

Pemrograman Visual C++ Konsep Dasar, Algoritma dan Aplikasi



Dr. Haryanto, M.Pd, M.T.

Undang-undang Republik Indonesia Nomor 19 Tahun 2002 tentang Hak Cipta

Lingkup Hak Cipta

Pasal 2:

1. Hak Cipta merupakan hak eksklusif bagi Pencipta atau Pemegang Hak Cipta untuk mengumumkan atau memper-banyak ciptaannya, yang timbul secara otomatis setelah suatu ciptaan dilahirkan tanpa mengurangi pembatasan menurut peraturan perundang-undangan yang berlaku.

Ketentuan Pidana

Pasal 72:

1. Barangsiapa dengan sengaja atau tanpa hak melakukan perbuatan sebagaimana dimaksudkan dalam Pasal 2 ayat (1) atau Pasal 49 ayat (1) dan (2) dipidanakan dengan pidana penjara masing-masing paling singkat 1 (satu) bulan dan/atau denda paling sedikit Rp 1.000.000,00 (satu juta rupiah), atau pidana penjara paling lama 7 (tujuh) tahun dan/atau denda paling banyak Rp 5.000.000.000,00 (lima miliar rupiah).
2. Barangsiapa dengan sengaja menyiarkan, memamerkan, mengedarkan, atau menjual kepada umum suatu ciptaan atau barang hasil Pelanggaran Hak Cipta atau Hak Terkait sebagaimana dimaksudkan dalam ayat (1) dipidanakan dengan pidana penjara paling lama 5 (lima) tahun dan/atau denda paling banyak Rp 500.000.000,00 (lima ratus juta rupiah).

PEMROGRAMAN VISUAL C++ KONSEP DASAR, ALGORITMA, DAN APLIKASI

Dr. Haryanto, M.Pd, M.T.



2012

**PEMROGRAMAN VISUAL C++
KONSEP DASAR, ALGORITMA, DAN APLIKASI**

Cetakan 1, Oktober 2012

Penulis:

Dr. Haryanto, M.Pd, M.T.

Editor: Teguh Setyawan

Tata Letak: Ari Setyo Wibowo

Desain Cover: Kuncoro W.D

Dicetak dan diterbitkan oleh:

UNY Press

Jl. Affandi (Gejayan), Gg. Alamanda, Kompleks FT
Kampus Karangmalang, Yogyakarta
Telp. (0274) 589346
Email: redaksi.unypress@gmail.com

ISBN 978-979-8418-83-9

Perpustakaan Nasional: Katalog dalam Terbitan (KDT)
110 hlm; 16 x 23 cm

KATA PENGANTAR

Segala puji bagi Allah S.W.T, bahwa hanya dengan karunia dan rahmat-Nya kami dengan sabar setelah melalui berbagai tantangan dan rintangan dapat menyelesaikan penulisan buku yang berjudul: "Pemrograman Visual C++ Konsep Dasar, Algoritma, dan Aplikasi".

Penyelesaian dan penyusunan buku ini tidak terlepas dari bantuan berbagai pihak. Oleh karena itu, terima kasih kami ucapkan kepada:

1. Dekan Fakultas Teknik UNY Yogyakarta yang telah memberi kesempatan, motivasi dan berbagai fasilitas untuk penulisan buku ini.
2. Pembantu Dekan 1, yang telah memfasilitasi dalam penulisan buku ini.
3. Ketua Jurusan Pendidikan Teknik Elektro FT UNY.
4. Keluarga besar jurusan Pendidikan Teknik Elektro dan Fakultas Teknik Universitas Negeri Yogyakarta, atas doadan bantuan moril, materil, dan spirituil bagi kami sekeluarga.
5. Keluarga yang penuh dengan kesabaran dan tak henti-hentinya memberi semangat untuk menyelesaikan buku ini.
6. Berbagai pihak yang tak dapat disebutkan satu per satu yang telah membantu kelancaran studi kami.

Kami menyadari tentu tak ada karya manusia yang sempurna, oleh karena itu kepada para pembaca saran konstruktif sangat kami harapkan. Semoga Allah S.W.T memberi pahala dan berbagai kenikmatan yang banyak serta rizqi yang barokah dunia dan akhirat, kepada pihak-pihak yang telah membantu penyelesaian dan penyempurnaan buku ini. amien amien amien ya robbal 'alamin.

Yogyakarta, Oktober 2012

Haryanto

DAFTAR ISI

Judul Buku.....	i
Kata Pengantar.....	v
Daftar Isi.....	vi
BAB IPemrograman Dasar C++.....	1
BAB IIPercabangan Program.....	17
BAB IIIPerulangan Program (<i>Looping</i>).....	25
BAB IVPointer.....	51
BAB VStruktur.....	54
BAB VIFungsi.....	63
BAB VIIPemrograman Berorientasi Objek.....	73
BAB VIIPemrograman Visual.....	83
DaftarPustaka.....	103
BiodataPenulis.....	104

BAB I

PEMROGRAMAN DASAR C++

A. Pendahuluan

Pemrograman dengan bahasa C++ dibangun dengan menggunakan perintah-perintah yang sedikit berbeda dengan bahasa C. *Software Visual C++* digunakan untuk pembuatan program yang melingkupi teknik pemrograman biasa, terstruktur, *objek oriented programming* (OOP) hingga pemrograman visual di lingkungan windows. Dalam buku ini akan dikembangkan teknik pemrograman biasa hingga pemrograman visual. Namun, sebelum melakukan pembuatan program, terlebih dahulu akan dikenalkan beberapa hal yang sangat diperlukan dalam rangka pengembangan program, yaitu: tipe data, variabel, dan operator.

Program-program pada buku ini diatur mulai dari pemrograman dengan teknik biasa dan tampilan di layar *console* (DOS) hingga teknik pemrograman visual dan tampilan di layar windows. Pengembangan program visual C++ menggunakan urutan perintah yang ditulis dengan serangkaian karakter yang disusun sedemikian rupa sehingga dimengerti oleh komputer. Karakter adalah elemen terkecil dari program C++ yang meliputi: huruf (a sampai z, A sampai Z), angka (0 sampai 9), simbol (misal * ! #), dan kode kontrol (*formfeed* dan *new line*).

Penyusunan karakter menjadi sebuah kata program dapat memiliki konsekuensi yang berbeda. Terdapat kata (*word*) yang dapat langsung dimengerti oleh program C++ (disebut dengan *reserve word*) dan kata yang tidak dapat dimengerti oleh program C++ (disebut dengan *identifier*). *Reserve word* tidak dapat digunakan sebagai *identifier*. Beberapa contoh *reserve word*, yaitu: *cin*, *cout*, *auto*, *char*, *float*, *int*, *if*, *static*, *switch*, *for*, *while*.

Identifier adalah kata yang dibuat oleh programmer (orang yang membuat program) yang digunakan untuk memberi nama variabel atau kelas (tentang variabel dan kelas akan dijelaskan kemudian). Adapun aturan untuk membuat *identifier* adalah sebagai berikut:

- Karakter pertama harus berupa huruf
- Karakter kedua dan seterusnya dapat berupa huruf, angka atau garis bawah

- Jika lebih dari satu kata harus digandeng
- Huruf kapital dan huruf kecil diperlakukan berbeda.

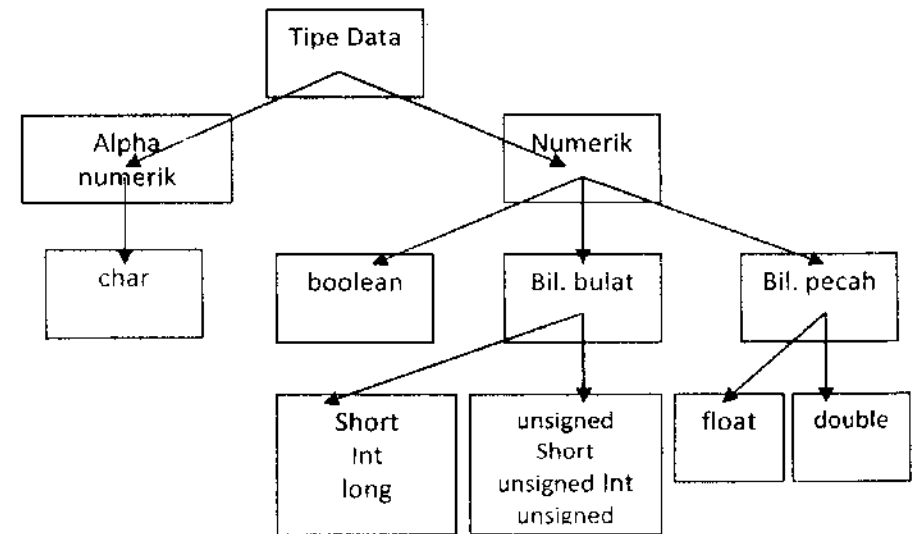
Contoh: *identifler*, X, Nama, SemesterSatu, Minggu1, IP_Kumulatif, TglSkrg

B. Tipe Data

Tipe data digunakan untuk menentukan jenis dan rentang nilai yang akan digunakan dalam pembuatan program. C++ memiliki beberapa tipe data seperti pada tabel 1.1.

Tipe Data	Ukuran (byte)	Jangkauan Nilai
Char	1	-128 s/d 127
Unsigned char	1	0 s/d 255
Short	2	-32.768 s/d 32.767
Unsigned Short	2	0 s/d 65.535
Long	4	-2.147.483.648 s/d 2.147.483.647
Unsigned Long	4	0 s/d 4.294.967.295
Int	4	Sama dengan long
Unsigned Int	4	Sama dengan unsigned long
Float	4	1.2E-38 s/d 3.4E381
Double	8	2.2E-308 s/d 1.8E308
Boolean	1	True atau false

Secara garis besar tipe data dapat digambarkan dalam sebuah diagram, sehingga terlihat pengelompokannya (lihat Gambar 1.1).



Gambar 1.1: Tipe Data

Tipe data digunakan untuk menyatakan nilai dari suatu konstanta (*literal*). Terdapat beberapa kelompok *literal*, yaitu: *literal* bilangan, *literal* logika, *literal* karakter, dan *literal* string. *Literal* dibedakan menjadi dua yaitu bilangan bulat (misal: 57, -212, 1255) dan bilangan pecah/real (misal: -21.12, 2.5E-03, 0.53, 1.2E05). Untuk menyatakan bilangan selain desimal, yaitu oktal dan heksadesimal digunakan aturan seperti contoh berikut: 017 menyatakan 17 oktal (basis 8), 0x17 menyatakan 17 heksadesimal (basis 16), 0x3E2 menyatakan 3E2 heksadesimal (basis 16). Di samping *literal* bilangan, bahasa C juga mengenal *literal* logika. Terdapat dua nilai logika, yaitu: *TRUE* atau 1 dan *FALSE* atau 0. Pada *literal* karakter penulisannya menggunakan tanda petik tunggal ('). Contoh: *literal* 'x' (karakter x), *literal* '#' (karakter simbol #), *literal* '7' (karakter bilangan 7). *Literal* string digunakan untuk menuliskan kata (kumpulan karakter). Penulisan *literal* string

menggunakan tanda petik ganda. Contoh: "Jalan Kenari No. 69", "Perumahan Tirtopermai No. 17 Nogotirto"

Bahasa C++ juga dilengkapi dengan kode karakter *escape* yaitu kode karakter yang diawali dengan tanda *counter slash* (\). Tabel 1.2 menyajikan beberapa kode karakter *escape*.

Tabel 1.2: kode karakter *escape*

Kode	Keterangan
\0	Karakter ber-ASCII nol (karakter null)
\a	Karakter audio (bel)
\b	<i>Backspace</i>
\f	<i>Formfeed</i>
\n	<i>Newline</i>
\r	<i>Carriage return</i>
\t	Tab
\v	Tab vertikal
\'	Petik tunggal
\"	Petik ganda
\?	Karakter ?
\ddd	Oktal (dd= 0 s.d 377)
\xdd	Heksadesimal (dd= 0 s.d FF atau ff)
'\n'	Karakter <i>newline</i>
'\x0a'	Karakter <i>formfeed</i>
'\12'	Karakter <i>formfeed</i>

Penggunaan *literal* dalam sebuah prgram disesuaikan dengan kebutuhan program yang akan dibuat. Dalam buku ini dalam membuat program selanjutnya kan dimulai dengan pembuatan algoritma terlebih dahulu. Algoritma adalah urutan langkah yang harus dilakukan oleh komputer saat mengeksekusi program. Contoh-contoh program yang akan dikembangkan di buku ini dalam rangka untuk memperjelas uraian akan diorientasikan pada aplikasi di bidang umum dan teknik elektro.

Contoh 1.1:

Algoritma:

1. Diketahui Komponen Resistor
2. Simbol R
3. Pita warna: Merah, merah, merah, emas
4. Hitung nilai resistor dan toleransinya

Program:

```
#include <iostream.h>
void main()
{
    cout << "Nama komponen: Resistor" << endl
        << "Simbol: " << 'R' << endl
        << "Kode warna: Merah merah merah emas" <<
endl
        << "Nilai: 2200 Ohm" << endl
        << "Toleransi: 5%" << endl;
}
```

Hasil 1.1:

Nama komponen: Resistor
 Simbol: R
 Kode warna: Merah merah merah emas
 Nilai: 2200 Ohm
 Toleransi: 5%

Penjelasan program:

Baris perintah *#include <iostream.h>* digunakan untuk memberitahu kepada komputer bahwa di dalam program terdapat peintah yang melibatkan library *iostream.h*
 Baris perintah *void main()* merupakan program utama dengan jenis fungsi tidak memakai nilai kembali (*void*), artinya bahwa program tidak spesifik untuk jenis bilangan tertentu.
 Baris perintah { berpasangan dengan } merupakan tanda awal dan akhir program.

Baris perintah `cout << "Nama komponen: Resistor" << endl` (*end line*) digunakan untuk menampilkan teks string Nama komponen: Resistor dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

Baris perintah `<< "Simbol: " << 'R' << endl` digunakan untuk menampilkan teks string Simbol: R dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

Baris perintah `<< "Kode warna: Merah merah merah emas" << endl` berfungsi untuk menampilkan teks string Kode warna: Merah merah merah emas dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

Baris perintah `<< "Nilai: 2200 Ohm" << endl` untuk menampilkan teks string Nilai: 2200 Ohm dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

Baris perintah `<< "Toleransi: 5%" << endl;` untuk menampilkan teks string Toleransi 5% dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

Contoh 1.2:

Algoritma:

1. Diketahui Komponen Kapasitor
2. Simbol C
3. Satuan Farad
4. Nilai 1000 μF

Program:

```
#include < iostream.h>
void main()
{ cout << "Nama komponen: " << " Kapasitor" << endl
  << "Simbol: \ 'C\ ' << endl
  << "Satuan: " << " Farad" << endl
  << "Nilai: 1000  $\mu\text{F}$ " << endl;
}
```

Hasil 1.2:

Nama komponen: Kapasitor
Simbol: 'C'
Satuan: Farad
Nilai: 1000 μF

Penjelasan program:

Baris perintah `#include <iostream.h>` digunakan untuk memberitahu kepada komputer bahwa di dalam program terdapat perintah yang melibatkan library `iostream.h`

Baris perintah `void main()` merupakan program utama dengan jenis fungsi tidak memakai nilai kembali (`void`), artinya bahwa program tidak spesifik untuk jenis bilangan tertentu.

Baris perintah `{` berpasangan dengan `}` merupakan tanda awal dan akhir program.

Baris perintah `cout << "Nama komponen: Kapasitor" << endl` digunakan untuk menampilkan teks string Nama komponen: Resistor dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

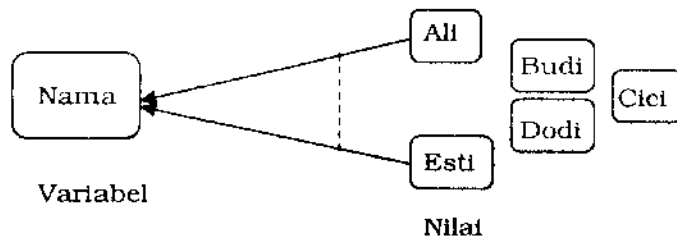
Baris perintah `<< "Simbol: " << 'C' << endl` digunakan untuk menampilkan teks string Simbol: R dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

Baris perintah `<< "Satuan: Farad" << endl` berfungsi untuk menampilkan teks string Kode warna: Merah merah merah emas dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

Baris perintah `<< "Nilai: 1000 μF " << endl` untuk menampilkan teks string Nilai: 2200 Ohm dan diikuti dengan perintah agar komputer membuat baris baru (*endl*).

C. Variabel

Keberadaan variabel dalam suatu program sangat penting. Variabel adalah simbol dari suatu tempat yang menyatakan lokasi di dalam memori komputer untuk digunakan mewakili suatu nilai yang dapat diubah-ubah.



Gambar 1.2: Variabel Berisi Nilai *String*

Pada Gambar 1.2 menyatakan bahwa variabel "Nama" berisi nilai-nilai *string* yang berubah-ubah, misal: Nama berisi "Ali", "Budi", atau dapat pula "Esti". Penggunaan variabel di dalam program harus dideklarasikan terlebih dahulu. Deklarasi variabel disesuaikan dengan jenis/tipe nilai/data yang akan diwakili.

