suatu linked list yang memiliki lebih dar 2 buah variabel pointer.



Gambar . Ilustrasi Multiple Linked List

Pelaksanaan Praktikum

a. Program 1

```

1  #include <iostream>
2  #include <conio.h>
3  #include <stdlib.h>
4  using namespace std;
5  struct TNode
6  {
7      int data;
8      TNode *next;
9  };
10 TNode *head, *tail;
11 void init()
12 {
13     head = NULL;
14     tail = NULL;
15 }
16 int isEmpty()
17 {
18     if(tail == NULL) return 1;
19     else return 0;
20 }
21 void insertDepan(int databaru)
22 {
23     TNode *baru;
24     baru = new TNode;
25     baru->data = databaru;
26     baru->next = NULL;
27
28     if(isEmpty()==1){
29         head=tail=baru;
30         tail->next=NULL;
31     }
32     else {
33         baru->next = head;
34         head = baru;
35     }
36     cout<<"Data masuk\n";
37     system("CLS");
38 }
39
40 void insertBelakang(int databaru)
41 {
42     TNode *baru,*bantu;
43     baru = new TNode;
44     baru->data = databaru;
45     baru->next = NULL;
46
47     if(isEmpty()==1){
48         head=baru;
49         tail=baru;
50         tail->next = NULL;

```

```

51 |     }
52 |     else{
53 |         tail->next = baru;
54 |         tail=baru;
55 |     }
56 |     cout<<"Data masuk\n";
57 |     system("CLS");
58 | }
59 |
60 | void tampil()
61 | {
62 |     TNode *bantu;
63 |     bantu = head;
64 |     if(isEmpty()==0){
65 |         cout<<"Data yang masuk: ";
66 |         while(bantu!=NULL){
67 |             cout<<bantu->data<<" ";
68 |             bantu=bantu->next;
69 |         }
70 |     }
71 |     else cout<<"Data yang masuk: Masih kosong\n";
72 | }
73 |
74 | void hapusDepan(){
75 |     int d;
76 |     if (isEmpty()==0){
77 |         if(head!=tail){
78 |             hapus = head;
79 |             d = hapus->data;
80 |             head = head->next;
81 |             delete hapus;
82 |         }
83 |         else {
84 |             d = tail->data;
85 |             head=tail=NULL;
86 |         }
87 |     }
88 |     cout<<d<<"terhapus";
89 | }
90 | else cout<<"Masih kosong\n";
91 | system("CLS");
92 | }
93 |
94 | void hapusBelakang()
95 | {
96 |     TNode *bantu,*hapus;
97 |     int d;
98 |     if (isEmpty()==0){
99 |         bantu = head;
100 |         if(head!=tail){
101 |             while(bantu->next!=tail){
102 |                 bantu = bantu->next;
103 |             }
104 |             hapus = tail;
105 |             tail=bantu;
106 |             d = hapus->data;
107 |             delete hapus;
108 |             tail->next = NULL;
109 |         }
110 |         else{
111 |             d = tail->data;
112 |             head=tail=NULL;
113 |         }
114 |         cout<<d<<" terhapus\n";
115 |     }
116 |     else cout<<"Masih kosong\n";
117 |     system("CLS");
118 | }
119 |
120 | main()
121 | {
122 |     int pil,dalabaru;
123 |     cout<<"*-----*<<<endl;
124 |     cout<<"* Single Linked List " <<<endl;
125 |     cout<<"*-----*<<<endl;

```



```

126 do{
127     cout<<"\n";
128     cout<<"\n*****";
129     cout<<"\n1. Insert Depan";
130     cout<<"\n2. Insert Belakang";
131     cout<<"\n3. Delete Depan";
132     cout<<"\n4. Delete Belakang";
133     cout<<"\n5. Tampil Data";
134     cout<<"\n\nSilahkan Masukan Pilihan Anda :";cin>>pil;
135     cout<<"\n";
136     switch (pil){
137         case 1:
138             {
139                 cout<<"Masukkan Data = ";
140                 cin>>databaru;
141                 insertDepan(databaru);
142                 break;
143             }
144         case 2:
145             {
146                 cout<<"Masukkan Data = ";
147                 cin>>databaru;
148                 insertBelakang(databaru);
149                 break;
150             }
151         case 3:
152             {
153                 hapusDepan();
154                 break;
155             }
156         case 4:
157             {
158                 hapusBelakang();
159                 break;
160             }
161         case 5:
162             {
163                 tampil();
164                 break;
165             }
166         default :
167             {
168                 cout<<"\n Maaf, Tidak ada dalam pilihan";
169             }
170     }
171 } while(pil>=1 && pil<= 5);
172 }
173

