MODUL PRAKTIKUM

ALGORITMA & PEMEROGRAMAN

S1 TEKNIK INFORMATIKA

CONTACT
NAME : Isep Lutpi nur

NPM : 2113191079

NO. WA : 085798132505

Pendahuluan

A. SEJARAH KEMUNCULAN ALGORITMA

Ditinjau dari asal-usul katanya, kata Algoritma sendiri mempunyai sejarah yang aneh. Orang hanya menemukan kata *algorism* yang berarti proses menghitung dengan angka arab. Para ahli bahasa berusaha menemukan asal kata ini namun hasilnya kurang memuaskan. Akhirnya ahli sejarah matematika menemukan asal kata tersebut yan berasal dari nama penulis buku arab yang terkenal yaitu *Abu Ja'far Muhammad Ibnu Musa Al-Khuwarizmi*.

Al-Khuwarizmi dibaca orang barat menjadi *algorism. Al-Khuwarizmi* menulis buku yang berjudul kitab *Al Jabar Wal-Muqabala* yang artinya "Buku Pemugaran Dan Pengurangan" (*The Book Of Restoration And Reduction*). Dari judul buku itu kita juga memperoleh kata "Aljabar" (Algebra). Perubahan kata dari *algorism* menjadi *algorithm* muncul karena kata *algorism* sering dikelirukan dengan *arithmetic* Sehingga akhiran -sm berubah menjadi –thm. Karena perhitungan dengan angka arab sudah menjadi hal yang biasa, maka lambat laun kata algorithm berangsur-angsur dipakai sebagai metode perhitungan (komputasi) secara umum, sehingga kehilangan makna kata asalnya. Dalam bahasa Indonesia kata *algorithm* diserap menjadi *algoritma.*

B. DEFINISI ALGORITMA

Berikut adalah beberapa definisi yang menjelaskan pengertian dari algoritma.

a. Algoritma adalah urutan langkah logis tertentu untuk memecahkan sesuatu masalah. Yang ditekankan adalah urutan langkah logis, yang berarti algoritma harus mengikuti urutan tertentu, tidak boleh melompat-lompat. (*Dari Microsoft Press Computer And Internet Dictionary 1997, 1998*).
b. Algoritma adalah alur pemikiran dalam menyelesaikan suatu pekerjaan yang dituangkan secara tertulis. Yang ditekankan pertama adalah alur pemikiran, sehingga algoritma seseorang dapat juga berbeda dari algoritma yang lain. Sedangkan penekanan kedua adalah tertulis, yang artinya dapat berupa kalimat, gambar, atau model tertentu. (*Dari Algoritma Dan Struktur Data Dengan C, C++, Dan Java Oleh Moh S Jukanihal1*).

Contoh algoritma dalam kehidupan nyata:

a. Jika seseorang ingin memasak atau membuat kue, baik itu melihat resep atapun tidak pasti akan melakukan suatu langkah-langkah tertentu sehingga masakan atau kuenya jadi.
b. Jika seseorang ingin mengirim surat kepada kenalannya di tempat lain, langkah yang harus di lakukan adalah:
    1. Menulis surat
    2. Surat dimasukan kedalam amplop tertutup
    3. Amplop ditempeli prangko secukupnya
    4. Pergi ke kantor POS untuk mengirimkannya

Dalam bidang komputer, algoritma sangat diperlukan dalam menyelesaikan berbagai masalah pemerograman, terutama dalam komputaso numeris. Tanpa algoritma yang dirancang baik maka peroses pemerograman akan menjadi salah, rusak, lambat dan tidak efesien. Pelaksana algoritma adalah komputer.

Manusia dan komputer berkomunikasi dengan cara: manusia memberikan perintah-perintah kepada komputer berupa instruksi-intruksi yang disebut program. Alat yang dipkai untuk membuat program tersebut adalah bahasa pemerograman.

Bahasa pemerograman sangat bermacam-macam: C, C++, Pascal, Java, C#, Basic, Perl, PHP, ASP, JSP, J#, J++ dan masih banyak bahasa yang lainnya. Dari berbagai bahasa pemerograman cara memberikan instruksinya berbeda-beda namun tujuan menghasilkan outputnya sama.

Kriteria Algoritma Menurut Donald E. Knuth

1. Input: algoritma dapat memiliki nol atau lebih inputan dari luar.
2. Output: algoritma harus memiliki minimal satu buah output keluaran.
3. Definiteness(pasti): algoritma memiliki instruksi-instruksi yang jelas dan tidak ambigu.
4. Finiteness(ada batas): algoritma harus memiliki titik berhenti(stopping role).
5. Effectifeness(tepat dan efisien): algoritma sebisa mungkin harus bias di laksanakan dan efektif. Contoh instruksi yang tidak efektif adalah: A=A+0 atau A=A*1

Namun ada beberapa program yang dirancang untuk unternitable: Contoh Sistem Operasi.

Jenis proses algoritma

1. Sequence Process: intruksi dikerjakan secara kuensial, berurutan.
2. Selection Process: instruksi di kerjakan jika memenuhi kriteria tertentu.
3. Iteration Process: instruksi di kerjakan selama memenuhi kondisi tertentu.
4. Concurrent Process: berberapa instruksi dikerjakan secara bersama.

Dalam Algoritma, tidak dipakai simbol-simbol / sintaks dari suatu bahasa pemerograman tertentu, melainkan bersifat umum dan tidak tergantung pada suatu bahasa pemerograman apapunjuga. Notasi-notasi algoritma dapat digunakan untuk seluruh bahasa pemerograman manapun.

C. BEDA ALGORITMA DAN PROGRAM

Program adalah kumpulan pernyataan komputer, sedangkan metode dan tahapan sistematis dalam program adalah Algoritma. Program ditulis dengan bahasa pemerograman. Jadi bias disebut bahwa program adalah suatu implementasi dari bahasa pemerograman. Beberapa pakar memberi formula bahwa:

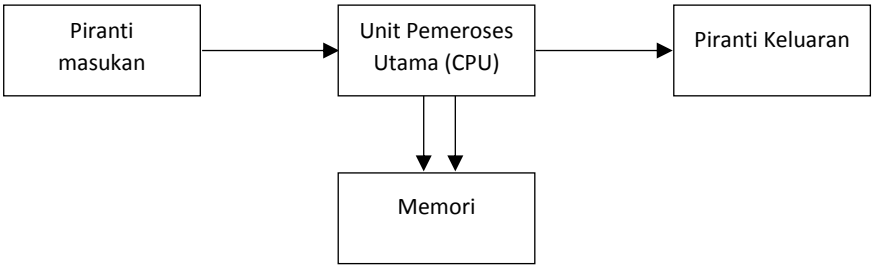
Program = Algoritma + bahasa (Struktur Data)

Bagaimanapun juaga struktur data dan algoritma berhubungan sangat erat pada sebuah program. Algoritma yang baik tanpa prmilihan struktur data yang tepat akan membuat sebuah program menjadi kurang baik, demikian juga sebaliknya. Orang yang menulis program disebut *Programmer.*

Tiap-tiap langkah dalam program disebut pernyataan atau instruksi. Jadi program tersusun atas sederetan instruksi. Bila suatu instruksi dilaksanakan, maka operasi-operasi yang bersesuaian dengan instruksi tersebut akan di kerjakan oleh komputer.

Secara garis besar komputer tersusun atas empat komponen utama, yakni:

1. Piranti masukan berfungsi untuk memasukan data atau program kedalam memori komputer.
2. Pitanti keluaran berfungsi untuk menampilkan hasil dari eksekusi program hasil komputer.
3. Unit pemeroses utama berfungsi mengerjakan operasi-operasi dasar.
4. Memori berfungsi untuk menyimpan program dan data atau informasi.



Gambar 1.1 Proses Eksekusi Program

Mekanisme eksekusi sebuah program adalah sebagai berikut:

1. Program disimpan dalam memori melalui piranti masukan.
2. Ketika sebuah program di eksekusi maka setiap instruksi program akan di kirim dari memori ke unit pemeroses utama. Unit pemeroses utama kemudian akan menjalankan operasi sesuai instruksi-intruksi yang di baca.
3. Apabila sebuah instruksi membutuhkan sebuah data, maka piranti masukan akan membaca data masukan, mengirimkan ke memori kemudian ke unit pemeroses utama untuk di proses.
4. Apabila di eksekusi program menghasilkan data keluaram, maka data keluaran akan disimpan di dalam memori, kemudian dikirim ke piranti keluaran.

D. BAHASA PEMEROGRAMAN

Belajar memperogram tidak sama dengan belajar bahasa pemerograman. Belajar memperogram adalah belajar metodologi pemcahan masalah, kemudian menuangkannya dalam suatu notasi tertentu yang mudah di baca dan dipahami. Sedangkan belajar bahasa pemerograman berarti belajar memakai suatu bahaasa aturan-aturan tata bahasa biasanya, pernyataan-pernyataannya, tata cara pengoperasian *compiler*-nya, dan memanfaatkan pernyataan tersebut untuk membuat program yang di tulis hanya dalam bahasa itu saja.

Sampai saat ini terdapat puluhan bahasa pemerogram, antara lain bahasa rakitan(*assembly*), Fortan, Cobol, Ada, PL/I, Algol, Pascal, C, C++, Basic, Prolog, LISP, PRG, bahasa-bahasa simulasi seperti CSMP, Simscript, GPSS, Dinamo. Berdasarkan terapannya, bahasa pemerograman dapat digolongkan atas dua kelompok besar:

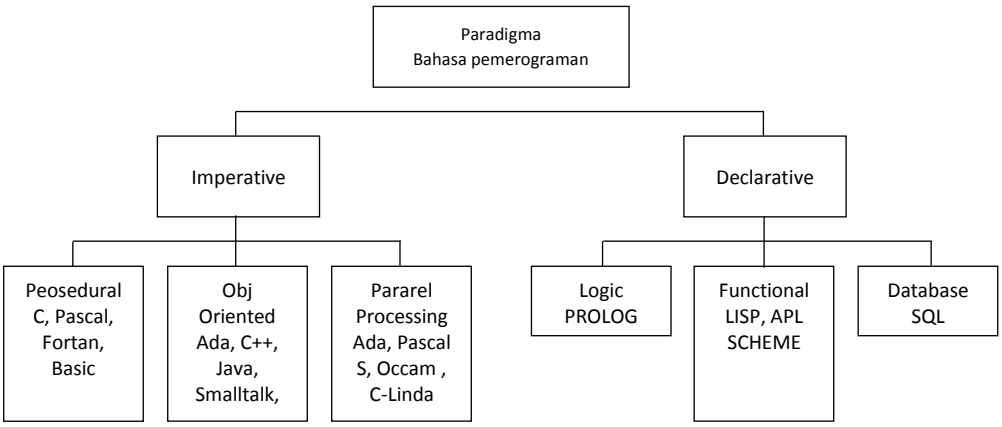
1. Bahasa pemerograman bertujuan khusus, yang masuk kelompok ini adalah Cobol (untuk terapan bisnis dan administrasi). Fortan (terapan komputasi ilmiah), bahasa rakitan terapan (terapan pemerograman mesin). Prolog (terapan kecerdasan buatan, bahasa-bahasa simulasi, dan sebagainya.
2. Bahasa pemerograman yang bertujuan umum, yang dapat diguankan untuk berbagai aplikasi. Yang termasuk kelompok bahasa ini adalah bahasa Pascal, Basic dan C, tentu saja pembagian ini tidak kaku Bahasa-bahasa bertujuan khusus tidak berarti tidak bisa digunakan untuk aplikasi lain. Cobol misalnya, dapat juga digunakan untuk terapan ilmiah, hanya saja kemampuannya

terbatas. Yang jelas, bahasa-bahasa pemerograman yang beda di kembangkan untuk bermacam-macam terapan yang berbedapula.