Tipe Variabel;
 Tipe Variabel1, Variabel2, ..., VariabelN-1,
 VariabelN;

Contoh:

```

char Nilai_Huruf;
float Skor, Nilai_Angka;
double PI=3.14;
double D=15, R=0.5*D;
  
```

Contoh 1.3:

Algoritma:

1. Diketahui nilai $PI=3,14$
2. Diameter $D=15$ dan $R=0,5 D$
3. Tampilkan nilai Diameter dan Jari-jari
4. Hitung nilai Keliling Lingkaran ($PI \times D$) dan Luas Lingkaran ($PI \times R^2$)

Program:

```

#include < iostream.h>
void main()
{
  const double PI = 3.14;
  double D = 15, R = 0.5*D;
  cout  << "Konstanta PI : " << PI << endl
  << "Diameter          : " << D << " Cm" << endl
  << "Jari-jari          : " << R << " Cm" << endl
  << "Keliling Lingkaran: " << PI*D << " Cm" << endl
  << "Luas Lingkaran: " << PI*R*R << " Cm^2" << endl;
}
  
```

Hasil 1.3:

```

Konstanta PI      : 3.14
Diameter          : 15 Cm
Jari-jari         : 7.5 Cm
Keliling Lingkaran : 47.1 Cm
Luas Lingkaran    : 176.625 Cm^2
  
```

Penjelasan program:

Baris perintah `#include <iostream.h>` digunakan untuk memberitahu kepada komputer bahwa di dalam program terdapat perintah yang melibatkan library `iostream.h`

Baris perintah `void main()` merupakan program utama dengan jenis fungsi tidak memakai nilai kembali (`void`), artinya bahwa program tidak spesifik untuk jenis bilangan tertentu.

Baris perintah `{` berpasangan dengan `}` merupakan tanda awal dan akhir program.

Baris perintah `const double PI = 3.14;` untuk pesan ke komputer membuat konstanta `PI` diberi nilai `3.14` bertipe `double`

Baris perintah `double D = 15, R = 0.5*D;` digunakan sama seperti di atas tetapi untuk variabel `D` dengan nilai `15` dan variabel `R` dengan nilai hasil kali `0.5` dengan `D`.

Baris perintah `cout << "Konstanta PI: " << PI << endl << "Diameter: " << D << " Cm" << endl << "Jari-jari: " << R << " Cm" << endl << "Keliling Lingkaran : " << PI*D << " Cm" << endl << "Luas Lingkaran: " << PI*R*R << " Cm^2" << endl;` berturut-turut

untuk menampilkan teks string Konstanta Pi: 3,14 Diameter:
15 Jari-jari: 7,5 Keliling Lingkaran 47,25 dan Luas Lingkaran:
177,19.

Contoh 1.4:

Algoritma:

1. Masukan nilai A, B, C
2. Hitung nilai $D=b^2 - 4 A C$
3. Tampilkan nilai D

Program:

```
#include < iostream.h>
void main()
{
    // Ini deklarasi variabel
    double A, B, C;
    double D;
    // Ini untuk meminta nilai dari keyboard
    cout << "Nilai A: "; cin >> A;
    cout << "Nilai B: "; cin >> B;
    cout << "Nilai C: "; cin >> C;
    // Menghitung nilai D
    D = B*B - 4*A*C;
    // Menampilkan nilai D
    cout << "Nilai D: " << D << endl;
}
```

Hasil 1.4:

Nilai A: 2
Nilai B: 6
Nilai C: 3
Nilai D: 12

Penjelasan program:

Baris perintah `#include <iostream.h>` digunakan untuk memberitahu kepada komputer bahwa di dalam program terdapat perintah yang melibatkan library `iostream.h`

Baris perintah `void main()` merupakan program utama dengan jenis fungsi tidak memakai nilai kembali (`void`), artinya bahwa program tidak spesifik untuk jenis bilangan tertentu.

Baris perintah `{` berpasangan dengan `}` merupakan tanda awal dan akhir program.

Baris perintah `double A, B, C, D;` digunakan untuk mendeklarasikan variabel A, B, C, dan D yang bertipe `double`.

Baris perintah `cout << "Nilai A: "; cin >> A; cout << "Nilai B: "; cin >> B; cout << "Nilai C: "; cin >> C;` untuk menampilkan teks string Nilai A, B, dan C (`cout`) serta meminta nilai dari pengguna yang dibaca melalui keyboard (`cin`).

Baris perintah `D=B*B-4*A*C` untuk menghitung nilai D berdasar rumus tersebut jika diketahui nilai A, B, C, dan D.

Baris perintah `cout << "Nilai D: " << D << endl << "Diameter: " << D << endl;` untuk menampilkan teks string Nilai D: (*hasil nilai D*)

D. Operator Aritmatika

Operator sering digunakan untuk perhitungan di dalam matematika. Operator merupakan simbol yang digunakan untuk menyatakan suatu operasi. Terdapat beberapa jenis operator, yaitu: operator aritmatika, operator relasi, dan operator logika. Tabel 1.3 menunjukkan ketiga jenis operator untuk menyatakan operasi berbagai operasi tersebut.

Tabel 1.3: Jenis-jenis Operator

Operator Aritmatika		Operator Relasi		Operator Logika		Operator Bit	
*	Perkalian	>	Lebih dari	&&	Dan	&	Bit and
/	Pembagian	<	Kurang dari		Atau		Bit or
%	Sisa bagi	>=	Lebih/Sama	!	Bukan	~	Bit not
+	Penjumlahan	<=	Kurang/Sama			^	Bit xor
-	pengurangan	!=	Tidak sama			<<	Geser kiri
		==	Sama dengan			>>	Geser kanan

Contoh 1.5:

Algoritma:

1. Masukkan nilai A dan B
2. $C \leftarrow A * B$
3. Tampilkan C

Program:

```
#include < iostream.h>
void main()
{
    // Ini deklarasi variabel
    int A, B, C;
    // Ini untuk meminta nilai dari keyboard
    cout << "Nilai A: "; cin >> A;
    cout << "Nilai B: "; cin >> B;
    // Menghitung nilai C
    C = A*B;
    // Menampilkan nilai C
    cout << "Nilai C: " << C << endl
}
```

Hasil 1.5:

Nilai A: 2
Nilai B: 6
Nilai C: 12

Dengan cara yang sama dapat dibuat program untuk pembagian, sisa bagi, penjumlahan dan pengurangan atau bahkan kombinasi dari beberapa operator. Untuk membantu dalam proses operasi aritmatika tersebut dapat digunakan tanda kurung agar hierarki operasi sesuai dengan yang diinginkan.

Contoh 1.6:

Algoritma:

1. Masukkan nilai A dan B
2. $C \leftarrow (A+B)*(A-B)/B$
3. Tampilkan C

Program:

```
#include < iostream.h>
void main()
{
    // Ini deklarasi variabel
    float A, B, C;

    // Ini untuk meminta nilai dari keyboard
    cout << "Nilai A: "; cin >> A;
    cout << "Nilai B: "; cin >> B;

    // Menghitung nilai C
    C = (A+B)*(A-B)/B;

    // Menampilkan nilai C
    cout << "Nilai C: " << C << endl
}
```

Hasil 1.6:

Nilai A: 4
Nilai B: 2
Nilai C: 6 $\rightarrow (4+2)*(4-2)/2 \rightarrow 6*2/2 \rightarrow 12/2 \rightarrow 6$

Contoh 1.7:

Algoritma:

1. Masukkan nilai A dan B
2. $C \leftarrow A > B$
3. Tampilkan C

Program:

```
#include <iostream.h>
void main()
{
    int A, B;
    bool C;

    cout << "Nilai A: "; cin >> A;
    cout << "Nilai B: "; cin >> B;

    C=A>B;
    cout << "Nilai C: " << C << endl;
}
```

Hasil 1.7:

Nilai A: 3
 Nilai B: 4
 Nilai C: 0 -----> C bernilai 0 karena 3>4 nilainya FALSE atau 0

Contoh 1.8:**Algoritma:**

1. Masukan nilai A dan B
2. $C \leftarrow A \& \& B$
3. Tampilkan C

Program:

```
#include <iostream.h>
void main()
{
    int A, B;
    bool C;
    cout << "Nilai A: "; cin >> A;
    cout << "Nilai B: "; cin >> B;
    C=A&&B;
}
```

```
    cout << "Nilai C: " << C << endl;
}
```

Hasil 1.8:

Nilai A: 0
 Nilai B: 1
 Nilai C: 0 -----> C bernilai 0 karena 0 AND 1 nilainya 0 (untuk 1 AND 1 = 1)

Contoh 1.9:**Algoritma:**

1. Masukan nilai A dan B
2. $C \leftarrow A \& B$
3. Tampilkan C

Program:

```
#include <iostream.h>
void main()
{
    int A, B;
    int C;

    cout << "Nilai A: "; cin >> A;
    cout << "Nilai B: "; cin >> B;

    C=A&B;
    cout << "Nilai C: " << C << endl;
}
```

Hasil 1.9:

Nilai A: 3
 Nilai B: 7
 Nilai C: 3 -----> C bernilai 7 karena 0011 AND 0111 hasilnya 0011 (untuk 1 AND 1 = 1)

Contoh 1.10:

Algoritma:

1. Masukan nilai A dan B
2. $C \leftarrow A|B$
3. Tampilkan C

Program:

```
#include <iostream.h>
void main()
{
    int A, B;
    int C;

    cout << "Nilai A: "; cin >> A;
    cout << "Nilai B: "; cin >> B;

    C=A|B;
    cout << "Nilai C: " << C << endl;
}
```

Hasil 1.10:

Nilai A: 3
Nilai B: 7
Nilai C: 7 -----> C bernilai 7 karena **0011 AND 0111** hasilnya **0111** (untuk 0 OR 1 = 1)

BAB II PERCABANGAN PROGRAM

A. Pendahuluan

Percabangan program dimaksudkan untuk melakukan pengambilan keputusan yang diakibatkan oleh hasil suatu operasi menyebabkan lebih dari satu tindakan. Keputusan tindakan yang diambil bergantung pada hasil operasi. Jika hasil operasi BENAR maka melakukan tindakan pertama dan jika hasil operasi SALAH maka melakukan tindakan kedua. Untuk menyatakan BENAR atau SALAH digunakan operator relasi dan operator logika.

B. Pernyataan Percabangan

Perintah percabangan dalam bahasa C++ sama dengan bahasa C. Terdapat beberapa macam pernyataan percabangan, yaitu:

1. IF <kondisi> <tindakan>
2. IF <kondisi> <tindakan_1> ELSE <tindakan_2>
3. IF <kondisi_1> <tindakan_1>
ELSE IF <kondisi_2> <tindakan_2>
....
ELSE IF <kondisi_N-1> <tindakan_N-1>
ELSE <tindakan_N>
4. <Ekspresi kondisi> ? <aksi benar> : <aksi salah>
5. Switch <kondisi>
{
Case 1: <tindakan_1>;
Break;
Case 2: <tindakan_2>;
Break;
....
Case N-1: <tindakan_N-1>;
Break;
Default: <tindakan N>;
}

Penggunaan perintah-perintah tersebut di atas seringkali dikaitkan dengan operator relasi dan operator logika. Operator relasi dipakai untuk membandingkan antara suatu nilai tertentu dengan

nilai yang lain. Operator logika dipakai untuk membentuk suatu keadaan berdasarkan sebuah atau dua buah ekspresi kondisi. Perbandingan pada operator relasi dan pembentukan keadaan pada operator logika tersebut di atas digambarkan dengan kondisi.

1. Perintah IF <KONDISI> <AKSI>

Perintah *if* <kondisi> <aksi> digunakan jika hanya sebuah ekspresi kondisi yang terjadi dan sebuah aksi.

Contoh 2.1:

Permasalahan: Diketahui dua buah bilangan ingin diketahui lebih dari atau kurang dari di antara kedua bilangan tersebut.

Algoritma:

- Masukan nilai A dan B
- Bandingkan nilai A dengan nilai B
- Tampilkan hasil perbandingan

Program:

```
#include <iostream.h>
void main()
{
    int a,b;

    cout << "Masukan nilai A: ";
    cin >> a;
    cout << "Masukan nilai B: ";
    cin >> b;

    if (a>b)
        cout << "Nilai A: " << a << " lebih dari nilai B: " << b << endl;
    if (a<b)
        cout << "Nilai A: " << a << " kurang dari nilai B: " << b << endl;
}
```

Hasil program:

Masukan nilai A: 5
Masukan nilai B: 7
Nilai A: 5 kurang dari nilai B: 7

2. Perintah IF <KONDISI> <AKSI_1> ELSE <AKSI_2>

Perintah *if* <kondisi> <aksi> *else* <aksi_2> digunakan hampir sama dengan perintah 1 di atas yakni jika hanya sebuah ekspresi kondisi yang terjadi dengan dua aksi (aksi_1 dan aksi_2).

Contoh 2.2:

Permasalahan: Diketahui sebuah bilangan ingin diketahui bahwa bilangan tersebut bilangan genap atau bilangan ganjil.

Algoritma:

- Masukan bilangan X
- Tentukan bilangan X genap atau ganjil
- Tampilkan hasilnya

Program:

```
#include <iostream.h>

void main()
{
    int x;

    cout << "Masukan sembarang bilangan X: ";
    cin >> x;

    if ((a % 2)==0)
        cout << "Bilangan X: " << x << " adalah bilangan genap" << endl;
    else
        cout << "Bilangan X: " << x << " adalah bilangan ganjil" << endl;
}
```

Hasil program:

Masukan sembarang bilangan X: 5
Bilangan X: 5 adalah bilangan ganjil

3. Perintah **IF** <KONDISI_1> <AKSI_1> **IF** **ELSE**<KONDISI_N-1> <AKSI_N-1> **ELSE** <AKSI_N>

Perintah *if* <kondisi_1> <aksi_1> *if else* <aksi_N-1> <aksi_N-1> *else* <aksi_N> digunakan jika terdapat N-1 ekspresi kondisi dan N aksi (aksi_1 sampai aksi_N).

Contoh 2.3:

Permasalahan: Diketahui sebuah Nilai angka (0 s.d 100) ingin dikonversi menjadi nilai huruf (A s.d E).

Algoritma:

- Masukan Nilai angka X
- Konversi nilai angka X menjadi nilai huruf
- Tampilkan hasilnya

Program:

```
#include <iostream.h>

void main()
{
    int x;

    cout << "Masukan sembarang nilai angka X: ";
    cin >> x;

    if (x >=80)
        cout << " Nilai angka X: " << x << " Nilai huruf: " << "A" << endl;
    else if (x >=66)
        cout << " Nilai angka X: " << x << " Nilai huruf: " << "B" << endl;
```

```
else if (x >=56)
    cout << " Nilai angka X: " << x << " Nilai huruf: " << "C" << endl;
else if (x >=40)
    cout << " Nilai angka X: " << x << " Nilai huruf: " << "D" << endl;
else
    cout << " Nilai angka X: " << x << " Nilai huruf: " << "E" << endl;
}
```

Hasil program:

Masukan sembarang bilangan X: 77
Nilai angka X: 77 Nilai huruf: B

4. Perintah <EKSPRESI KONDISI> ? <AKSI_BENAR> : <AKSI_SALAH>

Perintah <ekspresi kondisi> ? <aksi_benar> : <aksi_salah> hampir sama dengan perintah *if - else*. digunakan jika terdapat hanya sebuah ekspresi kondisi dan dua aksi yaitu aksi_benar dan aksi_salah.

Contoh 2.4:

Permasalahan: Diketahui dua buah bilangan X dan Y yang ingin diketahui bilangan mana yang lebih besar.

Algoritma:

- Masukan bilangan X dan Y
- Bandingkan X dengan Y → simpan di hasil.
- Tampilkan hasil

Program:

```
#include <iostream.h>
void main()
{
```

```

int x, y, hasil;

cout << "Masukan bilangan X: ";
cin >> x;
cout << "Masukan bilangan Y: ";
cin >> y;

hasil = (x>y) ? x : y;

cout << "Antara bilangan " << x << " dan " << y <<
endl;
cout << "yang lebih besar adalah " << hasil <<
endl;
}

```

Hasil program:

Masukan bilangan X: 7
Masukan bilangan Y: 9
Antara bilangan 7 dan 9
yang lebih besar adalah 9

5. Perintah switch <variabel> case nilai_1: pernyataan_1; break; ... case nilai_n-1: pernyataan_n-1; break; default: pernyataan ke-n;

Perintah *switch* <variabel> case nilai_1: pernyataan_1; break; ... case nilai_n-1: pernyataan_n-1; break; default: pernyataan ke-n; hampir sama dengan perintah *nested if*. Perintah tersebut digunakan jika terdapat sebuah variabel yang digunakan untuk mewakili lebih dari satu nilai.

Contoh 2.5:

Permasalahan: Diketahui sebuah variabel X yang digunakan untuk mewakili pilihan ibu kota dari suatu negara.

