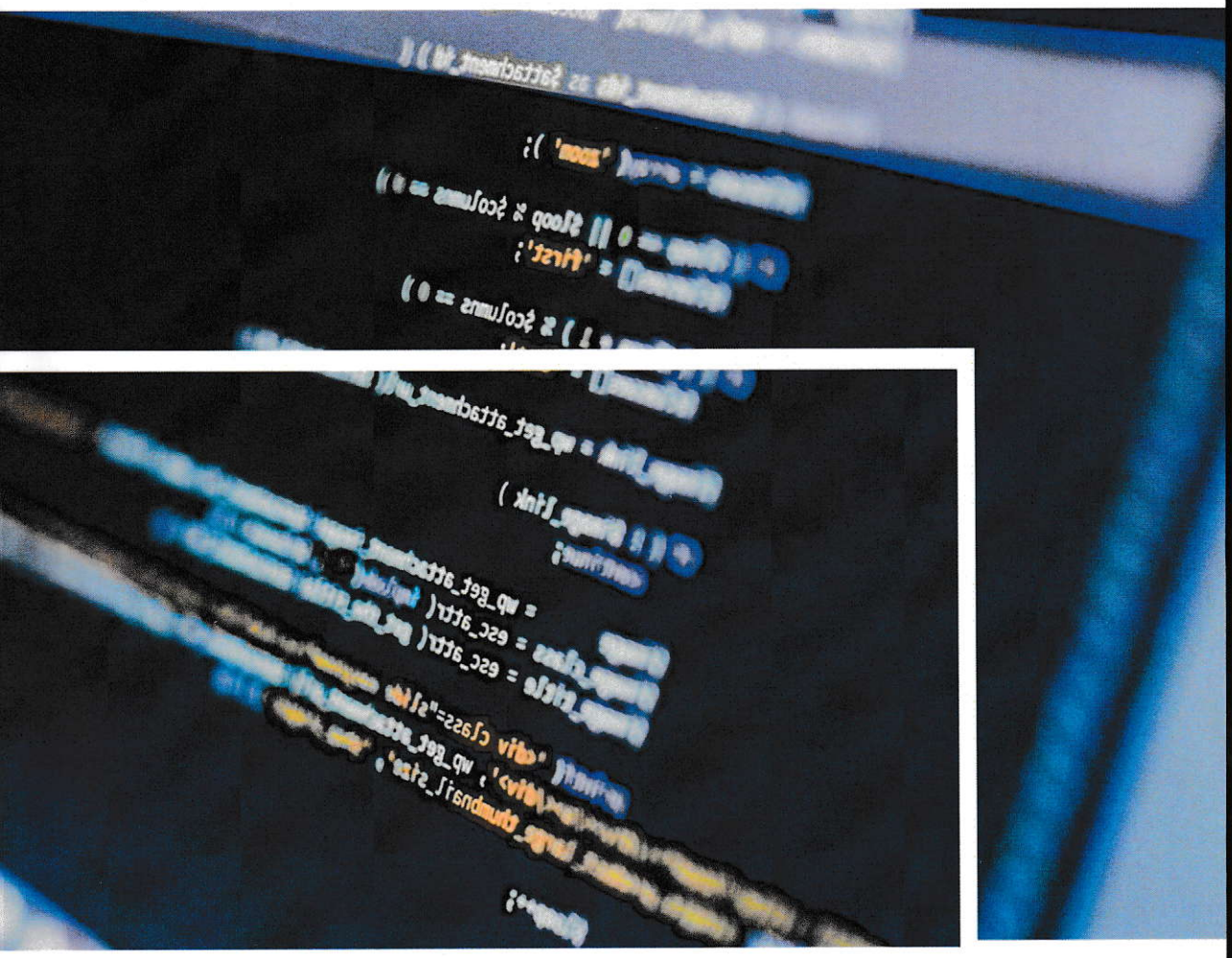
PEMROGRAMAN C++
DASAR & BERORIENTASI OBJEK



Indomedia Pustaka

Syafii, ST, MT, Ph.D

PEMROGRAMAN C++ DASAR DAN BERORIENTASI OBJEK

Syafii, PhD

Edisi Asli

Griya Kebonagung 2, Blok I2, No. 14
Kebonagung, Sukodono, Sidoarjo
Telp. : 0812-3250-3457
Website : www.indomediapustaka.com
E-mail : indomediapustaka.sby@gmail.com

Syafii

Pemrograman C++ Dasar dan Berorientasi Objek/Syafii
Edisi Pertama
—Sidoarjo: Indomedia Pustaka, 2019
Anggota IKAPI No. 195/JTI/2018
1 jil., 17 × 24 cm, 138 hal.

ISBN: 978-623-7137-33-7

1. Teknik
2. Pemrograman C++ Dasar dan Berorientasi Objek
I. Judul
II. Syafii

Dengan memanjatkan p
karunianya, sehingga bu
terselesaikan.

Buku pemograma
mata kuliah pemogran
pemograman merupaka
Dengan komputasi pers
diselesaikandengan mu
dalam komputasi adalal
persoalan dan pembuata
disajikan pemograman d
bahasa pemograman C+

Pada kesempatan in
yang telah membantu b
penulisan dan penerbit

Akhir kata kami berharap semoga buku ini bermanfaat bagi kita sebagai akademisi khususnya dan bagi yang pembaca pada umumnya. Saran dan kritik yang sifatnya membantu kelengkapan dan menuju kesempurnaan untuk edisi selanjutnya dari semua pihak sangat diharapkan.

Padang, 1 Mei 2019

Syafii, PhD

Daftar Isi

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1

Pendahuluan

1.1 Komputasi Pada Bidang Rekayasa

Penerapan teknologi digital dalam bidang rekayasa terus meningkat seiring dengan perkembangan teknologi komputer. Komputasi pada bidang rekayasa akan berhadapan dengan sistem yang besar, aplikasi real-time, perhitungan perulangan dan aplikasi interaktif lainnya. Oleh karena itu diperlukan analisis sistem rekayasa yang lebih cepat, akurat dan komprehensif untuk menilai benar dan memprediksi status sekarang dan masa depan dari sistem dalam rangka untuk membuat keputusan yang tepat. Kebutuhan tersebut telah melahirkan minat para peneliti untuk menerapkan teknologi komputasi dalam perhitungan bidang rekayasa dan pemodelannya seperti pemrograman Arduino, pemograman paralel, pemograman bebasis web dan pemograman berorientasi objek.

Kebutuhan operasional dan komersial industri memerlukan sistem informasi untuk tidak hanya melakukan fungsi tradisional tetapi juga mendukung banyak fungsi baru, khususnya untuk memenuhi kebutuhan persaingan dengan deregulasi global. Pesatnya perkembangan internet dan komputasi terdistribusi telah membuka pintu bagi solusi yang layak dan hemat biaya (Rong-Ceng Leou, 2002). Remote monitoring dan kontrol merupakan salah satu aplikasi yang paling menjanjikan dari internet sebagai media yang tepat untuk menyediakan akses remote. Banyak

program aplikasi telah dialihkan kepada platform baru berbasis Web juga merupakan salah satu darinya.

S. Chen, Jan pada tahun 2002 telah mengembangkan paket analisis berbasis Web dengan platform-independen. Tiga tingkatan arsitektur yang mengandung client tier, tingkat menengah, dan tingkat lanjut diadopsi dalam sistem ini. Sistem ini memasok klien/pengguna dengan user interface yang independent, fleksibel dan portabel, serta didesain dapat berbagi paket program dengan komputer lain via jaringan internet. Konsep ini telah diuji di bawah sistem tenaga standar dan menunjukkan hasil yang baik dan dapat diterima. Algoritma dan perangkat lunak masih perlu ditingkatkan dalam rangka untuk mempercepat waktu komputasi dan aplikasi yang user friendly. Setiap komputer dapat melakukan operasi-operasi dasar dalam pemrograman seperti operasi pembacaan data, operasi perbandingan, operasi aritmetika, dan sebagainya. Perkembangan teknologi komputer hanya merubah kecepatan, biaya, atau tingkat ketelitian tidak mengubah operasi-operasi dasar tersebut.

1.2 Pemograman Berorientasi objek

Sebelum diperkenalkan pemrograman berorientasi objek (OOP), perangkat lunak dibangun menggunakan pendekatan pemrograman struktural. Namun, pendekatan ini memiliki banyak kelemahan seperti biaya pembanguna tinggi, produktivitas yang rendah, kualitas perangkat lunak tidak terkendali, dan risiko untuk pindah ke teknologi baru [X. Cai, 2002]. OOP datang untuk menyelesaikan masalah ini di mana menggunakan pengembangan alamiah dari pemrograman terstruktur. Ciri utama dari pemrograman berorientasi obyek adalah pengenalan objek untuk menggantikan fungsi di mana objek memungkinkan kita untuk merangkum data dan fungsi. Hal ini memungkinkan fungsi untuk berperilaku berbeda tergantung pada parameter yang disediakan dan memungkinkan operator untuk membuat perangkat lunak yang lebih efisien [Jacobson, 1997]. Menggunakan kembali objek yang sama sangat mengurangi waktu pengembangan software dan meningkatkan produktivitas. Jacobson mengklasifikasikan objek- objek ke dalam tiga jenis [Jacobson, 1997]:

1. Objek entitas yang mewakili benda-benda di dunia nyata,
2. Objek kontrol yang dibuat untuk menangani operasi kompleks dalam
3. Objek antarmuka yang menangani pertukaran data dengan pengguna atau sistem lain.

Aplikasi pertama dari OOP untuk mengembangkan perangkat lunak analisis sistem tenaga dilaporkan oleh (Neyer A. F,1990). Dalam perngembangan tersebut, fitur dasar OOP diperkenalkan untuk membangun program aliran daya. Fokus utama adalah prinsip-prinsip desain OOP dan implementasi praktis untuk sistem

tenaga listrik. Model sistem tenaga dan operasi matriks jarang menggunakan OOP dipaparkan dalam (Hakavik B, 1994). Dari hasil tes numerik bahwa penerapan OOP cukup efisien dan dapat menjadi struktur standar rutin perbendaharaan numerik. Akan tetapi fitur OOP yang digunakan untuk mendapatkan fleksibilitas tersebut tidak signifikan meningkatkan waktu eksekusi.

Aplikasi yang komprehensif untuk teknologi objek komponen telah dilaporkan oleh Nor K. M, 2004. Dalam pengembangan tersebut, tool untuk memvisualisasi komponen juga dijelaskan. Tool tersebut dikembang dari integrasi banyak perangkat lunak berbasis komponen dengan memasukkan engine analisis dan komponen interface yang user frienly. Selanjutnya Mamdouh A Akher, 2005, mengembangkan aplikasi aliran daya tiga phasa dengan menggunakan objek komponen aliran daya satu satu phasa sebelumnya berbasis komponen simetris. Penelitian pengembangan aplikasi analisa sistem tenaga berbasis OOP terus berlanjut, terutama dalam menemukan model objek baru komponen sistem tenaga dan teknologi visualisasi yang user frienly.

1.3 Pemograman Parallel

Komputasi sistem rekayasa melibatkan perangkat keras dan perangkat lunak komputer. Beberapa tahun terakhir, teknologi komputer berkembang dengan cepat. Performansi mikroprosesor meningkat 52% per tahun sejak 1986 sampai 2002 [Callahan D, 2008]. Sekarang, performansi mikroprosesor meningkat dengan penambahan prosesor pada mesin komputer yang sama yang dikenal dengan sistem *multi-core*. Mesin dengan multi-prosesor sekarang telah menjadi standar semenjak kecepatan prosesor tunggal mendekati stabil atau meningkat sangat lambat dibandingkan dengan perkembagan yang lalu. Oleh karena itu, melihat kecenderungan perkembangan teknologi komputer saat ini, peningkatan performansi akan dapat dicapai dengan cara menjalankan program pada banyak prosesor secara parallel. Dengan kata lain, pendekatan banyak prosesor akan bermanfaat jika software dapat menjalankan banyak aktifitas pada priode waktu yang sama.

Aplikasi analisa sistem rekayasa harus dapat dijalankan serentak untuk mendapatkan manfaat dari perkembangan teknologi ini. Sayangnya, masih sangat sulit untuk mendapatkan algoritma yang benar-benar memberikan keuntungan dari banyak prosesor tersebut. Sebagian besar aplikasi analisa sistem tenaga berjalan menggunakan prosesor *single core*, sehingga tidak terlihat penambahan kecepatan meskipun dijalankan pada mesin multi-core. Sebenarnya program tersebut tetap berjalan dalam mode prosesor tunggal. Oleh karena itu, algoritma perlu diubah untuk mendapatkan manfaat dari perkembangan baru dalam teknologi komputer tersebut.

Beberapa aplikasi menggunakan pemrosesan paralel dalam perhitungan sistem tenaga telah dilakukan, seperti dengan menggunakan prosesor yang saling terinterkoneksi [Zang, 1996] dan klater komputer pribadi (PC) yang terhubung melalui komunikasi Ethernet [Tu F, 2002]. Sistem paralel seperti ini membutuhkan biaya yang mahal untuk menggunakannya dan waktu perhitungan juga tergantung pada kecepatan media komunikasi yang digunakan antara prosesor. Tingginya biaya hardware telah membuat keuntungan dari solusi paralel lebih cepat tidak berharga dan tidak praktis. Alternatif lain untuk mengurangi biaya dan waktu komunikasi adalah dengan menggunakan prosesor multi-core di komputer yang sama yang dikenal sebagai sistem paralel berbasis PC. Dalam waktu dekat semua komputer baru merupakan komputer paralel. Ini juga berarti bahwa biaya hardware adalah sama, apakah dasar PC standar digunakan sebagai sistem pengolahan serial atau sistem pemrosesan paralel.

Kinerja percepatan prosesor multi-core tergantung pada algoritma dan perangkat lunak. Permasalah harus didekomposisi menjadi tugas kecil dalam algoritma pemrograman paralel. Tugas-tugas ini dapat bekerja secara terpisah dari yang lain dan berjalan di bawah beberapa sistem prosesor. Masalah yang tidak bisa diurai menjadi tugas-tugas yang terpisah akan menggunakan teknik paralel loop. Kedua teknik paralel digunakan dalam pengembangan algoritma untuk mempercepat perhitungan analsia sistem tenaga. Teknologi komputasi paralel baru ini dapat memecahkan perhitungan sistem listrik secara lebih efisien.

BAB 2

Dasar Pemrograman C++

Saat ini terdapat banyak bahasa pemrograman seperti bahasa pascal, basic dan bahasa C. Pengguna bahasa pemrograman C dan C++ sangatlah banyak karenakan kemampuan bahasa C yang dapat dipakai dalam pembuatan aplikasi bidang rekayasa dan bidang terapan lainnya. Kelebihan bahasa pemorgraman C++ adalah kemampuan untuk melakukan pemrograman berorientasi objek (Object Oriented Programming atau OOP). Pada OOP, data dan instruksi dibungkus (encapsulation) menjadi satu. Kesatuan ini disebut kelas (class) dan inisiasi kelas pada saat run-time disebut objek (object). Data di dalam objek hanya dapat diakses oleh instruksi yang ada didalam objek itu saja.

2.1 Sejarah Pemograman C++

Sejarah Bahasa pemograman C++ dimulai dari perkembangan Bahasa C. Bahasa pemograman C pertama diirancang oleh Denis M. Ritchie tahun 1972 di Bell Labs. Bahasa C merupakan pengembangan dari bahasa BCPL oleh Martin Richard tahun 1967 dan bahasa B oleh Ken Thompson tahun 1970. Selanjutnya dibantu oleh Brian W. Kernighan, Ritchie menulis buku “The C Programming Language” pada tahun

1978. Sejak itu bahasa pemograman C lebih dikenal dengan nama K-R C atau C klasik. Versi C yang lebih baru adalah Ansi C pada tahun 1989 dan iso C 99 tahun 1999.

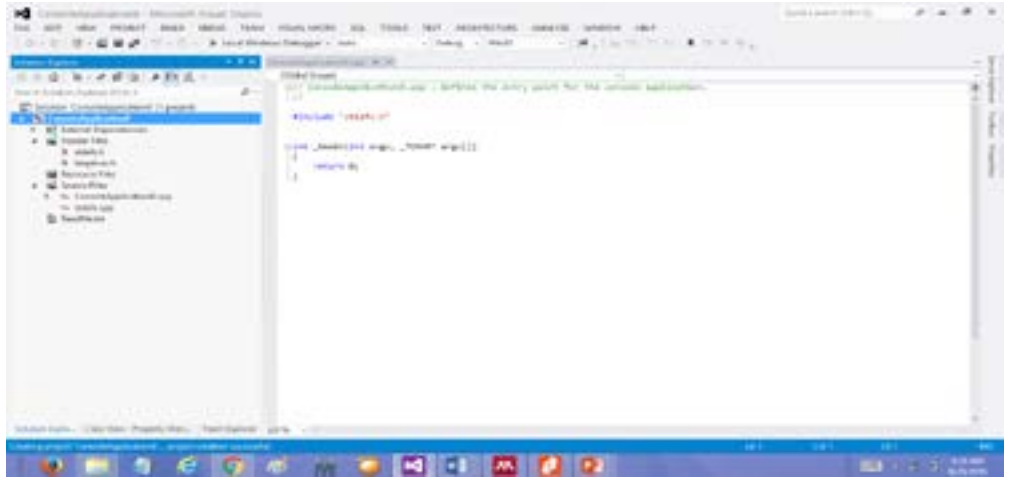
Bahasa C++ mulai diperkenalkan oleh Bjarne Stroustrup dari laboratorium AT&T / Bell pada tahun 1980 dengan nama asalnya "C with Classes". Selanjutnya Stroustroup's menulis buku " The C++ Programming Language" Bahasa C merupakan Bahasa tingkat menengah (a mid-level language). Bahasa C telah menjadi bahasa C plus OOP ditambah dengan sejumlah syntax baru, tipe statical dan dapat mendukung berbagai tipe pemograman.