Berdasarkan pada apakah notasi bahasa pemerograman lebih "dekat" ke mesin atau ke bahasa manusia , maka bahasa pemerograman di kelompokan menjadi dua macam:

1. Bahasa tingkat rendah Bahasa jenis ini dirancang agar setiap intruksinya langsung dikerjakan oleh komputer, tanpa harus melalui penerjemah (*translator*). Contohnya bahasa mesin. CPU mengambil instruksi dari memori, langsung mengerti dan mengerjakan operasinya Bahasa tingkat rendah bersifat primitif, sangat sederhana, orientasinya lebih dekat ke mesin dan sulit di pahami manusia. Sedangkan bahasa rakitan dimasukan ke dalam kelompok ini karena alasan notasi yang dipakai dalam bahasa ini lebih dekat ke mesin, meskipun untuk melaksanakan instruksinya masih perlu penerjemah ke dalam bahasa mesin.
2. Bahasa tingkat tinggi, yang membuat pemerograman lebih mudah di pahami, lebih "manusiawi", dan berorientasi ke bahasa manusia (Bahasa inggris). Hanya saja, program dalam bahasa tingkat tinggi tidak dapat langsung dilaksanakan oleh komputer. Ia perlu diterjemahkan dahulu oleh sebuah translator bahasa (yang disebut kompilator atau *compiler*) ke dalam bahasa mesin sebelum akhirnya di eksekusi oleh CPU. Contoh bahasa tingkat tinggi adalah Pascal, PL/I, Ada, Cobol, Basic, Fortan, C, C++, dan sebagainya.

Bahasa pemerograman bias juga dikelompokan berdasarkan berdasarkan pada tujuan dan fungsinya. Di antara lain adalah.



Gambar 1.2 Pembagian bahasa Pemerograman

Secara sistematis berikut diberikan kiat-kiat untuk belajar memperogram dan belajar bahasa pemerograman serta produk yang dapat di hasilkkan:

a. Belajar Memperogram

1. Belajar memperogram: belajar bahasa pemerograman. Belajar memperogram: belajar tentang strategi pemecahan masalah, metodologi dan statistika pemecahan masalah kemudian menuliskannya dalam notasi yang disepakati bersama.
2. Belajar memperogram: bersifat pemahaman persoalan, analisis dan sintesis.
3. Belajar memperogram, titik berat: *desiner* program.

b. Belajar bahasa Pemerograman

1. Belajar bahasa pemerograman: belajar memaknai sesuatu bahasa pemerograman, aturan sintaks , tata cara untuk memanfaatkan pernytaan yang spesifik untuk setiap bahasa.
2. Belajar Bahasa pemerograman, titik berat:coder.

Sejak dulu hingga sekarang tentu kita mengetahui bahwa di dunia komputer terdapat beraneka ragam Bahasa pemerograman. Karena begitu banyaknya jenis Bahasa pemerograman, Bahasa-bahasa tersebut juga di kelompokan berdasarkan kriteria tertentu antara kain:

1. Tiga level Bahasa yaitu high level (seperti Pascal dan Basic), midle level (seperti Bahasa C), dan down level (seperti Bahasa Assembly).
2. Procedural / functional programing, Object Oriented Programming, Dsb.

Dari kerangaman Bahasa tersebut namun memiliki bagian-bagian yang serupa, yang membedakan hanyalah tata Bahasa yang digunakannya.

Sekilas C dan C++

Berbicara tentang C++ tidak terlepas dari C, sebagai Bahasa pendahulunya. Pencipta C adalah *Brian W Kerninghan* dan Dennis M Ritchie pada sekitaran tahun 1972, dan sekitar satu dekade setelahnya diciptkanlah C++ oleh Bjarne Strousstrup Dari Laboratorium Bell, AT & T, pada tahun 1983. C++ cukup kompitabel dengan Bahasa pendahulunya C. pada mulanya C++ di sebut " a better C" nama C++ sendiri diberikan oleh Rick Mascitti pada tahun 1983, yang berasak dari operator increment pada Bahasa C.
Keistimewaan yang sangat berarti dari C++ ini adalah karena Bahasa ini mendukung pemerograman yang berorientasi obyek (OOP/ Object Oriented Programming).

Dalam hal ini, akan di bahas mengenai bagian-bagian Bahasa pemerograman procedural dengan contoh kasus Bahasa C. Bahasa pemerograman procedural merupakan Bahasa pemerograman yang melibatkan fungsi-fungsi atau prosedur-prosedur sebagai sub program untuk solusi dari suatu permasaalahan. Berbeda halnya dengan Bahasa pemerograman yang berorientasi obyek, yang menggunakan pendekatan obyek dalam menyelesaikan suatu persoalan.
Sebelum masuk lebih dalam mengenai struktur dalam pemerograman procedural. Kita akan membahas mengenai langkah-langkah sistematis dalam pembuatan suatu program, sebagai berikut:

1. Mendefinisikan permasalahan
2. Membuat rumusan untuk pemecahan masalah
3. Implementasi
4. Menguji coba dan membuat dokumentasi

1. Mengidentifikasikan Permasalahan

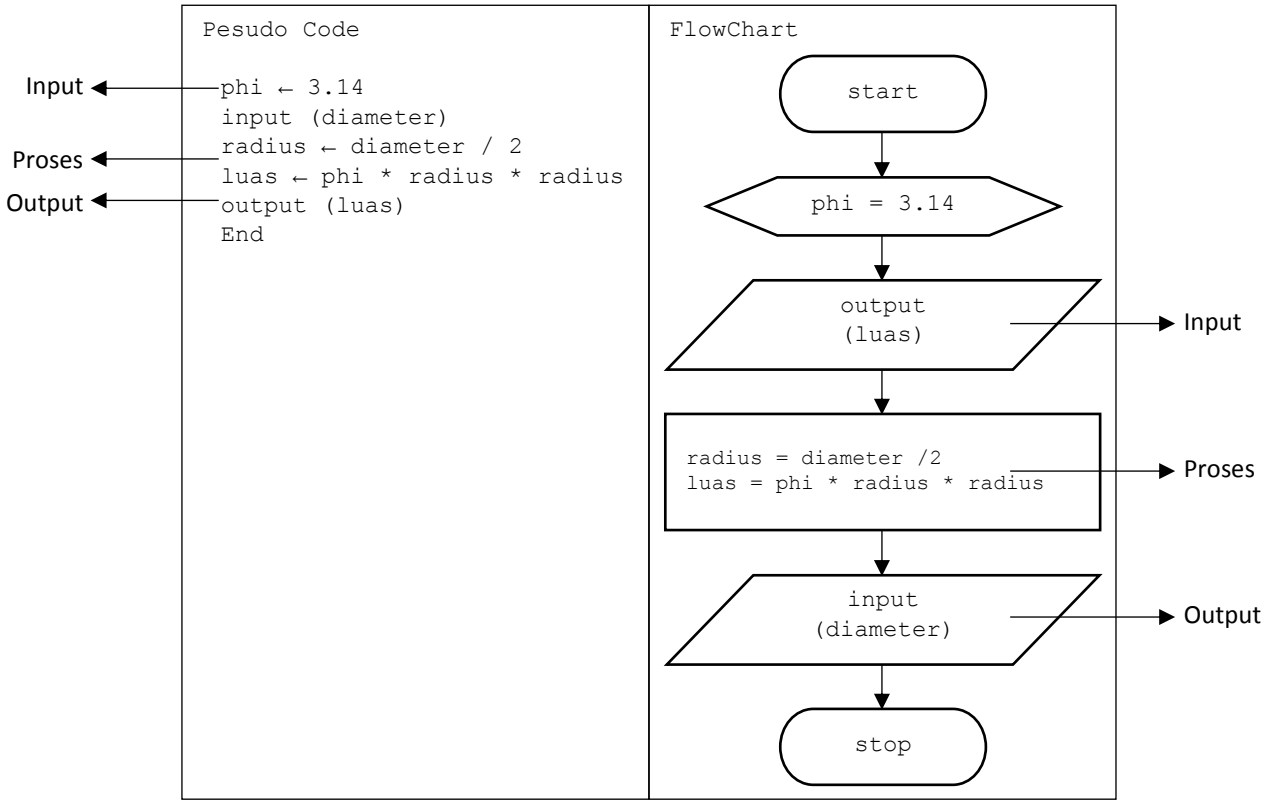
Yang dimaksud mengidentifikasikan permasalahan yaitu kita harus mengerti dengan baik mengenai permasalahan apa yang ingin di selesaikan. Contoh:

1. Permasalahan menghitung luas lingkaran, dengan data yang di ketahui adalah diameter lingkaran.
2. Permasalahan menampilkan bilangan dengan kelipatan tertentu dari 0 hingga range tertentu.

2. Membuat Rumusan Untuk Pemecahan Masalah

Setelah kita mengetahi dengan baik mengenai permasalahan yang ingin diselesaikan, langkah selanjutnya yaitu membuat rumusan algoritma untuk memecahkan masalah. Rumusan tersebut dapat di susun dalam bentuk pseudo code ataupun flowchart. Contoh:

Untuk mengetahui luas lingkaran



3. Implementasi

Apabila langkah 1 dan 2 belum melibatkan Bahasa pemerograman, maka langkah ke tiga ini telah memulai melibatkan Bahasa pemerograman yang ingin di gunakan. Di dalam memngimplementasi algoritma kita akan menentukan Bahasa pemerograman apa yang cocok atau ingin kita gunakan. Misalnya Pascal atau Delphi, Basic dan sebagainya, implementasi tersebut tentunya mengacu pada algoritma yang telah di susun pada langkah sebelumnya, baik itu variable-variable yang digunakan maupun alur program. Jika program diimplementasikan dengan Bahasa pemerograman yang bersifat visual dan event driven (melibatkan desai event-event) seperti Visual Basic atau Delphi, maka perlu pula diperhatikan langkah-langkah berikut:

1. Menambahkan obyek-obyek control pada Form seperti EditBox, ComboBox, Button Dll.
2. Mengatur posisi control , properties control, (seperti caption, warna, jenis, tulisan dan sebgianya), serta ukuran fokus obyek-obyek yang ada pada form.
3. Pemberian nama obyek control yang sesuai. Misalnya input untuk diameter diberi nama txtDiameter.
4. Menentukaan event-event control yang berpengaruh pada fungsionalitas program
5. Mulai koding

4. Menguji Coba Dan Mulai Membuat Dokumentasi

Setelah selesai mengimplementasi , langkah selanjutnya yaitu menguji program tersebut apakah telaj berjalan sesuai dengan tujuannya untuk memberi solusi dari suatu permasalahan . apabila

program belum berjalan dengan baik maka kita perlu mengkaji kembali rumusan / algoritma yang telah di buat pada langkah kedua, serta memperbaiki program yang munkin keliru.
Untuk memudahkan dalam memeriksa kesalahan suatu program atau memahami jalannya suatu program, kita juga perlu membuat dokumentasi dari program yang dibuat. Dokumentasi tersebut yang berisi informasi mulai dari tujuan/fungsi prigram, algoritma program, hingga cara mengguanakannya.

