MODUL PRAKTIKUM
ALGORITMA PEMROGRAMAN 2024

TIM PENYUSUN

DOSEN PENGAMPU MK :

Dr. Eng. Desy Purnami Singgih Putri, S.Si., M.Sc.

PROGRAM STUDI TEKNOLOGI INFORMASI

FAKULTAS TEKNIK

UNIVERSITAS UDAYANA

2024

DAFTAR ISI

MODUL 1
DASAR PEMROGRAMAN, FUNGSI, DAN PROSEDUR

A. TUJUAN PRAKTIKUM

1. Mampu menerapkan dasar pemrograman bahasa C/C++ dalam pembuatan program.
2. Mampu menggunakan dan menerapkan metode fungsi dan prosedur pada sebuah program.

B. TUGAS PENDAHULUAN

1. Sebutkan dan jelaskan secara rinci tipe data dasar yang digunakan dalam bahasa C/C++ dan keperluan memori untuk masing-masing tipe data serta berikan contoh pendeklarasian tipe data tersebut!
2. Jelaskan pengertian dan kegunaan dari *flowchart*! Lalu gambarkan dan jelaskan fungsi masing-masing simbol yang terdapat dalam *flowchart*!
3. Jelaskan tentang konsep variabel lokal dan variabel global serta perbedaannya!
4. Jelaskan tentang konsep fungsi dan prosedur, perbedaan keduanya, serta aplikasinya dalam pemrograman dengan C/C++!
5. Jelaskan tentang konsep *passing by value*, *passing by reference* dan perbedaannya, serta buatlah contoh program dengan Bahasa C/C++!
6. Jelaskan yang dimaksud dengan seni penulisan program beserta contohnya!
7. Buatlah *flowchart* dan *pseudocode* (notasi algoritma) dari program pada soal di bawah ini, lalu lakukan *trace* terhadap *flowchart* yang Anda buat sehingga yakin bahwa *flowchart* tersebut sudah benar.

C. PENGANTAR MATERI

1. Baris Komentar

Baris Komentar Baris komentar adalah baris-baris yang menjelaskan maksud dari perubah yang digunakan atau maksud dari program itu sendiri. Hal ini dimaksudkan untuk memudahkan pelacakan atas perubah yang digunakan apabila program yang digunakan cukup besar atau memudahkan orang lain memahami program yang kita buat. Dalam program, baris komentar diletakkan di antara tanda '`/*`' dan '`*/`' dan baris ini tidak dikerjakan oleh komputer, hanya dianggap sebagai baris kosong.

2. Struktur Bahasa C

Bentuk program C mirip dengan kebanyakan program bahasa tingkat tinggi lainnya. Bentuk programnya adalah judul program, daftar *header file*, deklarasi, deskripsi.

a. Judul Program

Judul program sifatnya sebagai dokumentasi saja, tidak signifikan terhadap proses program. Ditulis dalam bentuk baris komentar. Contoh :

```
/* Program Menghitung Rata-Rata */
```

b. Header File

Bahasa C menyediakan sejumlah *file* judul (*header file*) yaitu *file* yang umumnya berisi prototipe fungsi, definisi makro, variabel dan definisi tipe. *File* ini mempunyai ciri yaitu namanya diakhiri dengan *extension* .h. Contoh :

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

c. Deklarasi

Deklarasi adalah bagian untuk mendefinisikan semua nama yang dipakai dalam program. Nama tersebut dapat berupa nama tetapan (konstanta), nama variabel, nama tipe, nama prosedur, nama fungsi.

d. Deskripsi

Bagian inti dari suatu program yang berisi uraian langkah-langkah penyelesaian masalah. Program C pada hakikatnya tersusun atas sejumlah blok fungsi. Sebuah program minimal mengandung sebuah fungsi. Setiap fungsi terdiri dari satu atau beberapa pernyataan, yang secara keseluruhan dimaksudkan untuk melaksanakan tugas khusus. Bagian pernyataan fungsi (disebut tubuh fungsi) diawali dengan tanda '{' dan diakhiri dengan tanda '}'.

3. Variabel

Variabel dalam program digunakan untuk menyimpan suatu nilai tertentu di mana nilai tersebut dapat berubah-ubah. Setiap variabel mempunyai tipe dan hanya data yang bertipe sama dengan tipe variabel yang dapat disimpan di dalam variabel tersebut. Setiap variabel mempunyai nama. Pemisahan antar variabel dilakukan dengan memberikan tanda koma. Dari contoh di atas, variabel jumlah hanya boleh menerimadata yang bertipe integer (bulat), tidak boleh menerima data bertipe lainnya. Variabel `harga_per_unit` dan `total_biaya` hanya bisa diisi dengan bilangan *float* (pecahan).

```
int jumlah;
float harga_per_unit, total_biaya;
```

4. Konstanta

Berbeda dengan variabel yang isinya bisa berubah selama eksekusi program berlangsung, nilai suatu konstanta tidak bisa berubah. Contoh :

```
const int m = 8;
#define PAJAK 0.05;
```

5. Fungsi main()

Fungsi main() harus ada pada program, karena fungsi inilah yang menjadi titik awal dan titik akhir eksekusi program. Tanda '{' di awal fungsi menyatakan awal tubuh fungsi sekaligus awal eksekusi program, sedangkan tanda '}' di akhir fungsi merupakan akhir tubuh fungsi dan sekaligus akhir eksekusi program.