```

b. Program 2

```

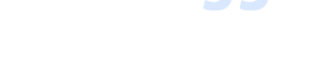
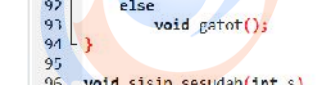
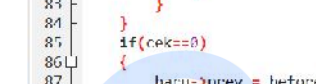
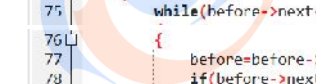
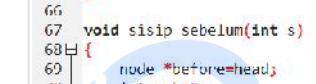
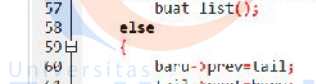
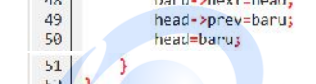
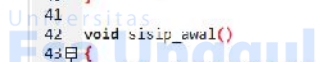
1 #include<stdio.h>
2 #include<stdlib.h>
3 #include<iostream>
4 #include<conio.h>
5 using namespace std;
6 typedef struct simpul node;
7 struct simpul
8 {
9     int data;
10    node *prev;
11    node *next;
12 };
13
14 node *head = NULL, *tail = NULL, *baru;
15
16 void allocate_node (int x)
17 {
18     baru = (node *) malloc (sizeof(node));
19     if (baru==NULL)
20     {
21         printf("Alokasi Gagal\n");
22         exit(1);
23     }
24     else
25     {

```

```

26     baru->data=x;
27     baru->next=NULL;
28     baru->prev=NULL;
29 }
30 }
31
32 //head = tail = baru;
33
34 void buat_list()
35 {
36     printf("Data Masih Kosong, List akan dibuat dengan data tersebut\n");
37     system("PAUSE");
38     head = tail = baru;
39 }
40
41 void sisip_awal()
42 {
43     if(head==NULL && tail==NULL)
44         buat_list();
45     else
46     {
47         baru->next=head;
48         head->prev=baru;
49         head=baru;
50     }
51 }
52
53 void sisip_akhir()
54 {
55     if(head==NULL && tail==NULL)
56         buat_list();
57     else
58     {
59         baru->prev=tail;
60         tail->next=baru;
61         tail=baru;
62     }
63 }
64
65 void sisip_sebelum(int s)
66 {
67     node *before=head;
68     int cek=0;
69     if(before->next==NULL)
70         cek=1;
71     else
72     {
73         while(before->next->data!=s)
74         {
75             before=before->next;
76             if(before->next==NULL)
77                 cek = 1;
78                 break;
79         }
80     }
81     if(cek==0)
82     {
83         baru->prev = before;
84         baru->next = before->next;
85         before->next->prev = baru;
86         before->next = baru;
87     }
88     else
89         void gatot();
90 }
91
92 void sisip_sesudah(int s)
93 {
94     node *after=tail;
95     int cek=0;
96     if(after->prev == NULL)

```



```

101     cek=1;
102     else
103     {
104         while(after->prev->data!=s)
105         {
106             after=after->prev;
107             if(after->prev==NULL)
108             {
109                 cek = 1;
110                 break;
111             }
112         }
113     }
114     if(cek==0)
115     {
116         baru->next = after;
117         baru->prev = after->next;
118         after->prev->next = baru;
119         after->prev = baru;
120     }
121     else
122         void gatot();
123 }
124
125 void free_node(node *p)
126 {
127     free(p);
128     printf("DATA TERHAPUS\n");
129 }
130
131 void hapus_satu()
132 {
133     node *hapus = head;
134     head = NULL;
135     tail = NULL;
136     free_node(hapus);
137 }
138
139 void hapus_awal()
140 {
141     node *hapus = head;
142     head->next->prev = NULL;
143     head = head->next;
144     free_node(hapus);
145 }
146
147 void hapus_akhir()
148 {
149     node *hapus = tail;
150     tail->prev->next = NULL;
151     tail = tail->prev;
152     free_node(hapus);
153 }
154
155 void hapus_Lengah(int s)
156 {
157     node *hapus=head;
158     int cek=0;
159     while(hapus->data!=s)
160     {
161         if(hapus->next==NULL)
162         {
163             cek=1;
164             break;
165         }
166         hapus = hapus->next;
167     }
168     if(cek==0)
169     {
170         hapus->prev->next = hapus->next;
171         hapus->next->prev = hapus->prev;
172         free_node(hapus);
173     }
174     else
175         void gadel();