Struktur Bahasa Program Procedural

Secara umum, Bahasa pemerograman yang berisikan prosedur terdiri dari blok/sub program. Yang memiliki dua bagian utama yaitu:

a. Bagian Deklarasi
b. Bagian Statement

1. Bagian Deklarasi

Bagian deklarasi merupakan bagian program untuk mendefinisikan tipe data suatu variable, konstanta, serta fungsi dan prosedur yang akan digunakan untuk memberi nilai awal suatu variable. Dengan kata lain, deklaraasi di gunakan untuk memperkenalkan suatu nama kepada Compiler program. Berikut contoh deklarasi:

```
Bahasa C:
int I, i2;
char s[100];
```

Penjelasan:

Deklarasi diawali dengan tipe data variable baru diikuti dengan nama variable (identifier). Suatu indentifier harus diawali dengan karakter bukan angka, tetapi tidak boleh mengandung karakter khusus seperti *,-+/\=<>.& dan sebagainya identifier bersifat case sensitive, sehingga variables dan akan dianggap variable dua identifier yang berbeda.

2. Bagian Statement

Bagian stetment merupakan bagian program yang berisi perintah yang akan di eksekusi/dijalankan. Dimulai dari deklarasi variable hingga akhir statement diawali dan diakhiri dengan tanda kurung kurawal { dan }. Berikut adalah potongan kode untuk implementasi menhitung luas lingkaran dengan **Bahasa C**.

```
include <stdio.h>

void main()
{     const phi = 3.14;
      float diameter, radius, luas;

      scanf("%f", &diameter);          → Input
      radius = diameter / 2.0;         → Proses
      luas = phi * radius * radius;
      printf("%f", luas);              → Output
}
```

Berikut adalah penjelasan baris demi baris dari potongan kode Bahasa C untuk contoh diatas.

1. #include<stdio.h>
   Baris di awal ini meng *include*kan header library stdio kedalam program. Seperti halnya Pascal Bahasa C cukup memiliki banyak library standar yang dapat di gunakan.
2. void main()
   Baris kedua ini menandakan awal dari blok statement utama. Pada Bahasa C blok program utama merupakan suatu fungsi/sub program yang diberi nama "main".
3. { const phi = 3.14;
   Pada awal baris ke tiga ini, terdapat kurung kurawal sebagai pembuka blok statement. Kemudiann reserved word **const** digunakan untuk mendeklarasikan kostanta phi.
4. float diameter, radius, luas;
   Baris ke empat ini digunakan untuk mendeklarasikan variable diameter, radius dan luas dengan tipe data float (bilangan pecahan)
5. scanf("%f", &diameter);
   Baris ke lima yang berisi perintah yang berfungsi untuk meminta input bertipe float dari user, dan kemudian nilainya disimpan kedalam variable diameter.
6. radius = diameter / 2.0;
7. luas = phi * radius * radius;
   Baris ke enam dan ke tujuh melakukan opersi matematika untuk menghitung luas lingkaran.
8. printf("%f", luas);
   Baris ini di gunakan untuk mencetak isi dari variable luas yang bertipe float.
9. }
   Baris ini menandakan akhir dari blok statement.

3. Aturan Leksial

Yang dimaksud dengan aturan leksial yaitu aturan yang digunakan dalam membentuk suatu deklarasi, definisi, maupun statement sehingga menjadi suatu program yang utuh. Aturan ini meliputi beberapa elemen antara lain:

a. Token
b. Komentar
c. Identifier
d. Keywords (Reserved Words)
e. Opertator

3.a. Token

Token yaitu elemen kecil pada Bahasa pemerograman yang memiliki arti penting bagi compiler. Yang termasuk token antara lain: identifier, Keywords (Reserved Words), Opertator dan sebagainya. Token yang satu dengan yang lain dipisahkan dengan satu atai lebih spasi, tab, baris baru, atau komentar.

3.b. Komentar

Komentar yaitu teks (kumpulan karakter) yang di abaikan oleh compiler. Komentar sangat berguna untuk catatan menenai bagian program tertentu sebagai referensi baik itu bagi programmer itu sendiri maupun orang lain yang membaca kode program tersebut.

Pada Bahasa Pascal, teks yang berada diantara kurung kurawal pembuka { dan kurung kurawal tutup } akan dianggap sebagai komentar. Selain itu dapat pula menggunakan tanda (* sebagai pembuka komentar, dan tanda *) sebagai penutup. Contoh:

Pada Bahasa C, teks yang berada di antar tanda /* dan tanda */ akan dianggap sebagai komentar. Dan untuk teks yang berada setelah tanda // juga akan dianggap komentar satu baris. Berikut adalah contoh penggunaan komentar pada Bahasa C:

```
/* program mencetak hello world
      oleh: saya */
void main(){
      // cetak hello world
      //    oleh: saya
      printf("hello world");
```

3.c. Identifier

Identifier merupakan kumpulan karakter yang dapat di gunakan sebagai penanda untuk nama variable, nama tipe data, fungsi, prosedur dan sebagainya. Aturan untuk penulisan identifier yaitu: satu identifier harus di awali oleh satu karakter non angka sebagai berikut:

_ 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

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

Selanjutnya boleh menggunkan karakter angka (0123456789) maupun karakter non angka tersebut diatas, namun tidak boleh menggunakan karakter khusus/spesialis seperti +-*/?!{}[] dan sebagainya. Berikut contoh identifier yang benar maupun salah.

Identifier	Keterangan
_Nama	Benar
No_Telepon	Benar
Bilangan2	Benar
4data	Salah, karena di awali oleh karakter angka; 4 data
Teks?	Salah, karena mengandung karakter khusus/special; Teks?

3.d. Keywords (Reserved Words)

Keywords atau Reserved Words merupakan kata-kata yang telah ada/didefinisikan oleh bahasa pemerograman yang bersangkutan. Kata-kata tersebut telah memiliki definisi yang sudah tetap dan dan tidak dapat diubah. Karena telah memiliki definisi tertentu, maka kata-kata ini tidak dapat digunakan sebagai identifier pada Bahasa C, yang termasuk Reserved Words antara lain:

break	case	char	const	continue	default	Do
double	else	enum	float	for	goto	if
inline	int	long	return	short	singed	sizeof
static	struct	switch	typedef	union	unsinged	void
while						

1-2

Landasan Teori:

Elemen Dasar C++
Cara Penulisan
Masukan Dan Keluaran
Karakter Dan String Literal
Keywords Dan Identifier
Konstanta

Untuk membuat membuat suatu program ada baiknya kita mengenal terlebih dahulu apa yang dimaksud preprocessor directive. Preprocessor ditandai dengan awalan #. Preprocessor selalu dijalankan terlebih dahulu pada saat proses kompilasi terjadi. Setiap program C++ mempunyai bentuk seperti di bawah, yaitu:

```
#preprocessor directive
main()
{
      // batang tubuh program utama
}
```

Melihat bentuk seperti itu dapat kita ambil kesimpulan bahwa batang tubuh utama berada di dalam fungsi main(). Berarti dalam setiap pembuatan program utama, maka dapat dipastikan seorang program menggunakan minimal sebuah fungsi. Pembahasan lebih lanjut mengenai fungsi akan diterangkan kemudian. Yang sekarang coba di tekankan adalah kita menuliskan program utama kita dalam sebuah fungsi main(). Jangan lupa bahwa C++ bersifat case sensitive sehingga nama hallo dan Hallo berbed artinya.

CARA PENULISAN

- Komentar

  Komentar tidak pernah di compile oleh compiler. Dalam C++ terdapat 2 jenis komentar yaitu:
  Jenis 1: /*komentar anda diletakan didalam ini
  Bias mengampit lebih dari satu baris*/
  Jenis 2://komentar anda diletakan disini (hanya bisa perbaris)

- Semicolon

  Tanda semicolom ";" digunakan untuk mengakhiri sebuah pernyataan. Setiap pernyataan harus di akhiri oleh tanda semicolon. Baris yang diawali dengan tanda #, seperti #include<iostream> tidak diakhiri dengan tanda semicolon, karena bentuk tersebut bukanlah suatu bentuk pernyataan, tetapi merupakan preprocceseor derective.

MASUKAN DAN KELUARAN DASAR

Pada C++ terdapat 2 jenis I/O dasar, yaitu;

1. cout (character out), standard keluaran
2. cin (character in), standart masukan

untuk dapat menggunakan keyword diatas, maka harus di tmbahkan #include<iostream> pada preprocessing directive

Contoh:

```
#include<iostream>
using namespace std;
main()
{
     char nama[100];  //Deklarasi nama variable
     cout << "Masukan nama anda: ";
     cin >> nama;     //Meminta user untuk menginisialisasi variable nama
     cout << "Nama anda adalah " << nama;
     return 0;
}
```

KARAKTER & STRING LITERAL

String literal adalah gabungan dari karakter

Contoh: "Belajar" → Literal String

"B" → Karakter

Panjang string strlen() → nama fungsi untuk menghitung panjang string Fungsi strlen() di deklarasikan dalam file string jadi jika anda ingin menggunakan fungsi strlrn(). Maka preprosesing directive #include<string> harus dimasukan dalam program diatas main().

Contoh:

```
#include<iostream>
#include<string>
using namespace std;
main()
{
  cout << strlen ("Selamat pagi.\n") << endl;
  cout << strlen ("Selamat pagi") << endl;
  cout << strlen ("Selamat") << endl;
  cout << strlen ("S") << endl;
  cout << strlen ("");
  return 0;
}
```

Keluarannya:

14
13
7
1
0

Perhatikan bahwa setiap akhir baris pernyataan diakhri dengan tanda titik – koma (semicolon) ";". Perhatikan bahwa "\n" dihitungkan satu karakter (sama kegunaanya seperti \n). Dalam C++ selain \n terdapat juga beberapa karakter khusus yang biasa di sebut *escape sequence characters* yairu:

Karakter	Keterangan
\0	Karakter ber ASCII nol (null)
\a	Karakter bell
\b	Karakter backspace

\f	Karakter ganti halaman (formfeed)
\n	Karakter garis baru (new line)
\r	Karakter carriage return (ke awal baris)
\t	Karakter tab horizontal
\v	Karakter tab vertical
\\	Karakter \
\"	Karakter "
\”	Karakter “
\?	Karakter ?
\ooo	Karakter yang nilai octalnya adalah ooo (3 digit octal)
\xhh	Karakter yang nilai hexadesimalnya adalah (2 digit hexadecimal)

KEYWORD dan IDENTIFIER

Dalam Bahasa pemerograman, suatu program dibuat dari elemen-elemen sintaks individual yang disebut token, yang memuat nama variable, konstanta, keyword, operator dan tanda baca.

Contoh:

```
#include<iostream>
using namespace std;
main()
{
 int n = 66;
 cout << n << endl; // sebagai variable
 return 0;
}
```

Keluarannya:

66

Program diatas memperlihatkan 15 token, yaitu:
main, (,), {, int, =, 66, ;, cout, <<, n, endl, return, 0 dan }
Token n adalah satu variable
Token 66, 0 adalah satu konstanta
Token int, return, dan endl adalah satu keyword
Token = dan << adalah operator
Token (,), {, ; dan } adalah tanda baca
Baris pertama berisi suatu preprosessing directive yang bukan bagian sebenarnya dari program.