Algoritma:

- a. Masukan bilangan X
- b. Jika $X = 1 \rightarrow$ Tampilkan pilihan 1.

c. Jika $X = N \rightarrow$ Tampilkan pilihan N

Program:

```

#include <iostream.h>
#include <string.h>
void main()
{
    int x;
    char negara[10], ibu_kota[10];

    cout << "Nama ibu kota Negara: \n";
    cout << "1. Indonesia \n";
    cout << "2. Jepang \n";
    cout << "3. Amerika \n";
    cout << "4. Inggris \n";
    cout << "\nPilihan Anda: ";
    cin >> x;

    switch (x)
    {
        case 1:
            strcpy(negara,"Indonesia");
            strcpy(ibu_kota,"Jakarta");
            break;
        case 2:
            strcpy(negara,"Jepang");
            strcpy(ibu_kota,"Tokyo");
            break;
        case 3:
            strcpy(negara,"Amerika");
            strcpy(ibu_kota,"New York");
            break;
        case 4:
            strcpy(negara,"Inggris");
            strcpy(ibu_kota,"London");
            break;
        default:
            strcpy(negara,"Tersebut");
            strcpy(ibu_kota,"Tak ada");
    }
}

```



```

)
cout << "Negara : " << negara << endl;
cout << "Ibu kota: " << ibu_kota << endl;
)

```

Hasil program:

Nama ibu kota negara:

1. Indonesia
2. Jepang
3. Amerika
4. Inggris

Pilihan Anda 3

Negara : Amerika

Ibu kota : New York

BAB III PERULANGAN PROGRAM (LOOPING)

Perulangan program merupakan suatu kondisi yang menyebabkan sebuah program melakukan tindakan yang sama/serupa berulang-ulang. Perintah program untuk keperluan tersebut terdapat tiga perintah, yaitu:

- A. *While*
- B. *Do - While*
- C. *For*

Ketiga perintah tersebut dapat digunakan secara sendiri-sendiri maupun secara bersama-sama bergantung pada programnya.

A. Perintah *While*

Perintah *while* digunakan jika sesuatu yang akan diulang-ulang dilakukan hingga suatu batas kondisi dicapai. Untuk menggunakan perintah ini, pertama harus ditentukan terlebih dahulu nilai suatu variabel pada kondisi awalnya yang akan dipakai untuk menentukan batas akhirnya. Kedua, kondisi batas akhir perulangan diberikan sebelum program melakukan tindakan yang diulang-ulang. Ketiga, pemberitahuan kepada program berupa perintah untuk menyatakan bahwa kondisi batas akhir tindakan yang diulang telah tercapai. Bentuk perintah *while* adalah sebagai berikut:

```

x=0;
while (x < 5 )
{
    x = x + 1;    // atau dapat ditulis x += 1;
    Blok program yang diulang;
}

```

Contoh Program 3.01

Permasalahan: Diinginkan untuk menampilkan nilai X dari 1 sampai N

Algoritma:

1. Masukkan bilangan untuk banyak pengulangannya $\rightarrow N$
2. Increment nilai i ($i=i+1$)
3. Tampilkan nilai i dari 1 sampai N
4. Hentikan jika nilai $i > N$

Program 3.01:

```
#include <iostream.h>

Void main()
{
    int i=0, N;

    cout << "Banyak pengulangan yang diinginkan (N):
";
    cin >> N;

    while (i<N)
    {
        i=i+1; //atau dapat ditulis i += 1;
        cout << "Pengulangan ke-" << i << endl;
    }
}
```

Hasil Program:

```
Banyak pengulangan yang diinginkan (N): 5
Pengulangan ke-1
Pengulangan ke-2
Pengulangan ke-3
Pengulangan ke-4
Pengulangan ke-5
```

Penjelasan program 3.01:

Baris perintah `#include <iostream.h>` digunakan untuk memberitahu kepada komputer agar mengikutsertakan *library iostream.h*

dalam program, karena terdapat perintah-perintah *input-output* yang menggunakan *library* tersebut.

Baris perintah `void main()` berfungsi untuk memberitahu kepada komputer bahwa program yang dibuat suatu fungsi yang diberi nama `main()` sebagai program utama. Kata `void` untuk menyatakan bahwa fungsi `main()` tersebut tidak memiliki nilai balikan tertentu sehingga tidak perlu diberi perintah `return 0` pada akhir program.

Baris perintah `{` berpasangan dengan `}` adalah untuk menandai awal dimulainya program dan akhir dari program.

Baris perintah `int i=0, N;` untuk memesan kepada komputer bahwa program yang dibuat akan melibatkan *variabel* i dan N yang jenisnya *integer* (bilangan bulat). Hal ini akan membawa konsekuensi kepada komputer pada penyediaan memori untuk menampung *variabel integer*.

Baris perintah `cout << "Banyak pengulangan yang diinginkan (N): ";` berfungsi untuk memberitahu komputer untuk menampilkan apa adanya kata-kata yang berada di dalam tanda petik ganda tersebut (Banyak pengulangan yang diinginkan (N):). Setelah perintah tersebut, tidak diikuti dengan perintah `"\n"` atau `endl` sehingga akan menyebabkan sesuatu yang diperintah selanjutnya akan ditampilkan menyambung dengan kata-kata yang ditampilkan sebelumnya.

Baris perintah `cin >> N;` digunakan untuk menyuruh kepada komputer agar meminta nilai masukan dari user yang diberikan/diketikkan melalui tombol keyboard.

Baris perintah `while (i<N)` untuk memberitahu kepada komputer bahwa baris/blok program yang berada di bawahnya agar diproses berulang selama kondisi $i < N$ terpenuhi.

Baris perintah `{` berpasangan dengan `}` untuk menandai blok program yang diproses berulang.

Baris perintah `i=i+1;` memberitahu komputer untuk melakukan operasi *increment* (menaikkan nilai). Perintah tersebut juga dapat diganti dengan perintah `i += 1;`

Baris perintah `cout << "Pengulangan ke-" << i << endl;` berfungsi untuk menampilkan kata-kata yang terdapat di dalam tanda petik ganda apa adanya, kemudian diikuti dengan menampilkan isi/nilai variabel i dan diakhiri dengan perintah `endl` untuk membuat agar baris perintah selanjutnya akan ditampilkan tidak menyambung dengan tampilan sebelumnya.

Pada program di atas di tampilkan nilai *i* yang nilai awalnya 0 diubah menjadi bernilai 1 dengan perintah *i = i + 1*. Selanjutnya, nilai *i* akan di uji kebenarannya melalui perintah *while (i<5)*. Jika kondisi bernilai salah, maka perintah menambah nilai *i* dilakukan (*i=i+1*) dan menampilkan nilai *i* melalui perintah *{cout << "Pengulangan ke-" << i << endl;}* juga dilakukan. Sebaliknya, jika kondisi bernilai benar maka perintah menampilkan nilai *i* akan dihentikan. Menampilkan nilai *i* akan berhenti jika besarnya telah melebihi nilai *N*.

B. Perintah Do - While

Perintah *do - while* penggunaannya hampir sama dengan perintah *while*, yaitu sesuatu yang akan diulang-ulang dilakukan hingga suatu batas kondisi dicapai. Syarat untuk menggunakan perintah ini juga sama bahwa, pertama harus ditentukan terlebih dahulu nilai suatu variabel pada kondisi awalnya. Kedua, kondisi batas akhir perulangan diberikan setelah program melakukan tindakan yang diulang-ulang. Ketiga, pemberitahuan kepada program berupa perintah untuk menyatakan bahwa kondisi batas akhir tindakan yang diulang telah tercapai. Bentuk perintah *do - while* adalah sebagai berikut:

```
X=0;
do
{
    X = X + 1; // atau dapat ditulis X += 1;
    Blok program yang diulang;
}
while (X < 5 )
```

Contoh Program 3.02

Permasalahan: Diinginkan untuk menampilkan nilai *X* dari 1 sampai *N* dengan step tertentu.

Algoritma:

1. Masukkan bilangan untuk banyak pengulangannya → *N*
2. Increment nilai *i* (*i=i+2*)

3. Tampilkan nilai *i* sampai *N* dengan step 2
4. Hentikan jika nilai *i*>*N*

Program 3.02:

```
#include <iostream.h>

Void main()
{
    int i=0, N;

    cout << "Banyak pengulangan yang diinginkan (N):
";
    cin >> N;

    do
    {
        i=i+2; // atau dapat ditulis i += 2;
        cout << "Nilai X ke-" << i << endl;
    }
    while (i<N);
}
```

Hasil Program:

```
Banyak pengulangan yang diinginkan (N): 10
Nilai X ke-2
Nilai X ke-4
Nilai X ke-6
Nilai X ke-8
Nilai X ke-10
```

Penjelasan program 3.02:

Baris perintah *#include <iostream.h>* digunakan untuk memberitahu kepada komputer agar mengikutsertakan library *iostream.h* dalam program, karena terdapat perintah-perintah input-output yang menggunakan library tersebut.

Baris perintah *void main()* berfungsi untuk memberitahu kepada komputer bahwa program yang dibuat suatu fungsi yang diberi

nama `main()` sebagai program utama. Kata `void` untuk menyatakan bahwa fungsi `main()` tersebut tidak memiliki nilai balikan tertentu sehingga tidak perlu diberi perintah `return 0` pada akhir program.

Baris perintah `{` berpasangan dengan `}` adalah untuk menandai awal dimulainya program dan akhir dari program.

Baris perintah `int i=0, N;` untuk memesan kepada komputer bahwa program yang dibuat akan melibatkan variabel `i` dan `N` yang jenisnya `integer` (bilangan bulat). Hal ini akan membawa konsekuensi kepada komputer pada penyediaan memori untuk menampung variabel `integer`.

Baris perintah `cout << "Banyak pengulangan yang diinginkan (N): ";` berfungsi untuk memberitahu komputer untuk menampilkan apa adanya kata-kata yang berada di dalam tanda petik ganda tersebut (Banyak pengulangan yang diinginkan (N):). Setelah perintah tersebut, tidak diikuti dengan perintah `"\n"` atau `endl` sehingga akan menyebabkan sesuatu yang diperintah selanjutnya akan ditampilkan menyambung dengan kata-kata yang ditampilkan sebelumnya.

Baris perintah `cin >> N;` digunakan untuk menyuruh kepada komputer agar meminta nilai masukan dari user yang diberikan/diketikkan melalui tombol `keyboard`.

Baris perintah `do` memberi tugas kepada komputer untuk melaksanakan perintah baris/blok program yang ada di bawahnya. Untuk blok program ditandai dengan `{` untuk awal blok dan `}` untuk akhir blok program.

Baris perintah `{` berpasangan dengan `}` untuk menandai blok program yang diproses berulang dengan pasangan perintah `do - while`.

Baris perintah `i=i+2;` memberitahu komputer untuk melakukan operasi `increment` (menaikkan nilai) dengan step 2. Perintah tersebut juga dapat diganti dengan perintah `i += 2;`

Baris perintah `cout << "Nilai X ke-" << i << endl;` berfungsi untuk menampilkan kata-kata yang terdapat di dalam tanda petik ganda apa adanya, kemudian diikuti dengan menampilkan isi/nilai variabel `i` dan diakhiri dengan perintah `endl` untuk membuat agar baris perintah selanjutnya akan ditampilkan tidak menyambung dengan tampilan sebelumnya.

Baris perintah `while (i<N)` untuk memberitahu kepada komputer bahwa baris/blok perintah program yang berada di atasnya

agar diproses berulang dan akan berhenti sampai kondisi `i<N` tidak terpenuhi.

Pada program di atas di tampilkan nilai `i` yang nilai awalnya 0 diubah menjadi bernilai 2 dengan perintah `i = 0 + 2`. Selanjutnya, nilai `i` akan di uji kebenarannya melalui perintah `while (i<10)`. Jika kondisi bernilai salah, maka perintah menambah nilai `i` dilakukan (`i=i+2`) dan menampilkan nilai `i` melalui perintah (`cout << "Nilai X ke-" << i << endl;`) juga dilakukan. Sebaliknya, jika kondisi bernilai benar maka perintah menampilkan nilai `i` akan dihentikan. Menampilkan nilai `i` akan berhenti jika besarnya telah melebihi nilai `N`.

C. Perintah *For*

Perintah `for` penggunaannya hampir sama dengan perintah `while` atau `do - while`, yaitu sesuatu yang akan diulang-ulang dilakukan hingga suatu batas nilai tertentu dicapai. Syarat untuk menggunakan perintah ini juga sama bahwa, pertama harus ditentukan terlebih dahulu nilai suatu variabel pada kondisi awalnya. Kedua, kondisi batas akhir perulangan diberikan sebelum program melakukan tindakan yang diulang-ulang. Ketiga, pemberitahuan kepada program berupa perintah untuk menyatakan bahwa kondisi batas akhir tindakan yang diulang telah tercapai. Bentuk perintah `for` adalah sebagai berikut:

```
for (i=0; i<N; i++)
{
    Blok program yang diulang;
}
```

Contoh Program 3.03

Permasalahan: Diinginkan untuk menampilkan nilai `X` dari 1 sampai `N`.

Algoritma:

1. Masukkan bilangan untuk banyak pengulangannya $\rightarrow N$
2. Increment nilai `i` (`i++` sama dengan `i=i+1`)
3. Tampilkan nilai `i` sampai `N`

4. Hentikan jika nilai $i > N$

Program 3.03:

```
#include <iostream.h>

Void main()
{
    int i, N;

    cout << "Banyak pengulangan yang diinginkan (N):
";
    cin >> N;

    for (i=0; i<N; i++)
    {
        cout <<"Nilai X ke-" << i+1 << endl;
    }
}
```

Hasil Program:

```
Banyak pengulangan yang diinginkan (N): 5
Nilai X ke-1
Nilai X ke-2
Nilai X ke-3
Nilai X ke-4
Nilai X ke-5
```

Penjelasan program 3.03:

Baris perintah `#include <iostream.h>` digunakan untuk memberitahu kepada komputer agar mengikutsertakan *library iostream.h* dalam program, karena terdapat perintah-perintah *input-output* yang menggunakan *library* tersebut.

Baris perintah `void main()` berfungsi untuk memberitahu kepada komputer bahwa program yang dibuat suatu fungsi yang diberi nama `main()` sebagai program utama. Kata `void` untuk menyatakan bahwa fungsi `main()` tersebut tidak memiliki nilai

balikan tertentu sehingga tidak perlu diberi perintah `return 0` pada akhir program.

Baris perintah `{` berpasangan dengan `}` adalah untuk menandai awal dimulainya program dan akhir dari program.

Baris perintah `int i=0, N;` untuk memesan kepada komputer bahwa program yang dibuat akan melibatkan *variabel i* dan *N* yang jenisnya *integer* (bilangan bulat). Hal ini akan membawa konsekuensi kepada komputer pada penyediaan memori untuk menampung *variabel integer*.

Baris perintah `cout << "Banyak pengulangan yang diinginkan (N): ";` berfungsi untuk memberitahu komputer untuk menampilkan apa adanya kata-kata yang berada di dalam tanda petik ganda tersebut (Banyak pengulangan yang diinginkan (N):). Setelah perintah tersebut, tidak diikuti dengan perintah `"\n"` atau `endl` sehingga akan menyebabkan sesuatu yang diperintah selanjutnya akan ditampilkan menyambung dengan kata-kata yang ditampilkan sebelumnya.

Baris perintah `cin >> N;` digunakan untuk menyuruh kepada komputer agar meminta nilai masukan dari *user* yang diberikan/diketikkan melalui tombol *keyboard*.

Baris perintah `for (i=0; i<N; i++)` untuk memberitahu kepada komputer bahwa baris/blok perintah program yang berada di bawahnya agar diproses berulang selama kondisi `i<N` terpenuhi.

Baris perintah `{` berpasangan dengan `}` untuk menandai blok perintah program yang diproses berulang.

Baris perintah `i=i+1;` memberitahu komputer untuk melakukan operasi *increment* (menaikkan nilai). Perintah tersebut juga dapat diganti dengan perintah `i += 1;`

Baris perintah `cout << "Nilai X ke-" << i+1 << endl;` berfungsi untuk menampilkan kata-kata yang terdapat di dalam tanda petik ganda apa adanya, kemudian diikuti dengan menampilkan isi/nilai *variabel i+1* (agar nilai *i* yang pertama adalah 1) dan diakhiri dengan perintah `endl` untuk membuat agar baris perintah selanjutnya akan ditampilkan tidak menyambung dengan tampilan sebelumnya.

Pada program di atas di tampilkan nilai *i* yang nilai awalnya 0 diubah menjadi bernilai 1 dengan perintah `cout << i+1`. Selanjutnya, nilai *i* akan di uji kebenarannya melalui perintah `for (i=0; i>N; i++)`.

Jika kondisi bernilai salah, maka perintah menambah nilai *i* dilakukan (*i++*) dan menampilkan nilai *i* melalui perintah (*cout << "Nilai X ke-" << i+1 << endl;*) juga dilakukan. Sebaliknya, jika kondisi bernilai benar maka perintah menampilkan nilai *i* akan dihentikan. Menampilkan nilai *i* akan berhenti jika besarnya telah melebihi nilai *N*.