```



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul



Universitas
Esa Unggul

```

176 | }
177 |
178 | void gatot()
179 | {
180 |     printf("SIMPUL BARU TIDAK DAPAT DISISIPKAN\n");
181 |     //system("PAUSE");
182 | }
183 |
184 | void gadel()
185 | {
186 |     printf("TIDAK ADA DATA YANG DIHAPUS\n");
187 |     //system("PAUSE");
188 | }
189 |
190 | void tampil()
191 | {
192 |     node *p= head;
193 |     printf("\nData Simpul ==> ");
194 |     while(p!=NULL)
195 |     {
196 |         printf("%d ", p->data);
197 |         p=p->nexlt;
198 |     }
199 |     printf("\n\n");
200 | }
201 |
202 | int main()
203 | {
204 |     head=baru;
205 |     int pilih, data, s;
206 |     char lagi='y';
207 |     while(lagi=='y')
208 |     {
209 |         system("CLS");
210 |         void awal();
211 |         printf("Program Double Linked List\n");
212 |         printf("*****");
213 |         tampil();
214 |         printf("Menu Pilihan : \n");
215 |         printf("1. Sisip Awal\n");
216 |         printf("2. Sisip Akhir\n");
217 |         printf("3. Sisip Sebelum Simpul\n");
218 |         printf("4. Sisip Sesudah Simpul\n");
219 |         printf("5. Hapus Awal\n");
220 |         printf("6. Hapus Akhir\n");
221 |         printf("7. Hapus Tengah\n");
222 |         printf("\nPilih No      : ");
223 |         scanf("%d", &pilih);
224 |         switch(pilih)
225 |         {
226 |             case 1 :
227 |                 printf("Masukkan data      : ");
228 |                 scanf("%d", &data);
229 |                 allocate_node(data);
230 |                 sisip_awal();
231 |                 break;
232 |             case 2 :
233 |                 printf("Masukkan data      : ");
234 |                 scanf("%d", &data);
235 |                 allocate_node(data);
236 |                 sisip_akhir();
237 |                 break;
238 |             case 3 :
239 |                 printf("Masukkan data      : ");
240 |                 scanf("%d", &data);
241 |                 allocate_node(data);
242 |                 if(head==NULL && tail==NULL)
243 |                     buat_list();
244 |                 else
245 |                 {
246 |                     printf("Dimasukkan sebelum : ");
247 |                     scanf("%d",&s);
248 |                     if(s==head->data)
249 |                         sisip_awal();
250 |                     else

```

```

251         sisip_scbelum(s);
252     }
253     break;
254 case 4 :
255     printf("Masukkan data      : ");
256     scanf("%d", &data);
257     allocate_node(data);
258     if(head==NULL && tail==NULL)
259         buat_list();
260     else
261     {
262         printf("Dimasukkan sesudah : ");
263         scanf("%d",&s);
264         if(s==tail->data)
265             sisip_akhir();
266         else
267             sisip_sesudah(s);
268     }
269     break;
270 case 5 :
271     if(head == NULL && tail == NULL)
272         gadel();
273     else if(head == tail)
274         hapus_satu();
275     else
276         hapus_awal();
277     break;
278 case 6 :
279     if(head == NULL && tail == NULL)
280         gadel();
281     else if(head == tail)
282         hapus_satu();
283     else
284         hapus_akhir();
285     break;
286 case 7 :
287     printf("Data yang dihapus : ");
288     scanf("%d", &data);
289     if(head == NULL && tail == NULL)
290         gadel();
291     else if(data == head->data && data == tail->data && head == tail)
292         hapus_satu();
293     else if(data == head->data)
294         hapus_awal();
295     else if(data == tail->data)
296         hapus_akhir();
297     else
298         hapus_tengah(data);
299     break;
300 }
301 fflush(stdin);
302 printf("Lagi (y/t) ? ");
303 scanf("%c", &lagi);
304 }
305 }

```

Latihan

1. Jelaskan maksud program dalam pelaksanaan praktikum diatas (pilih salah satu) !
2. Buatlah sebuah program linked list (metode bebas) yang berisikan menu sebagai berikut :
 - a. Insert depan
 - b. Insert tengah
 - c. Insert belakang
 - d. Delete depan
 - e. Delete tengah
 - f. Delete belakang
 - g. Tampilkan data
 - h. Keluar

PRAKTIKUM 8 STRUCT

Teori Singkat

Struktur adalah tipe data yang dapat menyimpan sejumlah data yang memiliki tipe data berbeda. Variabel yang menyusun Struktur dinamakan elemen struktur.