6. Fungsi printf()

Merupakan fungsi yang digunakan untuk menampilkan data ke layar. Dengan menggunakan fungsi ini, tampilan dapat diatur (diformat) dengan mudah. Bentuk umum dari fungsi ini : `printf(”string kontrol”, argumen1, argumen2, ....); String kontrol` dapat berupa keterangan beserta penentu format (seperti `%d`, `%f`). Argumen adalah data yang akan ditampilkan, dapat berupa variabel, konstanta, maupun ungkapan. Contoh :

```
/* Program Satu */
#include <stdio.h>

int main() {
    Printf(”Belajar Pemrograman Komputer”);
}
```

7. Fungsi scanf()

Merupakan fungsi yang digunakan untuk menampilkan data yang dimasukkan dari *keyboard*. Berikut ini merupakan contoh program menghitung luas dan keliling lingkaran dengan besar jari-jari lingkaran dimasukkan oleh *user* yang memanfaatkan fungsi `scanf()`.

```
/* Menghitung Luas dan Keliling Lingkaran */
#include<stdio.h>
#define PHI 3.14

int main() {
    float jari,luas,keliling;
    printf(”Masukan jari-jari lingkaran = ”);
    scanf(”%f”, &jari);

    luas=PHI*jari*jari;
    keliling=2*PHI*jari;
    printf(”Luas lingkaran = %f \n”,luas);
    printf(”Keliling lingkaran = %f \n”,keliling);
}
```

8. Konsep Penggunaan Modular Fungsi

Fungsi adalah satu blok kode yang dapat melakukan tugas tertentu atau satu blok instruksi yang dieksekusi ketika dipanggil dari bagian lain dalam suatu program. Fungsi berisi sejumlah pernyataan yang disimpan dalam sebuah nama. Nama tersebut selanjutnya dapat dipanggil berkali-kali di beberapa tempat dalam

program.

- Fungsi harus dideklarasikan di dalam program pemanggil/program utama, dengan tujuan supaya program pemanggil mengenal nama fungsi tersebut serta cara mengaksesnya.
- Deklarasi fungsi diakhiri *semicolon* (;).
- Fungsi didefinisikan dengan diawali tipe data keluaran fungsi (di depan nama fungsi), *default*-nya adalah integer.
- Pendefinisian fungsi tidak diakhiri *semicolon* (;).
- Aturan pemberian nama fungsi sama dengan aturan penulisan variabel.
- Blok fungsi diawali dengan '{' dan diakhiri dengan '}'.

Contoh Deklarasi Fungsi :

```
// Struktur Deklarasi Fungsi
tipe_data nama_fungsi(parameter a, parameter..., dst);

// Deklarasi Fungsi pada Program
int lebihbesar(int angka1, int angka2);
```

Contoh Pendefinisian Fungsi :

```
// Struktur Pendefinisian Fungsi
tipe_data nama_fungsi(parameter a, parameter b, dst){
          statement fungsi;
          return x;
}
```

Sintaks fungsi sebenarnya hampir sama dengan prosedur, hanya saja fungsi harus dideklarasikan beserta tipe datanya. Fungsi juga mengembalikan nilai sehingga tipe data dalam sebuah fungsi menunjukkan tipe dari data akhir deklarasi sebuah fungsi.

Contoh Program dengan Fungsi :

```
/*Program sederhana menggunakan fungsi yang mengembalikan
nilai*/

#include<stdio.h>
int lebihbesar(int a, int b); //deklarasi fungsi
```

```
int main(){
    int x,y,nilai;
    printf("Input nilai X: ");
    scanf("%d",&x);
    printf("Input nilai Y: ");
    scanf("%d",&y); nilai = lebihbesar(x,y); //memanggil fungsi
    printf("Nilai terbesar: %d\n",nilai);

    return 0;
}

int lebihbesar(int a, int b){
    if(a>b)
      return a;
    else
      return b;
}
```

9. Konsep Penggunaan Modular Prosedur

Perbedaan utama prosedur dengan fungsi adalah tidak adanya pengembalian nilai (*void*), selain itu dalam sintaks prosedur tidak ada deklarasi tipe data, melainkan di deklarasikan dengan *void*. Keuntungan dari pembuatan prosedur yaitu.

a. Memecah program menjadi lebih sederhana. Misalnya diperlukan proses pencarian berkali kali jika hanya terdiri dari satu program utama tanpa prosedur maka kode program pencarian akan beberapa kali ditulis ulang dan hasilnya dapat memperbesar ukuran *file*.

b. Blok program yang digunakan jelas jika akan digunakan pada program lain cukup dengan meng-*copy* yang membutuhkan dan program lain tersebut tinggal memanggil prosedur tersebut.

Contoh Deklarasi Prosedur :

```
// Struktur Deklarasi Prosedur
void nama_prosedur();

// Deklarasi Prosedur pada Program
void hitung();
```

Contoh Pendefinisian Prosedur :

```
// Struktur Pendefinisian Prosedur
void nama_prosedur(){
     statement prosedur;
}
```

Contoh Program dengan Prosedur :

```
/* Program sederhana menggunakan prosedur */

#include<stdio.h>
void input(); // deklarasi prosedur

int main(){
    printf("NIM NAMA NILAI \n");
    input(); //memanggil prosedur

    return 0;
}

void input(){
    float nilai;
    char nama[50], nim[10];

    printf("Masukan NIM : \n");
    scanf("%c",&nim);
    printf("Masukan Nama : \n");
    scanf("%c",&nama);
    printf("Masukan Nilai : \n");
    scanf("%d",&nilai);

    printf("NIM : %c\n",nim);
    printf("Nama : %c\n",nama);
    printf("Nilai : %d\n",nilai);
}
```