Contoh-contoh Program Aplikasi

Program 3.04:

```
#include <iostream.h>

void main()
{
    int i, N;
    double x, sx=0, rx;
    cout << "Program menjumlah data\n";
    cout << "dibuat oleh Alpha Beta\n\n";

    cout << "Masukan banyak data (N): ";
    cin >> N;

    cout << endl;
    i=0;
    while (i<N)
    {
        i++;
        cout << "Masukan data X ke-" << i << ": ";
        cin >> x;
        sx += x;
    }
    rx = sx/N;
    cout << "\nJumlah X = " << sx << endl;
    cout << "Rerata X = " << rx << endl;
}
```

Program 3.05

```
#include <iostream.h>
```

```
void main()
{
    int i, N;
    char Nama[20];

    cout << "Program meminta data Nama\n";
    cout << "dibuat oleh Alpha Beta\n\n";

    cout << "Masukkan banyak data (N): ";
    cin >> N;

    cout << endl;
    i=0;
    do
    {
        i++;
        cout << "Masukan nama ke-" << i << ": ";
        cin >> Nama;
    }
    while (i<N);
}
```

Program 3.06

```
#include <iostream.h>

void main()
{
    int i, Fak=0, N, n, s1=1, s2=1, s3=1;

    cout << " Program Kombinasi\n";
    cout << "dibuat oleh Alpha Beta\n";
    cout << "\n";

    cout << "Berapa nilai N: ";
    cin >> N;
    cout << "Berapa nilai n: ";
    cin >> n;
```

```

for (i=N; i>(N-n); i--)
    s1 *= i;
for (i=n; i>=n; i--)
    s2 *= i;

Fak = s1/s2;
cout << "Kombinasi = " << Fak << endl;
)

```

Program 3.07

```

#include <iostream.h>
#include <math.h>

void main()
(
    int i, N;
    double x, sx=0, rx=0;
    cout << "Program menghitung Rerata data\n";
    cout << "dibuat oleh Alpha Beta\n\n";
    cout << "Masukan banyak data (N): ";
    cin >> N;
    cout << endl;
    for (i=0; i<N; i++)
    (
        cout << "Masukan data X ke-" << i+1 << ": ";
        cin >> x;
        sx += x;
    )
    rx = sx/N;
    cout << "\nJumlah X      = " << sx << endl;
    cout << "Rerata data      = " << rx << endl;
)

```

Terkait dengan perintah perulangan di atas, terdapat dua perintah yang dapat digunakan untuk melengkapi, yaitu: *break* dan *continue*. Perintah *break* digunakan untuk keluar dari proses *looping* walaupun kondisi untuk proses *looping* belum saatnya berhenti. Adapun perintah *continue* merupakan kebalikan dari perintah *break*,

yaitu digunakan untuk melanjutkan proses *looping* meskipun kondisi untuk proses *looping* telah selesai.

D. Array

Array merupakan kumpulan data yang memiliki tipe data sama. Kumpulan data adalah sekelompok data yang tersusun secara teratur dan diwadahi dalam sebuah variabel. Tipe data adalah jenis data seperti *int* untuk bilangan bulat *float* atau *double* untuk tipe bilangan pecahan. Bentuk penulisan untuk jenis data *array* adalah sebagai berikut:

```

Tipe_Data_Array Nama_Variabel_Array[Banyak_Data];
Tipe_Data_Array
Nama_Variabel_Array[Banyak_Data1][Banyak_Data2];
...
Tipe_Data_Array
Nama_Variabel_Array[Banyak_Data1][Banyak_Data2][Banyak_
Data3];

```

Nama_Variabel_Array merupakan variabel *array* dengan tipe *array* Tipe_Data_Array memiliki ukuran data Banyak_Data. Berikut ini contoh variabel *array*.

```

int X[N]      →   int X[50]
float Nilai[M] →   float Nilai[50]

```

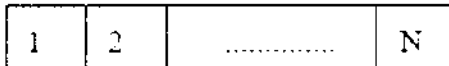
X dan Nilai merupakan variabel *array* dengan tipe data bilangan bulat untuk X dan bilangan pecahan untuk Nilai, sedangkan N dan M adalah ukuran maksimum banyaknya data yang dapat ditampung oleh variabel *array* tersebut. Misal, jika N atau M sama dengan 50, maka variabel X dan Nilai hanya mampu menampung 50 banyak data. Tipe data *array* meliputi dua macam, yaitu tipe data:

1. *Array* satu dimensi
2. *Array* multi dimensi

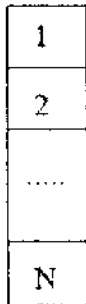
1. Array satu dimensi

Array satu dimensi merupakan *array* yang hanya memiliki sebuah elemen penyusun *array*. Variabel *array* dengan *array* satu dimensi elemen-elemennya dapat disusun membentuk satu baris data atau satu kolom data.

Array dengan elemen tersusun baris



Array dengan elemen tersusun kolom



Contoh *array* satu dimensi: `int Indeks_Prestasi_Mahasiswa[45]`

Contoh Program 3.08

Permasalahan: Jika diketahui sejumlah data *X* ingin diketahui nilai jumlah data *x* (*sx*), rata-rata data *X* (*rx*), dan simpang baku data *X* (*sd*).

Algoritma:

1. Masukkan banyaknya data *X*, simpan dalam variabel *N*
2. Untuk $i=0$, masukkan data *X* ke-1, simpan dalam variabel $x[i]$
3. Jumlahkan setiap masukan nilai data *X*, simpan dalam variabel *sx*
4. Ulangi langkah 2 selama $i < N$

5. Hitung nilai rata-rata data *X* $\Rightarrow Rx = sx/N$, simpan dalam variabel *rx*
6. Untuk $i=0$, mulai data *X* ke-1 hitung varian *X*, simpan dalam variabel *ssx*
7. Ulangi langkah 6 selama $X < N$
8. Hitung simpang baku data *X*, simpan dalam variabel *sd*
9. Tampilkan nilai Jumlah data (*sx*), rata-rata data (*rx*), dan simpang baku (*sd*).

Program komputer:

```
#include <iostream.h>
#include <math.h>

void main()
(
    int i, N;
    double x[10], sx=0, rx=0, sd=0, ssx=0;

    cout << "Program menghitung Rerata data\n";
    cout << "dibuat oleh Alpha Beta\n\n";

    cout << "Masukkan banyak data (N): ";
    cin >> N;

    cout << endl;
    for (i=0; i<N; i++)
    (
        cout << "Masukkan data X ke-" << i+1 << ": ";
        cin >> x[i];
        sx += x[i]; // atau sx = sx + x[i]
    )
    rx = sx/N;

    for (i=0; i<N; i++)
        ssx += (x[i]-rx) * (x[i]-rx);

    sd = sqrt(ssx/(N-1));
```



```

cout << "\nJumlah X = " << sx << endl;
cout << "Rerata data = " << rx << endl;
cout << "Simpang baku = " << sd << endl;

```

}

Penjelasan program 3.08

Baris perintah `#include <iostream.h>` digunakan untuk memberitahu kepada komputer agar mengikutsertakan *library iostream.h* dalam program, karena terdapat perintah-perintah *input-output* yang menggunakan *library* tersebut.

Baris perintah `#include<math.h>` diperlukan karena untuk memberitahu kepada komputer agar mengikutsertakan *library math.h*, karena di dalam program terdapat perintah yang menggunakan fasilitas yang terdapat di dalam *math.h*

Baris perintah `void main()` berfungsi untuk memberitahu kepada komputer bahwa program yang dibuat suatu fungsi yang diberi nama `main()` sebagai program utama. Kata `void` untuk menyatakan bahwa fungsi `main()` tersebut tidak memiliki nilai balikan tertentu sehingga tidak perlu diberi perintah `return 0` pada akhir program.

Baris perintah { berpasangan dengan } adalah untuk menandai awal dimulainya program dan akhir dari program.

Baris perintah `int i, N;` untuk memesan kepada komputer bahwa program yang dibuat akan melibatkan *variabel* `i` dan `N` yang jenisnya *integer* (bilangan bulat). Hal ini akan membawa konsekuensi kepada komputer pada penyediaan memori untuk menampung *variabel integer*.

Baris perintah `double x[10], sx=0, rx=0, sd=0, sxx=0;` untuk memesan kepada komputer bahwa di dalam program diidentifikasi variabel-variabel yang bertipe *double* atau bilangan pecah presisi ganda. Variabel tersebut adalah `x[10]` untuk variabel *array* satu dimensi berkapasitas maksimum 10 elemen. Variabel `sx=0` untuk menyimpan nilai jumlah `x` dan diawali dengan nilai 0. Variabel `rx=0` untuk menyimpan nilai rata-rata `x` dan diawali dengan nilai 0. Variabel `sd=0` untuk menyimpan nilai simpang baku `x` dan diawali dengan nilai 0. Variabel `sxx=0` untuk menyimpan nilai jumlah varians `x` dan diawali dengan nilai 0.

Baris perintah `cout << "Program menghitung Rerata data\n";` berfungsi untuk menampilkan nilai string atau teks Program menghitung Rerata data di layar. Sedangkan `\n` untuk memberi perintah kepada komputer agar membuat baris baru setelah menampilkan teks tersebut.

Baris perintah `cout << "dibuat oleh Alpha Beta\n\n";` sama seperti perintah di atas yakni untuk menampilkan teks string dibuat oleh Alpha Beta di layar. Kemudian perintah `\n\n` untuk membuat baris baru dua kali. Hal itu dimaksudkan agar tampilan program berikutnya berjarak satu baris kosong.

Baris perintah `cout << "Masukan banyak data (N): ";` untuk menampilkan teks string Masukan banyak data (N): di layar. Hal itu untuk memberitahu kepada pengguna bahwa komputer meminta masukan berapa banyak data (N) yang akan dihitung nilai rata-rata (`rx`), varian (`ssx`), dan simpang bakunya (`sb`).

Baris perintah `cin >> N;` untuk menyuruh komputer agar baca nilai yang diberikan pengguna melalui keyboard dan nilai tersebut ditampung pada variabel `N`.

Baris perintah `cout << endl;` digunakan agar komputer membuat baris baru pada tampilan berikutnya. Perintah tersebut sama dengan perintah `\n`.

Baris perintah `for (i=0; i<N; i++)` untuk memberitahu komputer agar melakukan proses berulang dimulai dari `i=0` sampai nilai `i<N`. Kenaikan nilai `i` dilakukan oleh perintah `i++` (*increment*).

Baris perintah { berpasangan dengan } untuk menandai awal dan akhir blok program yang diulang.

Baris perintah `cout << "Masukkan data X ke-" << i+1 << ": ";` untuk menampilkan nilai teks string Masukkan data X ke- diikuti dengan nilai `i+1` dan `:` Jika nilai `i` yang pertama 0 maka nilai `i` dengan perintah `0+1` menjadi sama dengan 1.

Baris perintah `cin >> x[i];` untuk menyuruh komputer agar baca nilai elemen data `x` yang diberikan pengguna melalui keyboard dan nilai tersebut ditampung pada variabel `x[i]`.

Baris perintah `sx += x[i];` berfungsi untuk menyuruh komputer agar menjumlahkan nilai setiap elemen masukan data `x` dan hasilnya ditampung pada variabel `sx`.

Baris perintah `rx = sx/N;` berfungsi untuk menyuruh komputer agar menghitung nilai rata-rata hasil penjumlahan elemen data `x`

dibagi dengan banyak data (N) dan hasilnya ditampung dalam variabel rx.

Baris perintah `for (i=0; i<N; i++)` untuk memberitahu komputer agar melakukan proses berulang dimulai dari `i=0` sampai nilai `i<N`. Kenaikan nilai `i` dilakukan oleh perintah `i++` (*increment*).

Baris perintah `ssx += (x[i]-rx) * (x[i]-rx);` untuk menyuruh komputer agar menghitung nilai varian `x` (jumlah dari $(X-x)^2$) dan hasilnya ditampung pada variabel `ssx`.

Baris perintah `sd = sqrt(ssx/(N-1));` agar komputer menghitung nilai simpang baku yaitu akar (`sqrt`) dari varian dan hasilnya ditampung pada variabel `sd`.

Baris perintah `cout << "\nJumlah X = " << sx << endl; cout << "Rerata data = " << rx << endl; cout << "Simpang baku = " << sd << endl;` berturut-turut adalah untuk menampilkan nilai jumlah data `x` (`sx`), rata-rata data `x` (`rx`), dan simpang baku data `x` (`sx`).

2. Array multi dimensi

Array multi dimensi merupakan array yang memiliki banyak elemen penyusun array. Variabel array dengan array multi dimensi elemen-elemennya dapat disusun membentuk sejumlah baris dan dengan sejumlah kolom data.

Array dengan elemen tersusun sejumlah baris dan sejumlah kolom

1,1	1,2	1,N
2,1	2,2	2,N
...
M,1	M,2	M,N

Contoh array multi dimensi

`int Data_X[45][50]` → nilai data X dua dimensi

`int Data_Y[45][50][40]` → nilai data X tiga dimensi

`int Data_Z[45][50][60][65]` → nilai data X empat dimensi

Contoh Program 3.09

Permasalahan: Jika diketahui matrik A dan matrik B ingin diketahui hasil jumlah matrik A dengan matrik B menjadi matrik C.

Algoritma:

1. Masukkan ordo matrik A dan matrik B, simpan dalam variabel M dan N
2. Untuk `i=0` dan `j=0`, masukkan elemen-elemen matrik A dan matrik B, simpan dalam variabel `A[i][j]` dan `B[i][j]`
3. Jumlahkan setiap elemen data matrik A dengan elemen data matrik B, simpan dalam variabel `C[i][j]`
4. Ulangi langkah 2 selama `i<M` dan `j<N`
5. Tampilkan nilai matrik C.

Program komputer:

```
#include <iostream.h>
void main()
{
    int i, j, M, N;
    int A[5][5], B[5][5], C[5][5]=0;

    cout << "Program Penjumlahan Matrik\n";
    cout << " dibuat oleh Alpha Beta\n\n";
    cout << "Masukkan Ordo Matrik: \n";
    cout << "Banyaknya baris : "; cin >> M;
    cout << "Banyaknya kolom: "; cin >> N;
    cout << endl;
    cout << "Masukan elemen matrik A:\n";

    for (i=0; i<M; i++)
        for (j=0; j<N; j++)
        {
            cout << "Masukkan elemen [" << i+1 << ", " <<
j+1 << "]: ";
            cin >> A[i][j];
        }

    cout << "Masukan elemen matrik B:\n";
```

```

for (i=0; i<M; i++)
    for (j=0; j<N; j++)
    {
        cout << "Masukkan elemen [" << i+1 << ", " <<
j+1 << "]: ";
        cin >> B[i][j];
    }

for (i=0; i<M; i++)
    for (j=0; j<N; j++)
        C[i][j] = A[i][j] + B[i][j];

Cout << "Matrik hasil penjumlahan: \n";
for (i=0; i<M; i++)
{
    for (j=0; j<N; j++)
        cout << C[i][j] << " ";
    cout << endl;
}
}

```

Penjelasan program 3.09

Baris perintah:

```

cout << "Masukkan Ordo Matrik: \n";
cout << "Banyaknya baris : "; cin >> M;
cout << "Banyaknya kolom: "; cin >> N;

```

digunakan untuk memberitahu kepada komputer agar menampilkan teks *string* Masukkan ordo matrik:, Banyak baris dan Banyak kolom diikuti membaca nilai masukan dari *keyboard* oleh pengguna untuk disimpan pada variabel M dan N.

Baris perintah

```

cout << "Masukkan elemen matrik A:\n";
for (i=0; i<M; i++)
    for (j=0; j<N; j++)
    {
        cout << "Masukkan elemen [" << i+1 << ", " << j+1 << "]: ";
        cin >> A[i][j];
    }

```

untuk menampilkan teks *string* Masukkan elemen Matrik A: kemudian menampilkan teks *string* Masukkan elemen [i,j] diikuti dengan meminta masukan nilai dari pengguna yang dibaca melalui *keyboard* dan disimpan ke variabel matrik A dan (juga sama untuk matrik B).

Baris perintah

```

for (i=0; i<M; i++)
    for (j=0; j<N; j++)
        C[i][j] = A[i][j] + B[i][j];

```

berfungsi untuk memberitahu kepada agar menjumlahkan setiap elemen matrik A dengan matrik B dan hasilnya disimpan pada variabel matrik C.

Baris perintah

```

cout << "Matrik hasil penjumlahan: \n";
for (i=0; i<M; i++)
{
    for (j=0; j<N; j++)
        cout << C[i][j] << " ";
    cout << endl;
}

```

adalah untuk menampilkan nilai teks *string* Matrik hasil penjumlahan dan menampilkan matrik C yang menampung hasil jumlah.

Catatan: Beberapa perintah program tidak dijelaskan secara detail, karena berdasar pada penjelasan-penjelasan program sebelumnya, pembaca dianggap telah mengetahui. Hal itu juga untuk menghemat kertas agar tidak menyampaikan sesuatu yang sama secara berulang-ulang.

Contoh Program 3.10

Permasalahan: Jika diketahui matrik A dan matrik B ingin diketahui hasil kali matrik A dengan matrik B menjadi matrik C.

Algoritma:

1. Masukkan ordo matrik A dan matrik B, simpan dalam variabel M dan N (syarat agar matrik dapat dikalikan maka banyak kolom matrik A = banyak baris matrik B)

2. Untuk $i=0$ dan $j=0$, masukkan elemen-elemen matrik A dan matrik B, simpan dalam variabel $A[i][j]$ dan $B[i][j]$
3. kalikan elemen data matrik A dengan elemen data matrik B, simpan dalam variabel $C[i][j]$
4. Ulangi langkah 2 selama $i < M$ dan $j < N$
5. Tampilkan nilai matrik C.