Hal yang perlu diingat :

1. Bentuk struktur data yang dapat menyimpan variabel-variabel dalam 1 nama, namun memiliki tipe data yang berbeda ataupun sama.
2. Variable-variabel tersebut memiliki kaitan satu sama yang lain.

Deklarasi Struktur

```

• struct nama_struktur
{
    elemen1;
    elemen2;
};

struct mahasiswa {
    char nim[15];
    char nama[15];
    float nilai;
};
        
```

Atau

↓

Contoh Deklarasi

```

• typedef struct {
    elemen1;
    elemen2;
} nama_struktur;

typedef struct {
    char nim[15];
    char nama[15];
    float nilai;
} mahasiswa;
        
```

Pengaksesan Elemen Struct

Pengaksesanelemenstructdilakukansecaraindividualdenganmenyebutkannamavariablist rukturdiikutidengantandatitik “.”.

Perhatikan contoh berikut:

```

mahasiswa mhs;
cout<<"Masukkan NIM : ";
cin>>mhs.nim;
cout<<"Masukkan Nama : ";
cin >>mhs.nama;
cout<<"Masukkan Nilai Akhir : "<< mhs.nilai;
        
```

Pelaksanaan Praktikum

Program

```

1  #include<iostream>
2  #include<conio.h>
3  #include<stdlib.h>
4  using namespace std;
5
6  struct mahasiswa
7  {
8      char nim[15];
9      char nama[15];
10     float nilai;
11 };
12 int main()
13 {
14     system("cls");
15     mahasiswa mhs;
16     cout<<"masukan NIM = ";
17     cin>>mhs.nim;
18     cout<<"masukan Nama = ";
19     cin>>mhs.nama;
20     cout<<"masukan Nilai Akhir = ";
21     cin>>mhs.nilai;
22     cout<<endl<<endl;
23     cout<<"NIM = "<<mhs.nim<<endl;
24     cout<<"Nama = "<<mhs.nama<<endl;
25     cout<<"Nilai Akhir = "<<mhs.nilai<<endl;
26     getch();
27 }

```

Latihan

1. Buatlah sebuah program untuk menghitung gaji harian, gaji perjam = 50000, bila jumlah jam kerja lebih dari 7 jam, maka lebihnya dianggap lembur dan gaji perjam lembur = 15 x gaji perjam. Gunakan Struct untuk membuat program tersebut !

PRAKTIKUM 9 POINTER

Teori Singkat

Pointer adalah suatu variabel penunjuk yang menunjuk pada suatu alamat memori komputer tertentu. Variabel pointer yang tidak menunjuk pada nilai apapun berartimemiliki nilai NULL, dan disebut sebagai **dangling pointer** karena nilainya tidakdiinisialisasi dan tidak dapat diprediksi.

1. Operator Alamat / Dereference Operator(&)

Setiap variabel yang dideklarasikan, disimpan dalam sebuah lokasi memori dan pengguna biasanya tidak mengetahui di alamat mana data tersebut disimpan. Dalam C++, untuk mengetahui alamat tempat penyimpanan data, dapat digunakan tanda ampersand(&) yang dapat diartikan “alamat”.

Contoh :

```
Bill = &Bil2;
```

dibaca: isi variabel bill sama dengan alamat bil2

2. Operator Reference (*)

Penggunaan operator ini, berarti mengakses nilai sebuah alamat yang ditunjukolehvariabel pointer.

Contoh :

```
Bill=*Bil2;
```

dibaca: bill sama dengan nilai yang ditunjuk oleh bil2.

Operasi Pointer

1. Operasi penugasan

Nilai dari suatu variabel pointer dapat disalin ke variabel pointer yang lain.

2. Operasi aritmatika

- Suatu variabel pointer hanya dapat dilakukan operasi aritmatika dengannilai integer saja.
- Operasi yang biasa dilakukan adalah operasi penambahan danpengurangan.
- Operasi penambahan dengan suatu nilai menunjukkan lokasi databerikutnya (index selanjutnya) dalam memori.
- Begitu juga operasi pengurangan.