D. SOAL PRAKTIKUM

1. Buatlah program kalkulator yang dapat mengalkulasi pertambahan, pengurangan, perkalian, pembagian, dan modulus. *Input* berupa dua bilangan. *Output* berupa hasil kalkulasi.
2. Buatlah program untuk menghitung luas dan keliling bangun datar berikut dengan menggunakan konsep prosedur :
    a. Segitiga sembarang
    b. Belah Ketupat
    c. Jajar genjang
    d. Trapesium

e. Lingkaran

3. Buatlah program untuk menghitung volume dan luas permukaan bangun ruang berikut dengan menggunakan konsep fungsi :

   a. Tabung
   b. Bola
   c. Limas Segiempat
   d. Prisma Segitiga
   e. Kerucut

MODUL II
PENYELEKSIAN KONDISI DAN PERULANGAN

E. TUJUAN PRAKTIKUM

1. Mampu menggunakan dan menerapkan struktur kontrol percabangan pada sebuah program untuk menyelesaikan suatu masalah.
2. Mampu menggunakan dan menerapkan struktur kontrol perulangan pada sebuah program untuk menyelesaikan suatu masalah.

F. TUGAS PENDAHULUAN

1. Jelaskan dan berikan contoh penggunaan *selection if*, *if-else*, dan *switch*, serta buatlah contoh *flowchart* serta sintaks penulisannya dalam C/C++!
2. Jelaskan dan berikan contoh penggunaan *repetition for, while,* dan *do-while* serta buatlah contoh *flowchart* serta sintaks penulisannya dalam C/C++!
3. Jelaskan perbedaan metode perulangan rekursif dan iteratif!
4. Buatlah diagram alir (*flowchart*), *pseudocode* (notasi algoritma), dan *trace* dari program-program pada soal-soal di bawah ini beserta *flowchart* masing-masing fungsi/prosedur pendukung yang digunakan!

G. PENGANTAR MATERI

1. Struktur Kontrol Percabangan If

Logika *if* digunakan untuk menyeleksi sebuah kondisi yang bernilai benar atau salah, apabila kondisi bernilai benar, maka *statement* atau pernyataan setelah *if* akan dijalankan, sedangkan apabila kondisi bernilai salah maka *statement* setelah *if* tidak akan dijalankan. Bentuk umum dari logika *if* adalah sebagai berikut.

```
if (kondisi) {
    true statement;
}
```

Berikut adalah contoh penulisan logika *if* dalam bahasa pemrograman C.

```
#include <stdio.h>
int main () {
    int panjang = 20 ;
    if (panjang > 7){
        printf (" panjang lebih besar dari 7");
    }
    return 0;
}
```

Kode tersebut terdapat *statement* apabila nilai panjang lebih besar dari 7, maka program akan memproses dan menampilkan *output* berupa kalimat panjang lebih besar dari 7.

2. Struktur Kontrol Percabangan If Else

Logika *if...else* ditambahkan sebuah *statement* yang akan dijalankan jika kondisi yang dicek bernilai salah, jadi pada logika *if...else* disiapkan dua *statement* perintah untuk menangani percabangan sebuah kondisi saat bernilai benar dan saat bernilai salah. Bentuk umum dari logika *if...else* adalah sebagai berikut.

```
if kondisi {
   true statement;
}
else {
   false statement;
}
```

Contoh penulisan logika *if…else* dalam bahasa pemrograman C.

```
#include <stdio.h>
int main () {
    int panjang = 4;
    if(panjang > 5){
      printf("Panjang lebih besar dari 5");
    }
    else{
      printf("Panjang tidak lebih besar dari 5");
    }
    return 0 ;
}
```

Kode program tersebut terdapat 2 *statement*, yang pertama apabila nilai lebih besar dari 5 maka program akan menghasilkan *output* panjang lebih besar dari 5. Setelah logika *if* terdapat *statement else* yang berarti apabila nilai lebih kecil dari

5 maka program akan menghasilkan *output* panjang tidak lebih besar dari 5.

3. Struktur Kontrol Percabangan Switch Case

Logika *switch* digunakan ketika seorang *programmer* harus menyeleksi banyak kondisi. Kelemahan dari *switch* ini adalah hanya bisa menyeleksi nilai dari sebuah variabel tidak bisa menyeleksi hasil dari operator kondisi seperti *if*, misalnya kondisi lebih kecil atau lebih besar. Bentuk umum dari logika *switch* adalah sebagai berikut.

```
Switch(kondisi)
      case ke-1 :
true statement;
break;
      case ke-2 :
true statement;
break;
      case ke-n :
           true statement;
           break;
      default     :
           false statement;
           break;
```

Contoh penulisan logika *switch* pada bahasa pemrograman C.

```
#include <stdio.h>
int main (){
    int hari;
    printf("Input angka 1 sampai 3 : ");
    scanf("%d",&hari);
    printf("\nAngka yang di input adalah : ");
    switch (hari)
           case 1:
printf("Satu");
break;
           case 2:
printf ("Dua");
break;
           case 3:
                printf ("Tiga");
                break;
           default:
                printf ("input tidak terdaftar");
                break;
    return 0;
}
```

Contoh penulisan dari logika *switch* tersebut terdapat beberapa *case*, di

mana setiap *case*-nya memiliki kondisi tertentu, apabila di *case* pertama kondisi tidak terpenuhi maka akan berlanjut ke *case* selanjutnya.