Saat ini bahasa C++ mulai banyak dikembangkan untuk membuat software operating systems seperti Windows, Linux dan juga untuk computer networking (distributed client/server applications), games, device drivers, and software embedded.

2.2 Tampilan Awal Visual C++

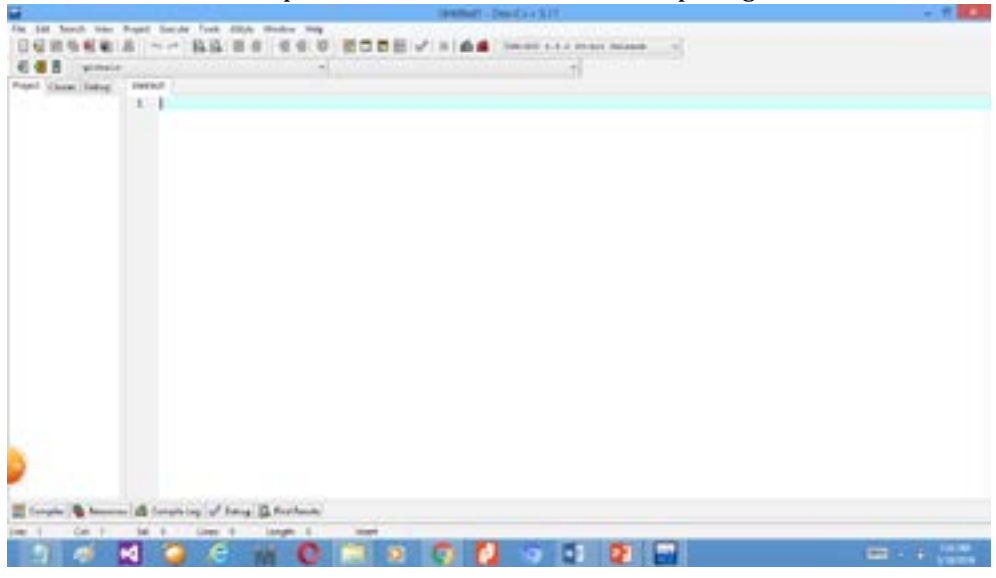
Visual C++ adalah sebuah produk Integrated Development Environment (IDE) untuk bahasa pemrograman C dan C++ yang dikembangkan Microsoft. Visual C++ merupakan salah satu bagian dari paket Microsoft Visual Studio. Bahasa ini merupakan bahasa pemrograman tingkat menengah sebagian sintaxnya masih dapat dipahami sebagaimana bahasa kita sehari-hari. Visual C++ adalah bahasa pemrograman yang cukup populer. Hampir semua file DLL pada sistem operasi Windows dibuat menggunakan bahasa ini.

Pada saat memulai menggunakan bahasa pemograman Visual C++, tampilan awal untuk penulisan kode program adalah sebagai berikut:



Gambar 2.1 Tampilan awal program Visual C++

Selain Visual C++, Bahasa C++ dapat juga dijalankan menggunakan compiler Dev C++. Dev-C ++ adalah lingkungan pengembangan terintegrasi (IDE) berfitur lengkap gratis yang didistribusikan di bawah Lisensi Publik Umum GNU untuk pemrograman dalam C dan C ++. Tampilan awal IDE Dev C++ adalah seperti gambar 2.2 berikut:



Gambar 2.2 Tampilan awal program Dev C++

2.3 Struktur dasar pemograman C++ adalah:

Setiap komputer dapat melakukan operasi-operasi dasar dalam pemrograman seperti operasi pembacaan data, operasi perbandingan, operasi aritmetika, dan sebagainya. Perkembangan teknologi komputer hanya merubah kecepatan, biaya, atau tingkat ketelitian tidak mengubah operasi-operasi dasar tersebut. Bahasa pemograman C++ termasuk dalam katagori bahasa pemograman tingkat menengah yang perlu diterjemahkan terlebih dahulu oleh sebuah translator bahasa (yang disebut kompilator atau compiler) ke dalam bahasa mesin sebelum akhirnya dieksekusi oleh CPU. Setiap instruksi dalam bahasa mesin merupakan kombinasi dari operasi dasar yang bersesuaian, dan menghasilkan efek yang sama pada setiap komputer.

Struktur dasar pemograman C++ adalah:

```
#include
main()
{
Perintah-perintah berdasarkan algoritma pemograman;
return 0;
}
```

Elemen-elemen bahasa Pemograman terdiri dari:

1. **Statemen**

   Statemen adalah sebuah baris kode formal yang menberi tahu komputer apa yang dilakukan.

   Program melakukan pekerjaan menggunakan statemen untuk:

   - Inisialisasi
   - Mendapatkan input
   - Proses data
   - Penampilan output
   - Penyimpanan data

   Sebagai contoh perhatikan program berikut:

   - Inisialisasi

     float fahr, celcius;
   - Mendapatkan input

     fahr= 50;
   - Proses data

     celcius = (fahr - 32) *5/9 ;
   - Penampilan hasil

     cout<< " Hasil Konversi = "<<celcius<<endln;

2. **Variabel**

   Sebuah unit memori yang dapat diubah selama eksekusi program. Setiap variabel harus diberi dengan nama yang unik.

   Syarat-syarat penamaan suatu variable adalah sebagai berikut:

   - Dibatasi sebesar 255 karakter.
   - Harus diawali oleh karakter alfabet.
   - Tidak boleh memiliki titik atau koma.
   - Nama tidak boleh berupa kata kunci (keyword)

3. **Konstanta**

   Cara deklarasi konstanta adalah sebagai berikut:

   - Const float pi = 3.14; atau
   - #define pi 3.14

   Dimana:

   - Const : kata wajib yang ada di depan definisi konstanta
   - pi : nama yang dipilih sebagai nama untuk konstanta
   - 3.14 : nilai yang diberikan pada konstanta

4. **Tipe data**

Tabel 2.1 Tipe data pemrograman C++

NO	Tipe Data	Ukuran	Range (jangkauan)	Format	Keterangan
1	char	1 byte	-128 s/d 127	%c	Karakter/string
2	int	2 byte	-32768 s/d 32767	%i, %d	Integer/bilangan bulat
3	Float	4 byte	-3.4E-38 s/d e.4E+38	%f	Float/bilangan pecahan
4	Double	8 byte	-1.7E-308 s/d 1.7E+308	%lf	Pecahan presisi ganda
5	void	0 byte	-	-	Tidak bertipe

5. **Operator dan ekspresi**

Simbol yang dapat menyebabkan perubahan pada ekspresi seperti operasi aritmatika.

Tanda sama dengan (=) merupan operator penugasan yang menberikan sebuag nilai pada sebuah variabel. Contoh:

V = I * R

Akan memberikan nilai pada variabel V dengan nilai perkalian variabel I dan R.

Tabel 2.2 Operator dan ekspresi pemrograman C++

C++ Operation	Operator	Algebraic Expression	C++ Expression
Addition	+	f + 7	f + 7
Substraction	-	p – c	p – c
Multiplication	*	Bm or b m	b * m
Division	/	x / y or or x ÷ y	x / y
Modulus	%	r mod s	r % s

Operator Pembanding

Tabel 2.3 Jenis operator pembanding pemrograman C++

Simbol	Deskripsi	Contoh Ekspresi	
		Benar	Salah
<	Kecil dari	-7.8 < 4.5	4.5 < -7.8
<=	Kecil dari atau sama dengan	0.67 <= 0.68	0.68 <= 0.67
>	Besar dari	20 > 10	10 > 20
	Besare dari atau sama dengan	1 >= 0	0 >= 1
<>	Tidak sama dengan	10 <> 10.01	10 <> 10
=	Sama dengan	7 = 7	7 = 6.99999

Selanjutnya pada bab ini akan dibahas dasar-dasar pemograman C++ yang akan digunakan dalam membuat aplikasi komputasi sistem rekayasa dan matematika.

Berikut ini merupkan operator penting yang sering digunakan dalam membuat kode program C++:

Operator	Keterangan
+	pertambahan
*	perkalian
%	Sisa pembagian
-	Pengurangan
/	Pembagian

Selain itu pemogramman C++ juga mengenal operator logika berikut:

Operator	Keterangan
&&	Operator Logika AND
||	Operator Logika OR
!	Operator Logika NOT

Operator logika tersebut digunakan untuk menghubungkan dua buah operasi relasi menjadi sebuah ungkapan kondisi. Hasil dari operator logika ini menghasilkan nilai numerik 1 (True) atau 0 (False).

Operasi Matematika:

Beberapa statement operasi matematikan yang sering digunakan dalam pemograman adalah sebagai berikut:

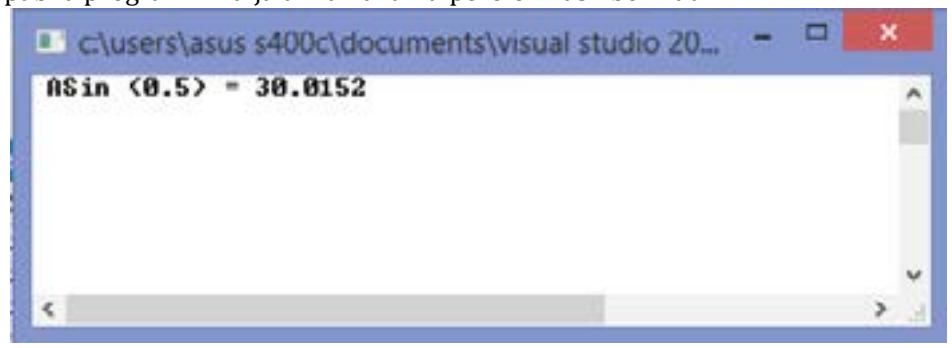
Tabel 2.4 Jenis operator matematika pemrograman C++

Operasi	Fungsi
sin (x)	Sin x
cos(x)	Cos c
tan(x)	Tan x
asin(x)	Arc sin x
acos(x)	Arc cos x
atan(x)	Arc tan x
pow(x,y)	x^y

Penggunaan arc sinus dalam bahasa pemograman C++ ditulis sebagai asin(x) dan mengembalikan hasil nilai sudut dalam radian. Untuk mendapatkan nilai sudut dalam derajat perlu dikalikan dengan 180/PI. Perhatikan contoh kode program berikut:


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

#define PI 3.14

int _tmain(int argc, _TCHAR* argv[])
{

    float A = asin(0.5);
    cout<<" ASin (0.5) = "<<A*180/PI<<endl<<endl;
    _getch();
    return 0;
}
```

Apabila program ini dijalankan akan diperoleh hasil berikut:



Gambar 2.3 Keluaran nilai sudut dari pi ke derajat

Operator Bitwise:

Operator bit merupakan operator yang dikomputasi berdasarkan nilai bit dari sebuah bilangan bulat. Contoh operator bit yang sering digunakan adalah:

Tabel 2.5 Jenis operator bitwise pemrograman C++

Symbol	Operator
&	Operator bit logika AND
|	Operator bit logika OR
^	Operator bit XOR (eXclusive OR)
<<	Operator pergeseran kiri
>>	Operator pergeseran kanan
~	Operator NOT

Contoh operator AND:

Operator bit dari 5 adalah 00000101
Operator bit dari 2 adalah 00000010
Maka nilai & adalah 00000000 = 0

Contoh operator OR:

Operator bit dari 5 adalah 00000101
Operator bit dari 2 adalah 00000010
Maka nilai | adalah 00000111 = 7

Contoh operator XOR:

Operator bit dari 5 adalah 00000101
Operator bit dari 2 adalah 00000010
Maka nilai ^ adalah 00000111 = 7

Contoh operator <<:

Operator bit dari 5 adalah 00000101
Maka nilai << 2 adalah 00010100 = 20

Contoh operator ~

Operator bit dari 5 adalah 00000101
Maka nilai ~ adalah 11111010 second complement = (-3)

2.4 Algoritma Pemograman

Definisi algoritma yang umum dijumpai di berbagai literatur adalah urutan logis langkah-langkah penyelesaian masalah. Berdasarkan algoritma selanjutnya dapat membantu pembuatan kode program komputer. Program komputer berisi urutan langkah-langkah penyelesaian masalah secara sistematis dapat ditulis menggunakan bahasa pemrograman. Urutan langkah-langkah penyelesaian masalah inilah yang dinamakan algoritma.
Sifat-sifat suatu algoritma adalah:

1. Setiap langkah harus terdefinisi dengan baik
2. Banyaknya operasi berhingga
3. Minimum satu keluaran
4. Bersifat umum

Cara penulisan terdiri dari komponen-komponen berikut:

1. Masukan
2. Langkah-langkah (proses)
3. Keluaran

Contoh soal:
Buat langkah-langkah perhitungan berulang berikut :

a. $D = a_1 + a_2 + a_3 + \ldots + a_n$
b. $H = b_1 \bullet b_2 \bullet b_3 \bullet \ldots \bullet b_n$

Jawab:

a. Masukan : a_i untuk i = 1, 2,...,n
   Proses:
   1. D = 0
   2. Untuk i = 1 sampai dengan n
      ↳ $D = D + a_i$
   Keluaran: D
b. Masukan : b_i untuk i = 1, 2,...,n
   Proses:
   1. H = 1
   2. Untuk i = 1 sampai dengan n
      ↳ $H = H * b_i$
   Keluaran: H

BAB 3

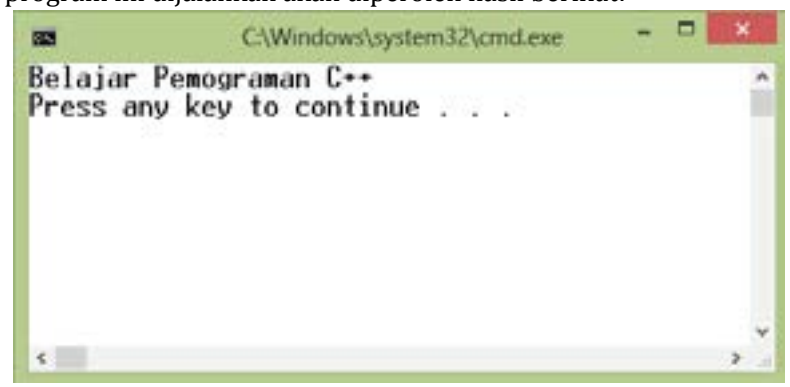
Perintah Masukan dan Keluaran

Masukan data dari keyboard menggunakan perintah cin>> , sedangkan keluaran ke tampilan dos (*console*) menggunakan perintah cout<<. Library yang digunakan iostream.h ditambahan dengan pernyataan using namespace std;. iostream adalah salah satu header file yang digunakan untuk fungsi input dan output yang ada di C++ dan using namespace std adalah perintah yang digunakan untuk mendeklarasikan/ memberitahukan kepada compiler C++ bahwa kita akan menggunakan semua fungsi/class/file yang terdapat dalam namespace std.

Perhatikan contoh berikut untuk menampilkan kalimat Belajar Pemograman C++ ke tampilan dos menggunakan pemograman Ms Visual C++ 2010.

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

int _tmain(int argc, _TCHAR* argv[])
{
    cout<<"Belajar Pemograman C++"<<endl;
    return 0;
}
```

Apabila program ini dijalankan akan diperoleh hasil berikut:



Gambar 3.1 Tampilan hasil running program

Untuk memperbaiki tampilan numerik gunakan library iomanip.h dengan beberapa sintaks penting berikut:
Contoh:
double f =3.14159
cout << setprecision(5) << f ;
Apabila program dijalankan hasilnya adalah:
3.1416
Selanjutnya mengatur jumlah digit numerik yang ditampilakan dapat dilakukan dengan:
cout << setw(10);
cout << 77 << endl;
Apabila program dijalankan hasilnya adalah:
77
Perintah berikut juga dapat digunakan untuk mencetak hasil tetap dengan lebar 5 digit dan rata kanan.
cout <<fixed <<setw(5) << right;

Untuk data yang banyak, perintah membaca data dari keyboard menjadi tidak efektif dan akan memerlukan waktu yang lama untuk proses input data. Alternatif lain adalah melakukan proses pembacaan data dari text file atau file .txt. Perintah untuk membaca data dari file .txt adalah fin>> dan keluaran ke file .txt menggunakan perintah fout<<. Perhatikan contoh berikut:
Data saluran suatu sistem tenaga sebagai berikut:

```
7 5
1 2 0.02 0.06 0.03
1 3 0.08 0.24 0.025
2 3 0.06 0.18 0.02
2 4 0.06 0.18 0.02
2 5 0.04 0.12 0.015
3 4 0.01 0.03 0.01
4 5 0.08 0.24 0.025
```

Data tersebut disusun mulai dari jumlah saluran dan jumlah bus, selanjutnya di bawah diberikan data parameter saluran sebanyak tujuh saluran dan simpan dalam file data5bus.txt. Maka untuk dapat membaca data tersebut, terlebih dahulu siapkan variabel untuk menampung data tersebut seperti berikut:

```
int i,Nb,Nl,sl[50],el[50];
double rl[50],xl[50],bl[50];
```

dimana:

- Variabel integer Nl dan Nb masing-masing untuk menampung jumlah saluran dan jumlah bus.
- Variabel array satu dimensi dengan tipe data integer untuk data bus kirim dan terima sl[50] dan el[50].
- Variabel array satu dimensi dengan tipe data double untuk reistansi, reaktansi dan suseptansi saluran: rl[50],xl[50],bl[50].