3. Operasi Logika

Pelaksanaan Praktikum

a. Program 1

```

1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 int main() {
5     int x, y; int *px;
6     x = 89;
7     y = x;
8     px = &x;
9     cout << "Nilai x = " << x << endl;
10    cout << "Nilai y = " << y << endl;
11    cout << "Alamat px = " << px << endl;
12    cout << "Nilai px = " << *px << endl;
13    getch();
14 }

```

b. Program 2

```

1 #include <iostream>
2 #include <conio.h>
3 using namespace std;
4 int main() {
5     int x;
6     int *px; //pointer ke variabel
7     int **ppx; //pointer ke pointer
8
9     x = 175;
10    px = &x;
11    ppx = &px;
12    cout << "Nilai x = " << x << endl;
13    cout << "Nilai px = " << *px << endl;
14    cout << "Nilai ppx = " << **ppx << endl;
15
16    getch();
17 }

```

c. Program 3

```

1 #include <iostream>
2 #include <conio.h>
3 #define MAX 5
4 using namespace std;
5 int main() {
6     int a[MAX];
7     int *pa; pa = a; //atau pa = &a[0]
8
9     for (int i = 0; i < MAX; i++) {
10        cout << "Masukkan Nilai " << i+1 << " : ";
11        cin >> a[i];
12    }
13    cout << endl;
14    for (int i = 0; i < MAX; i++) {
15        cout << "Nilai a[" << i << "] = " << *pa << endl;
16        pa++;
17    }
18    getch();
19 }

```

d. Program 4

```

1 #include <iostream>
2 #include <conio.h>
3 #define MAX 5
4 using namespace std;
5 int main() {
6     char nama[] = "Mia Kastina";
7     char *pNama = nama;
8
9     cout << "Nama = " << nama << endl;
10    cout << "pNama = " << pNama << endl;
11    pNama += 4; cout << "\nSetelah pNama += 4" << endl;
12    cout << "Nama = " << nama << endl;
13    cout << "pNama = " << pNama << endl;
14
15    getch();
16 }
    
```

Latihan

1. Deskripsikan program pelaksanaan praktikum tersebut !
2. Buatlah sebuah program yang menampilkan kalimat terbalik dari suatu kalimat yang telah diinputkan maupun data yang telah ada dalam program. Menggunakan pointer dan beberapa fungsi.



PRAKTIKUM 10& 11

STACK

Teori Singkat

Stack (Tumpukan) adalah list linier yang dikenali elemen puncaknya (TOP) dan Aturan penyisipan dan penghapusan elemennya tertentu. Penyisipan selalu dilakukan “di atas” TOP dan Penghapusan selalu dilakukan pada TOP.

Hal yang perlu diingat :

1. Stack disebut juga tumpukan dimana data hanya dapat dimasukkan dan diambil dari satu sisi.
2. Stack bersifat LIFO (Last In First Out).
Benda yang terakhir masuk ke dalam stack akan menjadi benda pertama yang dikeluarkan dari stack.

Operasi Utama Stack

a. Mendefinisikan stack dengan menggunakan struct

```
typedef struct stack // Mendefinisikan stack dengan
                    // menggunakan struct
{
    int top;
    char data [15][20]; // menampung 15 data dengan
                       // jumlah string max 20 huruf
}
```

b. Mendefinisikan max_stack untuk maksimum isi stack

```
#define max_stack 15
```

c. Membuat variable array sebagai implementasi stack

```
stack tumpuk;
```

Operasi Operasi Stack

1. Menambah (push)

```
void push(char d[20])
{
    tumpuk.top++;
    strcpy(tumpuk.data[tumpuk.top],d);
    printf("data berhasil dimasukkan");
}
```

2. Mengambil (pop)

```
void pop()
{
    printf ("data %sterambil",tumpuk.data[tumpuk.top]);
    tumpuk.top--;
}
```

3. Megecek apakah stack penuh (isFull)

```
int isFull()
{
    if (tumpuk.top==max_stack-1)
        return 1;
    else
        return 0;
}
```

4. Mengecek apakah stack kosong (isEmpty)

```
int isEmpty()
{
    if (tumpuk.top==-1)
        return 1;
    else
        return 0;
}
```

5. Membersihkan stack (clear)

```
void clear()
{
    tumpuk.top=-1;
    cout<<"semua data terhapus.";
}
```

6. Mencetak isi stack (print)

```
void print()
{
    for (int i=tumpuk.top;i>=0;i--)
    {
        Cout<<tumpuk.data[i];
    }
}
```