Program komputer:

```
#include <iostream.h>

void main()
{
    int i, j, k, M, N, O;
    int A[5][5], B[5][5], C[5][5];

    cout << "Program Perkalian Matrik\n";
    cout << "    dibuat oleh Alpha Beta\n\n";

    cout << "Masukkan Ordo Matrik: \n";
    cout << "Banyaknya baris mat A: "; cin >> M;
    cout << "Banyaknya kolom mat A: "; cin >> O;
    cout << "Banyaknya kolom mat B: "; cin >> N;

    cout << endl;
    cout << "Masukkan elemen matrik A:\n";
    for (i=0; i<M; i++)
        for (j=0; j<O; j++)
        {
            cout << "Masukkan elemen [" << i+1 << ", " <<
j+1 << "]: ";
            cin >> A[i][j];
        }

    cout << "\nMasukkan elemen matrik B:\n";
    for (i=0; i<O; i++)
        for (j=0; j<N; j++)
        {
            cout << "Masukkan elemen [" << i+1 << ", " <<
j+1 << "]: ";
```

```
        cin >> B[i][j];
    }

    for (i=0; i<M; i++)
        for (j=0; j<N; j++)
            C[i][j]=0;

    for (i=0; i<M; i++)
        for (j=0; j<N; j++)
            for (k=0; k<O; k++)
                C[i][j] += (A[i][k] * B[k][j]);

    cout << "Matrik hasil Perkalian Matrik: \n";
    for (i=0; i<M; i++)
    {
        for (j=0; j<N; j++)
            cout << C[i][j] << " ";
        cout << endl;
    }
}
```

Jumlah *array* dari suatu variabel yang dimasukkan dari peralatan masukkan dan belum diketahui banyaknya, dapat diketahui dengan perintah `sizeof`. Untuk menggunakan perintah tersebut dapat dilihat contoh di bawah ini.

```
#include <iostream.h>
void main()
{
    int Nilai[8]= {67, 75, 75, 70, 80, 85, 75, 70};
    cout << "Banyak elemen array dari Nilai = " <<
sizeof(Nilai)/sizeof(int) << endl;
}
```

Hasil program:

Banyak elemen array dari Nilai = 8

E. Array Karakter

Bekerja dengan karakter *array* dalam bahasa C sangat berbeda dengan bahasa komputer yang lain. Dalam bahasa C *array* karakter menjadi dasar untuk menentukan *array string*. Dengan kata lain, untuk *array* dengan elemen teks atau *string* tidak dapat diperlakukan seperti *array* karakter. Artinya, elemen *array string* merupakan kumpulan *array* karakter. Contoh *array string* "Haryanto Alfabet" maka akan dianggap *array* karakter yang membentuk *array string* tersebut.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
H	a	r	y	a	n	t	o		A	l	f	a	b	e	t	a

Jika akan memperoleh *string* tersebut maka harus digunakan dengan cara mengcopy *array* karakter tersebut.

Program:

```
#include <iostream.h>
#include <string.h>
void main()
{
    char Name[18], Nama[18] = "Haryanto Alfabet";

    cout << "Nama Anda adalah " << strcpy(Name, Nama)
<< endl;
    cout << "Banyaknya elemen array: " <<
sizeof(Name)-1 << endl;
}
```

Catatan: `strcpy` → perintah untuk mencopy data
`sizeof` → perintah untuk menghitung banyaknya elemen array.

Hasil program:

```
Nama Anda adalah Haryanto Alfabet
Banyaknya elemen array: 17
```

Jika programnya ditambahkan dengan sedikit perintah untuk mencetak dengan cara biasa yaitu dengan perintah perulangan, maka hasilnya dapat dilihat pada program berikut ini.

Program:

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

void main()
{
    char Name[18], Nama[18] = "Haryanto Alfabet";
    int i, N;
    cout << "Nama Anda adalah " << strcpy(Name, Nama)
<< endl;
    N = sizeof(Name);
    for (i=0; i<N; i++)
        cout << Nama[i];
    cout << endl;
    cout << "Banyaknya elemen array: " <<
sizeof(Name)-1 << endl;
}
```

Hasil program:

```
Nama Anda adalah Haryanto Alfabet
Haryanto Alfabet
Banyaknya elemen array: 17
```

Jika diinginkan *array* yang berupa data *string*, maka harus dibentuk dengan menggunakan cara seperti pada program berikut ini.

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

void main()
{
    char Name[20], Nama[10][20];
    int i, N;

    cout << "Masukkan Banyaknya data: "; cin >> N;
```

```

for (i=0; i<N; i++)
{
    cout << "Nama ke[" << i+1 << "]: ";
    cin >> Name;
    strcpy>Nama[i],Name);
}

for (i=0; i<N; i++)
    cout << "Nama ke[" << i+1 << "]: " <<
Nama[i] << endl;
}

```

Hasil program:

Masukkan Banyaknya data: 5 → misal diisi dengan 5 data

Nama ke[1]: Ali
Nama ke[2]: Dodi
Nama ke[3]: Fita

↳ dimasukkan satu persatu dari *keyboard*

Nama ke[4]: Nuri
Nama ke[5]: Budi

Nama ke[1]: Ali
Nama ke[2]: Dodi
Nama ke[3]: Fita

↳ ditampilkan setelah ke-5 data dimasukkan

Nama ke[4]: Nuri
Nama ke[5]: Budi

BAB 4 POINTER

A. Pengertian Pointer

Adalah suatu penunjuk yang memberitahu kepada kita tentang alamat dari suatu variabel. Dengan menggunakan *pointer* maka berdasar alamat dari suatu variabel dapat diperoleh data isi dari variabel tersebut. Untuk mendeklarasikan variabel *pointer* caranya adalah sebagai berikut:

Tipe_variabel *Variabel_pointer

Contoh:

int *pResistor;

Variabel *pointer* ditandai dengan simbol bintang (*). Nama variabel *pointer* pakai 'p' hanya untuk tanda bahwa variabel tersebut adalah *pointer* (*tidak harus diawali dengan huruf 'p'*). Cara variabel memperoleh nilai data dari suatu variabel, adalah dengan memberi simbol '&' di awal variabel yang akan diambil datanya. Bentuk penulisannya adalah sebagai berikut:

Variabel_pointer = &Variabel

Contoh:

int *pResistor, ResistorX=5600;
pResistor = &ResistorX

Variabel *pointer* pResistor dengan perintah tersebut di atas, maka variabel pResistor akan bernilai sama dengan variabel ResistorX yaitu 5600. Untuk memperjelas uraian di atas diberikan contoh program berikut ini.

Contoh Program 4.01

```
#include <iostream.h>
void main()
{
    int *pResistor, ResistorX;
    cout << "Masukkan nilai Resistor: "; cin >>
ResistorX;
    pResistor = &ResistorX;
    cout << "Nilai Resistor: " << ResistorX << endl;
    cout << "Nilai Resistor: " << *pResistor << endl;
}
```

Contoh Program 4.02

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

void main()
{
    char (*pNama)[20], Nama[10][20], Name[20];
    int *pNilai[10], Nilai[10], i, N;
    pNama = new char [10][20];

    cout << "Banyak data: "; cin >> N;

    for (i=0; i<N; i++)
    {
        cout << "Nama k[" << i+1 << "]: ";
        cin >> Name;
        strcpy(Nama[i], Name);
        strcpy(pNama[i], Name);
        cout << "Nilai      : ";
        cin >> Nilai[i];
        pNilai[i] = &Nilai[i];
    }

    cout << endl;
    for (i=0; i<N; i++)
    {
```

```
        cout << "Nama k[" << i+1 << "]: " << Nama[i]
<< endl;
        cout << "Nilai      : " << Nilai[i] << endl;
    }

    cout << endl;
    for (i=0; i<N; i++)
    {
        cout << "Nama k[" << i+1 << "]: " <<
pNama[i] << endl;
        cout << "Nilai      : " << *pNilai[i] <<
endl;
    }
}
```

BAB 5 STRUKTUR

Struktur mirip dengan *array* dalam hal fungsinya yaitu suatu variabel yang bertugas untuk menampung sejumlah nilai. Perbedaannya, jika *array* hanya untuk menampung sejumlah nilai yang memiliki tipe data yang sama, maka struktur berfungsi untuk menampung sejumlah nilai yang memiliki beragam tipe data. Contoh, struktur dapat menampung data *Nama* yang bertipe *string*, data *Gaji* yang bertipe *integer*, data *Jenis_Kelamin* yang bertipe *char*, dan lain-lain dalam satu kelompok. Struktur didefinisikan dengan bentuk sebagai berikut:

```
struct Nama_Struktur
{
    tipe_data_1 daftar variabel dengan tipe data ke_1;
    tipe_data_2 daftar variabel dengan tipe data ke_2;
    ...
    tipe_data_n daftar variabel dengan tipe data ke_n;
};
```

Setelah struktur didefinisikan, selanjutnya untuk mendeklarasikan di dalam program bahwa sekelompok atau sejumlah variabel adalah suatu struktur, maka menggunakan bentuk sebagai berikut:

```
Nama_Struktur Variabel_struktur;
```

Di dalam sebuah program, cara penggunaan/akses variabel yang telah dideklarasikan atau diidentifikasi adalah sebagai berikut:

```
Variabel_struktur.Variabelke_1;
Variabel_struktur.Variabelke_2;
...
Variabel_struktur.Variabelke_n;
```

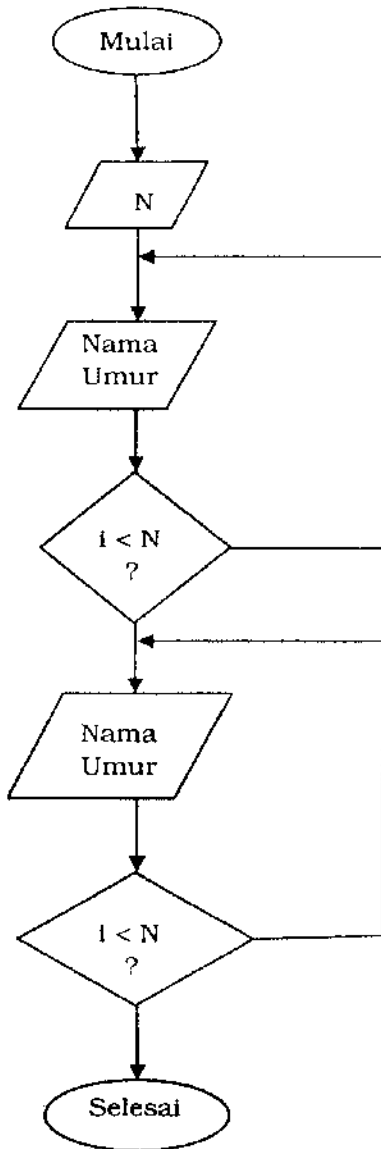
Contoh program:

Problem: dikehendaki meminta sejumlah data nama yang bertipe *string* dan data umur bertipe *integer*. Sebelum meminta data, dimasukkan terlebih dahulu banyaknya data (*N*) kemudian baru data *Nama* dan data *Umur*. Setelah semua data dimasukkan kemudian ditampilkan kembali semua data.

Algoritma:

1. Masukkan banyaknya data → *N*
2. Untuk *i = 1* sampai *i = N*,
 - a. Masukkan data nama → *Nama*
 - b. Masukkan data umur → *Umur*
3. Jika *i* masih kurang dari *N*, kembali ke langkah 2
4. Selesai.

Flowchart:



Program:

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

struct DataNama
{
    char Nama[18];
    int Umur;
};

struct DataNama Name[10], Umr[10];

void main()
{
    char Nam[18]=" ";
    int i, N;

    cout << "Banyaknya Data Nama: "; cin >> N;

    for (i=0; i<N; i++)
    {
        cout << "Nama ke[" << i+1 << "]: ";
        cin >> Nam;
        strcpy(Name[i].Nama, Nam);
        cout << "Umur      : ";
        cin >> Umr[i].Umur;
    }
    cout << endl;
    for (i=0; i<N; i++)
    {
        cout << "Nama ke[" << i+1 << "]: " <<
        Name[i].Nama << endl;
        cout << "Umur      : " << Umr[i].Umur <<
        endl;
    }

    cout << endl;
}
```

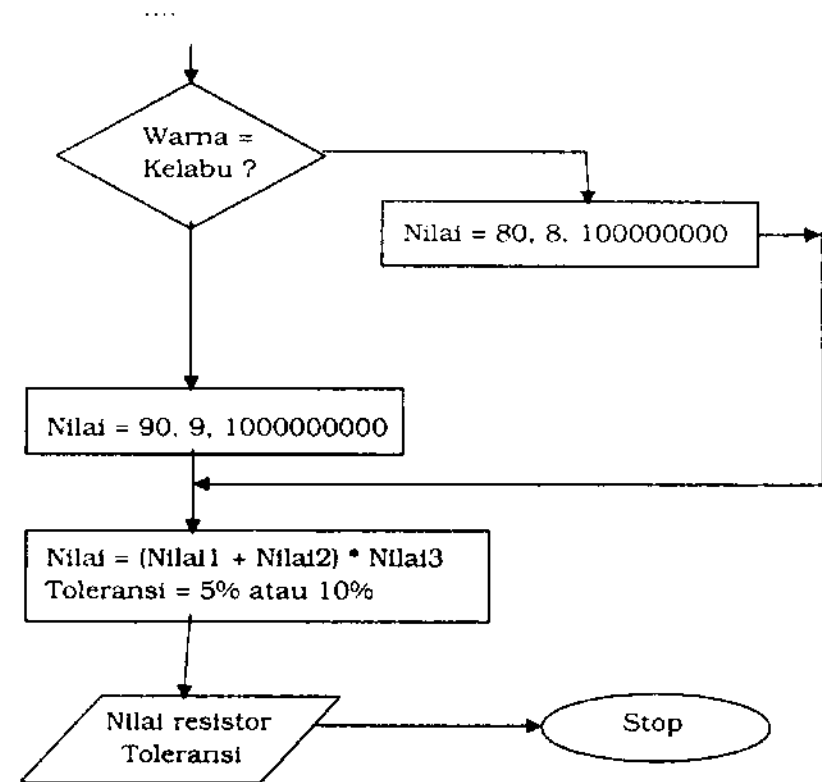
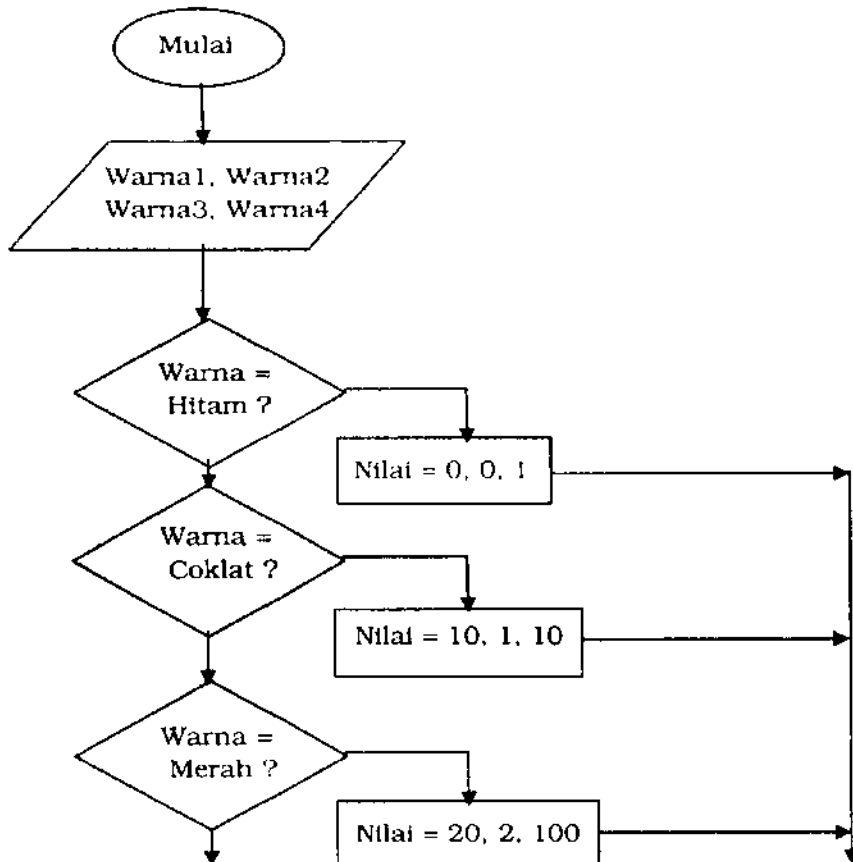
Contoh lain:

Problem: Ingin dihitung nilai resistor bila diketahui warna-warna pita pada resistor.