VARIABLE, DEKLARASI dan INISIALISASI

Variable adalah symbol dari suatu besaran yang mempresentasikan suatu lokasi didalam memori komputer. Informasi yang tersimpan dalam lokasi tersebut disebut nilai variable. Untuk memperoleh nilai dari suatu variable digunakan pernyataan penugasan (assignment stetment), yang mempunyai sintaks sebagai berikut:
Variable = ekspresi;
Yang akan diolah terlebih dahulu adalah ekspresi, baru hasilnya dimasukan kedalam variable.
Tanda “=” adalah operator penugasan

Contoh:

```
#include<iostream>
using namespace std;
main()
{
 int n;
 n = 66; // sama juga jika ditulis int n = 66;
 cout << n << endl; // n sebagai variable
 cout << n << endl; // end sebagai karakter
 return 0;
}
```

Keluarannya:
66

n

Deklarasi dari suatu variable adalah sebuah pernyataan yang memberikan informasi tentang varable kepada compiler C++. Sintaksnya adalah tipe variable, dengan tipe datanya yang didukung oleh C++, beberapa contohnya yaitu:

Tipe Data	Ukuran Memori (Byte)	Jangkauan Nilai	Jumlah Digit Presisi
char	1	-128 hingga +127	-
int	2	-32768 hingga +32787	-
long	4	-2147438648 hingga 2147438647	-
float	4	3,4E-38 hingga 3,4E38	6-7
double	8	1.7E-308hingga 1.7E308	15-16
long double	10	1.1E-4932 hingga 1.1E4932	19

NB: Untuk mrngrtahui ukuran memori dari suatu tipe digunakan fungsi sizeof(tipe). Tipe data dapat di ubah (type cast). Misalkan:

float x = 3.345;
int p = int(x);
maka nilai p adalah 3 (terjadi truncating)

Contoh deklarasi dan inisialisasi int

a,b,c;
int p = 55;

Dalam contoh kita mendeklarasikan tiga variable a, b dan c namun belum kita inisialisasi. Sedangkan variable p kita inisialisasi (diberikan nilai). Dalam C++ untuk dapat menggunakan suatu variable, variable tersebut minimal kita deklarasikan terlebih dahulu. Apa yang terjadi jika suatu variable telah di deklarasikan namun belum kita inisialisasi lalu kita mencetak variable tersebut ?

Contoh:

```
#include<iostream>
using namespace std;
main(){
 int n;
 cout << n << endl; // n sebagai variable return
 return 0;
}
```

Keluarnnya:
0
Dari mana angka 0 diperoleh ?
>> Jika variable tidak di inisialisasi, namun nilai keluarannya diminta, maka compiler dengan bijak akan maenampilkan nilai acak yang tergantung dari jenis compilernya.

KONSTANTA

1. Konstanta Oktal digit yang digunakan 0-7
2. Konstanta hexadecimal digit yang digunakan 0-9, A-F
3. Konstanta bernama
    a. Mengunakan keyword const
       Contoh const float PI = 3.14152965
       Berbeda dengan variable, konstanta tidak dapat di ubah jika telah di inisialisasi.
    b. Menggunakan #define
       Contoh #define PI 3.14152965
       Keuntungan menggunakan #define apabila dibAndingkan dengan const adalah kecepatan kompilasi, karena sebelum kompilasi dilaksanakan , compiler pertama mencari simbiol #define (oleh sebab itu mengapa # dikatakan preprocessor directive) dan mengganti semua nilai PI dengan nilai 3.14152965.

Materi Praktikum:

- Algoritma dan Blok Program Pada C++.
- variable dan Konstanta.
- Prepocessor directive, coments, cout dan cin.
- Hierarchical Input Proses Output.

Contoh 1

Ada seorang programmer ditugaskan untuk menghitung arus listrik yang mengalir pada tengangan 220 volt dengan daya listrik bervariasi 450 wat, 900 wat dan 1300 wat.

Algoritma: Menghitung Arus Listrik

kamus: voltase, wat, ampere
integer voltase ← 220
input (wat)
ampere ← wat/voltase
output (ampere)
Buatlah Program Menggunakan C++ Dari Algoritma Diatas

```
#include<iostream>
#include<stdio.h>
#include<conio.h>
using namespace std;
main()
{
	//Deklarasi konstanta
	int voltase = 220;
	int wat, ampere;

	cout << "Masukan daya listrik: ";
	cin >> wat;
	ampere = wat/voltase;
	cout << ampere;
	getch();
	return 0;
}
```

Hasilnya:
Jika daya 900 wat
Maka 900 dibagi dengan 220 sama dengan 4.090 A

Latihan 1

Diketahui Y = 9, G = 7 dan V sama dengan Y dikalikan 1 dan ditambahkan G berapakah V ?
Buatlah program menggunakan C++ dari soal latihan 1.

Latihan 2

Diketahui: namaku romeo, pacarku Juliet, gajiku = 500000, gaji pacarku = 400000
Ditanyakan: gajiku dan gaji pacarku adalah
Buatlah program menggunakan C++ dari soal latihan 2(gunakan #define untuk mendefinisikan konstanta).

Latihan 3
Seorang anak SD hendak menghitung luas segi empat.

Buatlah program menggunakan C++ untuk membantu anak tersebut. Dengan ketentuan :
Input : panjang dan lebar
Proses : luas = panjang X lebar
Output : luas

Latihan 4
Seorang ibu hendak menghitung jumlah uang belanja yang dihabiskan pada hari itu. Daftar belanjanya adalah:
Beras 5kg @kg = Rp. 7000
Daging ¼kg @kg = Rp. 1200
Sayuran seharga Rp. 5000
Kentang ½ @kg = Rp. 4500

Buatlah program menggunakan C++ untuk membntu ibu tersebut, dengan ketentuan:
Input : beras, sayuran dan kentang
Proses : …….
Output : total belanja

Latihan 5
Seorang anak SMP hendak menghitung luas segi empat dan keliling segi empat.

Buatlah program menggunakan C++ untuk membntu anak tersebut, dengan ketentuan:
Input : luas = …
Proses : keliling = …
Output : luas dan keliling ?

Latihan 6
Seorang guru SMU hendak Menghitung NEM dan rata-rata.
Jumlah mata pelajaran = 6 terdiri dari: Bahasa Indonesia, Bahasa Inggris, Matematika, Kimia, Fisika Dan Biologi.

Buatlah program menggunakan C++ untuk membntu guru tersebut, dengan ketentuan:
Input : Total NEM = …….
Proses : Rata – Rata NEM =……
Output : Total NEM dan Rata-Rata NEM ?

Latihan 7
Seorang murid sedang belajar matematika, mengitung koordinat titik, Diketahui koordinat titik A (5,6), Koordinat B (7,9). Ditanyakan koordinat titik tengah dari titik A dan titik B .

Buatlah program menggunakan C++ untuk membntu murid tersebut, dengan ketentuan:

Input : ……
Proses : ……
Output : Koordinat titik tengah A dan B

Tugas 1:
Buatlah program menggunakan C++dari latihan 7 gunakan fungsi bawaan C++ **printf** dan **scanf.**

Printf	:	Menampilkan tulisan di layar dari semua tipe data, masing masing data ditampilkan dengan format data yang telah ditentukan.
#include	:	<stdio.h>
Sintaks	:	int printf(cons char*(format[….,….,….]);
Scanf	:	Penerimaan atau pemasukan ketikan data dilengkapi dengan format tipe data yang sesuai formatnya mencakup semua tipe data, misalnya integer, longinteger, float, double, karakter dan string.
#include	:	<stdio.h>
Sintaks	:	Int scanf(“%format”,pointer_ke_variable_data);

3-4

Landasan Teori:

Operaotr Aritmatika
Operator Increment dan Decrement
Operator Biwise
Operator Relasi
Operator Logika
Operator Kondisi

Operator adala symbol yang bisa dilibatkan dalam program untuk melakukan sesuatu operasi atau manipulasi.

Contoh: a=b+c*d/4
a, b, c, d ← disebut operand
=, +, *, / ← disebut operator

`

OPERATOR ARITMATIKA

Operator	Deskripsi	Contoh
+	Penjumlahan (Add)	m + n
-	Pengurangan (Substrac)	m - n
*	Perkalian (Multiply)	m * n
/	Pembagian (Divide)	m / n
%	Sisa Pembagian Integer (Modulus)	m % n
-	Negasi (Negate)	-m

NB: Operator seperti operator negasi (-) disebut turnary operator, karena membutuhkan hanya satu operand.

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
main()
{
      int m = 82, n = 26;
      cout << m << " + " << n << " = " << m + n << endl;
      cout << m << " - " << n << " = " << m - n << endl;
      cout << m << " * " << n << " = " << m * n << endl;
      cout << m << " / " << n << " = " << m / n << endl;
      cout << m << " % " << n << " = " << m % n << endl;
      cout << " - " << m << " = " << -m << endl;
      getch();
      return 0;
}
```

Hasilnya
82 + 26 = 108
82 - 26 = 56
82 * 26 = 2123
82 / 26 = 3
82 % 26 = 4
- 82 = -82

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
main()
{
	int m = 66, n;
	n = ++m;
	cout << "m = " << m << " , n = " << n << endl;
	n = m++;
	cout << "m = " << m << " , n = " << n << endl;
	cout << "m = " << m++ << endl;
	cout << "m = " << m << endl;
	cout << "m = " << ++m << endl;
	getch();
	return 0;
}
```

Hasilnya:

m = 67 ,n = 67
m = 68 ,n = 67
m = 68
m = 69
m = 70

Penjelasan:

Dalam penugasan yang pertama, m adalah pre-increment, menaikan nilainya menjadi 67, yang selanjutnya dimaukan ke n.

Dalam penugasan kedua, m adalah post-increment sehingga 67 dimasukan dahulu ke n baru kemudian nilai m-nya dinaikan, itu sebabnya mengapa nilai m=68 dan n=67.

Dalam penugasan ketiga, m adalah post-increment sehingga nilai m(=68) ditampilkan dahulu (kelayar) baru kemudian nilai dinaikan menjadi 89.

Dalam penugasan keempat, m adalah pre-increment, sehingga nilai m dinaikan dahulu menjadi 70 baru kemudian ditampilkan kelayar.

Supaya lebih paham, perhatikan pula contoh dibawah.