Pelaksanaan Praktikum

Program

```

1 #include<iostream>
2 #include<conio.h>
3 #include<stdlib.h>
4 #define max 10
5 using namespace std;
6
7 struct Tumpukan
8 {
9     int atas;
10    int data[max];
11 }
12
13 T;
14
15 void awal()
16 {
17     T.atas=-1;
18 }
19
20 int kosong()
21 {
22     if(T.atas==max-1)
23         return 1;
24     else
25         return 0;
26 }
27
28 int penuh()
29 {
30     if(T.atas==max-1)
31         return 1;
32     else
33         return 0;
34 }
35
36 void input(int data)
37 {
38     if(kosong()==1)
39     {
40         T.atas++;
41         T.data[T.atas]=data;
42         cout<<"Data "<<T.data[T.atas]<<" masuk ke stack";
43     }
44     else if(penuh()==0)
45     {
46         T.atas++;
47         T.data[T.atas]=data;
48         cout<<"Data "<<T.data[T.atas]<<" masuk ke stack";
49     }
50     else
51         cout<<"Tumpukan penuh";
52 }
53
54 void hapus()
55 {
56     if(kosong()==0)
57     {
58         cout<<"Data teratas sudah terambil";
59         T.atas--;
60     }
61     else
62         cout<<"Data kosong";
63 }
64
65 void tampil()
66 {
67     if(kosong()==0)
68     {
69         for(int i=T.atas;i>=0;i--)

```

```

73 |         {
74 |             cout<<"\nTumpukan ke "<<i<< "="<<T.data[i];
75 |         }
76 |     }
77 |
78 |     else
79 |     {
80 |         cout<<"Tumpukan kosong";
81 |     }
82 | void bersih()
83 | {
84 |     T.atas=-1;
85 |
86 |     cout<<"Tumpukan kosong!";
87 | }
88 |
89 | int main()
90 | {
91 |     int pil,data;
92 |     awal();
93 |     do
94 |     {
95 |         system("cls");
96 |         cout<<"1. Input\n2. Hapus\n3. Tampil\n4. Bersihkan\n5. Keluar\nMasukkan pilihan :";
97 |         cin>>pil;
98 |         switch(pil)
99 |         {
100 |             case 1:cout<<"Masukkan data = ";
101 |                 cin>>data;
102 |                 input(data);
103 |                 break;
104 |
105 |             case 2:hapus();
106 |                 break;
107 |
108 |             case 3:tampil();
109 |                 break;
110 |
111 |             case 4:bersih();
112 |                 break;
113 |
114 |             case 5: cout<<"Terimakasih, tekan enter untuk keluar";
115 |         }
116 |         getch();
117 |     }
118 |     while(pil!=5);
119 | }
120 |

```

Latihan

1. Deskripsikan program pelaksanaan praktikum tersebut !
2. Buatlah sebuah program yang menunjukkan program tersebut sebuah program stack. Gunakan tampilan menarik dan penjelasan apa saja yang digunakan dalam program tersebut !

PRAKTIKUM 12& 13

QUEUE

Teori Singkat

Queue (Antrian) adalah suatu bentuk khusus dari List Linier dengan operasi penyisipan (*insertion*) hanya diperbolehkan pada salah satu sisi, yang disebut sisi belakang (REAR), dan operasi penghapusan (*deletion*) hanya diperbolehkan pada sisi yang lainnya.

Hal yang perlu diingat :

1. Queue disebut juga antrian dimana data masuk di satu sisi dan keluar di sisi yang lain.
2. Queue bersifat FIFO (First In First Out).

Operasi Utama Queue

1. Mendefinisikan queue dengan menggunakan struct dimana kita perlu menggunakan variable head dan tail sebagai penanda pada stack.

```
typedef struct queue //Mendefinisikan queue
                    dengan menggunakan struct
{
    int head;
    int tail;
    char data [15][20] // menampung 15 data dengan
                      jumlah
                      stringmax 20 huruf
};
```

2. Mendefinisikan max untuk maksimum isi queue
#define max 15
3. Membuat variable array sebagai implementasi queue
queue antri;

Operasi Operasi Queue

1. Enqueue (menginputkan data pada queue)

```
void enqueue(char d[20])
{
    antri.head=0;
    antri.tail++;
    strcpy(antri.data[antri.tail],d);
    printf("data berhasil dimasukkan");
}
```

2. Dequeue (mengambil data dari queue)

```
void dequeue()
{
    printf ("data %s
    terambil", antri.data[antri.head]);
    for (int i=antri.head;i<=antri.tail;i++)
        strcpy (antri.data[i],antri.data[i+1]);
    antri.tail--;
}
```