Algoritma:

1. Masukkan pita warna ke-1, ke-2, ke-3, dan ke-4
2. Konversi pita warna ke-1 sampai ke-4 ke nilai resistor
3. Tampilkan nilai resistor
4. Selesai

Flowchart:



Program:

```
#include <iostream.h>
#include <conio.h>
#include <string.h>
#include <ctype.h>

struct daftar
(
    char warna[15];
);
```

```

        nilai[i]=10000000;
if (strcmp(wrn[i].warna,"abuabu")==0)
    nilai[i]=100000000;
if (strcmp(wrn[i].warna,"putih")==0)
    nilai[i]=1000000000;
    )

if (i==3)
{
if (strcmp(wrn[i].warna,"emas")==0)
    nilai[i]=0.05;
if (strcmp(wrn[i].warna,"perak")==0)
    nilai[i]=0.1;
}

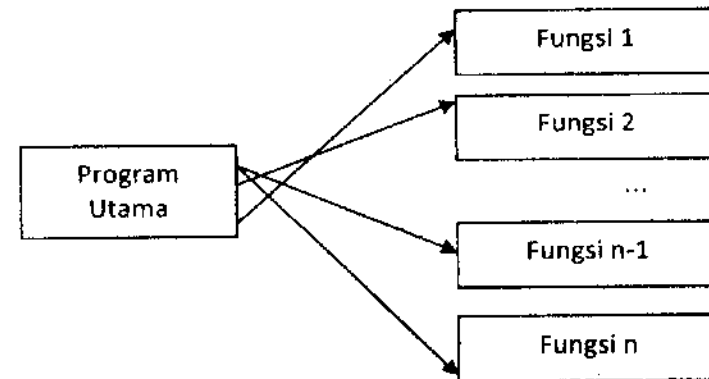
}

jadi=(nilai[0]+nilai[1])*nilai[2];
cout << "Nilai Resistor: " << jadi/1000 << " K Ohm
" << nilai[3]*100 << "%" << endl;
cout << "\nTekan tombol ENTER " << endl;
cout << "\n";
getche();
}

```

BAB 6 FUNGSI

Suatu fungsi diciptakan untuk menyederhanakan program, tujuannya agar program menjadi terstruktur, dan memudahkan untuk dilakukan pengecekan jika terjadi kesalahan. Pembuatan fungsi dalam suatu program sangatlah dianjurkan, sehingga sebuah program harus diupayakan terdiri dari sub-sub program sesuai dengan tugas masing-masing fungsi. Gambaran mengenai fungsi dan cara pembuatan fungsi adalah sebagai berikut:



Gambar 6.1:
Kedudukan Fungsi terhadap Program Utama

```

Tipe_Fungsi Nama_Fungsi(Parameter_Fungsi)
{
    Isi_Fungsi;
}

```

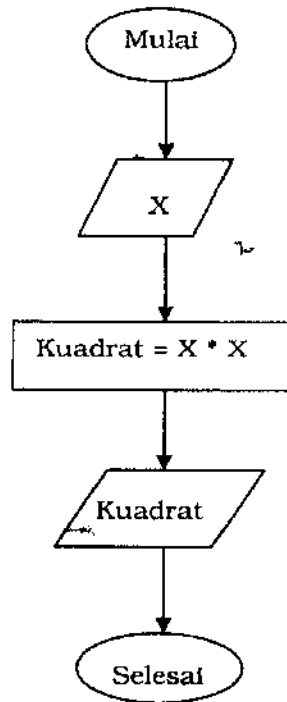
Contoh program 01:

Problem: diinginkan untuk membuat bilangan X dikuadratkan

Algoritma:

1. Masukkan nilai X
2. Nilai X dikuadratkan
3. Tampilkan hasil kuadrat X
4. Selesai

Flow chart:



Gambar 6.2:

Diagram Alir Menghitung Kuadrat

Program:

```
#include <iostream.h>
```

```
long Kuadrat(long a);

void main()
{
    long x;

    cout << "Masukkan nilai X: ";
    cin >> x;

    cout << Kuadrat(x) << endl;
}

long Kuadrat(long a)
{
    long hasil=a*a;
    return hasil;
}
```

Penjelasan program:

1. `#include <iostream.h>` untuk memberitahu komputer agar melibatkan pustaka *iostream*.
2. `long Kuadrat(long a);` perintah untuk mendeklarasikan fungsi tipe *long integer* dengan nama fungsi *Kuadrat* dan memiliki sebuah parameter variabel 'a' yang bertipe *long integer*.
3. `void main()` nama fungsi untuk program utama.
4. { perintah untuk memulai awal program yang harus berpasangan dengan tanda } di akhir program
5. `long x;` untuk mendeklarasikan variabel x yang bertipe *long integer*.
6. `cout << "Masukkan nilai X: ";` untuk menampilkan teks *Masukkan nilai X;*
7. `cin >> x;` untuk meminta nilai dari *keyboard* dan nilai yang dimasukkan akan disimpan dalam variabel x.
8. `cout << Kuadrat(x) << endl;` untuk memanggil fungsi *Kuadrat* yang telah diberi masukan nilai x dan menampilkan hasil berupa nilai x yang telah dikuadratkan.
9. `long Kuadrat(long a)` untuk menciptakan fungsi yang telah dideklarasikan di awal program. Di dalam fungsi ini dituliskan isi

fungsi, yaitu: `long hasil=a*a`; untuk menghitung $a \times a$ dan `return` hasil; untuk mengembalikan hasil perhitungan ke fungsi.

Contoh-contoh program:

1. Program: fungsi masukan proses dan tampil

```
#include <iostream.h>
#include <conio.h>
void Masukan();
void Proses();
void Tampil();

void main()
{
    Masukan();
    Proses();
    Tampil();
}

void Masukan()
{
    cout << "Bagian Ini untuk MEMASUKAN nilai" <<
endl;
    cout << "Tekan tombol ENTER " << endl;
    getch();
}

void Proses()
{
    cout << "Bagian Ini untuk MEMPROSES nilai" <<
endl;
    cout << "Tekan tombol ENTER" << endl;
    getch();
}

void Tampil()
{
```

```
    cout << "Bagian Ini untuk MENAMPILKAN nilai" <<
endl;
    cout << "Tekan tombol ENTER" << endl;
    getch();
}
```

Pada program di atas, fungsi yang dideklarasikan belum menggunakan *argument*. Fungsi yang seperti itu memang sudah dapat bekerja, tetapi belum mampu untuk mengoptimalkan program. Untuk itu sebaiknya suatu fungsi dilengkapi dengan *argument* agar program benar-benar menjadi lebih efisien.

2. Program: Meminta Nilai String

```
#include <iostream.h>
#include <string.h>
#include <conio.h>

struct daftar
{
    char pita[10];
} warn[4];

void masukan(struct daftar warna[4]);
void tampil(struct daftar warna[4]);

void main()
{
    masukan(warn);
    tampil(warn);
}

void masukan(struct daftar warna[4])
{
    int j;

    for (j=0; j<4; j++)
    {
        cout << "Masukan warna(" <<j+1<<" ) : ";
```

```

        cin >> warna[j].pita;
    }
}

void tampil(struct daftar warna[4])
{
    int j;

    for (j=0; j<4; j++)
        cout << "Masukan warna(" <<j+1<<") : " <<
warna[j].pita << endl;
}

```

3. Program: Aritmatika

```

#include <iostream.h>
#include <conio.h>

void Masukan(int *x);
void Proses(int, int, int *z);
void Tampil(int, int, int);

void main()
{
    int a, b, c;
    cout << "MASUKAN Nilai A" << endl;
    Masukan(&a);
    cout << "\nMASUKAN Nilai B" << endl;
    Masukan(&b);
    cout << "\nPROSES Nilai C=(A+B)" << endl;
    Proses(a,b,&c);
    cout << "\nTAMPILKAN Nilai C" << endl;
    Tampil(a,b,c);
    getch();
}

void Masukan(int *x)
{

```

```

    int p;
    cout << "Masukan Nilai: ";
    cin >> p;
    *x=p;
}

void Proses(int x, int y, int *z)
{
    int q;
    q=x+y;
    *z=q;
}

void Tampil(int x, int y, int z)
{
    cout <<x<<" + "<<y<<" = "<<z<<endl;
}

```

4. Program: Konversi Nilai

```

#include <iostream.h>
#include <string.h>
#include <ctype.h>
#include <conio.h>

struct daftrMhs
{
    char mhs[15];
};

struct dftNilai
{
    char nh;
};

void masukan(struct daftrMhs nama[5], int nilai[][5],
int bts[5], int *m);
void konversi(struct dftNilai value[5][5], int
nilai[][5], int bts[5], int m);
void tampil1(struct daftrMhs nama[5], int nilai[][5],
int bts[5], int m);

```

```

    int nilai[][5],
-----
"\t" << "N3\t" <<
-----
t";
1] << "\t";
-----

```

```

void tampil2(struct daftrMhs nama[5], struct dftNilai
value[][5], int bts[5], int m);

void main()
{   int p;       char sbr;       int batas[5],
    skor[5][5];
    struct daftrMhs nam[5]; struct dftNilai nh[5][5];

    cout << "Masukan Data Nama dan Nilai Mahasiswa
(ketik 0 jika selesai)\n";
    masukan(nam, skor, batas, &p);
    cout << "\nData Nama dan Nilai Mahasiswa SEBELUM
dikonversi\n";
    tampil1(nam, skor, batas, p);
    konversi(nh, skor, batas, p);
    cout << "\nData Nama dan Nilai Mahasiswa SETELAH
dikonversi\n";
    tampil2(nam, nh, batas, p);
    cout << "\nTekan sembarang tombol kemudian ENTER
!!!"; cin >> sbr;
}

void masukan(struct daftrMhs nama[5], int nilai[][5],
int bts[5], int *m)
{   int i=1, j; char nma[15];
    strcpy(nma, "0");
    while (i!=0)
    {   j=1;
        cout << "Nama Mhs ke[" << i << "]: "; cin
>> nama[i].mhs;
        strcpy(nma, nama[i].mhs);
        if (strcmp(nma, "0")==0) break;
        while (j!=0)
        {   cout << "\tNilai ke[" << j << "]: ";
            cin >> nilai[i][j];
            if (nilai[i][j]==0) break;
            j+=1;
        }
        bts[i]=j-1;
        i+=1;
    }
}

```

```

        cout << "\n";
    }
    *m=i;
}

void konversi(struct dftNilai value[5][5], int
nilai[][5], int bts[5], int m)
{   int i, j;
    for (i=0; i<m-1; i++)
        for (j=0; j<bts[i+1]; j++)
            (   if (nilai[i+1][j+1] >=80)
                value[i+1][j+1].nh='A';
                else if (nilai[i+1][j+1] >=66)
                value[i+1][j+1].nh='B';
                else if (nilai[i+1][j+1] >=55)
                value[i+1][j+1].nh='C';
                else if (nilai[i+1][j+1] >=40)
                value[i+1][j+1].nh='D';
                else value[i+1][j+1].nh='E';
            )
}

void tampil1(struct daftrMhs nama[5], int nilai[][5],
int bts[5], int m)
{   int i, j;
    cout << "-----
--\n";
    cout << "Nama\t" << "N1\t" << "N2\t" << "N3\t" <<
"N4\t" << "N5\n";
    cout << "-----
--\n";
    for (i=0; i<m-1; i++)
    {   cout << nama[i+1].mhs << "\t";
        for (j=0; j<bts[i+1]; j++)
            cout << nilai[i+1][j+1] << "\t";
        cout << endl;
    }
    cout << "-----
--\n\n";
}

```

```

void tampil2(struct daftarMhs nama[5], struct dftNilai
value[][5], int bts[5], int m)
(
    int i,j;
    cout << "-----\n";
--\n";
    cout << "Nama\t" << "N1\t" << "N2\t" << "N3\t" <<
"N4\t" << "N5\n";
    cout << "-----\n";
--\n";
    for (i=0; i<m-1; i++)
    {
        cout << nama[i+1].mhs << "\t";
        for (j=0; j<bts[i+1]; j++)
            cout << value[i+1][j+1].nh << "\t";
        cout << endl;
    }
    cout << "-----\n";
--\n\n";
)

```

BAB 7 PEMROGRAMAN BERORIENTASI OBJEK (OBJEK ORIENTED PROGRAMMING)

Hingga kini pemrograman yang telah diuraikan terdahulu adalah pemrograman tradisional dan amatiran. Pada materi terakhir, Anda rasakan bahwa bekerja dengan array, struktur, dan fungsi berparameter mulai rumit. Pada materi selanjutnya ini, yakni: OOP semoga menjadi tambahan pengalaman yang unik dan menantang untuk dipelajari lebih dalam dan menarik untuk dikembangkan. Mulai materi ini Anda akan merasakan bahwa makna sebenarnya hakekat belajar pemrograman yang professional.

Belajar OOP meliputi tiga macam materi, yaitu: *Encapsulation* (pengkapsulan), *Inheritance* (pewarisan), dan *Polymorphisme* (polimorfisme). Jangan cemas, sepertinya sangat rumit, tapi cobalah untuk awalnya barangkali kita ter bengong-bengong.

1. *Encapsulation* (Pengkapsulan)

Artinya menyembunyikan kerja kode Anda dan manghadirkan antarmuka yang dirumuskan dengan baik kepada dunia luar hanya yang diperlukan saja. Program yang baik tidak hanya efisien dan cepat eksekusi, tetapi juga mudah dipelihara dan diperbaiki.

2. *Inheritance* (Pewarisan)

Artinya kemampuan untuk meminjamkan sebagian kode untuk digunakan kembali dalam rangka untuk melakukan tugas khusus.

3. *Polymorphisme* (polimorfisme)

Artinya kemampuan untuk meminta objek yang berbeda untuk melaksanakan tugas yang sama dan membuat objek tahu untuk mencapainya dengan caranya sendiri.

Struktur program OOP terdiri dari komponen klas yang berfungsi untuk mengklasifikasikan jenis objek, misal: objek private (objek eksklusif yang nilai-nilainya tidak dapat diakses oleh objek lain), objek protekted (objek moderat yang nilai-nilainya dapat diakses oleh objek lain dengan cara yang khusus), dan objek publik (objek yang nilai-nilainya dapat diakses oleh objek lain). Di dalam

objek klas terdapat sekumpulan fungsi-fungsi untuk melakukan kerja sesuai dengan tugasnya. Cara mendeklarasikan klas adalah sebagai berikut:

```
class Nama_klas
{ tipe_objek_1:
    Fungsi_1;
    Fungsi_2;
    ...
    Fungsi_n;
tipe_objek_2:
    Fungsi_1;
    Fungsi_2;
    ...
    Fungsi_n;
...
tipe_objek_N:
    Fungsi_1;
    Fungsi_2;
    ...
    Fungsi_n;
}
```

Di samping berisi klas, program OOP juga dibangun dengan menggunakan konstruktor, yaitu fungsi khusus yang berada di dalam kelas. Nama konstruktor sama dengan nama klas. OOP memiliki konsep pada pengembangan program yang terstruktur dan hemat memori. Oleh sebab itu, dalam OOP sedapat mungkin program yang telah diciptakan akan segera di hapus kembali jika sudah tidak diperlukan. Dengan kata lain, setelah program dikonstruksi maka secepatnya didestruksi manakala sudah tidak diperlukan, sehingga hemat memori komputer. Dengan demikian, di dalam OOP teknik pemrograman yang selanjutnya dikembangkan adalah pembuatan konstruktor disusul kemudian destruktur. Berikut ini penulisan konstruktor dan destruktur.

BAB 7 PEMROGRAMAN BERORIENTASI OBJEK (OBJEK ORIENTED PROGRAMMING)

Hingga kini pemrograman yang telah diuraikan terdahulu adalah pemrograman tradisional dan amatiran. Pada materi terakhir, Anda rasakan bahwa bekerja dengan array, struktur, dan fungsi berparameter mulai rumit. Pada materi selanjutnya ini, yakni: OOP semoga menjadi tambahan pengalaman yang unik dan menantang untuk dipelajari lebih dalam dan menarik untuk dikembangkan. Mulai materi ini Anda akan merasakan bahwa makna sebenarnya hakekat belajar pemrograman yang professional.

Belajar OOP meliputi tiga macam materi, yaitu: *Encapsulation* (pengkapsulan), *Inheritance* (pewarisan), dan *Polymorphisme* (polimorfisme). Jangan cemas, sepertinya sangat rumit, tapi cobalah untuk awalnya barangkali kita terbingong-bengong.

1. *Encapsulation* (Pengkapsulan)

Artinya menyembunyikan kerja kode Anda dan manghadirkan antarmuka yang dirumuskan dengan baik kepada dunia luar hanya yang diperlukan saja. Program yang baik tidak hanya efisien dan cepat eksekusi, tetapi juga mudah dipelihara dan diperbaiki.

2. *Inheritance* (Pewarisan)

Artinya kemampuan untuk meminjamkan sebagian kode untuk digunakan kembali dalam rangka untuk melakukan tugas khusus.

3. *Polymorphisme* (polimorfisme)

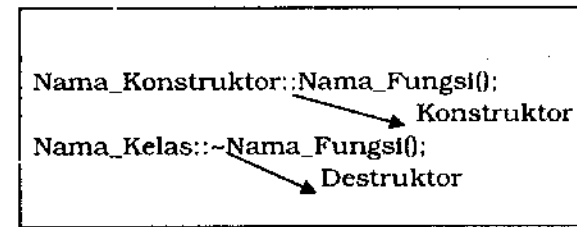
Artinya kemampuan untuk meminta objek yang berbeda untuk melaksanakan tugas yang sama dan membuat objek tahu untuk mencapainya dengan caranya sendiri.

Struktur program OOP terdiri dari komponen klas yang berfungsi untuk mengklasifikasikan jenis objek, misal: objek private (objek eksklusif yang nilai-nilainya tidak dapat diakses oleh objek lain), objek protekted (objek moderat yang nilai-nilainya dapat diakses oleh objek lain dengan cara yang khusus), dan objek publik (objek yang nilai-nilainya dapat diakses oleh objek lain). Di dalam

objek kelas terdapat sekumpulan fungsi-fungsi untuk melakukan kerja sesuai dengan tugasnya. Cara mendeklarasikan kelas adalah sebagai berikut:

```
class Nama_klas
{ tipe_objek_1:
    Fungsi_1;
    Fungsi_2;
    ...
    Fungsi_n;
    tipe_objek_2:
    Fungsi_1;
    Fungsi_2;
    ...
    Fungsi_n;
    ....
    tipe_objek_N:
    Fungsi_1;
    Fungsi_2;
    ...
    Fungsi_n;
}
```

Di samping berisi kelas, program OOP juga dibangun dengan menggunakan konstruktor, yaitu fungsi khusus yang berada di dalam kelas. Nama konstruktor sama dengan nama kelas. OOP memiliki konsep pada pengembangan program yang terstruktur dan hemat memori. Oleh sebab itu, dalam OOP sedapat mungkin program yang telah diciptakan akan segera di hapus kembali jika sudah tidak diperlukan. Dengan kata lain, setelah program dikonstruksi maka secepatnya didestruksi manakala sudah tidak diperlukan, sehingga hemat memori komputer. Dengan demikian, di dalam OOP teknik pemrograman yang selanjutnya dikembangkan adalah pembuatan konstruktor disusul kemudian destruktur. Berikut ini penulisan konstruktor dan destruktur.