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
main()
{
	int m = 5, n;
	n = ++m * --m;
	cout << "m = " << m << " , n = " << n << endl;
	cout << ++m << " " << ++m << " " << ++m << endl;
	getch();
	return 0;
}
```

Keluarannya:

M = 5 , n = 25
6 7 8

Penjelasan:
Dalam penugasan untuk n =, pertama kali m dinaikan (m++) menjadi 6, kemudian di turunkan kembali menjadi 5, karena keadaannya –m, sehingga nilai m sekarang menjadi 5 dan nilai m = 5 inilah yang dievaluasi pada saat penugasan perkalian dilakukan.
Pada baris terakhir, ketiga sub-eksptesi di evaluasi dari kanan ke kiri.

OPERATOR BITWISE

Operator	Deskripsi	Contoh
<<	Geser n bit ke kiri (left shift)	m << n
>>	Geser n bit ke kanan (right shift)	m >> n
&	Bitwise AND	m & n
|	Bitwise OR	m | n
^	Bitwise XOR	m ^ n
~	bitwise NOT	~m

NB: Seluruh operator bitwise hanya bisa dikenakan pada operand pertipe data int atau char

Berikut ini diberikan tabel kebenaran untuk operator logika P = A operator B

AND

A	B	P
0	0	0
0	1	0
1	0	0
1	1	1

OR

A	B	P
0	0	0
0	1	1
1	0	1
1	1	1

XOR

A	B	P
0	0	0
0	1	1
1	0	1
1	1	0

Note:
- A adalah pernyataan 1
- B adalah pernyataan 2
- 1 mewakili True
- 0 mewakili False

Contoh:
```
#include<iostream>
#include<conio.h>
using namespace std;
int main()
{
      int m = 82, n = 26;
      cout << m << " << 2" << " = " << ( m << 2 ) << endl;
      cout << m << " >> 2" << " = " << ( m >> 2 ) << endl;
      cout << m << " & " << n << " = " << (m & n) << endl;
      cout << m << " | " << n << " = " << (m | n) << endl;
      cout << m << " ^ " << n << " = " << (m ^ n) << endl;
      cout << "~" << m << " = " << ~m << endl;
      getch();
      return 0;
}
```

Keluarannya:
82<<2 = 328
82>>2 = 20
82&26 = 18
82|26 = 90
82^26 = 72
~82 = 83

Penjelasan:
Nilai keluaran diatas, tergantung dari jenis compiler yang digunakan. Hasil diatas merupakan keluaran dari compiler TurboC++.
Pada TurboC++ besai integer adalah 2 byte atau sama dengan 16 bit, untuk mengetahuinya digunakan perintah cout << sizeof(int) << endl; // untuk mengetaui besar dari int
Maka:
$82_{10} = 0000000001010010_2$ Dan
$26_{10} = 0000000000011010_2$
Sehingga:
$82 << 2 \rightarrow 0000000101001000_2 = 328_{10}$
$82 >> 2 \rightarrow 0000000000010100_2 = 20_{10}$
$82\&26 \rightarrow 0000000001010010_2$
0000000000011010_2
$----------\&$
$0000000000010010_2 = 18_{10}$
Dan begitu juga untuk operasi OR dan XOR.
~ 82 $\rightarrow$ digunakan 2"s complement, yaitu:
$82_{10} = 0000000001010010_2$
1111111110010010_2
$1111111110101110 = 65454_{10}$ nilai ini melebihi jangkauan maksimum integer yang berkisar di -32768 sampai 32767 sehingga nilai yang keluar yaitu 83.
Cara lain penulisan dengan menggunakan operator bitwise:
m = m << n $\Leftrightarrow$ m << = n
m = m >> n $\Leftrightarrow$ m >> = n
m = m & n $\Leftrightarrow$ m & n = m
m | n $\Leftrightarrow$ m |= n m = m
^n $\Leftrightarrow$ m ^= n

OPERATOR RELASI
Operator relasi digunakan untuk membAndingkan dua buah nilai. Operator ini bisa digunakan dalam instruksi percabangan.

Operator	Deskripsi
==	Sama dengan (bukan assignment)
!=	Tidak sama dengan
>	Lebih besar dari
<	Lebih kecil dari
>=	Lebih besa atau sama dengan
<=	Lebih kecil atau sama dengan

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
	int m = 5, n = 7;
	if(m == n){
	cout << m << " sama dengan " << n << endl;
	}else if(m != n){
	cout << m << " tidak sama dengan " << n << endl;
	}else if(m < n){
	cout << m << " lebih kecil dari " << n << endl;
	}else if(m > n){
	cout << m << " lebih kurang dari " << n << endl;
	}
	getch();
	return 0;
}
```

Kelurnnya:
5 tidak sama dengan 7

OPRATOR LOGIKA

Operator logika digunakan untuk menghubungkan dua atau lebih ungkapan menjadi sebuah uangkapan berkondisi.

Operator	Deskripsi	Contoh
&&	Logic AND	m && n
||	Logic OR	m || n
!	Logic NOT	!m

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
	int m = 166;
	cout << " (m>=0 && m<=150) → " << (m>=0 && m<=150) << endl;
	cout << " (m>=0 || m<=150) → " << (m>=0 || m<=150) << endl;
	getch();
	return 0;
}
```

Keluarannya:
(m>=0 && m<=150) → 0
(m>=0 && m<=150) → 1

Penjelasan:
Hasil / keluaran dari operator logika adalah 0 dan 1.
0 jika keluarannya salah dan 1 jika keluarannya benar.

OPERATOR KONDISI

Operator kondisi digunakan untuk memperoleh nilai dari dua kemungkinan
ungkapan1 ? ungkapan2 : ungkapan3
bila ungkapan1 benar maka hasilnya sama dengan ungkapan2 , bila tidak maka nilainya sama dengan ungkapan 3.

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
      int m = 26, n = 82;
      int min = m < n ? m : n;
      cout << "Bilangan terkecil adalah: " << min << endl;
      getch();
      return 0;
}
```

Keluarannya:
Bilangan terkecil adalah: 26
Operator relasi, logika dan kondisi akan banyak digunakan pada pernyataan berkondisi.

Materi Praktikum

- Pendeklarasian variable dan Konstanta.
- Prepocessor directive, coments, cout dan cin.
- Operator, printf, scanf, getch, if

Dibawah ini adalah contoh soal mengguanakan token diatas

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
      // deklarasi variable
      int age;
      char descrip[50];
      // percetakan user interface ke mmonitor
      printf("%s","Maasukan umur anda: " );
      // pemindahan isi variable
      scanf("%i", &age );
      // pengendalian nilai berdasarkan kondisi
      if( age <= 5){
            cout << "Balita" << endl;
      }else{
            cout << "Anak-anak" << endl;
      }
      // menahan layar
      getch();
      // mengembalikan proses
      return 0;
}
```

Latihan 8:

Diketahui sebuah algoritma dibawah ini, buatlah mengguanakan program C++:
(Gunakan dengan operator increment dan decrement).
Kamus: Data: F,G,H: Integer
Input (F,G,H)
F ← F + 10
G ← G – 1
H ← G
H ← H + 1
Output (F,G,H)

Latihan 9:

Diketahui sebuah algoritma dibawah ini, buatlah mengguanakan program C++:
Kamus: Z, R, I : Integer
Algoritma:
Z ← Z + 1
R ← I
Z ← R * 5
Output (R)

Latihan 10:
Diketahui sebuah algoritma dibawah ini, buatlah mengguanakan program C++:
(gunakan struktur kendali IF lihar bentuk dari pernyataan IF pada bab pernyataan)

Kamus:
X : Integer
Algoritma
X ← 5
IF X <> 5
X ← R *5
ELSE
R ←X
Output (R)

Latihan 11:
Butalah program C++ untuk menentukan tingkatan umur (gunakan if):
umur 0 s.d 5 : Balita
umur 6 s.d 12 : Anak-Anak
umur 13 s.d 17 : Remaja
umur > 18 : Dewasa

Tugas 2:
Butalah program C++ untuk menentukan bilangan ganjil dan genap

5-7

Landasan Teori:

Pernyataan (Staement): Pernyataan Ungkapan, Pernyataan Deklarasi, Pernyataan Kosong, Pernyataan Majemuk dan Pernyataan Berlabel

Pernyataan untuk melakukan suatu tindakan, yaitu:

PERNYATAAN UNGKAPAN

Pernyataan ini merupakan bentuk pernyataan yang paling sering digunakan. Pernyataan ini diakhiri dengan semicolon ";".

Contoh:

```
var = 166
var++;
```

PERNYATAAN DEKLARASI

Untuk menggunakan satu variable minimal variable tersebut di deklarasikan terlebih dahulu.

Contoh:

```
int var;
```

Merupakan contoh deklarasi sebuah variable var dengan tipe data integer (int).

PERNYATAAN KOSONG

Pernyataan ini tidak melaksanakan apapun

Contoh:

```
while(ada);
```

PERNYATAAN MAJEMUK

Merupakan sebuah pernyataan yang ada di dalam sebuah blok{}

Contoh:

```
for (var = 0; var < 10; var++){
      nilai1 = 100;
      if(!nilai2)nilai2 = 0;
      nilai3 = nilai1 + nilai2;
}
```

PERNYATAAN BERLABEL

Pernyataan **goto** diperlukan untuk melakukan suatu lompatan ke suatu pernyataan yang berlabel yang ditandai dengan tanda ":"

Contoh:

```
goto bawah;
pernyataan1;
pernyataan2;
bawah:
pernyataan3;
```

Pada contoh diatas, pada saat goto ditemukan maka program akan melompat ke pernyataan berlabel bawah dan melakukan pernyataan 3,

PERNYATAAN KONDISI (CONDITIONAL EXPRESSION)

Pernyataan kondisi dibagi menjadi.

1. **Pernyataan if**

Digunakan untuk mengambil keputusan

Bentuk umum:

if(kondisi) pernyataan1;

else pernyataan2;

Pernyataan1 dilaksanakan jika dan hanya jika kondisi yang diinginkan terpenuhi, jika tidak lakukan pernyataan2. Jika anda tidak menggunakan pernyataan else program tidak akan error, namun jikan anda mempergunakan else tanpa di dahului if maka program akan error. Jika pernyataan1 atau pernyataan2 terdiri dari satu baris maka tanda {} tidak diperlukan namun jika lebih maka diperlukan.

Bentuknya menjadi:

```
if(kondisi){
    Pernyataan1;
    Pernyataan1a;
    Pernyataan1b;
}
else{
    Pernyataan2;
    Pernyataan2a;
    Pernyataan2b;
}
```

Contoh:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
   int m = 166;
   if(m == 0) cout << "Nilainya sama denga nol \n";
   else{
         cout << "Nilainya sama denga nol \n";
         cout << "Nilainya sama denga " << m << endl;
   }
   getch();
   return 0;
}
```

Selain dari if.. else, juga dikenal bentuk if.. else if. Adapun bentuk perbedaannya diilustrasikan dibawah ini.

Contoh 1:

```
   #include<iostream>
   #include<conio.h>
   using namespace std;
   int main(){
```

```
	int m = 166;
	if(m > 1000) cout << m << " lebih besar dari 1000\n";
	if(m > 100) cout << m << " lebih besar dari 100\n";
	if(m > 10) cout << m << " lebih besar dari 10\n";
	getch();
	return 0;
}
```

Keluarannya:
166 lebih besar dari 100
166 lebih besar dari 10

Contoh 2:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
	int m = 166;
	if(m > 1000) cout << m << " lebih besar dari 1000\n";
	else if(m > 100) cout << m << " lebih besar dari 100\n";
	else if(m > 10) cout << m << " lebih besar dari 10\n";
	getch();
	return 0;
}
```

Keluarannya:
166 lebih besar dari 100
Mengapa ? karena contoh 2 sama saja ditulis seperti dibawah ini:

```
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
	int m = 166;
	if(m > 1000) cout << m << " lebih besar dari 1000\n";
	else{
		if(m > 100) cout << m << " lebih besar dari 100\n";
		else if(m > 10) cout << m << " lebih besar dari 10\n";
	}
	getch();
	return 0;
}
```

Contoh diatas disebut juga *nested conditional*

2. Pernyataan *switch*

Pernyataan if.. else if jamak dapat digunakan pernyataan *switch.* Bentuk umumnya adalah sebagai berikut.