Selanjutnya gunakan perintah fin>> untuk proses pembacaan data seperti kode program berikut:

```
ifstream fin ("data5.txt", ios::in);
fin>>Nl>>Nb;
for(i=1;i<=Nl;i++){
  fin>>sl[i]>>el[i]>>rl[i]>>xl[i]>>bl[i];
```

Sebelumnya tambahkan library fstream.h untuk proses input/output data dari file pada bagian atas program seperti berikut: `#include <fstream>.`

Untuk menghentikan tampilan dapat menggunakan perintah berikut diakhir program:

getch(); atau
_ getch() untuk Visual Studio versi 12.

dimana sebelumnya harus ditambahkan `#include<conio.h>` pada header project.

BAB 4

Perulangan dan Percabangan

4.1 Perulangan

Perulangan (repetisi) adalah kumpulan statemen yang dieksekusi berulang kali sampai kondisi yang ditentukan bernilai benar. Pernyataan perulangan dalam program komputer terdiri dari perulangan dengan for, while dan do while.
Syntax untuk Pernyataan perulangan for adalah sebagai berikut:
for (initialization; test; action)
Pernyataan yang diulang;
Syntax untuk Pernyataan perulangan while adalah sebagai berikut:
while (kondisi)
    Pernyataan yang diulang;

Syntax untuk Pernyataan perulangan do … while adalah sebagai berikut:
do
    pernyataan yang diulang;
while (kondisi)

Pernyataan perulangan yang paling sering digunakan adalah for. Pernyataan for memiliki tiga parameter yang diapit dalam tanda kurung ( ), yaitu nilai awal, akhir perulangan dan penambahan/pengurangan. Contoh perulangan dengan pernyataan for adalah:

```
for(i=1;i<=100;i++){
    Perintah yang diulang;
}
```

Dari contoh tersebut perintah yang akan diulang dilakukan sebanyak 100 kali dimulai dari i = 1 sampai dengan i = 100.

Pernyataan perulangan lain adalah while. Pernyataan while memiliki kondisi yang harus dipenuhi yang berada dalam tanda kurung ( ), jika kondisi bernilai benar akan dilakukan perintah yang diulang, jika tidak, maka keluar dari blok perulangan while .

Contoh perulangan dengan pernyataan while adalah:

```
Iterasi = 1;
while (Iterasi<=100){
    Perintah yang diulang;
    Iterasi=Iterasi+1;
}
```

Dari contoh di atas perintah yang akan diulang dilakukan sebanyak 100 kali dimulai dari Iterasi = 1 sampai dengan Iterasi = 100.

Contoh perulangan menggunakan perintah do while:

```
int a=1; int x=2;int i;
float Y= a*x*x*(x+7);
i=1;
do{
  cout<<"Nilai Y adalah = "<<Y<< endl<<endl;
  i=i+1;
} while (i<=5);
```

Perbedaan while dan do-while terletak pada pembacaan pertama program, while akan mencetak eksekusi setelah melakukan pengecekan kondisi, sedangkan pada do-while akan mencetak eksekusi terlebih dahulu baru melakukan pengecekan kondisi

Contoh perbedaan while dan do-while

```
    int a=1;
    do{
      cout<<"Nilai akan tercetak "<<a<< endl<<endl;
    } while (a=0);

    int b=1;
while (b=0){
    cout<<"Nilai tidak tercetak "<<b<< endl<<endl;
}
```

Maka output yang akan muncul adalah :
Nilai akan tercetak

Nilai tidak tercetak tidak akan muncul pada console command dikarenakan program akan langsung melewati kondisi pada while karena tidak memenuhi persyaratan kondisi.

Contoh 4.1:

Buat program untuk mencetak bilangan 1 s/d 50 dan menampilkan hasil perjumlahan totalnya diakhir perulangan menggunakan perintah:

a. for
b. while
v. do while

Jawab:

a. Perulangan **for**

```
for(i=1;i<=50;i++)
    cout<<i<<endl;
```

b. Perulangan **while**

```
int i,Sum;
Sum = 0;
i=1;
while(i<=50){
    cout<<i<<endl;
    Sum += i; // Sum = Sum +i;
    i+= 1; // i = i +1;

}
cout<<"Jumlah total adalah: "<<Sum<<endl;
```

c. Perulangan **do...while**

```
int i,Sum;
Sum = 0;
i=1;
do {
     cout<<i<<endl;
     Sum += i; // Sum = Sum +i;
     i+= 1; // i = i +1;

} while(i<=50);
cout<<"Jumlah total adalah: "<<Sum<<endl;
```

Contoh 4.2:

Buat program untuk menampilkan hasil perkalian dan perjumlahan berulang berikut:

$$D = \sum_{i=1}^{3} \left\{ \prod_{\substack{j=1 \\ j \neq i}}^{2} 4 * j \right\}$$

menggunakan perintah:

a. for
b. while
c. do while

Jawab:

a.

```
int i,j;
float D,faktor;
D=0.0;
for(i=1;i<=3;i++){
   faktor=1;
   for(j=1;j<=2;j++){
         if(i!=j){
               faktor=faktor*4*j;
         }
   }
D=D+faktor;
}
cout<<"D = "<<D<<endl;
```


b.

Contoh 4.3:

Buatlah program komputer perhitungan berulang berikut:

$$P_n(X) = \sum_{i=0}^{n} \left\{ \prod_{\substack{j=0 \\ j \neq i}}^{n} \left(\frac{x - x_j}{x_i - x_j} \right) \right\} \cdot f_i$$

Jawab:

Program komputer perhitungan berulang tersebut adalah:

```
PNZ=0.0;
for(i=0;i<N;i++){
    faktor=1;
    for(j=0;j<N;j++){
      if(i!=j){
            faktor=faktor*((z-x[j])/(x[i]-x[j]));
      }
    }
PNZ=PNZ+faktor*f[i];
}
```

Soal Latihan:

1. Buat program untuk mencetak bilangan 1, 3, 5, ... , 49 dan menampilkan hasil perjumlahan totalnya diakhir perulangan menggunakan perintah:
   a. for
   b. while

4.2 Percabangan

Pernyataan percabangan digunakan untuk memecahkan persoalan untuk mengambil keputusan di antara sejumlah pernyataan atau pilihan yang ada. Pernyataan percabangan terdiri dari:

if
if – else
switch – case

4.2.1 Pernyataan if

Bentuk umum dari pernyataan if adalah:

```
if (kondisi) {
pernyataan;
}
```

Contoh:

```
if(A=100)
   cout<<" Nilai A = 100 "<<endl;
```

Jika A bernilai 100 maka akan dijalankan perintah untuk mencetak Nilai A = 100, jika tidak maka perintah tersebut akan diabaikan.

4.2.2 Pernyataan if – else

Bentuk umum dari pernyataan if - else adalah:

```
if (kondisi) {
pernyataan 1;
}
else {
pernyataan 2;
}
```

Contoh program penggunaan percabangan dengan pernyataan if – else adalah:

```
int pilihan;
    cout<<" Program Operasi Matrik "<<endl;
    cout<<" 1. Perkalian Matrik."<<endl;
    cout<<" 2. Perjumlahan Matrik."<<endl;
    cout<<" Tentukan pilihan anda: ";
    cin>>pilihan;
    if (pilihan == 1)
      Perkalian();
    else
      Perjumlahan;
    return 0;
```

Pada contoh program diatas terdapat percabangan untuk memilih program operasi matrik, jika pilihan satu maka proses yang dilakukan adalah memanggil fungsi perkalian jika tidak maka fungsi yang di panggil adalah fungsi perjumlahan tanpa ada alternate pilihan lainnya.

Contoh percabangan lainnya adalah pada pemilihan perhitungan segitiga daya berikut:

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

int main()
{
   int pilih;
   float P, S, Q, Pf, sudut;
   cout<<" Program Segitiga Daya "<<endl<<endl;
   cout<<" Tentukan Pilihan Anda:"<<endl;
   cout<<" 1. Menghitung Daya Aktif"<<endl;
   cout<<" 2. Menghitung Daya Reaktif"<<endl;
   cout<<" 3. Menghitung Daya Semu"<<endl;
   cout<<" Tentukan Pilihan Anda:";cin>>pilih;
   if (pilih==1){
   cout<<endl<<" Besar Daya Semu ="; cin>>S;
   cout<<" Faktor Daya ="; cin>>Pf;
   P=S*Pf;
   cout<<" Daya Aktif adalah :"<<P<<" Watt"<<endl;
   }
   if (pilih==2){
   cout<<endl<<" Besar Daya Semu ="; cin>>S;
   cout<<" Faktor Daya ="; cin>>Pf;
   sudut=acos(Pf);
   Q=S*sin(sudut);
   cout<<" Daya Reaktif adalah :"<<Q<<" Var"<<endl;
   }
   if (pilih==3){
   cout<<endl<<" Besar Daya Aktif ="; cin>>P;
   cout<<" Besar Daya Reaktif ="; cin>>Q;
   S=sqrt(P*P+Q*Q);
   cout<<" Daya Semu adalah :"<<S<<" VA"<<endl;
   }
   return 0;
}
```

Pada header program terdapat perintah #include<math.h>, karena dalam kode program menggunakan perintah akar atau sqrt(). Perintah percabagan if ... else if digunakan untuk menentukan pilihan 1,2 atau 3 untuk perhitungan segitiga daya.

4.2.3 Pernyataan switch - case

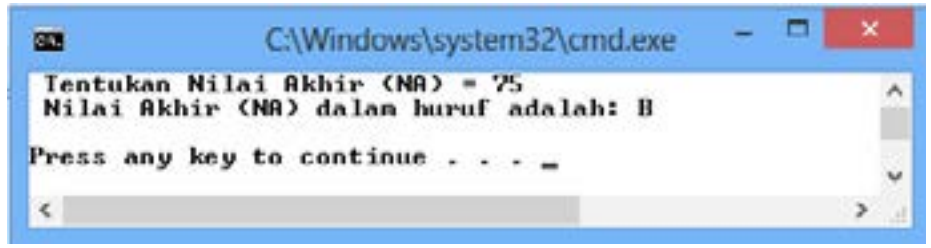
```
switch (syarat) {
      case kontanta1:
         pernyataan 1;
         break;
      case konstanta2:
         pernyataan 2;
         break;
default:
      perintah;
}
```

Contoh penggunakan percabangan menggunakan perintah switch case adalah seperti berikut:

Program ini akan mengubah nilai akhir mahasiswa menjadi nilai dalam bentuk huruf.

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

int _tmain(int argc, _TCHAR* argv[])
{
    float na;
    cout<<" Tentukan Nilai Akhir (NA) = "; cin>>na;
    cout<<" Nilai Akhir (NA) dalam huruf adalah: ";

    if(na >= 85){
        cout << "A";
    }else if(na >= 80){
        cout << "AB";
    }else if(na >= 70){
        cout << "B";
    }else if(na >= 65){
        cout << "BC";
    }else if(na >= 55){
        cout << "C";
    }else if(na >= 40){
        cout << "D";
    }else{
        cout << "E";
    }

    cout<<endl<<endl;
    return 0;
}
```

Jika Program ini dijalankan akan diperoleh tampilan dan hasil sebagai berikut:



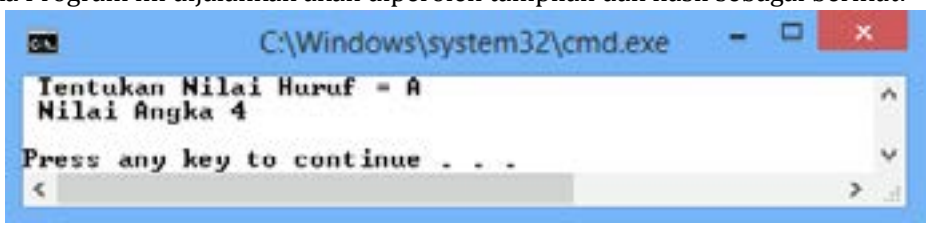
Gambar 4.1 Tampilan hasil running program percabangan

Sebaliknya untuk mengubah dari nilai huruf ke bobot angka menggunakan percabangan perintah switch case adalah seperti berikut:

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

int _tmain(int argc, _TCHAR* argv[])
{

   char Huruf;
   cout<<" Tentukan Nilai Huruf = "; cin>>Huruf;

   switch(Huruf)
   {
     case 'A':
     printf(" Nilai Angka 4\n");
     break;
     case 'B':
     printf(" Nilai Angka 3\n");
     break;
     case 'C':
     printf(" Nilai Angka 2\n");
     break;
     case 'D':
     printf(" Nilai Angka 1\n");
     break;
     case 'E':
     printf(" Nilai Angka 0\n");
     break;
     case 'a':
     printf(" Nilai Angka 4\n");
     break;
     case 'b':
     printf(" Nilai Angka 3\n");
     break;
     case 'c':
     printf(" Nilai Angka 2\n");
```

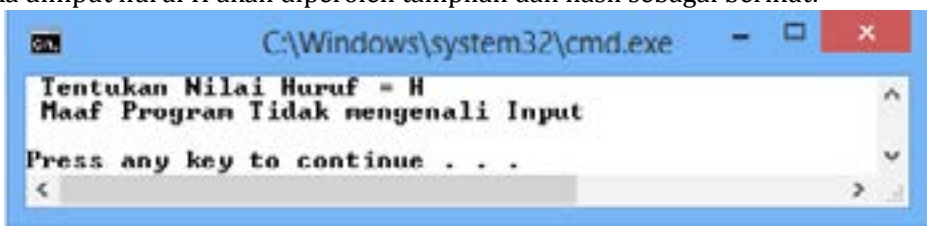
```
    break;
    case 'd':
    printf(" Nilai Angka 1\n");
    break;
    case 'e':
    printf(" Nilai Angka 0\n");
    break;
    default :
    printf("Mohon Maaf Program Tidak mengenali Input\n");
    }

  cout<<endl;
  return 0;
}
```

Perogram ini digunakan untuk mengubah nilai huruf kepada bobotnya. Pada kode program tersebut menggunakan tipe data char untuk menapung varibel huruf dan sebuah variable angka dengan tipe data integer yang merupakan bilanagn bulat dari 0 sampai dengan 4.
Jika Program ini dijalankan akan diperoleh tampilan dan hasil sebagai berikut:



Gambar 4.2 Program dengan input A-E dan a-e

Jika diinput huruf H akan diperoleh tampilan dan hasil sebagai berikut:



Gambar 4.3 Program dengan input yang tak dikenali

4.3 Perhitungan Waktu Komputasi

Terdapat beberapa perintah dalam C++ untuk mengakses waktu komputer, akan tetapi untuk ketelitian sampai orde millidetik dapat mengunakan perintah berikut:

```
volatile DWORD dwStart;
float T;
dwStart = GetTickCount();
.
.
T = (GetTickCount() - dwStart)/1000
Cout<<" Waktu Komputasi adalah: "<<T<<" seconds"<<endl;
```

Sebelumnya tambahkan library `#include <windows.h>` pada header program utama. Waktu kumputasi ditampung dalam variabel T dan satuannya adalah second.

Contoh soal:

1. Diketahui sekumpulan data seperti table berikut:

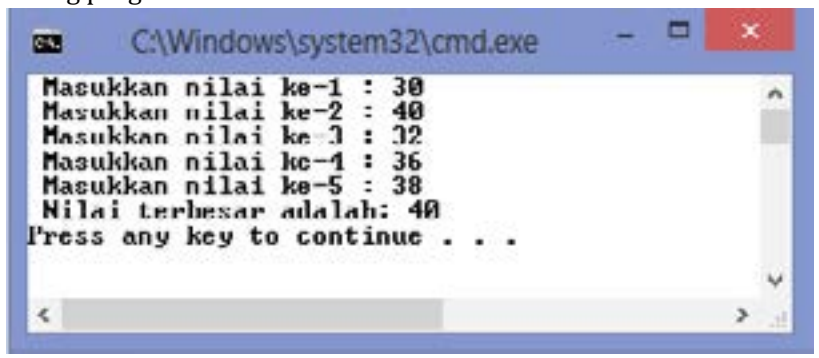
No	Nilai
1	30
2	40
3	32
4	36
5	38

Menggunakan perintah perulangan dan percabangan, buatlah program C++ untuk melacak nilai terbesar dari lima data nilai tersebut?
Jawab:

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

int _tmain(int argc, _TCHAR* argv[])
{
   int i;
   float A,B;
   B=0;
   for(i=1;i<=5;i++){
   cout<<" Masukkan nilai ke-"<<i<<" : ";
   cin>>A;
```

```
   if(A>B)
       B=A;
   }
   cout<<" Nilai terbesar adalah: "<<B<<endl;
   return 0;
}
```

Hasil running program adalah:



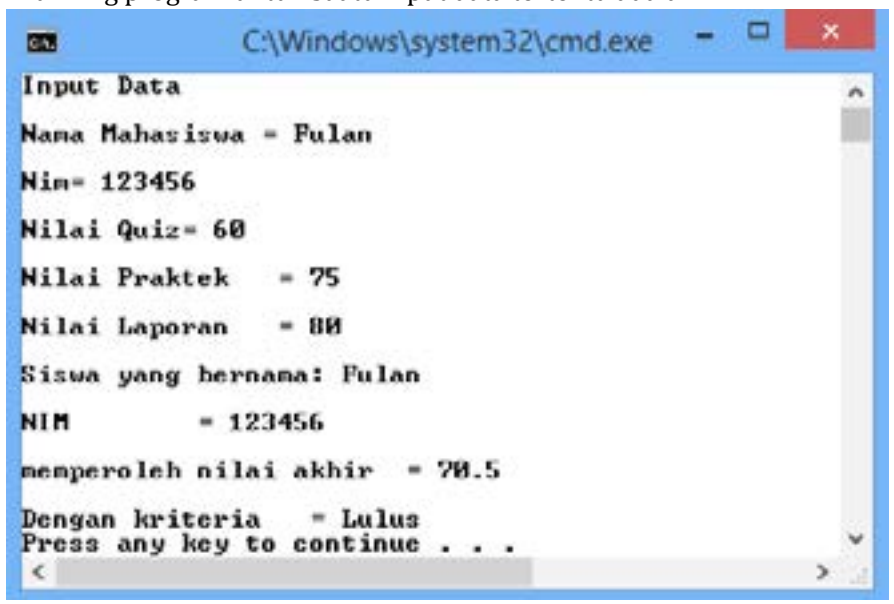
2. Buat program untuk mencetak bilangan 1, 3, 5, … , 49 dan menampilkan hasil perjumlahan totalnya diakhir perulangan menggunakan perintah for:
   Jawab:

```
int _tmain(int argc, _TCHAR* argv[])
{
   int i, k, Total;
   Total=0;
   for(i=0;i<=24;i++){
       k=2*i+1;
       cout<<k<<endl;
       Total=Total+k;
   }
   cout<<" Total = "<<Total<<endl;
   return 0;
}
```