3. isEmpty (mengecek apakah antrian kosong)

```
int isEmpty()
{
    if (antri.tail==-1)
    {
        antri.head=-1;
        return 1;
    }
    else
        return 0;
}
```

4. isFull (mengecek apakah antrian penuh)

```
int isFull()
{
    if (antri.tail==max-1)
        return 1;
    else
        return 0;
}
```

5. Clear (membersihkan seluruh isi antrian)

```
void clear()
{
    antri.head=antri.tail=-1;
    printf("semua data
    terhapus.");
}
```

6. Print(mencetak seluruh isi antrian)

```
void print()
{
    for (int
    i=0;i<=antri.tail;i++)
        printf ("%s\n",antri.data[i]);
}
```


Pelaksanaan Praktikum

Program

```

1  #include <iostream>
2  #include <conio.h>
3  #include <stdlib.h>
4  using namespace std;
5  char pil;
6  int jml;
7
8  struct node
9  {
10     char kar;
11     node *next;
12 };
13
14 node *tail, *now, *head;
15
16 void buatnodebaru()
17 {
18     tail=NULL;
19     head=NULL;
20 }
21
22 void push(char ch)
23 {
24     now = new nodes;
25     if(head == NULL)
26     {
27         now->kar=ch;
28         now->next=NULL;
29         tail = now;
30         head = now;
31     }
32     else
33     {
34         now->kar=ch;
35         now->next=NULL;
36         tail->next=now;
37         tail=now;
38     }
39 }
40
41 void enqueue()
42 {
43     int i;
44     char temp;
45     cout<<"Masukkan @ karakter : ";
46     cin>>temp;
47     push(temp);
48 }
49
50 void dequeue()
51 {
52     if(head==NULL)
53     {
54         cout<<"Antrean Kosong..";
55     }
56     else
57     {
58         jml = jml - 1;
59         char delete_element = head->kar;
60         node *temp;
61         temp = head;
62         head = head->next;
63         delete temp;
64     }
65 }
66
67 void tampil()
68 {
69     if (head == NULL)

```

```

70 | {
71 |     cout<<"\n\tMaaf data kosong";
72 | }
73 | else
74 | {
75 |     now = head;
76 |     cout<<"\n\tData ==> ";
77 |     while(now !=NULL)
78 |     {
79 |         cout<<now->kar;
80 |         now = now->next;
81 |         cout<<" ";
82 |     }
83 | }
84 | }
85 |
86 | void input()
87 | {
88 |     cout<<"\n1. Enqueue \n2. Dequeue \n3. Tampil \n4. Exit";
89 |     do
90 |     {
91 |         cout<<"\nMasukkan Pilihan Anda : ";
92 |         cin>>pil;
93 |
94 |         switch (pil)
95 |         {
96 |             case '1':
97 |                 if (jml<10)
98 |                 {
99 |                     jml = jml+1;
100 |                     enqueue();
101 |                 }
102 |             else
103 |             {
104 |                 cout<<"Antrean Penuh";
105 |                 getch();
106 |             }
107 |             break;
108 |             case '2':
109 |                 dequeue();
110 |             break;
111 |             case '3':
112 |                 tampil();
113 |             break;
114 |             case '4':
115 |                 exit(1);
116 |             break;
117 |         }
118 |     }
119 |     while(1);
120 | }
121 |
122 | int main()
123 | {
124 |     buatnodebaru();
125 |     jml=0;
126 |     system("cls");
127 |     input();
128 |     getch();
129 | }

```

Latihan

1. Deskripsikan program pelaksanaan praktikum tersebut !
2. Buatlah sebuah program yang menunjukkan program tersebut sebuah program queue. Gunakan tampilan menarik dan penjelasan apa saja yang digunakan dalam program tersebut !