```
switch(ekspresi)
{
   case costanta1:
        pernyataan1;
```

```
    case costanta2:
        pernyataan2;
    case costanta3:
        pernyataan3;
    case costantaN:
        pernyataanN;
    default:
        pernyataanlain;
}
```

Hal-hal yang perlu diperhatikan adalah :

1. Dibelakang keyword case harus diikuti oleh sebuah konstanta, tidak boleh diikuti oleh ekspresi atau variable.
2. Konstanta yang digunakan bertipe int atau char
3. Jika dibentuknya seperti diatas maka apabila *ekspresi sesuai* dengan konstanta2 maka pernyataan2, pernyataan3 sampai pernyataanlain dieksekusi. Untuk mencegah tersebut gunakan keyword *break;* jika keyword break digunakan maka setelah pernyataan2 dieksekusi program langsung keluar dari pernyataan *switch*. Selain digunakan dalam switch keyword *break* banyak digunakan untuk keluar dari pernyataan yang berulang (looping).
4. Pernyataanlain dieksekusi jika konstanta1 sampai konstantaN tidak ada yang memenuhi *ekspresi.*

Contoh:

```
// program untuk melihat nilai akhir test
// Nilai A jika nilai diatas 80, B jika nilai >= 70 dan < 80
// C jika nilai >= 50 dan < 70, D jika nilai >=30 dan < 50
// E jika nilai < 30
#include<iostream>
#include<conio.h>
using namespace std;
int main(){
      int nilai;
      cout << "Masukan nilai test: ";
      cin >> nilai;
      switch(nilai/10){
            case 10:
            case 9:
            case 8:
            cout << "A" << endl;
            break;
            case 7:
            cout << "B" << endl;
            break;
            case 6:
            case 5:
            cout << "C" << endl;
            break;
            case 4:
            case 3:
            cout << "D" << endl;
```

```
                break;
                case 2:
                case 1:
                case 0:
                cout << "E" << endl;
                break;
                default:
                cout << "Salah, Nilai diluar jangkauan";
                looping = false;
        }
        getch();
        return 0;
}
```

Keluaran:
Masukan nilai test: 45
D
Masukan nilai test: 450
Salah, Nilai diluar jangkauan
Masukan nilai test: *nilai_test*
Salah, Nilai diluar jangkauan

Ket : 45, 450 dan nilai_test adalah hasil input dari user

3. Pernyataan *while*

Digunakan untuk pengambilan keputusan dan looping
Bentuk:
while(kondisi){
    pernyataan
}
Jika kondisi tidak terpenuhi, maka pernyataan tidak akan dieksekusi.
Contoh:

```
#include<iostream>
#include<conio.h>
#define TINGGI 15
#define LEBAR 10
using namespace std;

// program menggambarkan karakter khusus pada sebuah
// koordinat yang ditentukan
int main(){
        char matrix [TINGGI] [LEBAR];
        int x, y;
        for(y=0; y<TINGGI; y++){
                for(x=0; x<LEBAR; x++){
                        cout << "Ketik koordinat dalam bentuk x,y (4 2) .\n";
                        cout << "Gunakan bilangan negatif untuk berhenti .\n";
                }
        }

        while(x>=0 && y>=0){
                for(y=0; y<TINGGI; y++){
                        for(x=0; x<LEBAR; x++){
```

```
                        cout << matrix [x] [y];
                        }
                cout << "\n\n";
                }
                cout << " koordinat: ";
                cin >> x >> y;
        }
        getch();
        return 0;
}
```

Penjelasan:

Program ini adalah menggambar karakter [] jika di compile di Turbo C++ atau menggambar ♂ jika di compile di broland C++. (ditunjukan oleh karakter "xBO").

Karena adanya pernyataan *while(x>=0 && y>=0).* Maka program akan langsung mengeksekusi pernyataan:

Cout << "Koordinat: ";

Cin >> x >> y;

Matrix[x-1][y-1]="\xBO';

Jika user memasukan bilangan negatif. Pada program diatas terdapat fungsi getch(). Gunanya untuk menghentikan keluaran program sampai user menekan tombol keyboard. Untuk menggunakannya file conio harus di include.

4. Pernyataan *do… while*

Pernyataan do…while mirip seperti pernyataan while, hanya saja pada do…while pernyataan yang terdapat didalamnya minimal akan sekali di eksekusi.

Bentuk:

```
do{
        pernyataan;
} while (kondisi);
```

Terlihat walau kondisi tidak terpenuhi, maka pernyatan minimal akan di eksekusi sekali.

Contoh:

```
#include <iostream>
#include <conio.h>
#include <math.h>
using namespace std;

//program konversi bilangan desimal ke biner
int main(){
      int p, n, i = 0;
      cout << "Masukan bilangan desimal: ";
      cin >> p;
      double A[100];
      do{
            A[++i] = p%2;
            p = p/2;
            floor(p);
```

```
        }while(p>1);
        cout << "Nilai binernya: ";
        cout << p;
        for(n=i; n>=1; n--){
              cout << A[n];
        }
        getch();
        return 0;
}
```

Penjelasan:
Coba anda masukan bilangan negatif itulah letak kesalahan program ini (sekaligus untuk menunjukan sifat dari do…while). Jika anda memasukan bilangan negatf , maka program ini akan menghasilkan nilai biner yang sesuai dengan nilai desimal yang diasumsikan.

5. Pernyataan *for*

Pernyataan for digunakan untuk melakukan loping, pada umumnya looping yang dilakukan oleh for telah diketahui batas awal, syarat looping dan perubahannya.

Bentuk:
for (inisialisasi; kondisi; perubahan){
    pernyataan;
}

Selama kondisi terpenuhi maka pernyataan akan terus dieksekusi. Bila pernyataan yang terdiri atas satu baris pernyataan, maka tanda kurung {} tidak diperlukan.

Contoh:

```
int main(){
      for(int x=1; x<=100; x++) cout << x << endl;
      getch();
      return 0;
}

#include <iostream>
#include <conio.h>
using namespace std;

//program mencetak angka 1-100
int main(){
      for(int x=1; x<=100; x++){
            cout << x << endl;
      }
      getch();
      return 0;
}

Bagaimana jika program diatas diubah menjadi
```

```
#include<iostream>
using namespace std;

int main(){
      for(int x=1; ; x++) cout << x << endl;
}
```

Program diatas akan menampilkan bilangan yang banyaknya tak terhingga sehingga dapat membuat komputer anda berhenti bekerja. Contoh diatas juga merupakan prinsip membuat boom program (contohnya : bom mail) pernyataan *for* dapat berada didalam pernyataan *for* lainnya yang biasa disebut *nested for.*

Contohnya:

```
// program membuat segi tiga pascal
#include <iostream>
#include <conio.h>
#include <iomanip>
using namespace std;

main(){
      unsinged int n, a, b, x, s[100], p[100];
      cout << "Masukan nilai n: "; cin >> n;

      for(a=0,x=0; a<=0; a++, x+=2){
            cout << setw(3*n-x);
            s[a] = 1;
            p[a] = 1;
            for(b=0; b<=a; b++){
                  if(b<1 || b==a) cout << "1" << setw(4);
                  else{
                        s[b]=p[b];
                        p[b]=s[b-1]+s[b];
                        cout << p[b] << setw(4);
                  }
            }
            cout << endl;
      }
      getch();
      return 0;
}
```

PERNYATAAN BREAK

Pernyataan *break* akan selalu terlihat bila menggunakan pernyataan *switch.* Bila pernyataan ini dieksekusi, maka akn mengakhiri loop dan akan menghentikan iterasi pada saat tersebut.

PERNYATAAN CONTINUE

Petnyataan *continue* digunakan untuk pergi ke bagian bawah dari blookloop untuk memulai iterasi berikutnya

Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;

int main(){
      int n;
      for( ; ; ){
            cout << "Masukan bilangan integer: "; cin >> n;
            if(n%2 == 0) continue;
            else if(n%5 == 0) break;
            cout << "Lanjutkan loop selanjutnya. \n";
      }
      cout << "Akhir dari loop. \n";
      getch();
      return 0;
}
```

Keluarannya
Masukan bilangan integer: 9
Lanjutkan loop berikutnya
Masukan bilangan integer: 8
Masukan bilangan integer: 5
Akhir Dari loop

Materi Praktikum

- Pernyataan (Statement)
- Penyelesaian kondisi : if tunggal, if – else, if bersarang (nested if), switch tunggal, switch bersarang
- Penyelesaian kondisi berulang (looping) : For, While, While-Do, Loop Didalam Loop, Continue Dan Break;
- Operator

Dibawah ini adalah contoh soal mengguanakan token diatas

Latihan 12

Buatlah program dengan C++ dari algoritma sbb(gunakan IF tunggal dan switch):

Input : 2 bilangan integer (x dan Y)

Proses : jika x > y maka tampilkan tulisan “x lebih besar dari y”
jika y > x maka tampilkan tulisan “y lebih besar dari x”

Output : sama dengan proses

Latihan 13

Buatlah program dengan C++ dari algoritma sbb(gunakan IF else tunggal dan switch):

Input : tiga sisi segitiga (s1, s2, s3)

Proses : jika 1 = s2 atau 1 = s3 atau 3= s2 tampilkan tulisan “Segitiga sama kaki“
Jika 1 = s2 dan 2 = s3 dan 1 = s3 tampilkan tulisan “Segitiga sama sisi“
Jika 1 ≠ s2 dan 2 ≠ s3 dan 1 ≠ s3 tampilkan tulisan “segitiga sembarang“

Output : sama dengan proses

Latihan 14

Taxi menetapkan harganya berdasarkan kilo meter. Jika kilo meter yang diempuh kurang dari 7 km harga per km-nya adalah Rp. 3000. Jika lebih dari 7 km harga per km-nya adalah Rp. 2000. Buatlah programnya.

Buatlah program dengan C++ dari algoritma sbb(gunakan IF else tunggal dan switch):

Latihan 15

System kalender yang kita pakai sekarang adalah system kalender Georgian yang ditemukan oleh PopeGregory XIII pada tahun 1852. Menurut Georgian tahun kabisat adalah:

Tahun yang angkanya habis dibagi 4, kecuali tahun abad.

Tahun abad yang merupakan tahun kabisat adalah tahun yang habis dibagi 400.

Catatan: tahun abad adalah tahun yang akhirannya 000 contoh 1000, 1900, 1800, 2000, dll.

Jadi menurut Georgian tahun 2000, 1980, 1984 adalah tahun kabisat sedangkan tahun 1900, 1700 bukan tahun kabisat karena tidak habis dibagi 400.

Buatlah program dengan C++ (gunakan IF else tunggal dan switch):

Latihan 16

PertAndingan Sepak bola memiliki aturan sebagai berikut.

Main ke -1	Main ke -2	Skor
Menang(M)	Menang(M)	2
Menang(M)	Kalah(K)	1
Kalah(K)	Kalah(K)	1
Kalah(K)	Kalah(K)	0

Aturan Mendali:
Jika skor = 2 maka tampilkan tulisan “Mendali Emas”
Jika skor = 1 (K) (M) Maka tampilkan tulisan”Mendali Perak”
Jika skor = 1 (M) (K) Maka tampilkan tulisan”Mendali Perunggu”
Jika skor = 0 Maka tampilkan tulisan”Kalah nih ciyeeeee..”
Buatlah program dengan C++ (gunakan IF else tunggal dan switch):
Input : Main Ke-1, Main ke-2
Proses : lihat tabel diatas dan aturan mendali
Output : sama dengan proses

Tugas 3: Sesi 4

Biro tout “lpkia.ac.id” membuat aturan untuk wisata group sebagai berikut:

Tujuan	Kode	Batas Min Peserta	Biaya/Peserta
Pangandaran	pa	100	Rp. 150.000
Broubudur	bo	85	Rp. 100.000
Taman Mini	tm	50	Rp. 80.000

Buatlah program dengan C++
Input : kode tujuan, jumlah peserta
Proses : Lihat table
Jika jumlah pesert dibawah batas minimal maka harus membayar extra 15% dari biaya diatas
Total harga jika batas min terpenuhi = ……..
Total harga jika batas min tidak terpenuhi = ……..
Output : Total Harga

Latihan 17
Buatlah program untuk menampilkan deretan angka 1 s.d 10 (gunakan penyelesaian kondisi berulang (loopin): For, While, While-Do).