4. Struktur Kontrol Perulangan For

Perulangan *for* berguna untuk mengulang kegiatan eksekusi terhadap satu atau sejumlah pernyataan dengan *range* atau syarat tertentu. Perulangan *for* digunakan untuk mengulang *statement* yang sudah pasti berapa kali pengulangannya dengan kata lain berapa kali *statement* akan diulangi sudah diketahui dan didefinisikan kedalam perintah *for*.

Sintaks dasar penulisan perulangan *for*:

```
for (instalasi variabel; kondisi; update_variabel){
     pernyataan ketika variabel memenuhi kondisi;
}
```

Contoh program dengan perulangan *for:*

```
//program sederhana menampilkan deret bilangan
#include <stdio.h>
#include <stdlib.h>

int main()
{
    int bilangan, deret;

    printf("Deret bilangan \n");
    printf("Input:");
    scanf("%d", &bilangan);
    for (deret=1; deret<=bilangan; deret++){
         printf("%d\n", a);
    }
}
```

5. Struktur Kontrol Perulangan Do While

Pengecekan kondisi pada perulangan *do-while* tidak berada di awal melainkan di akhir sintaks yang mengakibatkan pernyataan akan dieksekusi oleh program minimal sekali, jika variabel memenuhi kondisi, maka program akan lompat ke bagian awal dan mengeksekusinya lagi hingga kondisinya bernilai salah.

Sintaks dasar penulisan perulangan *do-while*:

```
do{
    statement diulang;
} while(kondisi);
```

Contoh program dengan perulangan *do-while:*

```
#include <stdio.h>

int main()
{
    int deret=1,bilangan;
    printf("Masukkan Bilangan : ");
    scanf("%d",&bilangan);

    do
    {
       printf("%d\n", deret);
       deret++;
    } while (deret<=bilangan);

    return 0;
}
```

6. Struktur Kontrol Perulangan While

Perulangan *while* melakukan pengecekan kondisi pada bagian awal. Perulangan hanya dilakukan jika kondisi yang didefinisikan di dalamnya terpenuhi (bernilai benar), jika kondisi yang didefinisikan tidak terpenuhi (bernilai salah) maka *statement-statement* yang terdapat dalam blok perulangan pun tidak pernah dieksekusi oleh program.

Sintaks dasar penulisan perulangan *while*:

```
while(kondisi){
      statement yang diulang;
      perubahan nilai;
}
```

Contoh program dengan perulangan *while:*

```
#include <stdio.h>

int main()
{
    int deret=1, bilangan;

    printf("Deret Bilangan \n" );
```

```
    printf("Input : " );
    scanf("%d", &bilangan);

    while (deret<=bilangan){
           printf("%d \n", deret);
           deret++;
    }
}
```

H. SOAL PRAKTIKUM

1. Buatlah program untuk menentukan Zodiak dengan meng-*input*-kan tanggal dan bulan lahir berdasarkan Ketentuan dari jangkauan tanggal zodiaknya adalah seperti berikut:

Zodiak	Jangkauan Tanggal
Aries	21 Maret - 19 April
Taurus	20 April - 20 Mei
Gemini	21 Mei - 20 Juni
Cancer	21 Juni - 22 Juli
Leo	23 Juli - 22 Agustus
Virgo	23 Agustus - 22 September
Libra	23 September - 22 Oktober
Scorpio	23 Oktober - 21 November
Sagitarius	22 November - 21 Desember
Capricorn	22 Desember - 19 Januari
Aquarius	20 Januari - 18 Februari
Pisces	19 Februari - 20 Maret

2. Buatlah program untuk menentukan Nilai Akhir (NA) mata kuliah berupa huruf dengan ketentuan sebagai berikut :

   Nilai Angka = (Absensi (maks. 15) x 5%) + (Nilai 3 Tugas x 20%) + (Nilai *Quiz* x 15%) + (Nilai UTS x 30%) + (Nilai UAS x 30%). Nilai huruf ditentukan dengan ketentuan berikut :

   $0 \leq NA < 45 \rightarrow E$

   $45 \leq NA < 50 \rightarrow D$

   $50 \leq NA < 55 \rightarrow D+$

   $55 \leq NA < 60 \rightarrow C$

$60 \leq$ NA $< 65 \quad \rightarrow$ C+

$65 \leq$ NA $< 75 \quad \rightarrow$ B

$75 \leq$ NA $< 80 \quad \rightarrow$ B+

$80 \leq$ NA $< 100 \quad \rightarrow$ A

Input program berupa banyak kehadiran, nilai tugas sebanyak 3 kali, nilai *quiz*, nilai UTS, dan nilai UAS. *Output* berupa total nilai angka dan nilai huruf yang didapat dengan ketentuan di atas.