Contoh program 01:

Problem: Dikehendaki untuk meminta data Mahasiswa berupa NIM, NAMA, dan JENIS_KELAMIN dengan menggunakan program yang dikembangkan menggunakan teknik OOP.

Algoritma:

1. Masukkan data nomor induk mahasiswa → NIM
2. Masukkan data nama mahasiswa → NAMA
3. Masukkan data jenis kelamin → JNSKEL
4. Selesai

Program:

```
#include <iostream.h>
#include <string.h>
#include <ctype.h>

class Mahasiswa
{
private:
    int nim;
    char nama[25];
    char jnskel[5];
public:
    void inisialisasi(int nim, char nama[], char jnskel[]);
    void isinim(int nim);
    void isinama(char nama[]);
    void isijskel(char jnskel[]);
}
```

```

int perolehnim();
char *perolehnama();
char *perolehjnskel();
};

void main()
{
    int nmr=0;
    char nam[25];
    char jns[5];
    Mahasiswa MhsMekatronika;
    MhsMekatronika.inisialisasi(nmr, nam, jns);

    cout << "Mahasiswa Mekatronika: " << endl;
    cout << "-----\n";
    cout << "NIM      : " << MhsMekatronika.perolehnim()
<< endl;
    cout << "Nama      : " <<
MhsMekatronika.perolehnama() << endl;
    cout << "Kelamin: " <<
MhsMekatronika.perolehjnskel()<< endl;
}

void Mahasiswa::inisialisasi(int nim, char nama[], char
jnskel[])
{
    cout << "NIM : "; cin >> Mahasiswa::nim;
    cout << "Nama: "; cin >> Mahasiswa::nama;
    cout << "Jenis kelamin laki-laki (T/F): "; cin >>
Mahasiswa::jnskel;
}

void Mahasiswa::isinim(int nim)
{    Mahasiswa::nim = nim;    }

void Mahasiswa::isinama(char nama[])
{    strcpy(Mahasiswa::nama, nama);    }

void Mahasiswa::isijskel(char jnskel[])
{    strcpy(Mahasiswa::jnskel, jnskel);    }

```

```

int Mahasiswa::perolehnim()
{    return Mahasiswa::nim;    }

char *Mahasiswa::perolehnama()
{    return Mahasiswa::nama;    }

char *Mahasiswa::perolehjnskel()
{    return Mahasiswa::jnskel;    }

```

Penjelasan program:

1. Perintah program di bawah ini adalah untuk melibatkan pustaka yang diperlukan dalam program.

```

#include <iostream.h>
#include <string.h>
#include <ctype.h>

```

2. Deklarasi kelas dituliskan sebagai berikut:

```

class Mahasiswa
{
private:
    int nim;
    char nama[25];
    char jnskel[5];
public:
    void inisialisasi(int nim, char nama[], char jnskel[]);
    void isinim(int nim);
    void isinama(char nama[]);
    void isijskel(char jnskel[]);
    int perolehnim();
    char *perolehnama();
    char *perolehjnskel();
};

```

Klas di atas memiliki dua objek yang bertipe private dan publik. Masing-masing objek memiliki beberapa fungsi yang bekerja sesuai dengan tugasnya.

3. Program di bawah ini merupakan fungsi utama. Di fungsi utama, di samping diidentifikasi kebutuhan variabel, juga digunakan untuk memanggil atau mengakses sub-sub fungsi.

```
void main()
{
    int nmr=0;
    char nam[25];
    char jns[5];
    Mahasiswa MhsMekatronika;
    MhsMekatronika.inisialisasi(nmr, nam, jns);

    cout << "Mahasiswa Mekatronika: " << endl;
    cout << "-----\n";
    cout << "NIM   : " << MhsMekatronika.perolehnim() << endl;
    cout << "Nama   : " << MhsMekatronika.perolehnama() <<
endl;
    cout << "Kelamin: " << MhsMekatronika.perolehjnskel()<<
endl;
}
```

4. Program di bawah ini merupakan penciptaan objek yang telah dideklarasikan pada bagian klas di atas.

```
void Mahasiswa::inisialisasi(int nim, char nama[], char jnskel[])
{
    cout << "NIM : "; cin >> Mahasiswa::nim;
    cout << "Nama: "; cin >> Mahasiswa::nama;
    cout << "Jenis kelamin Laki-laki (T/F): "; cin >>
Mahasiswa::jnskel;
}

void Mahasiswa::isinim(int nim)
{ Mahasiswa::nim = nim; }

void Mahasiswa::isinama(char nama[])
```

```
{ strcpy(Mahasiswa::nama, nama); }

void Mahasiswa::isijnskel(char jnskel[])
{ strcpy(Mahasiswa::jnskel, jnskel); }

int Mahasiswa::perolehnim()
{ return Mahasiswa::nim; }

char *Mahasiswa::perolehnama()
{ return Mahasiswa::nama; }

char *Mahasiswa::perolehjnskel()
{ return Mahasiswa::jnskel; }
```

Contoh-contoh program:

1. Program: Meminta Identitas

```
#include <iostream.h>
class Kontak
{
public:
    char nama[25];
    int umur;
    char phone[12];
    void TambahKontak(char nama[], int umur, char
phone[]);
    void Tampil(char nama[], int umur, char phone[]);
};
const int MaksKontak=10;
void main()
{
    Kontak daftar[MaksKontak];
    char nam[25], phon[12];
    int umr=0, i=0, j=0;
    char jwb;
    while (i < MaksKontak)
    {
        cout << "Tambah kontak [Y]es [N]o: ";
```

```

        cin >> jwb;
        if (jwb=='N' || jwb=='n')
            break;
        daftar[i].TambahKontak(nam, umr, phon);
        i+=1;
    }
    while (j<i)
    {
        daftar[j].Tampil(nam, umr, phon);
        j+=1;
    }
}

void Kontak::TambahKontak(char nama[], int umur, char
phone[])
{
    cout << "Nama : "; cin >> Kontak::nama;
    cout << "Umur : "; cin >> Kontak::umur;
    cout << "Phone: "; cin >> Kontak::phone;
}

void Kontak::Tampil(char nama[], int umur, char phone[])
{
    cout << "Nama : " << Kontak::nama << endl;
    cout << "Umur : " << Kontak::umur << endl;
    cout << "Phone: " << Kontak::phone << endl;
}

```

2. Program: Meminta Identitas (Versi Lain)

```

#include <iostream.h>
#include <string.h>

class Kontak
{
private:
    char nama[25];
    int umur;
    char phone[12];
public:

```

```

        void TambahKontak(char nama[], int umur, char
phone[]);
        void Tampil(char nama[], int umur, char phone[]);
        void BacaNama(char nama[]);
        void BacaUmur(int umur);
        void BacaPhone(char phone[]);
        char *DapatNama();
        int DapatUmur();
        char *DapatPhone();
    };
    const int MaksKontak=10;

void main()
{
    Kontak daftar[MaksKontak];
    char nam[25], phon[12];
    int umr=0, i=0, j=0;
    char jwb;
    while (i < MaksKontak)
    {
        cout << "Tambah kontak [Y]es [N]o: ";
        cin >> jwb;
        if (jwb=='N' || jwb=='n')
            break;
        daftar[i].TambahKontak(nam, umr, phon);
        i+=1;
    }
    cout << "-----\n";
    cout << "Nama\t" << "Umur\t" << "Phone\n";
    cout << "-----\n";
    while (j<i)
    {
        daftar[j].Tampil(nam, umr, phon);
        j+=1;
    }
    cout << "-----\n";
}

void Kontak::TambahKontak(char nama[], int umur, char
phone[])
{
    cout << "Nama : "; cin >> Kontak::nama;

```

```

        cout << "Umur : "; cin >> Kontak::umur;
        cout << "Phone: "; cin >> Kontak::phone;
    }

void Kontak::Tampil(char nama[], int umur, char phone[])
{
    cout << DapatNama() << "\t";
    cout << DapatUmur() << "\t";
    cout << DapatPhone() << endl;
}

void Kontak::BacaNama(char nama[])
{
    strcpy(Kontak::nama,nama);
}

void Kontak::BacaPhone(char phone[])
{
    strcpy(Kontak::phone,phone);
}

void Kontak::BacaUmur(int umur)
{
    Kontak::umur=umur;
}

char *Kontak::DapatNama()
{
    return Kontak::nama;
}

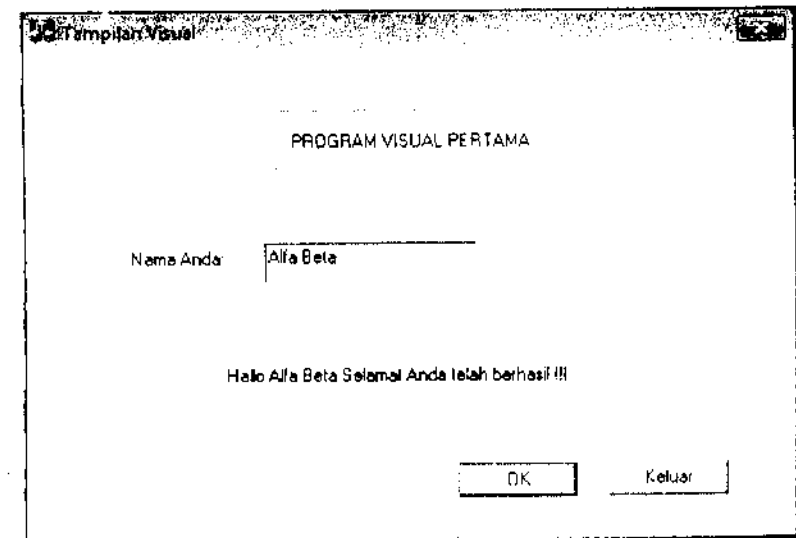
char *Kontak::DapatPhone()
{
    return Kontak::phone;
}

int Kontak::DapatUmur()
{
    return Kontak::umur;
}

```

BAB 8 PEMROGRAMAN VISUAL

Pemrograman yang telah dilakukan pada bab-bab terdahulu tampilan hasilnya berada pada lingkungan layar sistem operasi (*non windows*). Pada pemrograman visual ini akan dibahas pembuatan program yang memiliki tampilan visual di lingkungan *windows*. Berikut ini contoh tampilan hasil program dengan model visual.



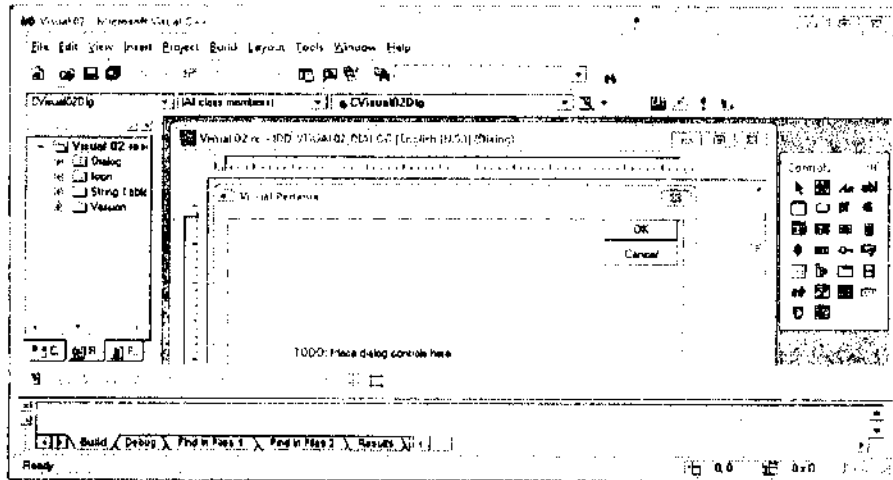
Gambar 8.1:
Contoh Tampilan Program Visual

Tampilan di atas, pertama-tama dimasukkan nama "Alfa Beta" pada tempat yang telah disediakan. Kemudian dengan meng-klik tombol "OK" akan tampil teks string "Halo Alfa Beta Selamat Anda telah berhasil !!!". Untuk keluar atau mengakhiri tampilan dengan meng-klik tombol "Keluar".

A. Memulai Menyiapkan Program Visual

Ampilan program seperti pada Gambar 8.1. untuk memulainya diperlukan langkah-langkah sebagai berikut:

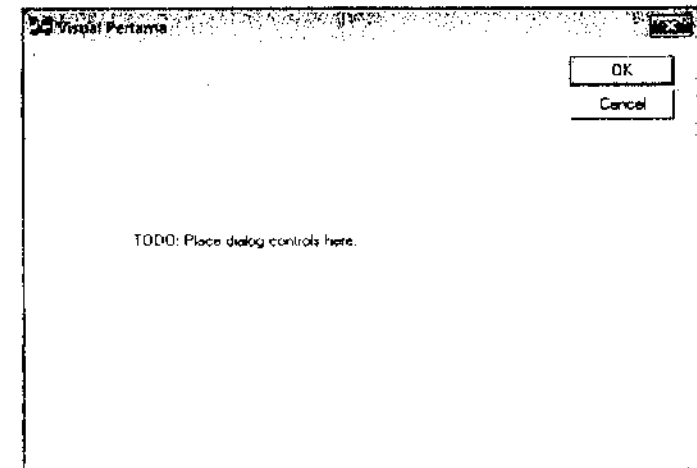
1. Panggil program Visual ++
2. Klik *File* kemudian pilih *New*
3. Pada menu *Project* pilih *MFC App Wizard (exe)*
4. Isi *Project Name* dengan nama proyek, misal: Visual 01
5. Tentukan lokasi untuk tempat penyimpanan hasil akhir proyek
6. Klik tombol *OK*
7. Klik tombol radio *Dialog Base* kemudian klik *Next*
8. Klik pilihan *Active Controls* agar tanda centang hilang pada pilihan tersebut
9. Klik *Next* kemudian pilih radio *As a statically linked library*.
10. Klik *Next*
11. Klik *Finish*
12. Klik *OK* maka akan tampil program visual C yang siap untuk pembuatan proyek seperti pada Gambar 8.2.



Gambar 8.2:
Tampilan Awal Visual C untuk Pembuatan Proyek

Program di atas meskipun belum ada program yang dibuat, tetapi sudah dapat diuji hasilnya. Untuk mengetahuinya, harus

dilakukan pengkompilasian dengan meng-klik Menu *Build* dan pilih *Rebuild All*. Jika program berhasil dikompilasi dan tidak ada kesalahan, maka pada bagian bawah akan ada informasi *0 error(s) 0 warning(s)*. Setelah berhasil dikompilasi selanjutnya di eksekusi dengan perintah *Build* kemudian *Execute program*. Hasilnya akan terlihat seperti pada Gambar 8.3.



Gambar 8.3:
Hasil Tampilan Program

Langkah-langkah yang dilakukan pada bagian A di atas harus selalu dilakukan setiap kali akan membuat program visual yang baru. Oleh sebab itu, bagi Anda yang baru memulai belajar visual C++ dengan teknik pengembangan program di lingkungan windows, maka langkah-langkah tersebut di atas harus hafal. Tapi jangan khawatir seiring dengan jam terbang Anda dalam membuat proyek, maka Anda akan hafal dengan sendirinya.

B. Membuat Program Visual

Bagian berikut ini akan dimulai pembuatan proyek pertama membuat program yang dapat bekerja di lingkungan windows. Proyek pertama yang perlu dibuat adalah program yang melibatkan komponen *Button*, *Static text*, dan *Edit text*. Langkah-langkah yang perlu dilakukan adalah analisis kebutuhan komponen, mendisain

tampilan program, menentukan variabel-variabel dan fungsi, mengembangkan kode program, pengujian program.

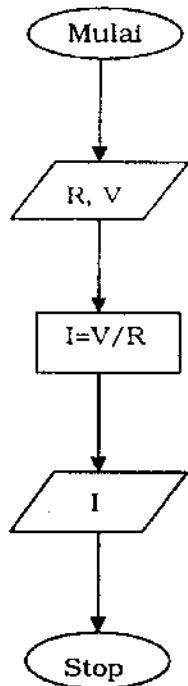
Contoh program:

Problem: Ingin diketahui nilai Arus (I) yang mengalir pada suatu rangkaian arus searah dengan beban tunggal jika nilai Resistor (R) dan nilai Tegangan (V) yang bekerja pada rangkaian diketahui.

Algoritma:

1. Masukkan nilai resistor dan tegangan $\rightarrow R, V$
2. Hitung nilai $I = V/R$
3. Selesai

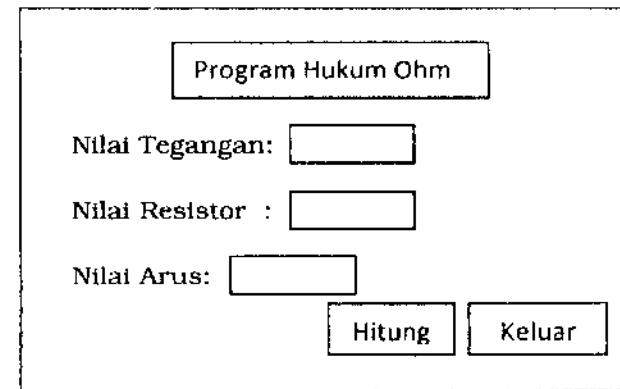
Flow chart:



Gambar 8.4:

Diagram Alir Program Menghitung Arus

Disain Prototipe Program



Pembuatan kode program:

Pada disain prototipe di atas Nilai arus akan diketahui jika nilai tegangan dan nilai arus telah diketahui. Untuk menghitung nilai arus dilakukan oleh button hitung saat di-klik, maka berarti program yang dibuat harus ditulis pada fungsi hitung. Button keluar berfungsi untuk mengakhiri program.