PRAKTIKUM 14 BINARY TREE

Teori Singkat

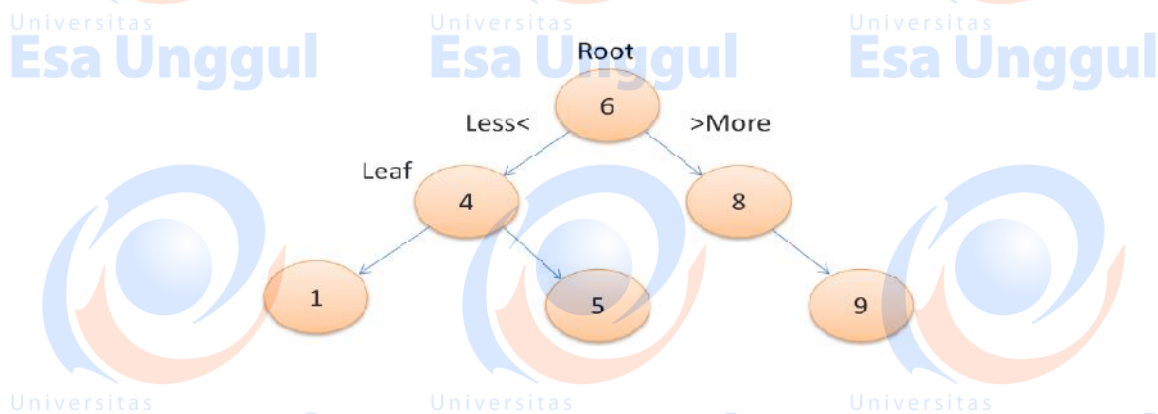
Binary Tree (Pohon Biner) yaitu pohon yang setiap simpul/node-nya paling banyak mempunyai dua buah subpohon.

Contoh implementasi :

untuk membuat pohon silsilah keluarga, ungkapan aritmatika yang setiap operatornya dipasang sebagai simpul pencabangan dan operand-operandnya sebagai subpohon, dll.

Binary tree dapat diimplementasikan dalam C++ dengan menggunakan double linkedlist.

Ilustrasi Binary Tree



Pembentukan Tree

Langkah-langkah Pembentukan Binary Tree

1. Siapkan node baru
 - a. alokasikan memory-nya
 - b. masukkan info-nya
 - c. set pointer kiri & kanan = NULL
2. Sisipkan pada posisi yang tepat
 - **penelusuran** → utk menentukan posisi yang tepat; info yang nilainya lebih besar dari parent akan ditelusuri di sebelah kanan, yang lebih kecil dari parent akan ditelusuri di sebelah kiri
 - **penempatan** → info yang nilainya lebih dari parent akan ditempatkan di sebelahkanan, yang lebih kecil di sebelah kiri

Operasi Umum Tree

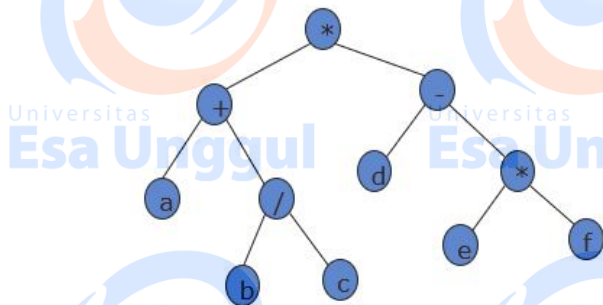
Salah satu operasi yang paling umum dilakukan terhadap sebuah tree adalah kunjungan (traversing)

Ada 3 urutan dasar yang dapat digunakan untuk mengunjungi pohon, yaitu :

1. **PreOrder** : Cetak isi node yang dikunjungi, kunjungi Left Child, kunjungi Right Child.
2. **InOrder** : Kunjungi Left Child, cetak isi node yang dikunjungi, kunjungi Right Child.

3. **PostOrder** : Kunjungi Left Child, kunjungi Right Child cetak isi node yang dikunjungi.

Logical Aritmatika



Hasil :

1. PreOrder : $*+a/bc-d*ef$

2. InOrder : $a+b/c*d-e*f$

3. PostOrder : $abc/+def*-*$

Pelaksanaan Praktikum

Program

```

1 #include<iostream>
2 #include<stdio.h>
3 #include<conio.h>
4 using namespace std;
5
6 struct node
7 {
8     char data;
9     node *kiri;
10    node *kanan;
11 };
12
13 node *akar=NULL;
14
15 node *addNode(node **akar, char isi) {
16     if((*akar)==NULL){
17         node *baru;
18         baru= new node;
19         baru->data = isi;
20         baru->kiri = NULL;
21         baru->kanan = NULL;
22         (*akar)=baru;
23     }
24 }
25
26 node *preOrder(node *akar) {
27     if(akar !=NULL) {
28         cout<<" "<<akar->data;
29         preOrder(akar->kiri);
30         preOrder(akar->kanan);
31     }
32 }
33
34 node *inOrder(node *akar) {
35     if(akar !=NULL) {
36         inOrder(akar->kiri);
37         cout<<" "<<akar->data;
38         inOrder(akar->kanan);
39     }
40 }
41
42 node *postOrder(node *akar) {
43     if(akar !=NULL) {

```