3. Buatlah satu program konversi yang dapat melakukan konversi sebagai berikut :
   a. Bilangan desimal ke biner
   b. Bilangan biner ke desimal
4. Buatlah program untuk menampilkan n deret bilangan *Fibonacci* menggunakan metode rekursif dan metode iteratif! *Input* berupa sembarang bilangan n dan *output* berupa n deret bilangan *Fibonacci* dengan kedua metode.
5. Buatlah program untuk menghitung banyaknya langkah minimum yang diperlukan dalam memindahkan n buah cakram pada kasus menara Hanoi dengan menggunakan metode rekursif dan iteratif! *Input* berupa n jumlah cakram dan *output* berupa langkah penyelesaian dan jumlah langkah minimum.
6. Buatlah sebuah program menghitung angsuran yang harus dibayar berdasarkan pokok pinjaman, besar bunga (menurun) dan lama pinjaman. Gunakan rumus berikut untuk menentukan bunga menurun :

```
p = Pokok Pinjaman
i = Besar bunga dalam setahun
t = Lama pinjaman dalam bulan
Angsuran Pokok = p/t
Bunga Bulan ke y = (p - ((y-1) x Angsuran Pokok)) x i/12
```

Input berupa pokok pinjaman, besar bunga (%), dan lama pinjaman. *Output*

harus berisi rincian angsuran setiap bulan ke 1,2,dst, bunga per bulan, angsuran pokok, dan total angsuran.

Contoh :

```
// Input Program
Pokok Pinjaman        : 1000000
Besar Bunga(%)        : 10
Lama Pinjaman(bulan)  : 12

// Output Program
  Bulan      Bunga      Angsuran Pokok    Angsuran Perbulan
    1       Rp. 8333       Rp. 83333          Rp. 91666
    2       Rp. 7638       Rp. 83333          Rp. 90972
   ...        ....           .....              .....
   12         ....           .....              .....
Total Bunga                              Rp. .........
Total Angsuran                           Rp. .........
```

MODUL III
ARRAY DAN POINTER

A. TUJUAN PRAKTIKUM

1. Mampu menggunakan dan menerapkan fungsi dari struktur data *array* pada sebuah program untuk dapat menyelesaikan masalah.
2. Mampu menggunakan dan menerapkan fungsi dari *pointer* pada sebuah program untuk dapat menyelesaikan masalah.

B. TUGAS PENDAHULUAN

1. Jelaskan tentang *array* dan deklarasi *array* dalam C/C++ serta tipe data yang mungkin digunakan dalam *array*. Berikan contoh deklarasi *array* dalam C/C++ lalu hitung keperluan *memory*-nya!
2. Jelaskan apa yang dimaksud dengan *pointer* serta berikan contoh pendeklarasian dan penggunaan *pointer* dalam bahasa C/C++!
3. Jelaskan secara rinci perbedaan *array* dan *pointer*!
4. Buatlah contoh cara membangkitkan bilangan *random* dalam C/C++!
5. Jelaskan apa yang dimaksud dengan *sorting* dan *searching*!
6. Jelaskan mekanisme metode pengurutan dan pencarian berikut serta berikan contoh pengurutan data yang dilakukan dengan metode tersebut.
   a. *Insertion sort*
   b. *Bubble sort*
   c. *Quick sort*
   d. *Sequential search*
   e. *Binary search*
7. Buatlah *flowchart* dan *pseudocode* (notasi algoritma) dari masing-masing metode di atas, lalu buatlah *trace* dari *flowchart* yang Anda buat.
8. Jelaskan kompleksitas waktu untuk masing-masing metode pada soal 6!

C. PENGANTAR MATERI

1. Array

Array merupakan koleksi data di mana setiap elemen memakai nama dan tipe yang sama serta setiap elemen diakses dengan membedakan indeks *array*-nya. Pada C, data *array* akan disimpan dalam memori yang berurutan. Elemen pertama mempunyai indeks bernilai 0.

Bentuk : **tipe_data nama_var[ukuran];**

Keterangan :

Tipe : menyatakan jenis elemen *array* misal `int`, `char`, dan lain-lain.

Ukuran : menyatakan jumlah maksimal elemen *array*.

a. Array Satu Dimensi

Array Satu Dimensi adalah kumpulan elemen-elemen yang identik, yang tersusun dalam satu baris. Elemen tersebut memiliki tipe data yang sama, tetapi isi dari elemen tersebut bisa berbeda.

Contoh pendeklarasian :

```
int nilai [5];
```

Contoh Program *Array* Satu Dimensi :

```
/* Program menginput data dengan array */
#include<stdio.h>
main()
{
     char nama[10] [25];
     char nim[10] [25];
     int  nilai[10];
     int i,n;
     /* ------- menentukan banyaknya data yg akan diinputkan ---
     -*/
     printf("Banyak data :");
     scanf("%i",&n);
     printf("\n");
     /* --- input data sesuai dengan banyaknya data yg
     ditentukan ---
     */
     for(i=1;i<=n;i++)
     {
printf("Data ke-%i \n",i);
printf("Nama : ");
scanf("%s",&nama[i]);
printf("Nim : ");
```

```
scanf("%s",&nim[i]);
printf("Nilai : ");
scanf("%i",&nilai[i]);
printf("\n");
     }
     /* --- menampilkan data sesuai dengan yg diinputkan
     --- */
     printf("---------------------------------------\n");
     for(i=1;i<=n;i++)
     {
          printf("Data ke-%i \n",i);
          printf("Nim : %s\n",nim[i]);
          printf("Nama : %s\n",nama[i]);
          printf("Nilai : %i\n",nilai[i]);
          printf("\n");
     }
}
```

b. Array Dua Dimensi

Array Dua Dimensi merupakan sebuah variabel yang menyimpan sekumpulan data yang memiliki tipe sama dan elemen yang akan diakses melalui 2 indeks atau subskrip yaitu indeks baris dan indeks kolom.