1. Langkah pertama adalah meletakkan komponen-komponen 3 *static text*, 3 *edit text*, dan 2 *button* dari *controls box* ke dalam *windows* program dengan cara *men-drag* komponen-komponen tersebut.
2. Mengatur teks agar kata-kata yang tampak seperti yang kita harapkan. Untuk itu lakukan dengan cara meng-klik kanan pada komponen yang akan diatur kemudian pilih *properties*. Misal pada komponen *static* yang akan diganti menjadi PROGRAM HUKUM OHM.
 - a. Ganti teks *static* pada *Caption* dengan teks PROGRAM HUKUM OHM
 - b. Klik *style* kemudian atur *align text* menjadi *center*
 - c. Klik *center vertically* dan *sunken*
 - d. Klik tanda X untuk mengakhiri

Dengan cara yang sama hal itu dapat dilakukan untuk teks *static* yang lain agar menjadi: *Nilai Tegangan*, *Nilai Resistor*, dan *Nilai Arus*. Hal yang sama juga dapat dilakukan kepada *button* yang semula OK menjadi *Hitung* dan *Cancel* menjadi *Keluar*.

3. Untuk keperluan pembuatan program maka yang berperan adalah komponen *edit text* dan *button* Hitung. Oleh sebab itu, komponen *edit text* harus diset untuk variabel-variabel program. Caranya adalah sebagai berikut:
 - a. Klik kanan *edit text* 1 yang berada di samping Nilai tegangan kemudian pilih *properties*.
 - b. Pastikan pada pilihan *General* atur agar pada ID kata IDC_EDIT1 diganti menjadi IDC_TEGANGAN.
 - c. Klik tanda X untuk mengakhiri

Dengan cara yang sama maka komponen *edit text* yang lain dapat diganti menjadi IDC_RESISTOR DAN IDC_ARUS.

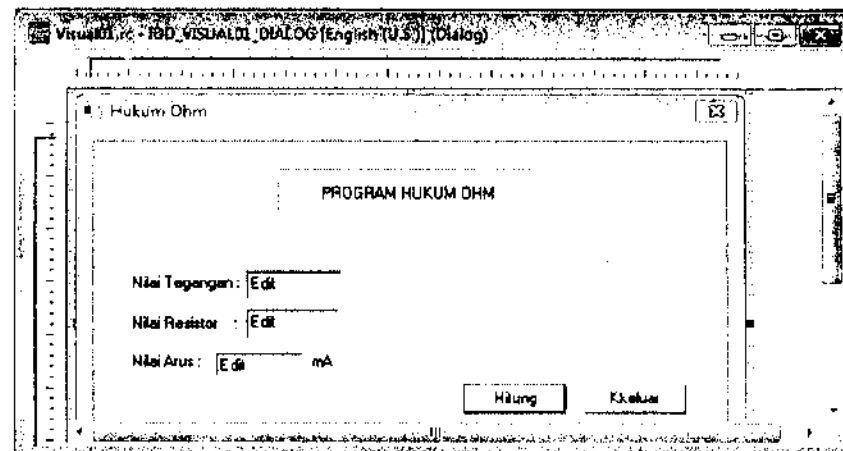
4. Tugas selanjutnya adalah membuat variabel dan menulis fungsi program pada *button* Hitung. Langkah-langkahnya adalah sebagai berikut:
 - a. Klik kanan pada *button* Hitung kemudian pilih *member variables*
 - b. Klik IDC_ARUS pada *IDs Controls* dan klik *Add Variables*
 - c. Ketik *m_I* pada kotak *Member variable name* dan pilih *float* pada *Variabel type*
 - d. Klik OK

Dengan cara yang sama lakukan pada IDC_RESISTOR dan IDC_TEGANGAN sehingga diperoleh *m_R*, *float* dan *m_V* *float*.

5. Klik *Message maps* kemudian pilih IDOK dan klik BN_CLICKED pada *Messages* dilanjutkan klik *Add function*, OK dan klik *Edit Code*
6. Ketik program sebagai berikut:

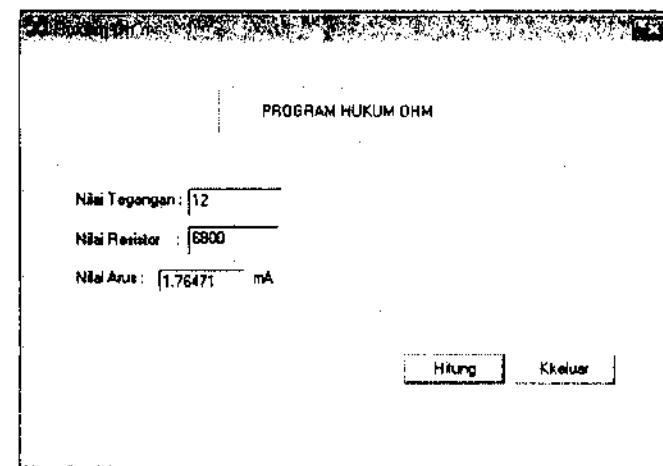

```
void CVisual01Dlg::OnOK()
{
    UpdateData(TRUE);
    m_I = m_V/m_R*1000;
    UpdateData(FALSE);
}

```
7. Klik tanda X untuk mengakhiri



Gambar 8.5:
Hasil Disain Tampilan Program

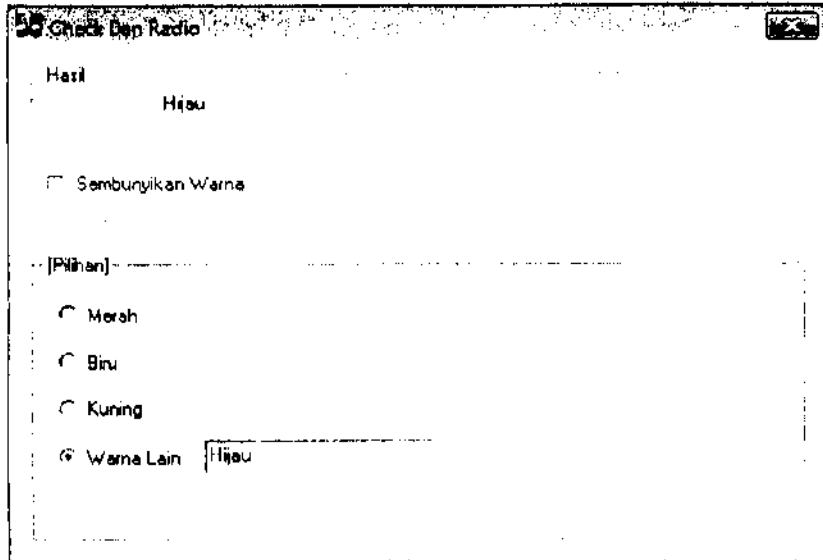
8. Jika disain tampilan dan kode program telah ditulis dengan lengkap, maka langkah selanjutnya adalah menguji keberhasilan program tersebut dengan melakukan kompilasi (*Build*) dan mengeksekusi program (*execute program*). Gambar 8.6 adalah hasil tampilan setelah dieksekusi:



Gambar 8.6:
Hasil Jadi Tampilan Program

C. Radio Button dan Check Box

Program berikut adalah persoalan yang akan dikembangkan untuk menampilkan contoh penggunaan *radio button*, *check box* dan *group box*. Tampilan jadi yang akan dibuat adalah sebagai berikut:



Gambar 8.7:
Hasil Jadi Tampilan Program

Untuk keperluan tersebut diperlukan komponen-komponen dua *group box*, sebuah *check box*, dan empat *radio button*. Komponen *group box* yang pertama untuk mengelompokkan *static text1* dengan *check box*. Komponen *group box* kedua untuk mengelompokkan empat *radio button* dan sebuah *edit text*. *Static text1* digunakan untuk menampilkan pilihan dan atau isian warna pada *radio button* dan *edit text*. Sedangkan *check box* digunakan untuk menampakkan dan menyembunyikan tampilan pada *static text1*.

Berdasar gambaran program di atas selanjutnya ambil komponen-komponen objek pada *control box* untuk diletakkan dan

ditatur pada jendela tampilan seperti pada Gambar 8.7. Kemudian diidentifikasi variabel-variabel dan fungsi-fungsi yang diperlukan pada program tersebut. Berikut ini penulisan kode sumber untuk program tersebut.





1. Kode program fungsi pada *Check box* untuk menampilkan dan menyembunyikan teks.

```
void CCheckDanRadioDlg::OnWarnaCheck1()  
{  
    UpdateData(TRUE);  
    if (m_WarnaCheck==TRUE)  
        GetDlgItem(IDC_WARNA_STATIC)-  
>ShowWindow(SW_HIDE);  
    else  
        GetDlgItem(IDC_WARNA_STATIC)-  
>ShowWindow(SW_SHOW);  
    UpdateData(FALSE);  
}
```

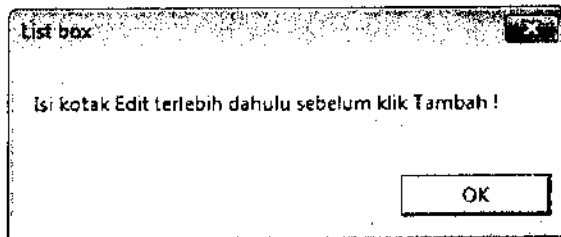
2. Kode program untuk menampilkan pilihan yang di-klik dari *radio button*

```
void CCheckDanRadioDlg::OnMerahRadiol1()  
{  
    UpdateData(TRUE);  
    switch(m_MerahRadiol1)  
    {  
        case 0:  
            m_WarnaStatic="Merah";  
            break;  
        case 1:  
            m_WarnaStatic="Biru";  
            break;  
        case 2:  
            m_WarnaStatic="Kuning";  
            break;  
        case 3:  
            m_WarnaStatic="Warna Lain";
```

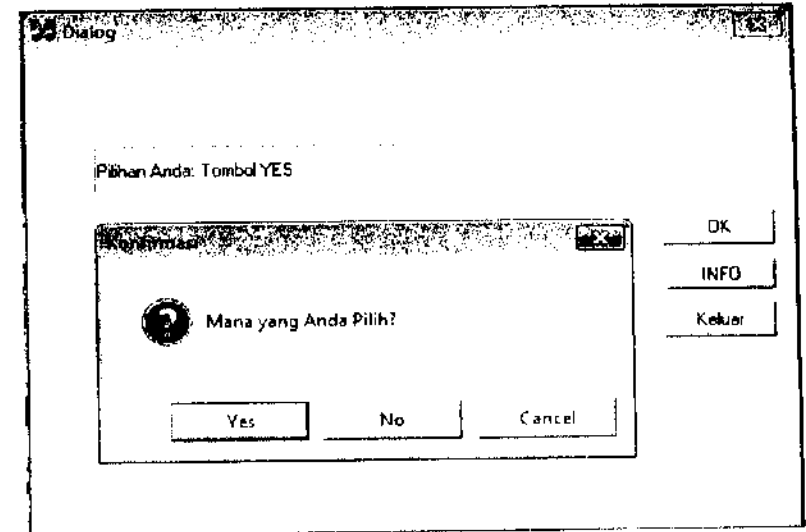
Tabel: Datar Kontrol dan *Icon*

Kontrol	Gambar	Fungsi
MB_ICONEXCLAMATION		Peringatan
MB_ICONINFORMATION		Informasi
MB_ICONQUESTION		Pertanyaan
MB_ICONSTOP		Kesalahan

Contoh tampilan dialog box:

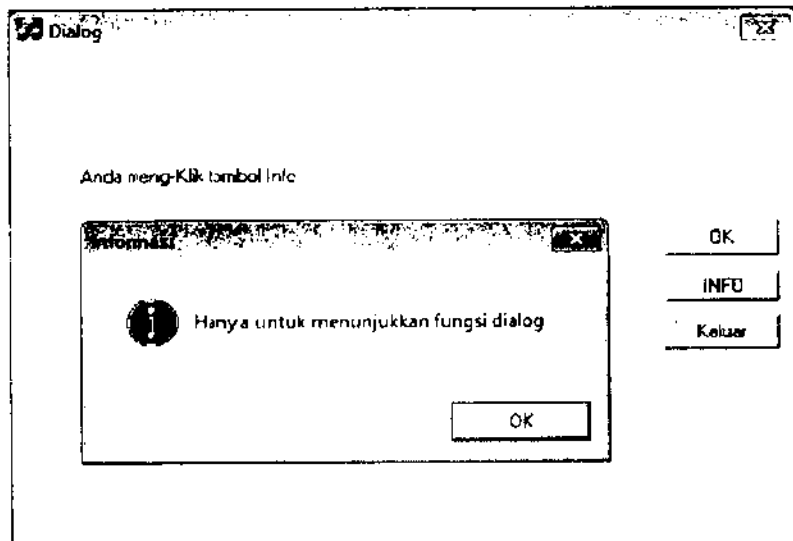


Gambar 8.11 contoh permasalahan yang akan dibuat programnya, yaitu: bila di-klik tombol Yes, muncul pesan "Tombol Yes yang di Klik", bila di-klik tombol No, muncul pesan "Tombol No yang di Klik", dan bila di-klik tombol *Cancel*, muncul pesan "Tombol *Cancel* yang di Klik". Kemudian jika tombol *INFO* yang di-klik maka akan muncul pesan "Anda meng-klik tombol Info".



Gambar 8.11:
Tampilan Program Dialog

Langkah yang pertama harus dilakukan adalah mengidentifikasi bahwa untuk mengembangkan program tersebut diperlukan sebuah objek *static text*, dan tiga buah button. Objek *static text* digunakan untuk menampilkan informasi mengenai tombol apa yang ditekan. Sedangkan tombol button digunakan untuk memunculkan *dialog box*. Button *OK* untuk memunculkan *dialog box* dengan tiga pilihan tombol yaitu: tombol *YES*, *NO* dan *CANCEL*. Button *Info* untuk memunculkan dialog informasi, dan button 'Keluar' untuk keluar dari program.



Gambar 8.12:
Tampilan Program Dialog

Setelah diidentifikasi kebutuhan objek, selanjutnya yang diperlukan adalah identifikasi variabel dan fungsi. Dalam program tersebut terdapat sebuah variabel `m_info` untuk menampilkan teks di *static text*. Terdapat dua fungsi yang dibuat, yaitu fungsi `OK` dan fungsi `Info`. Pembuatan variabel dan fungsi dilakukan di menu *view*, *class wizard*. Program secara keseluruhan adalah sebagai berikut:

1. Program Fungsi Tombol OK

Program ini memerlukan variabel pilih yang bertipe *integer*. Variabel ini berfungsi untuk menampung ID yang berkaitan dengan tombol yang ditekan pada kotak dialog. Kotak dialog yang dimunculkan memiliki tiga buah tombol, yaitu `YES`, `NO`, dan `CANCEL`. Untuk memunculkan ketiga tombol tersebut digunakan perintah `MB_YESNOCANCEL`. Adapun jenis icon yang dimunculkan adalah pertanyaan. Hal itu dimunculkan dengan perintah `MB_ICONQUESTION`.

Penekanan terhadap salah satu tombol menyebabkan variabel pilih menyimpan nilai yang menunjukkan ID *message box*. Bila yang ditekan tombol `YES` maka variabel pilih akan berisi nilai `IDYES`. Bila

yang ditekan tombol `NO` maka variabel pilih akan berisi nilai `IDNO`. Bila yang ditekan tombol `CANCEL` maka variabel pilih akan berisi nilai `IDCANCEL`.

```
void CDialogDlg::OnButton1()
{
    int pilih = MessageBox("Mana yang Anda
        Pilih?", "Konfirmasi", MB_YESNOCANCEL |
        MB_ICONQUESTION);
    switch (pilih)
    {
        case IDYES:
            m_Info = "Pilihan Anda: Tombol YES";
            break;
        case IDNO:
            m_Info = "Pilihan Anda: Tombol NO";
            break;
        case IDCANCEL:
            m_Info = "Pilihan Anda: Tombol CANCEL";
            break;
    }
    UpdateData(FALSE);
}
```

Pada perintah selanjutnya menyebutkan bila yang ditekan tombol `YES` maka objek *static text* melalui variabel `m_Info` akan diberi nilai *string* "Pilihan Anda: Tombol YES". Bila yang ditekan tombol `NO` maka objek *static text* melalui variabel `m_Info` akan diberi nilai *string* "Pilihan Anda: Tombol NO". Bila yang ditekan tombol `CANCEL` maka objek *static text* melalui variabel `m_Info` akan diberi nilai *string* "Pilihan Anda: Tombol CANCEL".

2. Program Fungsi Tombol INFO

Program tombol fungsi `Info` digunakan untuk memunculkan kotak dialog untuk menampilkan teks "Hanya untuk menunjukkan fungsi dialog". Dialog yang dimunculkan memiliki sebuah tombol `OK`. Hal itu diperintah dengan instruksi `MB_OK`. Sedangkan icon yang dimunculkan adalah icon informasi, yakni dengan perintah `MB_ICONINFORMATION`. Bersamaan dengan meng-klik tombol `Info`,