Latihan 18
Buatlah program untuk menampilkan deret angka bilangan ganjil dan genap dari 1 s.d 10. (gunakan penyelesaian kondisi berulang (loopin): For, While, While-Do).

Latihan 19
Buatlah program untuk menampilkan deret angka 1 s.d 100 carilah angka 9 diantara deret tersebut.

Latihan 20
Buatlah program untuk menghitung berapakah total nilai rupiah yang harus disiapkan oleh juru bayar pada saat ia melakukan penggajian. Asumsikan bahwan data yang akan di input dari keyboard adalah nama karyawan dan nilai gajinya berdihnya dimana jumlah karyawan-nya ada 10 orang.(gunakan penyelesaian kondisi berulang (loopin): For, While, While-Do).

10-11

Landasan Teori:
Fungsi dan deklarasinya
Fungsi bawaan C++

Fungsi adalah sekumpulan perintah operasi program yang dapat menerima argumen input dan dapat memberikan hasil output yang dapat berupa nilai maupun hasil operasi.
Nama fungsi yang didefinisikan sendiri oleh pemerogram tidak boleh sama dengan built-in function pada compiler C++.
Fungsi digunakan agar pemerogram menghindari penulisan bagian program (kode) berulang-ulang. Dapat menyusun kode program agar terlihat lebih rapid an kemudahan dalam bugging program.

FUNGSI, DEKLARASI DAN SEJENISNYA
Pemerogram dapat membuat fungsi yang didefinisikan sendiri olehnya.

Contoh:

```
// fungsi kuadrat
// tipe_return nama_fungsi(tipe_argument argument)
float kuadrat(float x){
return x*x;
}
```

Fungsi yang didefinisikan sendiri oleh pemerogram terdiri atas dua bagian, yaitu judul(*header*) dan isi(*body*). Judul dari sebuah fungsi terdiri dari tipe return(float). Nama fungsi (kuadrat). Dan list parameter (float x). jadi judul untuk fungsi kuadrat adalah.

```
float kuadrat(float x)
```

isi dari sebuah fungsi adalah blok kode yang mengikuti judulnya. Berisi kode yang menjalankan aksi dari fungsi, termasuk pernyataan *return* yang akan menbuat nilai fungsi dikembalikan ke yang memanggilnya, isi fungsi kuadrat() adalah.

```
{
return x*x;
}
```

Barisnya diisi dari fungsi cukup besar, meskipun demikian judulnya tetap hanya berada dalam satu baris. Isi dari sebuah fungsi dapat memanggil fungsi itu sendiri (disebut *rekrusif*) atau memanggil fungsi lainnya.
Pernyataan *return* dari sebuah fungsi mempunyai dua manfaat, yaitu akan mengakhiri fungsi dan mengenbalikan nilainyake program pemanggil. Bentuk umum pernyataan return adalah:

```
return ekspresi
```

Dengan *ekspresi* adalah sebuah *ekspresi* yang nilainya dinyatakan untuk sebuah variable yang tipenya sama seperti tipe *return*-nya void.

Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;

void sayHello(string nama); // deklarasi fungsinya sayHello()

int main(){
      string n;
      cout << "Masukan nama anda: ";
      getline(cin,n);
      sayHello(n);
      getch();
      return 0;
}

void sayHello(string nama){
      cout << "Selamat datang " << nama;
}
```

Pengertian deklarasi fungsi berbeda dengan definisi fungsi. Suatu deklarasi fungsi adalah suatu judul fungsi yang sederhana yang diikuti oleh tanda semicolon (;). Sedangkan definisi fungsi adalah fungsi yang lengkap terdiri dari judul dan isinya. Suatu deklarasi fungsi disebut juga prototype fungsi.

Suatu deklarasi fungsi layaknya suatu deklarasi variable. Yang memberitahu compiler semua informasi yang dibutuhkan untuk mengkompilasi file *Compiler* tidak mengetahui bagaimana fungsi bekerja, yang perlu diketahui adalah nama fungsi, jumlah dan tipe parameternya, dan tipe balikannya(*return*). Hal ini merupakan informasi yang dimuat secara lengkap dalam judul fungsi.

Jika seperti sebuah deklarasi variable, suatu deklarasi fungsi harus muncul diatas semua nama fungsi yang digunakannya. Berbeda dengan definisi fungsi, yang dapat diletakan terpisah dari deklarasinya dan dapat muncul dimana saja diluar fungsi main() dan biasanya dituliskan setelah fungsi main() atau dalam file terpisah yang ingin digunakan tinggal menambahkan preprocessor *#include "nama_file"* pada file utama jika definisi fungsi diletakan diatas fungsi main() maka deklarasi fungsi tidak diperlukan

Variable-variable yang dilist dalam parameter fungsi disebut juga parameter-parameter formal atau argument-argumen formal. Variable lokal seperti hanya ada selama eksekusi fungsi yang bersangkutan. Dalam contoh dibawah parameter-parameter formalnya adalah x dan y.

Variable yang dilist dalam pemanggilan fungsi disebut parameter-parameter actual atau argument-argument actual. Sama seperti variable lainnya dalam program utama. Variable-variable tersebut harus dideklarasikan sebelum digunakan dalam pemanggilan, dalam contoh dibawah, parameter-parameter aktualnya adalah m dan n.

Contohnya:

```
// penggunaan fungsi rekursif
// program menecek bilangan integer atau bukan
#include <iostream>
#include <conio.h>
```

```
#include <math.h>
using namespace std;

void cekInt(double n);
int main(){
      double angka;
      cout << "Masukan sebuah angka: ";
      cin >> angka;
      cekInt(angka);
      getch();
      return 0;
}
void cekInt(double n){
      if(n>1) cekInt(-1);
      else if(n<1) cekInt(-n-1);
      else{
            if(n>0 && n<1) cout << n << "\t Bukan bilangan bulat \n";
            else cout << n << "\t Bilangan bulat \n";
      }
}
```

Keluaran:
Masukan sebuah angka: 57
Bilangan Bulat
Masukan sebuah angka: 0.57
Bukan bilangan bulat
Masukan sebuah angka: -24
Bilangan bulat

NILAI BAWAAN UNTUK ARGUMEN FUNGSI

Salah satu keistimewaan C++ yang sangat bermanfaat dalam pemerograman adalah adanya kemampuan untuk menyetel nilai *default* Argumen fungsi. Argument-argumen yang memiliki nilai bawan nantinya tidak dapat disertakan didalam pemanggilan fungsi dan dengan sendirinya C++ akan menggunakan nilai bawaan dari argument yang tidak disediakan.

Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;
void sayHello(int n=1);

int main(){
      sayHello();
      getch();
      return 0;
}
void sayHello(int n=1){
      for(int m=0; m<n; m++) cout << "Halloo.....\n";
}
```

Penjelasn:
Jika pada program, argumennya sayHello tidak diberikan, maka program akan menampilkan
Halloo…..
Sebanyak satu kali, namun jika argument pada fungsi sayHello diberkan, misalkan sayHello(4), maka program akan menampilkan.
Halloo…..
Halloo…..
Halloo…..
Halloo…..

Itulah yang disebut nilai default pada fungsi.

MELEWATKAN ARGUMEN DENGAN REFERENSI
Lihat sesii mengenai array dan pointer.

FUNGSI BAWAAAN C++
Anda dapat menggunakan fungsi bawaan C++ , misalkan fungsi-fungsi matematika, pengolah kata dan banyak lagi. Sebenarnya (mungkin tidak terasa bagi anda) main juga adalah fungsi, jadi tanpa anda sadari anda telah mengunakan fungsi.

Untuk menggunakan fungsi-fungsi tersebut anda harus meng-include file dimana fungsi tersebut didefinisikan.

Misalkan:

- Fungsi-fungsi matematika, anda harus meng-include file math.h
- Fungsi-fungsi pengolah string dan karakter, anda harus meng-include file string
- Fungsi clscr(), getch() dalam file conio.h

Materi Praktikum

- Fungsi
- Fungsi bawaan C++

Latihan 21
Buatlah program dengan C++ untuk menentukan bilangan prima dan bukan prima (gunakan Fungsi).

Latihan 22
Buatlah program dengan C++ untuk membuat fungsi nama bulan dari januari sampai dengan desember.

Latihan 23
Buatlah program dengan C++ untuk membuat fungsi kwetansi.

Sesi 12-14

Landasan Teori:
Array
String
r

ARRAY

Array adalah kumpulan data-data bertipe sama dengan menggunakan nama yang sama. Dengan menggunakan array, sejumlah nama variable dapat memakai nama yang sama. Antara satu variable dengan variable yang lain didalam array dibedakan berdasarkan *subscrip.* Sebuah *subscript* berupa bilangan didalam kurung siku. Melalui *subscript* inilah masing-masing elemen array dapat diakses. Nilai *subscript* pertama secara default adalah 0.

C++ tidak mengecek array. Bila anda menyatakan int x[10], ini artinya 10 elemen yang dimulai dari 0. Karena itu elemen terakhir array adalah x[9]. Bila anda salah mereferensikannya dengan x[10], anda akan mendapatkan harga yang tidak terpakai. Akan lebih buruk lagi jika anda memberikan harga ke x[10], yang tidak dapat diterima.

1. **Reperentasi Array**

   Misalkan kita memiliki sekumpulan data *ujian* seorang siswa. *Ujian* perbaikan pertama bernilai 90, kemudian 95, 78, 85. Sekarang kita ingin menyusunya sebagai satu data kumpulan *ujian* seorang siswa. Dalam array kita menyusunya sebagai berikut:

   ujian[0] = 90;
   ujian[1] = 95;
   ujian[2] = 78;
   ujian[3] = 85;
   Perhatikan:
   - Tanda kurung [] digunakan untuk menandakan element array
   - Perhitungan element array dimulai dari 0, bukan 1

   Empat pernyataan diatas memberikan nilai kepada array *ujian*. Tetapi sebelum kita memberikan nilai kepada array, kita harus mendeklarasikannya terlebih dahulu, yaitu:

   ```
   Int ujian[4];
   ```

   Perhatikan bahwa nilai 4 yang berada didalam tanda kurung menunjukan jumlah elemen array, bukan menunjukan elemen array yang ke-4, jadi elemen array *ujian* dimulai dari 0 sampai 3

   Program juga dapat menginisialisasi array sekaligus mendeklarasikannya, sebagai contoh:

   int ujian = {,90,95,78,85};

   Elemen terakhir dari array diisi dengan karakter “\0”. Karaktrer ini memberitahu compiler bahwa akhir dari elemen array telah dicapai. Walaupun pemerogram tidak dapat melihat karakter ini secara eksplisit, namun compiler mengetahui dan membutuhkannya.

Sekarang kita akan membuat daftar beberapa nama pahlawan di Indonesia

```
char pehlawan[3][15];
char pehlawan[0][15] ="Soekarno"
char pehlawan[1][15] ="Diponegoro"
char pehlawan[2][15] ="Soedirman"
```

Array diatas terlihat berbeda dengan contoh array pertama kita. Perhatikan bahwa pada array *pahlawan* memiliki dua buah tanda kurung [ ] [ ]. Array seperti itu disebut array dua dimensi. Tanda kurung pertama menyatakan total elemen yang dapat dimiliki oleh array pahlawan dan tanda kurung kedua menyatakan total elemen yang dapat dimiliki setiap elemen array pahlawan. Dalam contoh diatas , tanda kurung kedua menyatakan karakter yang menyatakan nama pahlawan.