Contoh pendeklarasian *Array* Dua Dimensi :

```
int matrixA[10][10];
```

Contoh program *Array* Dua Dimensi :

```
/* Program membuat matriks */
#include<stdio.h>
main()
{
     int A[100] [100];
     int m,n,i,j;
     /* ------menentukan banyaknya baris & kolom matriks
      */
     printf("Matriks berordo m x n \n");
     printf("      \n\n");
     printf("Masukkan banyaknya baris (m) : ");
     scanf("%d", &m);
     printf("Masukkan banyaknya kolom (n) : ");
     scanf("%d", &n);
     printf("\n");
     /* ------input elemen matriks   */
     for(i=0;i<m;i++)
     {
          for(j=0;j<n;j++)
          {
               printf("Elemen matriks A[%d,%d] : ",i+1,j+1);
```

```
                scanf("%d",&A[i][j]);
           }
      }
      /* ------menampilkan elemen matriks   */
      printf("\n"); printf("Matriks A = \n\n");
      for(i=0;i<m;i++)
      {
           for(j=0;j<n;j++)
           {
                printf("%3d",A[i][j]);
           }
           printf("\n");
      }
}
```

2. 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 *asterisk* atau bintang (*) di depan 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 variabel *pointer* seperti halnya deklarasi variabel lainnya hanya ditambahkan tanda '*' pada depan nama variabel.

```
int *b;
```

Untuk mendapatkan alamat memori *pointer* (*address of*) maka perintah yang digunakan adalah menambahkan tanda '&' di depan variabel.

```
...........
int b=10;

Printf("Alamat Memory b = %d",&b);
...........
```

Untuk mendapatkan isi atau nilai dari variabel *pointer* maka perintah yang digunakan cukup nama variabelnya saja.

```
...........
```

```
int b=10;

Printf(“Nilai variabel b = %d”,b);
...........
```

Untuk mendapatkan isi atau nilai dari alamat yang terdapat pada isi *pointer* (*value pointed by*) maka perintah yang digunakan adalah menambahkan tanda ‘*’ di depan variabel.

```
...........
int b=10;

Printf(“Isi Alamat Memory b = %d”,*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");
}
```

D. SOAL PRAKTIKUM

1. Implementasikan *flowchart* yang Anda kerjakan pada soal pendahuluan no. 6 ke dalam program C/C++. Tambahkan pencatat waktu yang bisa digunakan untuk menghitung waktu proses masing-masing metode pengurutan dan pencarian data tersebut. Lakukan uji coba terhadap program dengan pembangkitan bilangan acak dengan banyak bilangan 1000, 16000, dan 64000 data. Lalu catatlah waktu yang diperlukan untuk mengurutkan masing-masing jumlah data dan mencari data pada program tersebut kemudian bandingkan dengan kompleksitas masing-masing metode.
2. Buatlah program untuk melakukan operasi matematika pada matriks (perkalian, penjumlahan, *transpose*)!

3. Buatlah program statistika untuk menghitung nilai median, modus dan *mean* dari n data yang di-*input*-kan dengan menggunakan *pointer*!
4. Buatlah 1000, 16000, dan 64000 data acak yang disimpan dalam sebuah *array*! Lakukan pengurutan kepada sekumpulan data tadi dengan metode *bubble sort* cara biasa dan menggunakan *pointer*! catat berapa waktu yang dibutuhkan untuk masing-masing proses pengurutan!
5. Buatlah program yang dapat melakukan enkripsi dan deskripsi kata/kalimat dengan pergeseran abjad sebagai kunci enkripsi dan deskripsi. *Input* berupa jumlah pergeseran dan *plaintext* jika melakukan enkripsi, dan *input* berupa jumlah pergeseran dan *ciphertext* jika melakukan deskripsi. *Output* berupa hasil enkripsi atau deskripsi.

   Contoh Enkripsi :

```
// Input Program
Jumlah Pergeseran : 3
Plaintext         : data alprog

// Output Program
Hasil Enkripsi    : GDWD DOSURJ
```

   Contoh Deskripsi :

```
// Input Program
Jumlah Pergeseran : 3
Ciphertext        : GDWD DOSURJ

// Output Program
Hasil Deskripsi   : data alprog
```

   Proses Pergesaran :

D E F G H I J K L M N O P Q R S T U V W X Y Z A B C -> cipher

a b c d e f g h i j k l m n o p q r s t u v w x y z -> *plaintext*

   Kunci enkripsi/deskripsi adalah *ciphertext* mengalami pergeseran sebanyak 3 huruf ke belakang, untuk *plaintext* tinggal mengikut *cipher* sesuai dengan kata yang di-*input*-kan.

MODUL IV
STRUKTUR DAN OPERASI FILE

A. TUJUAN PRAKTIKUM

1. Mampu menggunakan dan menerapkan fungsi dari tipe data bentukan *struct* pada sebuah program untuk menyelesaikan suatu masalah.
2. Mampu menggunakan dan menerapkan fungsi dari operasi *file* pada sebuah program untuk menyelesaikan suatu masalah.

B. TUGAS PENDAHULUAN

1 Jelaskan tentang pengertian serta kegunaan *struct*!

2 Apakah perbedaan antara *struct* yang bersifat *private* dan *public*?

3 Berikan contoh penggunaan *struct* secara *private* maupun *public* dalam C/C++!

4 Jelaskan apa yang dimaksud dengan *union* serta jelaskan perbedaan *union* dengan *struct*!

5 Jelaskan macam-macam akses *modifier* yang ada pada dalam C/C++!

6 Jelaskan tentang penggunaan tipe data *string* beserta dengan fungsi-fungsi yang dapat di pakai untuk mengkonversi tipe data *string* ke tipe data lain.