3. Buat program untuk menampilkan hasil studi mahasiswa dengan inptan Nama, Nim, Nilai Quiz (35%), Nilai Praktek 50% dan Nilai Laporan (15%) dan menampilkan hasil perjumlahan total nilan untuk menentukan kelulusan mahasiswa dengan kriteria jika nilai akhirnya besar atau sama dengan 50 berarti Lulus, sedangkan kecil dari 50 Tidak Lulus?
   Jawab:


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

int _tmain(int argc, _TCHAR* argv[])
{
    string a, b;
    float nilai_quiz, nilai_praktek, nilai_laporan, na;
    cout << "Input Data " << endl << endl;
    cout << "Nama Mahasiswa = "; getline(cin,a);
    cout << "\nNim= "; getline(cin,b);
    cout << "\nNilai Quiz= "; cin >> nilai_quiz;
    cout << "\nNilai Praktek   = "; cin >> nilai_praktek;
    cout << "\nNilai Laporan   = "; cin >> nilai_laporan;
    nilai_quiz = nilai_quiz * 0.35;
    nilai_praktek = nilai_praktek * 0.5;
    nilai_laporan = nilai_laporan * 0.15;
    na = nilai_quiz + nilai_praktek + nilai_laporan;

    cout << endl;
    cout << "Siswa yang bernama: "<< a <<endl;
    cout << "\nNIM         = " << b << endl;
    cout << "\nmemperoleh nilai akhir  = " << na << endl;
    cout << "\nDengan kriteria   = ";
    if(na >= 50){
        cout << "Lulus"<<endl;
     }else{
        cout << "Tidak Lulus"<<endl;
    }

   return 0;
}
```

Hasil running program untuk suatu input data tertentu adalah:



BAB 5

Fungsi dan Prosedur

Fungsi merupakan bagian dari kode program yang terpisah dari program utama. Bagian kode program ini di-execute jika diperlukan untuk melakukan tindakan khusus dalam program. Fungsi banyak dilibatkan dalam program dengan tujuan untuk mengurangi duplikasi pengkodean dan untuk mempermudah pemahaman. Penulisan fungsi terdiri dari atas:

1. Function Prototype Syntax
   return_type function_name ([type [parameterName]]...);

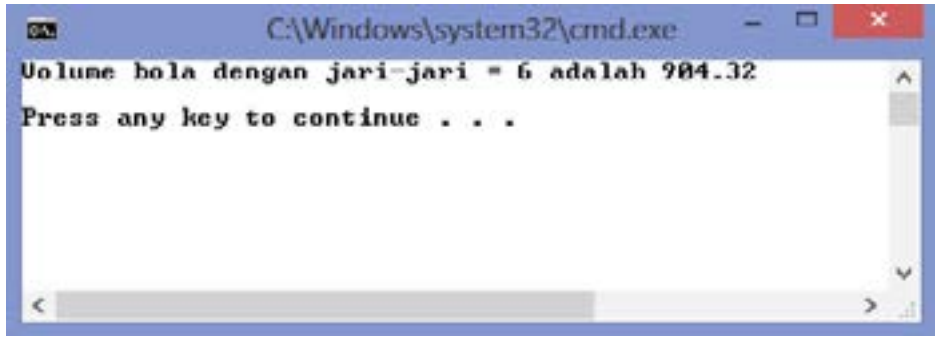
2. Function Definition Syntax
   return_type function_name ([type parameterName]...)
   {
   statements;
   }
   Return type = int, float, double, bool

Function Prototype Syntax biasanya ditulis diatas program utama atau pada file .h sedangkan Function Definition Syntax biasanya ditulis di bawah program utama atau dalam file .cpp.

Contoh 5.1
Fungsi menghitung volume bola adalah:

```
#define PI    3.14
float volume_bola(float r);

int _tmain(int argc, _TCHAR* argv[])
{
   float volume, jejari = 6.0;
   volume = volume_bola(jejari);
   cout<<"Volume bola dengan jari-jari = "<<jejari<<" adalah
   "<<volume;

   return 0;
}

float volume_bola(float r)
{
   return (4.0/3.0 * PI * r * r * r);
}
```

Jika program dijalankan akan diperoleh hasil seperti berikut:



Gambar 5.1 Tampilan hasil running program menggunakan fungsi

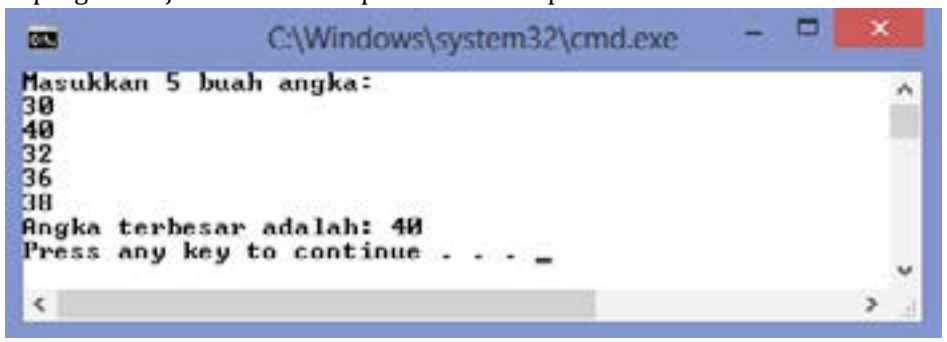
Contoh. 5.2
Diketahui sekumpulan data seperti table berikut:

No	Nilai
1	30
2	40
3	32
4	36
5	38

Menggunakan perintah perulangan, percabangan dan fungsi, buatlah program C++ untuk melacak nilai terbesar dari lima data nilai tersebut?

```
int _tmain(int argc, _TCHAR* argv[])
{
   double terbesar(double x, double y);
   double angka;
   double maks;
   int hitung;

   cout<<"Masukkan 5 angka:"<<endl;
   angka=0;
   maks=angka;
   for(hitung=0;hitung<5;hitung++)
   {
       cin>>angka;
       maks=terbesar(maks,angka);
   }
   cout<<"Angka terbesar adalah: "<<maks<<endl;

   return 0;
}
double terbesar(double x, double y)
{
   if(x>=y)
       return x;
   else
       return y;
}
```

Jika program dijalankan akan diperoleh hasil seperti berikut:



Gambar 5.2 Tampilan hasil running program mencari nilai terbesar

Contoh lain adalah program untuk menghitung turunan dengan fungsi dy/dx = (x - y)/2 menggunakan metode RK4 dengan nilai awal y_0 pada x_0 seperti berikut:

```
float dydx(float x, float y)
{
    return((x - y)/2);
}

float rungeKutta(float x0, float y0, float x, float h)
{
int n = (int)((x - x0) / h);

float k1, k2, k3, k4, k5;

float y = y0;
for (int i=1; i<=n; i++)
{
    k1 = h*dydx(x0, y);
    k2 = h*dydx(x0 + 0.5*h, y + 0.5*k1);
    k3 = h*dydx(x0 + 0.5*h, y + 0.5*k2);
    k4 = h*dydx(x0 + h, y + k3);

    // Update nilai y
    y = y + (1.0/6.0)*(k1 + 2*k2 + 2*k3 + k4);;

    // Update nilai x berikutnya
    x0 = x0 + h;
}

return y;
}
```

Pada main program cukup ditulis sebagai berikut:

```
int main()
{
    float x0 = 0, y = 1, x = 2, h = 0.2;
    cout<<" Nilai y pada x adalah:"<<rungeKutta(x0, y, x, h));
    return 0;
}
```

5.1 Menggunakan fungsi void

Void adalah sebuah fungsi (function) yang ada dalam sebuah bahasa pemrograman C. Fungsi ini juga disebut sebagai prosedur (procedure). Fungsi ini tidak mengembalikan nilai keluaran hasil proses tersebut, ini kenapa fungsi ini disebut void, secara harfiah berarti kosong.

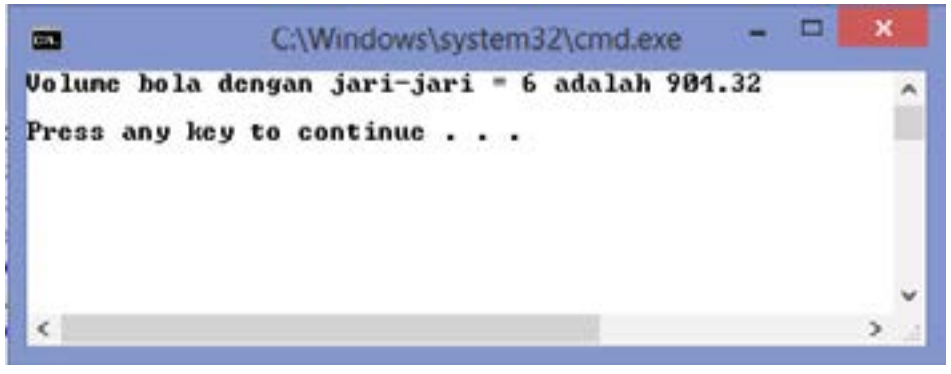
Ciri Ciri Void

Untuk ciri ciri dari function ini adalah :

1. Tidak membutuhkan atau memerlukan tipe data (data type) dalam deklarasi
2. Tidak memiliki nilai kembalian (return value) atau tidak menggunakan kata kunci return

```
#define PI 3.14

void volume_bola(float r);

int _tmain(int argc, _TCHAR* argv[])
{
    float jejari = 6.0;
    volume_bola(jejari);
    return 0;
}

void volume_bola(float r)
{
    float V;
    V=(4.0/3.0 * PI * r * r * r);
    cout<<"Volume bola dengan jari-jari = "<<r<<" adalah
    "<<V<<endl;
}
```

Jika program dijalankan akan diperoleh hasil seperti berikut:



Gambar 5.3 Tampilan hasil running program dengan fungsi Void

Hasil dari fungsi non void sama dengan fungsi void, yang membedakan void dan non void adalah cara penulisan dimana non void menggunakan kata"return" pada definisi fungsinya.

5.2 Variabel Lokal dan Variabel Global

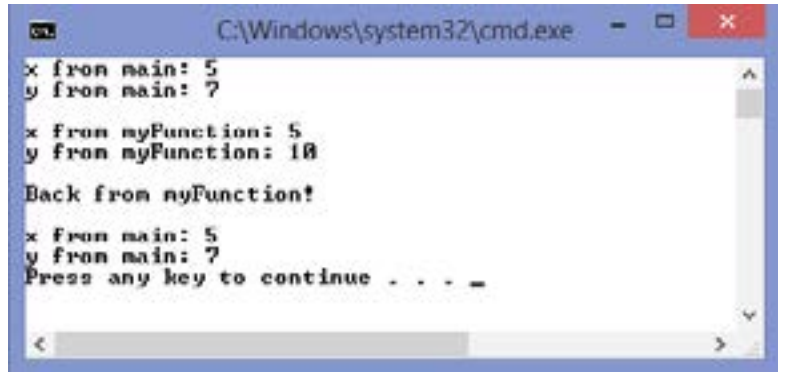
Sebuah variabel yang dapat dikenali oleh semua lingkungan dalam program, maka variabel tersebut harus dideklarasikan sebagai variabel yang bersifat global. Struktur program dalam bahasa C++ selalu ada fungsi utama dengan nama main() . Apabila kita mendeklarasikan sebuah variabel di atas fungsi main(), maka compiler akan menganggap variabel tersebut sebagai variabel global. Variabel global ini dapat digunakan pada beberapa fungsi / prosedur, Hal ini betujuan untuk menghemat penulisan, karena tidak perlu lagi berkali - kali menuliskan variabel yang sama pada beberapa fungsi / prosedur.

Variabel lokal adalah variabel yang hanya dikenali oleh sebuah fungsi / prosedur saja (hanya dikenali pada fungsi / prosedur tempat variabel tersebut dideklarasikan). Hal tersebut karena proses deklarasi variabel lokal dilakukan di dalam lingkup fungsi yang dimaksud. Misalnya 2 buah fungsi memiliki variabel lokal yang sama misal x, maka variabel x dalam kedua fungsi tidak akan saling mengenali sehingga tidak terjadi tumpang tindih atau kesalahan.

Kode program berikut memperlihatkan perbedaan antara variable lokal dengan variable global:

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

void myFunction(int y);

int x = 5, y = 7; //Global variable

int _tmain(int argc, _TCHAR* argv[])
{
   cout << "x from main: " << x << "\n";
   cout << "y from main: " << y << "\n\n";
   myFunction(10);
   cout << "Back from myFunction!\n\n";
   cout << "x from main: " << x << "\n";
   cout << "y from main: " << y << "\n";
   return 0;
}

void myFunction(int y)
{
   //int y = 10; // Local variable
   cout << "x from myFunction: " << x << "\n";
   cout << "y from myFunction: " << y << "\n\n";
}
```

Jika program dijalankan akan diperoleh hasil seperti berikut:



Gambar 5.9 Tampilan hasil running program global dan local variable

5.3 Parameter pass by value dan Parameter pass by reference

Dalam pemrograman C++, terdapat dua jenis perlewatan nilai pada fungsi yaitu berdasarkan alamat atau berdasarkan nilai pada fungsi tersebut, apabila perlewatan dilakukan berdasarkan alamat, maka parameter formal akan mengambil alamat pada parameter actual untuk dilakukan komputasi, sedangkan pada perlewatan

berdasarkan nilai maka parameter formal hanya akan mengambil data berdasarkan nilai pada parameter formal saja.

Contoh program pass by reference :

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

void myFunction(int &x, int &y)
{
   int x = x >> 2;
int y = y >> 2;
cout << "x inside function " << x << "\n";
   cout << "y inside function: " << y << "\n\n";

}

int _tmain(int argc, _TCHAR* argv[])
{
   int x = 5;
   int y = 10;
   cout << "x before function " << x << "\n";
   cout << "y before function: " << y << "\n\n";
   myFunction(x,y);
   cout << "x after function: " << x << "\n";
   cout << "y after function: " << y << "\n";
   return 0;
}
```

Dari program diatas akan menghasilkan output :

```
x before function 5
y before function: 10

x inside function 1
y inside function: 2

x after function 1
y after function: 2
```


Sedangkan apabila dilakukan eksekusi dengan parameter pass by value dengan menghilangkan tanda (&) pada parameter formal fungsi, maka program akan menghasilkan output :

```
x before function 5
y before function: 10

x inside function 1
y inside function: 2

x after function 5
y after function: 10
```

Terlihat pass by value tidak merubah nilai x dan y meskipun telah dilakukan eksekusi fungsi pada fungsi main, namun pada pass by reference karena alamat yang dilewatkan adalah alamat x dan y, maka apabila nilai yang ada pada alamat x dan y berubah, maka nilai yang ditampilkan juga akan berubah

Soal Latihan:

Buatlah program dalam bahasa C++ menggunakan fungsi untuk:

1. Menghitung luas segi tiga dengan inputan **panjang alas** dan **tinggi**
2. Menghitung luas permukaan kubus dengan inputan **panjang sisi** kubus
3. Menghitung luas permukaan tabung tertutup, dengan inputan **jari-jari** dan **tinggi** tabung. Dimana:
   Luas Permukaan Tabung = 2 x luas alas + Luas selimut tabung
   = 2 (π r^2)+ 2 π r t = 2 π r (r + t)

BAB 6

Variabel ARRAY dan Aplikasinya

6.1 Variabel Array

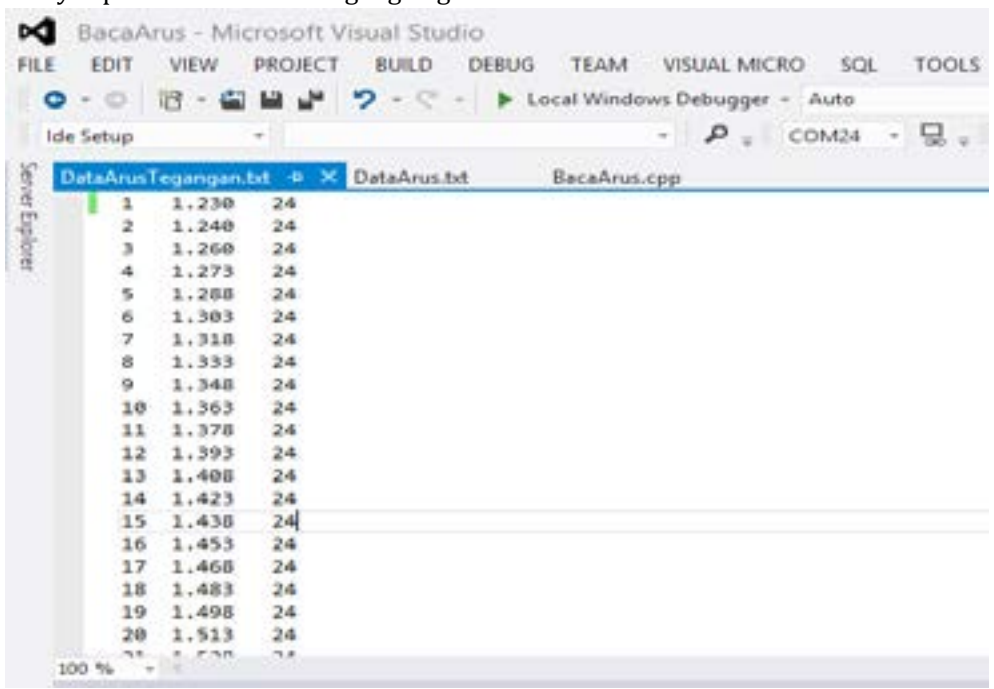
Array adalah kumpulan data yang terstruktur dengan nama variable sama dan bertipe sama tetapi mempunyai indeks yang berbeda. Setiap variable dapat berisi data yang sama atau berbeda tetapi tipe data harus sama. Urutan indeks array dimulai dari hitungan 0,1,2 dst. Tipe data variable array pada umumnya adalah char , string, int, float dan double.

Variable array harus dideklarasikan terlebih dahulu, sebelum digunakan. Bentuk deklarasi sebagai berikut :