2. Menghitung Jumlah Elemen Array

Karena fungsi *sizeof()* mengembalikan jumlah byte yang sesuai dengan argumennya, maka operator tersebut dapat digunakan untuk menemukan jumlah elemen array, misalnya

```
int array[] = {26, 7, 82, 166; cout << sizeof(array)/sizeof(int)
```

Array mengembalikan nilai 4. Yaitu sama dengan jumlah elelmen yang dimiliki array array.

3. Melewatkan Array Sebagai Fungsi Argumentasi

Array dapat dikirim dan dikembalikan oleh fungsi. Pada saat array dikirim kedalam fungsi, nilai aktualnya dapa dimanipulasi

Contohnya:

```
#include <iostream>
#include <conio.h>
using namespace std;

void ubah(int x[]);
int main(){
      int ujian[]= {90, 95, 78, 85};
      ubah(ujian);
      cout << "Elemen kedua dari array ujian adalah: " << ujian[1] <<
endl;
      getch()
      return 0;
}

void ubah(int x[]){
      x[
}
```

Keluarannya:

Elemen kedua dari array ujian adalah: 100

POINTER

Pointer adalah variable yang berisi alamat memori variable lain secara tidak langsung menunjuk ke variable tersebut.

Analoginya – sebagai contoh – Andi berteman dengan Budi. Lalu anda ingin mengetahui jumlah keluarga Budi untuk keperluan sensus penduduk. Anda tidak mengetahui alamat Budi namun anda mengenal Andi, untuk mencarii jumlah keluarga Budi, makka pertama-tama anda pergi kerumah Andi, misalnya dirumah no 8321. Sesampai di Andi. Andi memberitahukan kepada anda bahwa alamat Budi pada alamat 8921. Kemudian anda pergi ke rumah Budi lalu mencatat jumlah keluarga yang dimiliki Budi yaitu lima orang (misalkan).

Dalam contoh diatas, Andi bertindak sebagai pointer, Andi tidak memberitahukan jumlah keluarga Budi, tetapi Andi memberitahu alamat Budi, di alamat 9821(alamat Budi) itulah anda mengetahui jumlah keluarga Budi. Jika alamat ditunjukan dengan symbol & dan isi dari ditunjukan symbol *, maka hubungan diatas adalah:

Nama	Alamat	Isi
Andi	8321	821 = &Budi
Budi	9821	5 =*(&Budi)

Dalam bentuk pointer ditulis:
Andi = &Budi; // baris 1
Budi = *(Budi); // baris 2

Subsitusi pernyataan baris 2
Andi = *Andi;

Contoh program yang menggunakan hal tersebut:

```
#include <iostream>
#include <conio.h>
using namespace std;

int main(){
      int *Andi; // Andi sebagai pointer
      int Budi = 5; // Budi bukan pointer, perhatikan pada *
      Andi = &Budi; //  Isi dari andi yaitu alamat Budi
      cout << "Isi alamat Andi: " << Andi << endl;
      cout << "Isi alamat Budi: " << Budi << endl;
      cout << "Isi alamat Budi: " << *Andi << endl;
      cout << "Alamat memori Andi: " << &Andi << endl;
      cout << "Alamat memori Budi: " << &Budi << endl;
      getch();
      return 0;
}
```

Keluarannya:
Isi alamat Andi: 0x22fe44
Isi alamat Budi: 5
Isi alamat Budi: 5
Alamat memori Andi: 0x22fe48
Alamat memori Budi: 0x22fe44

1. Pointer –Array

Dalam bab sebelumnya kita telah membahas array, sekarang kita akan melihat bagaimana data disimpan di memori dalam sebuah array.

Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;

int main(){
      int n;
      int array[4] = {10, 20, 30, 40};
      for(n=0; n<4; n++){
            cout << "Array [" << n << "] = " << array[n] << endl;
            cout << "\tMenggunakan pointer = " << *&array[n] << endl;
            cout << "\tDisimpan dalam = " << &array[n] << endl;
      }
      getch();
      return 0;
}
```

Keluarannya:

Array [0] = 10
	Menggunakan pointer = 10
	Disimpan dalam = 0x22fe30
Array [1] = 20
	Menggunakan pointer = 20
	Disimpan dalam = 0x22fe34
Array [2] = 30
	Menggunakan pointer = 30
	Disimpan dalam = 0x22fe38
Array [3] = 40
	Menggunakan pointer = 40
	Disimpan dalam = 0x22fe3c

Penjelasan:

Seperti yang anda lihat, setiap array yang disimpan dalam 2 byte memori karena kita menggunakan data integer. Perhatikan pula pointer dalam pengaksesan nilai setiap elemen array dan pengaksesan alamat setiap array.

- Alamat setiap elemen array dapat diperoleh dengan cara &array[n] atau array +n
- Isi dari setiap elemen array dapat diperoleh dengan cara array[n] atau *(arrai+n)

Dibawah ini adalah contoh pengaksesan memori dan isi memori dengan menggunakan cara kedua

Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;
```

```
int main(){
      int n;
      int array[4] = {10, 20, 30, 40};
      for(n=0; n<4; n++){
            cout << "Array [" << n << "] = " << array[n] << endl;
            cout << "\tMenggunakan pointer = " << *(array+n) << endl;
            cout << "\tDisimpan dalam = " << array+n << endl;
      }
      getch();
      return 0;
}
```

Keluarannya:

Array [0] = 10
    Menggunakan pointer = 10
    Disimpan dalam = 0x22fe30
Array [1] = 20
    Menggunakan pointer = 20
    Disimpan dalam = 0x22fe34
Array [2] = 30
    Menggunakan pointer = 30
    Disimpan dalam = 0x22fe38
Array [3] = 40
    Menggunakan pointer = 40
    Disimpan dalam = 0x22fe3c

Mengapa hasil antara dua contoh diatas sama namun sintaks-nya berbeda ? karena array itu sebenarnya telah menunjuk kealamat memori setiap elemen array cukup dengan array+n dengan n bilangan bulat (integer)

2. Pointer –String

String merupakan bentuk khusus dari array. Oleh karena itu operasi pointer-array tidak jauh beda dengan operasi pointeri-string.

Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;

int main(){
      char nama[5] = "Andi";
      cout << "Nama awal: " << nama << endl;
      char*ptr;
      ptr = nama;
      *(ptr+3) = 'y';
      cout << "Nama menjadi: " << nama << endl;
      getch();
      return 0;
}
```

Keluarannya:
Nama awal: Andi
Nama menjadi: Andy
Jadi:

- Strung adalah array(susunan) dari karakter-karakter
- String dapat diakses dan dapat dimanipulasi oleh pointer
- Alamat awal string dapat diperoleh dari namanya

3. Pointer Sebagai Argumren String

Jika pointer dikirim sebagai argument, maka nilai aktualnya dapat dimodifikasi.
Contoh:

```
#include <iostream>
#include <conio.h>
using namespace std;

void ubah(char *x);
int main(){
	char *ptr, nama[5] = "Andi";
	ptr = nama; // ptr sebagai pointer ke variable nama
	cout << "Nama awal: " << nama << endl;
	ubah(ptr);
	cout << "Nama menjadi: " << nama << endl;
	getch();
	return 0;
}
void ubah(char *x){
	*(x+3) = 'y';
}
```

Keluarannya:
Nama awal: Andi
Nama menjadi: Andy

4. Alias

Alias adalah nama lain dari suatu variable. Jika suatu perubah terjadi pada variable alias maka akan berpengaru pada variable asli dan begitu juga sebliknya.
Contoh:

```
/*==============================
#include <iostream>
#include <conio.h>
using namespace std;

int main(){
	int uang = 10000;
	int &duit = uang;
	cout << "Nilai uang Rp." << uang << endl;
	cout << "Nilai duit Rp." << duit << endl;
	uang = 9000;
```

```
      cout << "Uang dibelikan eskrim Rp.1000 nilainya menjadi Rp." <<
uang << endl;
      cout << "Nilai duit juga berubah menjadi Rp." << duit << endl;
      getch();
      return 0;
}
```

Keluarannya:
Nilai uang Rp.10000
Nilai duit Rp.10000
Uang dibelikan eskrim Rp.1000 nilainya menjadi Rp.9000
Nilai duit juga berubah menjadi Rp.9000

Penjelasan:
Perubahan pada uang menyebabkan perubahan pada duit karena duit memiliki alamat memori yang sama dengan uang. Jadi jika isi dari alamat memori uang atau duit berubah, maka nilai variable duit atau uang juga akan ikut berubah

5. Argument baris perintah

Seringkali kita menggunakan perintah *editfile.txt* pada DOS atau perintah *vfile.txt* pada Unix. Yang dimaksud dengan argument baris perintah yaitu *file.txt* hal itu dapat dibuat dengan C++ dengan menyatakan argument berikut pada fungsi main()

```
int main(int argc, char const *argv[])
{
      .........
}

atau

main(int argc, char const *argv[])
{
      .........
}
```

Keterangan:

- Argc: berisi jumlah parameter baris ditambah 1
- Argv: berisi daftar argument dan program, dengan rincian sebagai berikut:
  - Arg[0] Menunjukan nama program, lengkap dengan alamat path
  - Arg[1] Menunjukan argument pertama (kalua ada)
  - Arg[n] Menunjukan argument ke-n (kalua ada)

Contoh:

```
// beri nama tes.cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main(int argc, char const *argv[])
{
for(int a=0; a<argc; a++){
```

```
      cout << "argv[" << a << "] = " << argv[a] << endl;
}
return 0;
}
```

Penjelasan:
Setelah di compile link akan muncul file tes.exe, misalkan anda simpan di d:\tes.exe buka command prompt, pindah ke direktori d:\ ketikan tes argument1 argumen2 argumen3, maka akan mucul tampilan

argv[0] = D:\TES.EXE
argv[1] = argumen1
argv[2] = argumen2
argv[3] = argumen3

Dibawah ini diberikan cintoh penggunaan argument baris perintah yang lain, supaya anda lebih memahami

Contoh:

```
//program mengubah nilai desimal ke biner
//simpan dengan nama dec2bin.cpp
#include <iostream>
#include <conio.h>
#include <math.h>
#include <stdlib.h>
using namespace std;


int main(int argc, char const *argv[]){
if(argc !=2 ){
      cerr <<"Pemakaian : dec2bin angka";
      exit(1);
}
int p = atoi(argv[1]), n, i = 0;
double A[100];
do{
      A[++i] = p%2;
      p = p/2;
      floor(p);
}while(p>1);
cout << "Nilai binernya: ";
cout << p;
for(n=i; n>=1; n--){
      cout << A[n];
}
getch();
return 0;
}
```

Jika anda mengetik *dec2bin4*, maka outputnya *Nilai binernya: 100*

Materi Praktikum

- Array

Latihan 24

Buatlah program untuk mengurutkan data:
Diketahui negara (Malaysia, Indonesia, Brunei, Fhilipina, Singapura, Birma) urutkan berdasarkan abjad (gunakan array).

Latihan 25

Buatlah program untuk menghitung berapakah total nilai rupiah yang harus disiapkan oleh juru bayar pada saat ia melakukan penggajian. Asumsikan bahwan data yang akan di input dari keyboard adalah nama karyawan dan nilai gajinya berdihnya dimana jumlah karyawan-nya ada 10 orang.(gunakan penyelesaian kondisi berulang (loopin): For, While, While-Do, Array Dan Pointer).

Latihan 26

Diketahui deretan angka sebagai berikut:

3	6	4	8	1	2	7	5	9	10

Urutkan angka tersebut menggunakan teknik algoritma bubleshort, selection, insert