7 Jelaskan dan berikan contoh penggunaan `FILE` dan *statement* lain yang digunakan dalam operasi *file*, dalam C++.

8 Buatlah *flowchart* dan *pseudocode* (notasi algoritma) dari masing-masing soal dibawah, lalu buat *trace* terhadap *flowchart* yang Anda buat!

C. PENGANTAR MATERI

1. Struct

Struct adalah kumpulan data yang memiliki tipe data yang berbeda. Secara pendeklarasian, *struct* sangat berbeda dengan *array* yang hanya memiliki satu buah tipe data untuk setiap kumpulannya. *Struct* digunakan apabila data yang ingin dikelompokkan memiliki tipe data yang berbeda. Pemanfaatan *struct* memberi

manfaat yaitu data akan tertata lebih rapi dan mudah di-*maintenance*. Pendeklarasian *struct* sebagai berikut.

```
struct data_keluarga
{
   long int nik;
   char nama[100];
   char jenis_kelamin[100];
};
data_keluarga keluarga1, keluarga2;
```

Deklarasi di atas merupakan suatu tipe data yang bernama `data_keluarga` di mana setiap data yang akan dideklarasikan menggunakan tipe data `data_keluarga` akan mempunyai *field* `nik`, `nama`, dan `jenis_kelamin`. Untuk dapat menggunakan tipe data tersebut sebuah variabel harus dideklarasikan menggunakan nama *struct*-nya. Bentuk umum pendeklarasian variabel *struct* adalah sebagai berikut.

```
struct data_keluarga keluarga1, keluarga2;
```

Deklarasi dua variabel di atas bernama `keluarga1` dan `keluarga2`, di mana setiap variabel tersebut mempunyai *field* sesuai dengan `data_keluarga`. Hal yang harus diperhatikan selain deklarasi variabel adalah cara untuk mengisi dan memanggil nilai yang ada di dalam sebuah *struct*. Cara untuk mengisi dan memanggil nilai yang ada pada *struct* adalah sebagai berikut.

```
//Untuk mengisi nilai struct
scanf("%d",&keluarga1.nik);

//Untuk memanggil nilai struct
Printf("%d",keluarga1.nik);
```

2. Union

Union adalah *user defined type* yang mirip seperti *struct* di mana memiliki kumpulan data dengan tipe data yang sesuai keinginan *user*, namun seluruh *field*-nya disimpan pada memori yang sama. Dengan begitu *union* memiliki keunggulan pada penggunaan memori yang lebih sedikit. Namun, keunggulannya juga menjadi kelemahan di mana jika salah satu *field* di ganti maka seluruh *field* akan berubah. Secara pendeklarasian, *union* sangat mirip dengan *struct*, hanya berbeda di tipenya yaitu *union.* Berikut adalah contoh pendeklarasian dan alamat memori pada *union* dan *struct*.

```
// Pendeklarasian struct
struct ab
{
   int a;
   char b;
};

// Alamat memory struct
alamat ab.a = 6295624;
alamat ab.b = 6295628;

// Pendeklarasian union
union bc
{
   int b;
   char c;
};

// Alamat memory union
alamat ab.b = 6295616;
alamat ab.c = 6295616;
```

Terlihat bahwa pada *struct* tiap *field*-nya memiliki alamat yang berbeda, namun pada *union* tiap *field*-nya berada pada satu alamat memori yang sama. Maka dari itu, union sangat cocok digunakan untuk penggunaan yang membutuhkan beragam tipe data yang dikelompokkan dan tidak saling dibutuhkan pada waktu yang sama serta pada program yang mengedepankan penggunaan memori yang efisien. Cara untuk mengisi dan memanggil nilai yang ada pada *union* dapat dilihat sebagai berikut.

```
//Untuk mengisi nilai union
scanf("%d",&keluarga1.nik);
```

```
//Untuk memanggil nilai
Printf("%d",keluarga1.nik);
```

Kode di atas merupakan cara pengisian dan pemanggilan pada *union*. Terlihat bahwa dalam mengisi maupun memanggil tetap sama seperti *struct*.

3. Operasi File

File merupakan suatu operasi di dalam bahasa pemrograman C ataupun C++ yang memungkinkan untuk menyimpan suatu data secara tetap dan dengan jangka waktu yang lama. Penggunaan `FILE` tidak memerlukan suatu *header* dan saat pendeklarasiannya menggunakan suatu *pointer*. Contoh pendeklarasian `FILE` yaitu sebagai berikut.

```
// Pendeklarasian FILE
FILE *pointerFile;
```

Operasi yang biasanya melibatkan `FILE` yaitu membuat *file* baru, membuka *file*, mengambil data dari suatu *file*, menyimpan data ke dalam *file*, dan menutup *file*. Contoh program yang mempergunakan `FILE` untuk membuat *file* baru lalu membuka *file* baru tersebut dan menyimpan suatu data ke dalamnya serta terakhir menutup *file* tersebut yaitu sebagai berikut.

```
// Pendeklarasian FILE FILE *pointerFile;

// Membuat file baru sekaligus membukanya
pointerFile = fopen("namaFile.txt", "w");

// Menyimpan data ke dalam file
fprintf(pointerFile, "%s", "Data yang disimpan");

// Menutup file fclose(pointerFile);
```

D. SOAL PRAKTIKUM

1 Buatlah sebuah program untuk menghitung gaji harian menggunakan *Struct*! Diketahui gaji perjam = 10.625, dan bila jumlah jam kerja lebih dari 8 jam, maka kelebihannya akan dianggap sebagai lembur dan gaji perjam lembur = 8 x gaji perjam. *Input* berupa jumlah jam kerja dan *output* harus menampilkan rincian dari perhitungan gaji harian!