```
Tipe_data  nama_array[dimensi_array]
```

Dimensi array menyatakan banyaknya data menyatakan indeks variable. Misalnya int A[20], artinya variable array A dapat menampung 20+1 data dengan tipe integer dengan nama sama tetapi indek berbeda-beda.

Perhatikan contoh pendeklarasian dan penggunaan variable array untuk menyimpan data arus dan tegangan gambar 6.1 berikut:



Gambar 6.1 Data arus dan tegangan yang disimpan dalam .txt file

Program untuk membaca data arus tegangan dari file adalah sebagai berikut:

```
int no[101],i;
float Arus[101],Teg[101],Daya[101],Rata2;

ifstream fin ("DataArusTegangan.txt",ios::in);

for(i=1;i<=100;i++)
   fin>>no[i]>>Arus[i]>>Teg[i];

for(i=1;i<=100;i++)
   Daya[i]=Arus[i]*Teg[i];

// Tampilan hasil pembacaan
for(i=1;i<=100;i++)
   cout<<no[i]<<"  "<<Arus[i]<<"  "<<Teg[i]<<"  "<<Daya[i]
   <<endl;
```


Sedangkan untuk menghitung nilai rata-rata arus adalah:

```
Rata2=0;
   for(i=1;i<=100;i++)
       Rata2=Rata2+Arus[i];

Rata2=Rata2/100;
cout<<" Nilai rata-rata arus adalah "<<Rata2<<endl;
```

Variabel array dalam C++ terdiri dari tiga jenis dimensi yaitu:

1. Array Berdimensi Satu. Array satu dimensi merupakan kelompok data yang hanya terdiri dari bari satu baris atau satu kolom saja.

   Deklarasi array :
   Tipe_array nama_array[ukuran]

2. Array Berdimensi Dua. Array dua dimensi mempunyai 2 buah indeks dan sering digambarkan sebagai sebuah matriks, merupakan perluasan dari array satu dimensi. Indeks pertama menunjukkan baris dan indeks kedua menunjukan kolom.

   Deklarasi array :
   Tipe_array nama_array [baris][kolom]

3. Array Berdimensi Tiga. Array berdimensi tiga tersusun dari 3 sisi atau 3 indeks di mana indeks pertama menunjukkan baris ,indeks kedua menunjukkan kolom dan lebar menunjukkan tinggi. Array dimensi tiga pada umumnya digunakan untuk kasus ruang 3 dimensi seperti kubus, balok, dll.

   Deklarasi array :
   Tipe_array nama_array [ukuran 1][ukuran 2] [ukuran 3]

6.2 Aplikasi Array untuk Operasi Matrik

Matrik adalah susunan bilangan atau elemen matrik dalam bentuk persegi dan diapit oleh tanda kurung siku "[ ]". Suatu matrik dinotasikan dengan huruf kapital tebal. Sebuah matrik mempunyai ukuran yang disebut dengan ordo sebagai indeks pada notasi matrik. Matrik berordo n x m berarti matrik tersebut terdiri dari n baris dan m kolom. Selain itu matrik memiliki elemen matrik yang memiliki indeks sebagai tanda posisi elemen didalam matrik. Elemen matrik berindek i,j berarti bilangan tersebut terletak pada barik ke-i dan kolom ke-j suatu matrik.

$$A = \begin{bmatrix} a_1 & a_{12} & \dots & a_{1n} \\ a_{12} & a_2 & \dots & a_{2n} \\ \dots & \dots & a_{n-1n-1} & \dots \\ a_{n1} & a_{n2} & \dots & a_n \end{bmatrix}$$

A adalah notasi matriks sedang a_{nm} adalah elemen matriks.

Dalam bidang rekayasa biasanya variable arry digunakan untuk operasi matrik dan penyelesaian persamaan linear berikut:

Ax = b (6.1)

Dimana:
A adalah matriks berukuran n x n dengan elemen matrik a_{ij}
x adalah matriks berukuran n x 1 dengan elemen matrik x_j
b adalah matriks berukuran n x 1 (disebut juga vektor kolom) dengan elemen matrik b_j

Penyelesaian numerik sistem persamaan linear ***Ax = b*** dapat dilakukan dengan dua cara, yaitu metode langsung dan metode tidak langsung. Penyelesaian Sistem Persamaan Linear secara numerik menggunakan metode langsung dapat diklasifikasikan sebagai berikut :

1. Metode Eliminasi Gauss, prinsipnya: merupakan operasi eliminasi elemen-elemen pembentuk matriknya sedemikian rupa sehingga dapat terbentuk matriks segitiga atas, dan akhirnya solusinya diselesaikan menggunakan teknik substitusi mundur.
2. Dekomposisi LU, prinsipnya: melakukan dekomposisi matriks *A* terlebih dahulu sehingga dapat terbentuk matriks-matrik segitiga atas dan bawah, kemudian secara mudah dapat diselesaikan menggunakan teknik substitusi maju atau mundur.
3. Invers Matrik, untuk mencari selesaian SPL digunakan invers matrik

6.2.1. Penjumlahan dan pengurangan matriks

Suatu matrik $A = [a_{ij}]$ dan matrik $B = [b_{ij}]$ dengan dimensi *m´n*, maka untuk operasi penjumlahan atau pengurangan $(A \pm B)$ dari kedua matriks tersebut, menghasilakan suatu matriks $C = [c_{ij}]$ dengan dimensi *m´n*, dimana setiap elemen matriks *C* adalah jumlah atau selisih dari elemen-elemen yang berkaitan dari *A* dan *B*.

$$\begin{bmatrix} a_1 & a_{12} & \dots & a_{1m} \\ a_{12} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_m \end{bmatrix} + \begin{bmatrix} b_1 & b_{12} & \dots & b_{1m} \\ b_{12} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_m \end{bmatrix} = \begin{bmatrix} c_1 & c_{12} & \dots & c_{1m} \\ c_{12} & c_{22} & \dots & c_{2m} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_m \end{bmatrix}$$

$c_{11} = a_{11} + b_{11}$
$c_{12} = a_{12} + b_{12}$
dan seterusnya.

Untuk operasi pengurangan matrik:

$$\begin{bmatrix} a_1 & a_{12} & \dots & a_{1m} \\ a_{12} & a_{22} & \dots & a_{2m} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_m \end{bmatrix} - \begin{bmatrix} b_1 & b_{12} & \dots & b_{1m} \\ b_{12} & b_{22} & \dots & b_{2m} \\ \dots & \dots & \dots & \dots \\ b_{n1} & b_{n2} & \dots & b_m \end{bmatrix} = \begin{bmatrix} c_1 & c_{12} & \dots & c_{1m} \\ c_{12} & c_{22} & \dots & c_{2m} \\ \dots & \dots & \dots & \dots \\ c_{n1} & c_{n2} & \dots & c_m \end{bmatrix}$$

$c_{11} = a_{11} - b_{11}$
$c_{12} = a_{12} - b_{12}$
dan seterusnya.

Kode program perjumlahan dan pengurangan matrik dimulai dari:

```
//Perulangan untuk inputan Matriks A
   for(i=1;i<=n;i++){
       for (j=1;j<=n;j++){
              cout<<"A["<<i<<"]["<<j<<"] =";cin >> a[i][j];
       }
   }   cout<<endl<<endl;

   //Perulangan untuk inputan Matriks B
   for(i=1;i<=n;i++){
       for (j=1;j<=n;j++){
              cout<<"B["<<i<<"]["<<j<<"] =";cin >> b[i][j];
       }
   }
```

Program untuk perjumlahan matrik adalah:

```
   for (i=1;i<=n;i++)
       for (j=1;j<=n;j++)
              c[i][j]=a[i][j]+b[i][j];
```

Sedangkan program untuk pengurangan matrik adalah:

```
for (i=1;i<=n;i++)
    for (j=1;j<=n;j++)
          c[i][j]=a[i][j]-b[i][j];
```

Selanjutnya tambahkan program untuk menampilkan hasil perjumlahan atau pengurangan matrik yang ditamung dalam variable array C berikut:

```
for(i=1;i<=n;i++)
    for(j=1;j<=n;j++){
          cout<<" C["<<i<<"]["<<j<<"]= "<<c[i][j]<<endl;
    }
```

6.2.2. Perkalian Matrik

A . x = b

$$\begin{bmatrix} a_1 & a_2 & \dots & a_{1n} \\ a_2 & a_2 & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_n \end{bmatrix} \quad \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}$$

Cara pengoperasian perkalian matriks dengan vektor :

$$b_1 = a_{11}x_1 + a_{12}x_2 + \dots + a_{1n}x_n$$

$$= \sum_{j=1}^{n} a_{1j}.x_j$$

$$b_i = \sum_{j=1}^{n} a_{ij}.x_j$$

untuk: i = 1, 2, 3, ..., n;

$b_i = 0$

j = 1, 2, 3, ..., n;

$b_i = b_i + a_{ij} . x_j$

Program lengkap perkalian matrik adalah:

```
float A[10][10],b[10],c[10];
int n,i,j;
cout<<" Tentukan Ordo Matrik A = ";
cin>>n;
for(i=1;i<=n;i++)
```



```
    for(j=1;j<=n;j++){
          cout<<" A["<<i<<"]["<<j<<"]= ";
          cin>>A[i][j];
    }
for(i=1;i<=n;i++){
    cout<<" b["<<i<<"]= ";
    cin>>b[i];
}
for(i=1;i<=n;i++){
    c[i]=0;
    for(j=1;j<=n;j++){
          c[i]=c[i]+A[i][j]*b[j];
    }
}
for(i=1;i<=n;i++){
    cout<<" c["<<i<<"]= "<<c[i]<<endl;
}
```

6.3 Aplikasi Invers Matrik

Perhatikan sistem persamaan linear berikut:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \bullet \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} \qquad (6.2)$$

Untuk mengitung nilai vector x dilakukan proses berikut:
Bentuk persamaan baru dimana persamaan baru berbentuk sama dengan persamaan asal akan tetapi x_2 dan b_2 telah dipertukarkan menjadi:

$$\begin{bmatrix} a'_{11} & a'_{12} \\ a'_{21} & a'_{22} \end{bmatrix} \bullet \begin{bmatrix} x_1 \\ b_2 \end{bmatrix} = \begin{bmatrix} b_1 \\ x_2 \end{bmatrix} \qquad (6.3)$$

dengan:

$a'_{11} = a_{11} - a_{12}\, a_{22}^{-1} a_{21}$

$a'_{12} = a_{12}\, a_{22}^{-1}$

$a'_{21} = - a_{22}^{-1}\, a_{21}$

$a'_{22} = a_{22}^{-1}$

Proses yang sama juga dapat dilakukan untuk mempertukarkan x_1 dan b_1.
Algoritma invers matrik tersebut dikenal dengan metode Shipley and Coleman dimana kode program proses invers matriknya adalah:

```
int i,j,k,l;
 for (i=1;i<=n;i++)
 {
   a[i][i] = 1.0/a[i][i];
   for (j=1;j<=n;j++)
   {
      if (j!=i)
      {
         a[j][i] = a[j][i] * a[i][i];
         for (k=1;k<=n;k++)
         {
            if (k!=i)
            {
               a[j][k] = a[j][k] - a[j][i]*a[i][k];
               if (j==n)
               {
                  a[i][k] = -a[i][i]* a[i][k];
               }
            }
         }
      }
   }
}
k = n-1;
for (l = 1;l<=k;l++)
{
  a[n][l] = -a[n][n]*a[n][l];
}

for(i=1;i<=n;i++)
   for(j=1;j<=n;j++)
       cout<<" a["<<i<<"]["<<j<<"] = "<<a[i][j]<<endl;
```

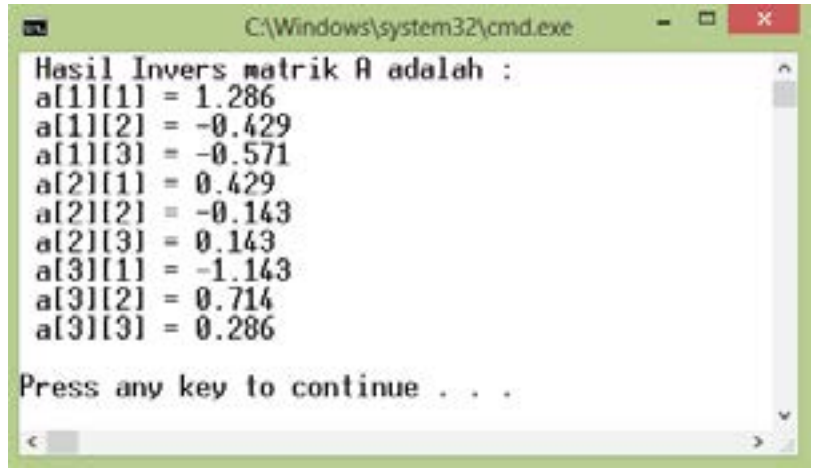

Jika Matrik A adalah:

$$A = \begin{bmatrix} 1 & 2 & 1 \\ 2 & 2 & 3 \\ -1 & 3 & 0 \end{bmatrix}$$

dan vektor b adalah:

$$b = \begin{bmatrix} 0 \\ 3 \\ -4 \end{bmatrix}$$

Maka untuk menentukan vektor x, diperlukan invers matrik A menggunaan algoritma invers matrik tersebut. Hasil invers matrik adalah seperti diperlihatkan pada gambar 2.2



Gambar 6.2 Hasil invers matrik

Setelah invers matrik A diperoleh, langkah selanjutnya dalah perkalian matrik untuk memperoleh vektor x, seperti berikut:

$$[x] = [A]^{-1}\,[b] \qquad (6.4)$$

Maka diperoleh:

$$x = \begin{bmatrix} 1 \\ -1 \\ 1 \end{bmatrix}$$

Pada beberapa aplikasi diperlukan matrik A dinyatakan sebagai bilangan komplek dimana setiap elemen matrik terdiri dari bagian real dan bagian imajiner. Untuk melakukan invers komplek algoritma Shipley and Coleman dapat dikembangkan menjadi sperti berkut:

```
int i,j,k,l;
   double temp1;
   for (i=1;i<=n;i++)
    {
    temp1 = g[i][i]/(g[i][i]*g[i][i]+b[i][i]*b[i][i]);
b[i][i] = -b[i][i]/(g[i][i]*g[i][i]+b[i][i]*b[i][i]);
  g[i][i] = temp1;
    for (j=1;j<=n;j++)
      {
      if (j!=i)
        {
        temp1 = (g[j][i]*g[i][i])-(b[j][i]*b[i][i]);
  b[j][i] = (g[j][i]*b[i][i])+(b[j][i]*g[i][i]);
  g[j][i] = temp1;
        for (k=1;k<=n;k++)
          {
          if (k!=i)
            {
            temp1 = g[j][k]-g[j][i]*g[i][k]+b[j][i]*b[i][k];
  b[j][k] = b[j][k]-g[j][i]*b[i][k]-b[j][i]*g[i][k];
g[j][k] = temp1;
          if (j==n)
            {
            temp1 = -g[i][i]*g[i][k]+b[i][i]*b[i][k];
  b[i][k] = -b[i][i]*g[i][k]-g[i][i]*b[i][k];
  g[i][k] =temp1;
            }
          }
        }
      }
    }
  }
  k = n-1;
  for (l = 1;l<=k;l++)
    {
    temp1 = -g[n][n]*g[n][l]-b[n][n]*b[n][l];
  b[n][l] = -b[n][n]*g[n][l]-g[n][n]*b[n][l];
  g[n][l] = temp1;
    }
```

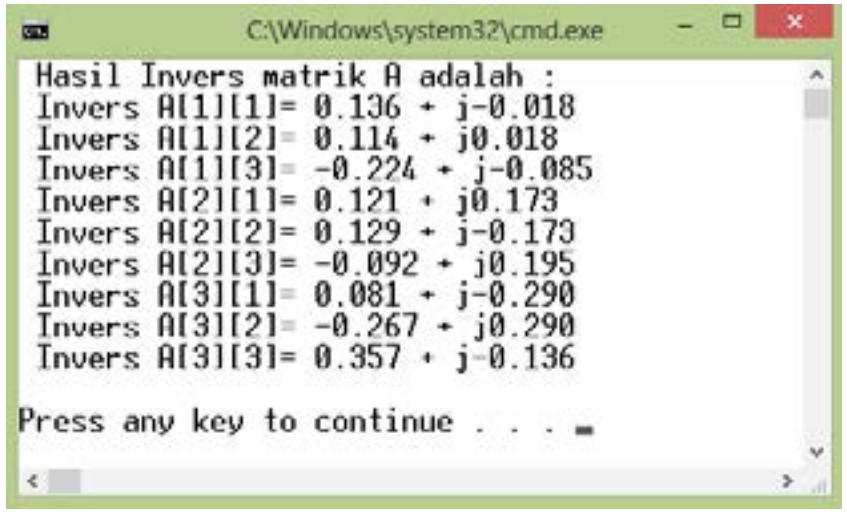

Jika Matrik A adalah:

$$A = \begin{bmatrix} 3+j2 & 2-j & 1+j \\ 2-j & 5-j & 3-2j \\ -1+j & 3-j2 & 2-j \end{bmatrix}$$

dan vektor b adalah:

$$b = \begin{bmatrix} 2+j \\ 3-2j \\ -4+j \end{bmatrix}$$

Maka untuk menentukan vektor x, diperlukan invers komplek dari matrik A menggunaan algoritma invers komplek di atas. Hasil invers matrik A adalah seperti diperlihatkan pada gambar 6.3



Gambar 6.3 Hasil invers matrik komplek

Input data matrik komplek terdiri dari bagian real dan imajiner seperti berikut:
Bagian real:

```
a[1][1] = 3; a[1][2] = 2; a[1][3] = 1;
a[2][1] = 2; a[2][2] = 5; a[2][3] = 3;
a[3][1] =-1; a[3][2] = 3; a[3][3] = 2;
```

Bagian imajiner :

```
b[1][1] = 2; b[1][2] =-1; b[1][3] = 1;
b[2][1] =-1; b[2][2] =-1; b[2][3] =-2;
b[3][1] = 1; b[3][2] =-2; b[3][3] =-1;
```

Kemudian program dicompile dan jalankan untuk memperoleh output invers matrik A. Setelah invers matrik A diperoleh, langkah selanjutnya dalah perkalian matrik untuk memperoleh vektor x, seperti berikut:

```
[x] = [A]-1 [b]
```

Maka diperoleh:

$$x=\begin{bmatrix} 1.651+j0.0 \\ 0.283-j1.8 \\ -1.018+j1.2 \end{bmatrix}$$

Program untuk tampilan invers matrik komplek dapat juga menggunakan coding berikut:

```
cout<< " Diagonal elements : "<<endl;
for (i=1;i<=nb;i++)
  {
  for (j=1;j<=nb;j++)
   {
     if (i==j)
   cout<<" Z["<<i<<"]["<<j<<"]= "<<g[i][j]<<" +j "<<b[i][j]<<endl;
   }
  }
cout<< " Off diagonal elements : "<<endl;
for (i=1;i<=nb;i++)
  {
  for (j=1;j<=nb;j++)
   {
     if (i!=j)
   cout <<" Z["<<i<<"]["<<j<<"]= "<<g[i][j]<<" +j "<<b[i][j]<<endl;
   }
  }
```


6.4 Metode Iteratif

Proses penyelesaian sistem persamaan linear dengan metode iteratif secara numerik harus ada tebakan awal, skema iterasi dan kriteria penghentian. Proses iterasi dilakukan sampai dicapai suatu nilai selisih yang sekecil mungkin sesuai dengan toleransi yang diberikan. Setiap harga x_i yang baru dihasilkan segera dapat segera dipakai pada persamaan berikutnya untuk menentukan harga x_{i+1} yang lainnya.

Untuk menyelesaikan sistem persamaan linier *A x = b* dengan *A* adalah matriks koefisien *n x n, b* vector konstan *n x 1*, dan *x* vektor *n x 1* yang perlu dicari, maka proses yang diulang adalah persamaan umum (6.5) berikut:

$$x_i^{(k+1)} = \frac{b_i - \sum_{j=1}^{i-1} a_j x_j^{k+1} - \sum_{j=i+1}^{n} a_j x_j^k}{a_i} \qquad (6.5)$$

Metode ini dikenal dengan metode Gauss Seidel.

Contoh:

Diketahui sistem persamaan linear Ax = b yaitu:

$$10x_1 - x_2 + 2x_3 = 6$$
$$-x_1 + 11x_2 - x_3 + 3x_4 = 25$$
$$2x_1 - x_2 + 10x_3 - x_4 = -11$$
$$3x_2 - x_3 + 8x_4 = 15$$

Nilai awal $(0\ 0\ 0\ 0)^T$

Hitung $[x_1, x_2, x_3, x_4]$ menggunakan Metode Iteratif Gauss Seidel

Jawab:

$$x_1 = \frac{x_2}{10} - \frac{x_3}{5} + \frac{3}{5}$$

$$x_2 = \frac{x_1}{11} + \frac{x_3}{11} - \frac{3x_4}{11} + \frac{25}{11}$$

$$x_3 = \frac{-x_1}{5} + \frac{x_2}{10} + \frac{x_4}{10} - \frac{11}{10}$$

$$x_4 = \frac{-3x_2}{8} + \frac{x_3}{8} + \frac{15}{8}$$

Iterasi 1,
Hampiran pertama terhadap penyelesaian SPL tersebut adalah

$$x_1 = \frac{3}{5} = 0.6$$

$$x_2 = \frac{0.6}{1} + \frac{7}{1} = 2.3727$$

$$x_3 = \frac{-0.6}{5} + \frac{2.2727}{10} - \frac{1}{10} = -1.1$$

$$x_4 = \frac{-3.2.2727}{8} + \frac{-1.1}{8} + \frac{5}{8} = 1.8750$$

Hasil iterasi pertama ini menjadi nilai awal proses iterasi berikutnya. Proses yang sama dilankutkan untuk iterasi ke 2. Hasil perhitungan dapat dilihat pada tabel berikut:

Iterasi	x1	x2	x3	x4
1	0.6	2.32727	-0.98727	0.878864
2	1.03018	2.03694	-1.01446	0.984341
3	1.00659	2.00356	-1.00253	0.998351
4	1.00086	2.0003	-1.00031	0.99985
5	1.00009	2.00002	-1.00003	0.999988
6	1.00001	2	-1	0.999999
7	1	2	-1	1

Menggunakan selisih < 0.00001, maka setelah iterasi ke-7 diperoleh hampiran penyelesaian *x* = $(1\ 2\ -1\ 1)^T$. Nilai ini sudah sama dengan nilai sebenarnya, *yaitu x =* $(1\ 2\ -1\ 1)^T$.

Berdasarkan langkah-langkah diatas dapat disusun algoritma Metode Gauss Seidel sebagai berikut:

Masukan: a_{ij} ; i.j = 1, 2, …., n
b_i ; i = 1,2,……,n
x_i ; i = 0
Eps

Langkah-langkah:

1. Untuk i = 1, 2, 3, ..., n.
   - $x_i^{baru} = b_i$
   - Untuk j = 1,2,3,...,n
     - Jika j ≠i,
       - $x_i^{baru} = x_i^{baru} - \sum a_j x_j^0$
   - $x_i^{baru} = \frac{x_i^{baru}}{a_i}$
   - Deltax = $|x_i^{baru} - x_i^0|$
   - Update nilai $x_i^0 = x_i^{baru}$
2. Jika maksimum $|x_i^{k+1} - x_i^k| <$ Eps, Selesai, Print hasil
3. Kembali ke langkah 1

Keluaran: x_i ; untuk i = 1,2,…,n
Contoh 2.4:
Berdasarkan algoritma Gauss Seidel selesaikan SPL berikut:

$$\begin{bmatrix} 3 & 2 & 1 \\ 2 & 4 & 1 \\ 0 & 3 & 5 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} 4 \\ 5 \\ -2 \end{bmatrix}$$

Nilai awal $[x_1\ x_2\ x_3] = [0\ 0\ 0]$
Jawab :
Iterasi 1
i = 1
$x_1^{baru} = b_1 = 4$
j = 2
$x_1^{baru} = x_1^{baru} - a_{12} . x_2^0 = 4 - 2\ (0) = 4$
j = 3
$x_1^{baru} = x_1^{baru} - a_{13} . x_3^0 = 4 - 1.(0) = 4$
$x_1^{baru} = x_1^{baru} / a_{11} = 4/3$
Deltax = $|x_1^{baru} - x_1^0|$

$= |4/3 - 0| = 4/3$

Update $x_1^0 = 4/3$
i = 2

$x_2^{baru} = b_2 = 5$

j = 1

$x_2^{baru} = x_2^{baru} - a_{21} . x_1^0 = 5 - 2\ (4/3)$

j = 3

$x_2^{baru} = x_2^{baru} - a_{23} . x_3^0 = 5 - 2\ (4/3) - 1.(0)$

$x_2^{baru} = x_2^{baru} / a_{22} = (5 - 2\ (4/3) - 1.(0))/4$

$= 7/12$

Deltax $= |x_2^{baru} - x_2^0|$

$= \left|7/12 - 0\right| = 7/12$

Update $x_2^0 = 4/3 = 7/12$

i = 3

$x_3^{baru} = b_3 = -2$

j = 1

$x_3^{baru} = x_3^{baru} - a_{31} . x_1^0 = -2-0(4/3)$

j = 2

$x_3^{baru} = x_3^{baru} - a_{32} . x_2^0 = -2-0(4/3)-3(7/12)$

$x_3^{baru} = x_3^{baru} / a_{33} = -2-0(4/3)-3(7/12)/5$

$= -45/(12.5)$

$= -9/12 = -3/4$

Deltax $= |x_3^{baru} - x_3^0|$

$= \left|-3/4 - 0\right| = -3/4$

Update $x_3^0 = -3/4$

Selanjunya periksa maksimum Deltax apakah sudah < Epsilon, Jika sudah tampilkan hasil perhitungan x_i^{baru} untuk i = 1,2,3, jika tidak lanjutkan proses iterasi 2.

Proses yang sama dilankutkan untuk iterasi ke 2. Hasil perhitungan dapat dilihat pada tabel berikut:

Iterasi	x1	x2	x3
1	1.33333	0.583333	-0.75
2	1.19444	0.840278	-0.90417
3	1.07454	0.938773	-0.96326
4	1.02857	0.97653	-0.98592
5	1.01095	0.991003	-0.9946
6	1.0042	0.996551	-0.99793
7	1.00161	0.998678	-0.99921
8	1.00062	0.999493	-0.9997
9	1.00024	0.999806	-0.99988

Iterasi	x1	x2	x3
10	1.00009	0.999925	-0.99996
11	1.00003	0.999971	-0.99998
12	1.00001	0.999989	-0.99999
13	1.00001	0.999996	-1

Menggunakan selisih < 0.00001, maka setelah iterasi ke-13 diperoleh hampiran penyelesaian $x = (1.00001\ 0.999996\ -1)^T$. Nilai ini sudah mendekati nilai sebenarnya, *yaitu* $x = (1\ 1\ -1)^T$.

Program C++

Metode Gauss Seidel

```
S=1.0;
while (abs( S ) > 0.00001){
   for ( i = 0; i < n; i ++ ){
   Xbaru[i] = b[i];
       for ( j = 0; j < n; j ++ ){
             if ( j != i )
             Xbaru[i] = Xbaru[i] - a[i][j] * X[j];
       }
       Xbaru[i] = Xbaru[i] / a[i][i];
       DeltaX[i] = Xbaru[i] - X[i];
       X[i] = Xbaru[i]; // update vektor X
   }
   Deltamax=0.0;
   for (i=0;i<n;i++) {
       if (abs(DeltaX[i])>Deltamax)
             Deltamax=abs(DeltaX[i]);
   }
   S=Deltamax;
}
```

Contoh input data:

```
float a[3][3]={{3,2,1},
               {2,4,1},
                {0,3,5}};
float b[3]={4,5,-2};
float X[3]={0,0,0};
```

BAB 7

Pointer

7.1. Pengertian

Pointer adalah variabel yang berisi alamat memori suatu variabel, berbeda dengan variable biasa yang berisi nilai tertentu. Dengan kata lain, pointer berisi alamat dari variable yang mempunyai nilai tertentu. Dengan demikian, ada variabel dalam proses yang secara langsung menunjuk ke suatu nilai tertentu, dan ada variabel yang tidak langsung menunjuk ke nilai atau disebut variabel pointer. Terdapat 2 macam operator pointer yang disediakan dalam bahasa C++ yaitu Operator dereference (&) dan Operator reference (*).

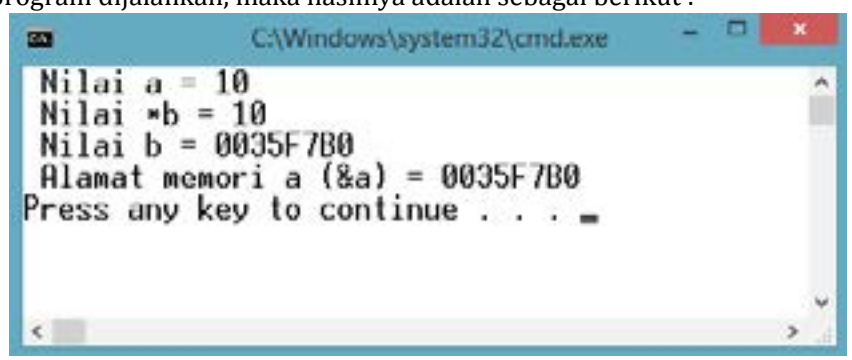
Fungsi dari Operator dereference (&) dan Operator reference (*) adalah, Operator dereference digunakan Jika kita mempunyai sebuah variable dengan tipe data tertentu, maka untuk mendapatkan alamat memori dari variable tersebut adalah dengan menggunakan operator & (Operator dereference). Apabila dideklarasikan sebuah variabel dengan tipe data tertentu, maka untuk mendapatkan alamat dari variabel tersebut adalah dengan menggunakan operator &. Alamat inilah yang kemudian akan disimpan kedalam variabel yang bertipe pointer. Sedangkan operator reference digunakan untuk mendeklarasikan variable sebagai pointer, dalam mendeklarasikan sebuah variabel menjadi pointer kita hanya menambahkan tanda

asterisk (*) di depan nama variable tersebut. Operator reference (*) juga digunakan untuk menampilkan (mengeluarkan) nilai dari alamat memori yang di tunjuk.
Adapun bentuk umum dari pernyataan variabel pointer dalam C++ adalah :
Tipe_data *variabel_name;
Int *p;
float *A;
Dengan :
Tipe_data adalah tipe dasar pointer
Variabel_name adalah nama variabel pointer
* adalah variabel pada alamatnya yang ditentukan oleh operand.

Selanjutnya untuk menyatakan alamat memori dapat menggunakan operator &.
Contoh 3.1:
Perhatikan kode program berikut:

```
int _tmain(int argc, _TCHAR* argv[])
{
   int a = 10, *b;
   b = &a;
   cout<<" Nilai a = "<<a<<endl;
   cout<<" Nilai *b = "<<*b<<endl;
   cout<<" Nilai b = "<<b<<endl;
   cout<<" Alamat memori a (&a) = "<<&a<<endl;

   return 0;
}
```

Bila program dijalankan, maka hasilnya adalah sebagai berikut :

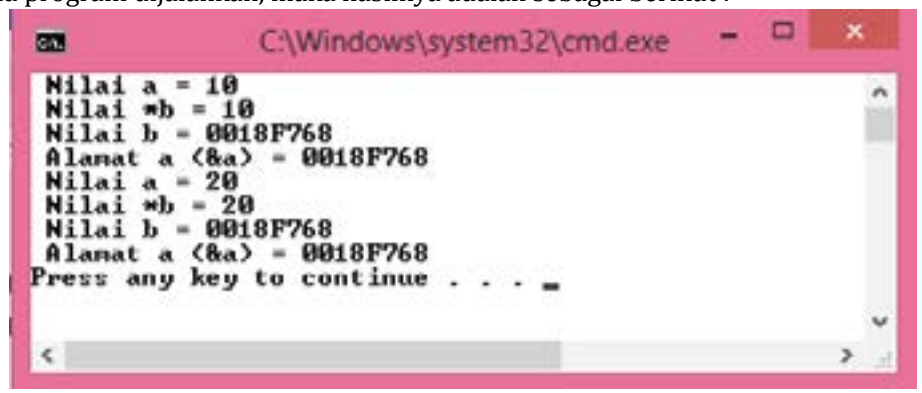


Gambar 7.1 Hasil running program pointer

Selanjutnya jika nilai pada alamat memori yang ditunjuk variable b diberikan 20, maka nilai a menjadi 20 perhatikan kode program berikut:

```
int _tmain(int argc, _TCHAR* argv[])
{
int a=10,*b;
   b=&a;
   cout<<" Nilai a = "<<a<<endl;
   cout<<" Nilai *b = "<<*b<<endl;
   cout<<" Nilai b = "<<b<<endl;
   cout<<" Alamat a (&a) = "<<&a<<endl;
   *b=20;
   cout<<" Nilai a = "<<a<<endl;
   cout<<" Nilai *b = "<<*b<<endl;
   cout<<" Nilai b = "<<b<<endl;
   cout<<" Alamat a (&a) = "<<&a<<endl;
   return 0;
}
```

Bila program dijalankan, maka hasilnya adalah sebagai berikut :



Gambar 7.2 Hasil running program

Dengan menggunakan pointer dan operator dereferencing, nilai dari suatu variabel dalam memori dapat dengan mudah dimanipulasi. Hal ini membuat pointer menjadi sangat powerful dalam pemrograman ANSI/C, terutama dalam pengiriman variabel-variabel dengan memori besar via parameter fungsi. Tetapi kesalahan dalam menggunakan pointer juga dapat mengakibatkan bugs atau error. Oleh karena itu, penggunaan pointer dalam pemrograman harus dilakukan secara benar dan hati-hati.

Berikut contoh aplikasi menggunakan variable pointer:

```
cout<<"PROGRAM PENGOPERASIAN 10 NILAI"<<endl<<endl;
int nilai1, nilai2, nilai3, nilai4, nilai5, nilai6, nilai7, nilai8,
   nilai9, nilai10;
float average, sum;
int *a,*b,*c,*d,*e,*f,*g,*h,*i,*j, k;

cout<<"Masukan nilai 1==> ";
cin>> nilai1;
cout<<"Masukan nilai 2==> ";
cin>> nilai2;
cout<<"Masukan nilai 3==> ";
cin>> nilai3;
cout<<"Masukan nilai 4==> ";
cin>> nilai4;
cout<<"Masukan nilai 5==> ";
cin>> nilai5;
cout<<"Masukan nilai 6==> ";
cin>> nilai6;
cout<<"Masukan nilai 7==> ";
cin>> nilai7;
cout<<"Masukan nilai 8==> ";
cin>> nilai8;
cout<<"Masukan nilai 9==> ";
cin>> nilai9;
cout<<"Masukan nilai 10==> ";
cin>> nilai10;

a=&nilai1;
b=&nilai2;
c=&nilai3;
d=&nilai4;
e=&nilai5;
f=&nilai6;
g=&nilai7;
h=&nilai8;
i=&nilai9;
j=&nilai10;

sum=*a+*b+*c+*d+*e+*f+*g+*h+*i+*j;
average=sum/10;

cout<<"Jumlah dari 10 nilai tersebut adalah =  "<<sum<<endl;
cout<<"Rata-rata dari 10 nilai tersebut adalah =
  "<<average<<endl<<endl<<endl;
```


7.2 Pointer Array

Suatu array berikut:

A[4] artinya ada elemen array A[0], A[1], A[2] dan A[3]

7.3 Alokasi Memori Dinamis

Pendeklarasian memori dinamis dalam Bahasa C terdiri dari empat fungsi yaitu: calloc, malloc, realloc dan free, tetapi dalam C++ menggunakan dua operator yaitu new dan delete.

Matrik satu dimensi

Membuat pointer b.