2 Buatlah sebuah program untuk menghitung luas permukaan dan volume kubus di mana ada 2 kubus yaitu kubus dengan *struct* dan kubus dengan union. Masing-masing kubus memiliki *field* sisi, luas permukaan, dan volume. Gunakan 2 buah variabel yaitu kubus union dan kubus *struct* saja pada operasi aritmatika dan menampilkan *output*-nya.

Contoh:

```
//input program
Sisi kubus     : 6

//output program
Kubus dengan Struct
Sisi           : 6
luas permukaan: 216
volume         : 216

Kubus dengan Union
sisi           : 6
luas permukaan: 216
volume         : 216

Ukuran memory pada kubus struct: 12
Ukuran memory pada kubus union : 4
```

3 Buatlah sebuah program untuk meng-*input*-kan kata kemudian tentukan banyak huruf dalam kata tersebut dan tentukan apakah kata tersebut merupakan palindrom (kata yang jika penulisan dibalik tidak mengubah makna awal) atau bukan palindrom. Simpan hasilnya dalam sebuah *file* txt (kamuskata.txt) yang isinya tidak akan hilang jika program di-*close* dan dapat ditambahkan dengan data baru terus menerus.

Contoh :

```
// Input Program
Kata : malam
```

Output pada kamuskata.txt :

```
Malam(5) : m=2 a=2 l=1 : Palindrom
Senin(5) : s=1 e=1 n=2 i=1 : Bukan Palindrom
```

4 Buatlah sebuah sistem informasi data mahasiswa dari bahasa

pemrograman C/C++. Program tersebut berisi NIM dan Nama Mahasiswa. Data pada program tersebut disimpan dalam sebuah *file* (*file* txt) di mana *file* tersebut dapat ditempatkan pada *drive* mana pun. Data dalam program tidak hilang jika program dimatikan, dapat melakukan pencarian data berdasarkan NIM, dan dapat di-*update* terus menerus, selain itu program ini tidak memungkinkan ada pen-*double*-an data mahasiswa berdasarkan NIM mahasiswa (gunakan *searching* untuk menentukan apakah data baru yang ingin ditambah sudah ada atau belum) saat ada *input* data baru. Program Sistem Informasi tersebut diharapkan mampu untuk menciptakan sebuah *file*, membaca *file* dan mencetak sebuah *file* untuk dapat di-*update* terus menerus.

5 Buatlah program dari permasalahan berikut ini.

Masalah:

Anda adalah seorang asisten dari mata kuliah Algoritma dan Pemrograman di suatu kelas. Pada suatu hari diadakan *quiz* di kelas tersebut. Anda mengoreksi semua jawaban mahasiswa dan menyimpannya ke dalam *file* asistenin.txt bertipe *text*. Namun karena hasil koreksi Anda belum ter-*sorting*, Anda menjadi bingung untuk melakukan *ranking* terhadap hasil tersebut sehingga Anda berencana mengubah hasil nilai tersebut menjadi ter-*sorting*. Anda kemudian menyimpannya ke dalam *file* asistenout.txt yang bertipe *text* agar dapat dilihat di waktu lain. Buatlah program yang membaca *file* asistenin.txt kemudian mengurutkan dari nilai terbesar ke terkecil dan mencetak hasilnya ke dalam asistenout.txt.

Input:

Pada *file input* asistenin.txt terdapat data berupa nama dan nilainya, setiap data menghabiskan 2 baris. Contoh *file input* sebagai berikut.

```
Laria
10
Nadani
12
Saiton
98
Shasha
```

```
99
Fikry
67
Kapuk
99
Modo
0
```

Output:

Berupa isi *file* asistenout.txt.

```
Nilai Mata Kuliah Algoritma dan Pemrograman
1. Shasha 99
2. Kapuk 99
3. Saiton 98
4. Fikry 67
5. Nadani 12
6. Laria 10
7. Modo 0
```

Gunakan *struct* untuk mengelompokkan data dan memudahkan operasi seperti contoh berikut :

```
struct mahasiswa {
     char nama[100];
     int nilai;
};
```

Gunakan *sorting* yang sudah di bahas di modul sebelumnya untuk melakukan pengurutan data berdasarkan nilai.

STRUKTUR LAPORAN

HALAMAN JUDUL

KATA PENGANTAR

ABSTRAK

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE PROGRAM

BAB I PENDAHULUAN

1.1 Latar Belakang Masalah

1.2 Rumusan Masalah

1.3 Tujuan Praktikum

1.4 Manfaat Praktikum

1.5 Batasan Masalah

1.6 Sistematika Penulisan

BAB II TINJAUAN PUSTAKA

2.1 Dasar Penggunaan Bahasa Pemrograman

2.1.1 Konsep Dasar Pemrograman

2.1.2 Pengenalan Bahasa Pemrograman C

2.1.3 Tipe Data dalam Bahasa C

2.1.4 Algoritma

2.1.5 Flowchart

2.1.6 Penggunaan Statement Pemilihan

2.1.7 Penggunaan Statement Perulangan

2.2 Fungsi dan Prosedur

2.2.1 Variabel Lokal dan Variabel Global

2.2.2 Konsep Fungsi dan Prosedur

2.2.3 Konsep Passing by value dan Passing by reference