```
double *b;
b=(double*)calloc(n+1,sizeof(double));
```

Matrik dua dimensi

```
double **a;
try
{
  a = new  double*[n];          // Setup baris.
  for (int i = 1; i < (n+1); i++)
  a[i] = new double[n];         // Setup kolom
}
catch (std::bad_alloc)          // Blok jika terjadi bad_alloc.
{
     cout << "Could not allocate";
     exit(-1);
}
```

Sedangakan dalam bahasa C++:

Contoh:

Variable array b

```
int *b;

b = new int[20];

delete [] b;
```

Sedangkan untuk matrik 2 dimensi:

```
Double **A;
try
    {
      A = new  double*[n];
      for (int i = 1; i < (n+1); i++)
      A[i] = new double[n];
    }
    catch (std::bad_alloc)
    {
         cout << "Could not allocate";
         exit(-1);
    }
```

Atau dapat juga menggunakan:

```
double** A= new double*[n];
   for(i=0;i<=n;i++)
       A[i]=new double[n];
```

untuk menghapus memori:

```
for (int i = 0; i < n; ++i)
    delete [] A[i];
delete [] A;
```


BAB 8

Pemrograman Arduino

8.1 Pengertian

Arduino merupakan piranti keras mikrokontroller yang dapat berfungsi untuk membaca nilai masukan pada sensor, mengendalikan motor, maupun untuk mengirimkan pesan teks pada jaringan internet. Arduino banyak digunakan karena pengoperasian yang mudah serta bersifat open source dan juga menggunakan bahasa pemrograman turunan yang telah banyak dipakai yaitu C++.

Gambar 8.1 Arduino Uno

Dalam menggunakan Arduino, terlebih dahulu harus memiliki mikrokontroller Arduino, jenis-jenis mikrokontroller Arduino yang sering dijumpai adalah :

1. Arduino Uno
2. Arduino Due
3. Arduino Mega
4. Arduino Nano
5. Arduino Leonardo

Selain jenis Arduino diatas, terdapat juga modul tambahan yang berfungsi untuk memudahkan penggunanya untuk menggunakan fitur Arduino, antara lain :

1. Ethernet Shield
2. Xbee shield
3. Motor Shield
4. Relay Shield
5. RGB Shield

Untuk mengupload program kedalam arduino diperlukan sebuah software ARDUINO IDE, tampilan program arduino IDE dapat dilihat seperti gambar dibawah:

Gambar 8.2 Tampilan awal Arduino IDE

Sebelum mengupload program pada Arduino IDE, terlebih dahulu IDE harus mengenali jenis Arduino yang dipakai. Arduino yang telah terhubung dengan PC akan terbaca pada menu Tools > Ports.

Program yang telah diupload dapat dilihat pada computer/PC dengan membuka tampilan serial monitor pada Arduino IDE dengan mengakses menu Tools > Serial Monitor

Gambar 8.3 Tampilan serial monitor Arduino IDE

8.2 Fungsi Standar Pemrograman Arduino

8.2.1 Void Setup dan Void Loop

Void setup() merupakan fungsi yang dijalankan sekali ketika Arduino dihidupkan pertama kali. Fungsi dari void setup() ini adalah untuk menginisialisasi variable, pinMode, ataupun library yang akan digunakan dalam program. Void setup() juga akan berjalan ketika tombol reset pada Arduino dijalankan.
Contoh kode :

```
int pin_led = 5; // pin digital 5 diinisialisasi

void setup() {
  Serial.begin(9600); // set kecepatan transmisi 9600
 pinMode(pin_led, OUTPUT); //pin no 5 diperkenalkan sebagai output
}
```

Void loop() merupakan jenis fungsi yang akan dijalankan berulang-ulang setelah fungsi void setup() dijalankan. Contoh program yang akan menghidupkan LED pada pin 5 dengan delay 1 detik.

```
int pin_led = 5; // pin digital 5 diinisialisasi

void setup() {
  Serial.begin(9600); // set kecepatan transmisi 9600
 pinMode(pin_led, OUTPUT); //pin no 5 diperkenalkan sebagai output
}
```



```
void loop() {
  digitalWrite(pin_led, HIGH); // mengatur pin 5 menjadi kondisi ON
   delay(1000); //menghentikan sementara perintah program selama
   1 detik
}
```

8.2.2 Serial Komunikasi

Fungsi serial diperlukan untuk menjalin komunikasi antara Arduino dengan sensor, ataupun Arduino dengan PC. Jumlah serial yang terdapat pada Arduino tergantung dari jenis Arduino tersebut. Tabel dibawah merupakan jumlah dari serial komunikasi (UART) yang ada pada Arduino

Tabel 8.1 Jenis serial (UART) pada arduino

Arduino	Serial PIN	Serial1 PIN	Serial 2 PIN	Serial 3 PIN
Uno, Nano, Mini	0(RX), 1(TX)	-	-	-
Mega	0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)
Leonardo, Micro, Yún	0(RX), 1(TX)	-	-	-
Uno WiFi Rev.2	Terkoneksi dengan USB Port	0(RX), 1(TX)	Terkoneksi ke NINA	-
MKR boards	-	13(RX), 14(TX)	-	-
Zero	Terkoneksi ke port Programming	0(RX), 1(TX)	-	-
Due	0(RX), 1(TX)	19(RX), 18(TX)	17(RX), 16(TX)	15(RX), 14(TX)
101	-	0(RX), 1(TX)	-	-

Dalam menggunakan serial komunikasi Arduino, terlebih dahulu kode Serial. begin() harus diletakkan didalam fungsi void setup(). Serial.begin() merupakan standar untuk mengatur kecepatan transmisi data serial. Kecepatan tranmisi umumnya digunakan dengan nilai 9600, namun nilai lain dapat digunakan seperti: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600, atau 115200.

Serial.print() merupakan jenis fungsi serial yang sering digunakan untuk menampilkan data keluaran atau variable pada Arduino untuk ditampilkan pada PC. Syntax penulisan Serial.print() adalah :

```
Serial.print(val);
Serial.print(variabel, format_data);
```

Format data merupakan parameter opsional, yang terdapat pada Arduino antara lain :

1. BIN(binary, atau base 2),
2. OCT(octal, atau base 8),
3. DEC(decimal, atau base 10),

4. HEX(hexadecimal, atau base 16)
5. Penggunaan koma belakang dari desimal

Contoh program Serial.print() :

```
Serial.print(20, DEC); // akan menampilkan 20
Serial.print(20, BIN); // akan menampilkan 10100
Serial.print(20, OCT); // akan menampilkan 24
Serial.print(20, HEX); // akan menampilkan 14
Serial.print(20.5599, 2); // akan menampilkan 20.55
Serial.print(20.5599, 3); // akan menampilkan 20.559
```

8.2.3 Digital I/O

Digital I/O merupakan proses pembacaan atau pemberian tegangan secara digital. Nilai tegangan hanya berupa aktif (HIGH) maupun mati (LOW).

1. pinMode()
   Merupakan proses konfigurasi pin pada Arduino. Konfigurasi pin dapat berupa OUTPUT maupun INPUT. Apabila konfigurasi pinMode pada Arduino tidak dijalankan, ada kemungkinan LED yang diaktifkan melalui proses OUTPUT akan berkedip saja.
   Syntax : `pinMode(pin, mode)`

2. digitalRead()
   Merupakan proses pembacaan nilai pada pin digital pada Arduino, nilai yang terbaca berupa keadaan HIGH atau LOW.
   Syntax : `digitalRead(pin)`

3. digitalWrite()
   Merupakan proses pemberian tegangan HIGH atau LOW pada pin yang dideklarasikan. parameter HIGH dapat berupa 3.3 Volt atau 5 Volt tergantung dari jenis Arduino yang dipakai, sedangkan untuk parameter LOW nilai tegangan berupa 0 Volt
   Syntax : `digitalWrite(jenis_pin, parameter)`
   `Dengan nilai parameter HIGH atau LOW`

8.2.4 Analog I/O

Analog I/O merupakan proses pembacaan atau pemberian tegangan secara analog. Berbeda dari digital I/O yang hanya memiliki 2 jenis nilai, pada Analog I/O nilai yang diatur dapat berupa 0-1023.

1. analogRead()
   Merupakan proses pembacaan nilai pada pin analog Arduino, nilai yang terbaca pada Arduino merupakan proses konversi dari analog ke digital dengan menggunakan ADC 10 bit. Sehingga proses proyeksi tegangan apabila Arduino beroperasi pada 5 volt/1024 adalah 0.0049 volts (4.9 mV) setiap satu unit. Untuk jenis Arduino yang menggunakan mikrokontroller ATmega (UNO, NANO, MEGA dan Mini) memerlukan waktu 100 micro sekon (0.0001 s) untuk membaca sebuah nilai analog.

Berikut adalah spesifikasi pin-pin analog pada Arduino:

Tabel 8.2 Spesifikasi pin pada jenis arduino

Arduino	Tegangan Operasi	PIN yang tersedia	Jenis ADC
Uno	5 Volt	A0 hingga A5	10 bit
Mini, Nano	5 Volt	A0 hingga A7	10 bit
Mega, Mega2560, MegaADK	5 Volt	A0 hingga A14	10 bit
Micro	5 Volt	A0 hingga A11*	10 bit
Leonardo	5 Volt	A0 hingga A11*	10 bit
Zero	3.3 Volt	A0 hingga A5	12 bit**
Due	3.3 Volt	A0 hingga A11	12 bit**
MKR Family	3.3 Volt	A0 hingga A6	12 bit**

* pin A6-A11 pada jenis Arduino ini berada pada pin 4, 6, 8, 9, 10, and 12
** nilai default dengan fungsi analogRead() pada jenis Arduino ini adalah 10 bit, namun apabila ingin menggunakan ADC 12 bit diperlukan pemanggilan fungsi analogReadResolution()

2. analogWrite()
   Merupakan proses pemberian tegangan analog oleh Arduino dengan menggunakan gelombang PWM. Kegunaan dari analogWrite adalah untuk mengatur kecepatan motor, mengatur tingkat cahaya LED.

Tabel 8.3 Spesifikasi pin PWM pada arduino

Arduino	PIN PWM	Frekuensi PWM
Uno, Nano, Mini	3, 5, 6, 9, 10, 11	490 Hz (pin 5 dan 6: 980 Hz)
Mega	2 - 13, 44 - 46	490 Hz (pin 4 dan 13: 980 Hz)
Leonardo, Micro, Yún	3, 5, 6, 9, 10, 11, 13	490 Hz (pin 3 dan 11: 980 Hz)
Uno WiFi Rev.2	3, 5, 6, 9, 10	976 Hz

Arduino	PIN PWM	Frekuensi PWM
MKR boards *	0 - 8, 10, A3 (18), A4 (19)	732 Hz
MKR1000 WiFi *	0 - 8, 10, 11, A3 (18), A4 (19)	732 Hz
Zero *	3 - 13, A0 (14), A1 (15)	732 Hz
Due **	2-13	1000 Hz
101	3, 5, 6, 9	pin 3 dan 9: 490 Hz, pin 5 dan 6: 980 Hz

* pada Arduino ZERO, MKR terdapat Digital to Analog Converter yang dapat menghasilkan analog murni pada PIN DAC0 (A0)
** pada Arduino Due terdapat Digital to Analog Converter yang dapat menghasilkan analog murni pada PIN DAC0 (A0), dan DAC1

Dalam menggunakan analogWrite, tidak perlu memanggil fungsi pinMode() pada pin tertentu untuk dijadikan OUTPUT, karena fitur analogWrite dapat diakses oleh pin apapun.
Syntax : `analogWrite(pin, value)`

8.3 Struktur Pemrograman Pada Arduino

Setiap program Arduino (biasa disebut *sketch*). *Sketch* merupakan *source code* yang berisi logika dan algoritma yang akan di upload ke dalam IC mikrokontroler (Arduino).

1. Structure
   Structure dasar dari bahasa pemrograman arduino adalah sederhana yang terdiri dari dari 2 program utama yang harus ada dalam pemrograman arduino.
   Void setup ( )
   {
   // Statement
   }
   Void loop ( )
   {
   // Statement
   }

Dimana Void Setup ( ) suatu bagian untuk inisialisasi yang hanya dijalankan sekali di awal program, sedangkan loop ( ) untuk mengeksekusi bagian program yang akan dijalankan berulang ulang untuk selamanya. Statement digunakan untuk membuat suatu catatan pada program.

2. Setup ( )
   Fungsi setup ( ) hanya di panggil satu kali ketika program pertama kali di jalankan. Ini digunakan untuk pendefinisian mode pin atau memulai komunikasi serial. Fungsi setup ( ) harus diikut sertakan dalam program walaupun tidak ada statement yang dijalankan.
   void setup ( )
   {
   pinMode(7, OUTPUT);// artinya 'pin 7' disetting sebagia output
   }

3. Loop
   Setelah melakukan fungsi setup ( ) maka secara langsung akan melakukan fungsi loop ( ) secara berurutan dan melakukan instruksi-instruksi yang ada dalam fungsi loop ( ).
   void loop ( )
   {
   digitalWrite(7, HIGH); // 'Pin7' kondisi berlogika 1 (hidup)
   delay(1000); // Waktu tunda program selama 1 detik
   digitalWrite(7, LOW); // 'Pin7' kondisi berlogika 0 (mati)
   delay(1000); // Waktu tunda program selama 1 detik

4. PinMode ( )
   Digunakan untuk melakukan konfigurasi secara spesifik fungsi dari sebuah pin, apakah sebagi input atau output.
   πınMode(0, INPUT) konfigurasi pin 0 Arduino sebagai pin input
   pinMode(7, OUTPUT) konfigurasi pin 13 Arduino sebagai pin output

5. digitalRead( )
   Digunakan untuk membaca nilai pin digital yang spesifik, apakah bernilai HIGH atau LOW.

6. digitalWrite( )
   Selain membaca nilai ada juga function untuk menuliskan atau memberikan nilai pada suatu pin digital secara spesifik.
   digitalWrite(7,HIGH) memberikan nilai digital HIGH pada pin 7 Arduino

7. delay( )
   Fungsi delay digunakan untuk memberikan waktu tundaan (dalam satu milisecond) untuk mengerjakan satu baris program ke baris selanjutnya.

8. analogRead( )
   Selain berfungsi membaca nilai digital, juga digunakan untuk membaca nilai analog. Dengan menggunakan function analogread( ) untuk membaca nilai analog melalui pin analog.

9. Function
   Function (fungsi) adalah blok pemrograman yang mempunyai nama dan mempunyai statement yang akan dieksekusi ketika function di panggil.
   Cara pendeklarasian function
   Type functionName(parameters)
   {
   // Statement ;
   }

10. { } Curly Braces
    Curly braces mendefenisikan awal dan akhir dari sebuah blok fungsi. Apabila ketika memprograman dan programmer lupa member curly brace tutup makan ketika di- compile akan terdapat laporan error.

11. Semicolon
    Semicolon harus diberikan pada setiap statement program yang kita buat ini merupakan pembatasan setiap statement program yang dibuat.

12. /*.../* blok comment
    Semua statement yang ditulis dalam blok comment tidak akan dieksekusi dan tidak akan di compile sehingga tidak mempengaruhi besar program yang dibuat untuk di masukkan dalam board arduino.
13. // Line comment
    Sama halnya dengan blok comment, line comment pun sama hanya saja yang dijadikan komentar adalah perbaris.

Type – Type Data

1. Byte
   Type byte ini dapat menyimpan 8-bit nilai angka bilangan asli tanpa koma, tipe byte ini memiliki range 0 – 255.
   Byte biteVariable = 180; // mendeklarasikan 'biteVariable' sebagai type byte
2. Integer
   integer adalah tipe data yang utama untuk menyimpan nilai bilangan bulat tanpa koma. Penyimpanan integer sebesar 16-bit dengan range 32.767 sampai -32.768.

Int integerVariable = 1600; // mendeklarasikan 'integerVariable' sebagai type integer

3. Long
   Perluasan ukuran untuk long integer, penyimpan long integer sebesar 32-bit dengan range 2.147.483.647 sampai -2.147.483.648
   Long log variabel = 500000; // mendeklarasikan 'longVariabel' sebagai type long
4. Float
   Float adalah tipe data yang dapat menampung nilai decimal, float merupakan penyimpan yang lebih besar dari integer dan dapat menyimpan sebesar 32-bit dengan range 3.4028235E+38 sampai -3.4028235E+38
   Float floatVariable = 3.14; // mendeklarasikan 'floatVariable' sebagai type float
5. Array
   Array adalah kumpulan nilai yangdapat di akses dengan index number, nilai yangterdapat dalam array dapat di panggil dengan cara menuliskan nama array dan index number.
   Contoh :
   Int arraysName[] = {nilai0, nilai1, nilai2. . .}
   Int arrayKampus[] = {1,2,3,4,5}
   X = arrayKampus[5]; // x sekarang sama dengan 5

Flow Control

1. If
   Operator if ,operator ini mengset sebuah kondisi seperti nilai analog sudah berada dibawah nilai yang kita kehendaki atau belum, apabila terpenuhi maka akan mengeksekusi baris program yang ada dalam bracket kalau tidak terpenuhi maka akan mengabaikan baris program yang ada dalam brackets.
   Contoh :
   If (someVariabel?? value)
   {
   //Dosomething;
   }

2. If...else
   Operator if else mengeset sebuah kondisi apabila tidak sesuai dengan kondisi yang pertama maka akan mengeksekusi baris program yang ada di else.
   If (inputPin ==HIGH)
   {
   // rencana A;

}
Else
{
//rencana B;
}

3. For
   Operator for digunakan dalam blok pengulangan tertutup
   Contoh :
   For (basah, kering, lembab)

4. While
   Operator while akan terus mengulang baris perintah yang ada dalam bracket sampai ekspresi sebagai kondisi pengulangan bernilai salah
   While (someVariabel ?? value)
   {
   //doSomething.
   }

5. Do....While
   Sama halnya dengan while( ) hanya saja pada operator Do...While tidak melakukan pengecekan pada awal tapi di akhir, sehingga otomatis akan melakukan satu kali baris perintah walaupun pada awalnya sudah terpenuhi.
   Do
   {
   // doSomething;
   }
   While (someVariabel?? value);

Contoh Kode program Arduino untuk mengaktifkan LED pada pin 7:

```
const int pin_led =  7
int kondisi_led = LOW;
unsigned long waktu_sebelumnya = 0
const long interval = 1000; // 1000ms = 1 detik

void setup() {
  // set the digital pin sebagai output:
  pinMode(pin, OUTPUT);
}
```



```
void loop() {
  unsigned long waktu_sekarang = millis(); // menyimpan
  if (waktu_sekarang - waktu_sebelumnya >= interval) {
    // menimpa waktu sebelumnya dengan waktu sekarang
    waktu_sebelumnya = waktu_sekarang;
    if (kondisi_led == LOW) {
      kondisi_led = HIGH;
    } else {
      kondisi_led = LOW;
    }
    digitalWrite(pin_led, kondisi_led);
  }
}
```

Penjelasan :

1. Const int pin_led merupakan pendeklarasian pin pada Arduino yang terhubung dengan LED, apabila pin terhubung pada no 7, maka ditulis 7. Fungsi const agar tidak terjadi perubahan pada variable pin_led
2. Kondisi awal LED diset mati, sehingga ditulis LOW.
3. Milis berfungsi untuk mengambil waktu secara milisekon ketika pertama kali Arduino diaktifkan. Dengan mengurangkan waktu yang telah berlalu dengan waktu sekarang maka akan didapatkan interval waktu tergantung keinginan sehingga dapat dibuat kondisi apabila telah melebihi waktu interval maka led akan diaktifkan, sedangkan apabila tidak terpenuhi LED akan tetap mati

BAB 9

Pemograman Beroreantasi Objek

9.1 Pengantar Pemograman Berorientasi Objek

Pemograman berorientasi objek (object oriented programming atau OOP) merupakan paradigma baru dalam rekayasa perangkat lunak yang didasarkan pada objek dan kelas. Pemograman berorientasi objek merupakan metodologi pemograman terbaik yang ada saat ini dalam rekayasa perangkat lunak. Object-oriented memandang software bagian per bagian dan menggambarkan satu bagian tersebut dalam satu objek.

Konsep output oriented mulai dikembangkan pada pertengahan 1960-an dengan sebuah bahasa program Simula kemudian dilanjutkan di era 70-an dengan Smalltalk berorientasi pada proses. Pada pertengahan 80-an, telah dikembangkan teknik pemograman berorientasi pada data (entity relationship). Pemograman berorientasi objek berlanjut pada tahun 90an, banyak pengembang software menggunakan konsep OOP seperti Java dan lain-lain. Di tahun 2002, versi terakhir dari Visual Studio, Microsoft memperkenalkan bahasa OOP baru yaitu C# (dibaca C-sharp) serta upgrade Visual C++, dan ini merupakan sebuah bahasa pemograman membangun software berbasis OOP.

Teknik berorientasi objek menganalogikan sistem aplikasi seperti kehidupan nyata yang terdiri atas objek-objek. Dengan demikian keunggulan teknik berorientasi objek adalah bahwa model yang dibuat akan sangat mendekati dunia nyata yang masalahnya akan dipecahkan oleh sistem yang dibangun. Model objek, atribut dan perlakuannya bisa langsung diambil dari objek yang ada di dunia nyata. Sebagai contoh objek sepeda memiliki atribut: pedal, roda, jeruji, warna, jumlah roda dan memiliki tingkah laku (behavior) seperti: kecepatanya menaik, kecepatannya menurun, perpindahan gigi sepeda.

Aplikasi dari pemograman berorientasi objek bukan sekedar prosedur melainkan sebagai objek dan real entity. Objek yang dimaksud disini memiliki pengertian suatu modul yang mengkombinasikan antara data dan kode program yang bekerja sama dalam program dengan melewatkan proses satu sama lain. Jadi *object oriented programming* merupakan cara yang paling efisien untuk menulis program komputer yang sangat mudah untuk di kombinasikan dan untuk dipergunakan kembali.

Pada pemrograman berorientasi obyek, atribut suatu objek disimpan pada variabel dan tingkah laku dinyatakan pada method sebagai fungsi atau prosedur. Sehingga suatu objek akan memiliki atribut dan sejumlah fungsi. Selanjutnya teknik berorientasi objek lebih memfokuskan kepada manipulasi objek-objek tersebut. Dengan menggunakan OOP maka dalam melakukan pemecahan suatu masalah tidak melihat bagaimana cara menyelesaikan suatu masalah tersebut (terstruktur) tetapi objek-objek apa yang dapat melakukan pemecahan masalah tersebut.

9.2 Kelas dan Objek

Kelas adalah bentuk penyederhanaan dari suatu permasalahan yang berkaitan dengan objek. Oleh karena itu kelas dapat didefinisikan sebagai sesuatu yang mempunyai data dan fungsi. Suatu kelas tidak terlepas dari pada suatu *object*. Objek merupakan penerjemahan dari *subgroup – subgroup* sehingga menjadi unit – unit. Suatu objek di dalam c++ merupakan suatu variabel yang didefinisikan sendiri oleh pemrogram, yang berisi data dan kode program untuk memanipulasi data. Pendeklarasian suatu kelas menggunakan kata kunci **class.**

Dalam suatu kelas atribut disebut sebagai variabel dan behavior disebut juga sebagai methods. Methods adalah serangkaian statements dalam suatu class yang menghandel suatu tugas tertentu. Cara objek berkomunikasi dengan objek lain adalah dengan menggunakan fungsi. State diimplementasikan menggunakan properties/ variabel yang selanjutnya disebut sebagai member dari sebuah objek. Member dari sebuah obyek memiliki aturan pengaksesan, terdapat tiga level aturan akses yaitu:

1. Tingkat akses private
   Akses *private* berarti bahwa variabel yang digunakan hanya dapat diakses oleh kelas yang memilikinya. Dengan mendeklarasikan variabel menggunakan akses *private*, berarti variabel tersebut tidak boleh diakses atau digunakan oleh kelas-kelas lain yang terdapat didalam program. Sebuah variabel yang dideklarasikan *private* hanya dapat diakses oleh variabel yang merupakan member dari kelas tersebut. Untuk mendeklarasikan suatu data dengan tingkat akses *private*, digunakan kata kunci `private.` Berikut contoh program deklarasi variabel private dalam kelas matrik:

```
#pragma once
class Matrik
{
private:
    int i,k,j,l;
public:
    Matrik(void);
    ~Matrik(void);
    void Matrik::invm(float a[100][100],int n);
};
```

   Variabel `i,k,j,l` dideklarasikan *private*. sehingga, kelas yang berada dalam *package* yang sama tidak dapat mengakses variabel tersebut. Begitu juga dengan sub kelas tidak dapat mengakses variabel tersebut.

2. Tingkat Akses Protected
   Suatu variabel yang dideklarasikan dengan tingkat akses *protected* dapat diakses oleh kelas yang memilikinya dan juga oleh kelas-kelas yang masih memiliki hubungan turunan. Untuk mendeklarasikan suatu data protected digunakan kata kunci `protected.`

```
class FastDecoupled :
    public LFBase
{
private:
void DevelopeB1();
void DevelopeB2();

protected:
int i,j,k;
float  B1[100][100],B1_inv[100][100],B2[100][100],B2_inv[100]
    [100];
float psch[100],qsch[100],delta_theta[100],delta_v[100];
```

```
public:
FastDecoupled(void);
~FastDecoupled(void);
void FastDecoupled::Calculate();
};
```

Dari contoh diatas, variabel `B1[100][100], B1_inv[100][100], B2[100][100], B2_inv[100][100]` dideklarasikan *protected*. Sehingga, kelas yang berada dalam *package* yang sama dapat mengakses atribut tersebut. Begitu juga dengan sub kelas yang merupakan subclass dari kelas FastDecoupled dapat mengakses variabel tersebut.

3. Tingkat akses Public
   Tingkat akses *public* merupakan kebalikan dari tingkat akses *private*. Pada tingkat akses *public*, variabel yang bersifat *public* dapat diakses oleh semua bagian dalam program. Dengan kata lain, variabel yang dideklarasikan dengan tingkat akses *public* dapat dikenali dan diakses oleh semua kelas yang ada di program, baik yang merupakan kelas turunan maupun kelas yang tidak memiliki hubungan sama sekali. Untuk mendeklarasikan suatu data dengan tingkat akses public, digunakan kata kunci `public`.

4. Tingkat akses Friend
   Friend Class merupakan sebuah kelas yang dapat mengakses dan protected members dari kelas lain class yang dideklarasikan sebagai friend.
   Contoh:

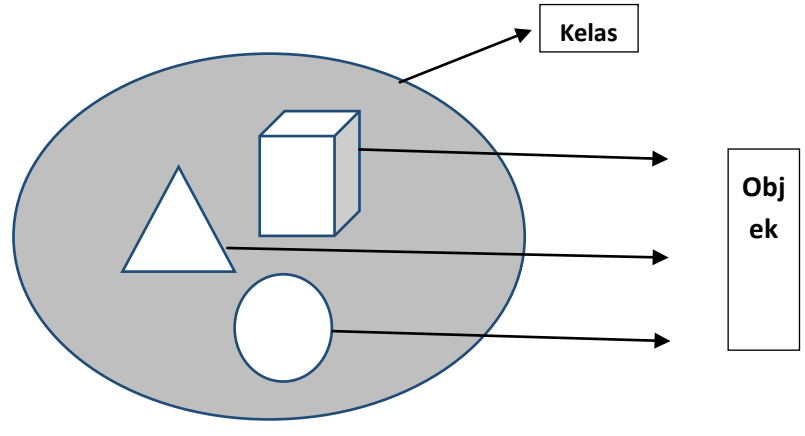
```
Class nama_kelas
{
        Private:
Int member1, member2;
        Friend class teman_kelas;
}
```

Dari program diatas teman_kelas dapat mengakses private member yaitu member 1 dan 2, namun nama_kelas tidak dapat mengakses private member dari teman_kelas.

Dalam suatu kelas terdapat konstruktor dan destruktor. Konstruktor merupakan suatu fungsi dari anggota suatu kelas yang memiliki nama yang sama dengan nama kelas tempat fungsi itu berada. Konstruktor digunakan untuk mengalokasikan ruang suatu objek dan untuk memberikan nilai awal. Konstruktor yang dideklasrasikan tanpa parameter disebut konstruktor default.

Destruktor merupakan suatu fungsi anggota yang dijalankan secara otomatis ketika suatu objek akan terbebas dari memori karena keberadaannya telah menyelesaikan tugas. Fungsi ini dideklarasikan dengan nama yang sama dengan kelas dan diawali karakter tilde (~). Destruktor digunakan secara khusus manakala suatu objek menggunakan memori dinamis selama keberadaannya dan kemudian melepaskan memori itu setelah tidak menggunakannya lagi.

Objek adalah sebuah *structure* yang menggabungkan data dan method untuk bekerja bersama-sama.



Gambar 9.1 Kelas dan Objek

9.3 Karakteristik Object Oriented Programming

Karakteristik Object Oriented Programming adalah

a. Pewarisan (Inheritance)
   Inheritance adalah karakteristik OOP yang memungkinkan suatu class dibuat turunannya. Sehingga class turunan tersebut bisa menggunakan data member ataupun *method* milik kelas induk. Hal ini sering dianalogikan sebagai "anak mewarisi sifat orangtuanya (induknya)".
   Sifat inheritance :
   1. Kelas Induk, sering juga disebut *base class*, atau *class* saja.
   2. Kelas Turunan, sering juga disebut *derived class*, atau *subclass*.

   Dalam proses penurunan sebuah kelas, harus benar dalam pemberian hak akses data dan fungsi terhadap kelas turunan. Berikut hal – hal yang perlu diketahui dalam membuat sebuah kelas turunan.

Apabila kelas diturunkan sebagai public dari kelas induknya, maka :

- Bagian public yang terdapat pada kelas induk tetap akan menjadi bagian public pada kelas turunannya.
- Bagian protected yang terdapat pada kelas induk tetap akan menjadi bagian protected pada kelas turunannya.
- Bagian private yang terdapat pada kelas induk tetap tidak dapat diakses oleh kelas turunannya.

Apabila kelas diturunkan sebagai private dari kelas induknya, maka :

- Bagian public yang terdapat pada kelas induk akan menjadi bagian private pada kelas turunannya.
- Bagian protected yang terdapat pada kelas induk akan menjadi bagian private pada kelas turunannya.
- Bagian private yang terdapat pada kelas induk tetap tidak dapat diakses oleh kelas turunannya.

b. Polimorphisme
Polimorphisme dapat diterjemahkan sebagai banyak bentuk dimana dimana objek – objek yang berbeda memberikan respons terhadap suatu pesan yang sama dan sesuai dengan sifat masing – masing. Dengan kata lain *Polimorphisme* adalah kemampuan mengungkap suatu hal yang berbeda melalui satu cara yang sama.

c. Pembungkusan (Encapsulation)
Encapsulation adalah karakteristik OOP dimana suatu informasi (bisa variabel atau method) disembunyikan dari aspek eksternal. Pembungkusan bisa dilakukan dengan menggunakan hak akses private dan protected.

d. Abstraction
Abstraction adalah proses pengabstrakan atau penyembunyian detail program yang sangat rumit sehingga tidak perlu untuk mempermasalahkan pembuatannya.

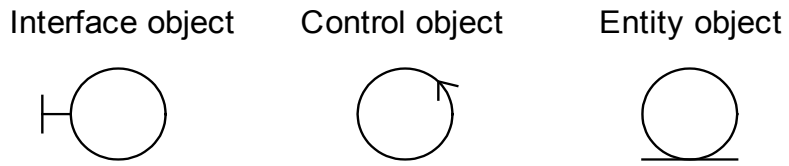
e. Reusability
Reusability adalah karakteristik OOP dimana objek dan variabel yang dirancang pada suatu perangkat lunak dapat digunakan untuk program lainnya. Dengan demikian penambahan aplikasi tidak perlu mengulang program dari awal, tetapi dapat menggunakan objek yang sama untuk aplikai yang lain.

9.4 Klasifikasi Objek

Jacobson mengklasifikasikan obyek menjadi tiga jenis (Jacobson et al., 1997):

i. Objek entitas yang mewakili benda-benda di dunia nyata ,
ii. Kontrol objek yang dibuat untuk menangani operasi kompleks yang tidak cocok secara alami dalam satu objek atau kelas ,
iii. Objek antarmuka yang menangani pertukaran data dengan pengguna atau sistem lainnya .

Tiga objek berdasarkan dari diagram UML adalah :



Gambar 9.2 Klasifikasi Objek berdasarkan Diagram UML

1. Interface Object
   Interface object merupakan objek input/output yang nantinya berfungsi membaca data dan menampilkan hasil dari perhitungan yang dicari pada program .
   Seperti: Read Data, Add, Print Hasil dan lainnya
2. Control Object (Kelas Aplikasi)
   Control Object merupakan objek aplikasi berupa algoritma atau proses dari perhitungan penyelesaian suatu masalah.
   Seperti : Menghitung Admitansi, Skema iterasi, Operasi Matriks, dan lainnya.
3. Entity Objek
   Entity object merupakan *object* yang berisikan kelas-kelas komponen yang akan digunakan oleh program.

Unified Modelling Language (UML) adalah suatu alat untuk memvisualisasikan dan mendokumentasikan hasil analisa dan desain yang berisi sintak dalam memodelkan sistem secara visual (Braun, et. al. 2001). Juga merupakan satu kumpulan konvensi pemodelan yang digunakan untuk menentukan atau menggambarkan sebuah sistem software yang terkait dengan objek (Whitten, et. al. 2004). Secara filosofi UML diilhami oleh konsep yang telah ada yaitu konsep permodelan Object Oriented karena konsep ini menganalogikan sistem seperti kehidupan nyata yang didominasi oleh obyek dan digambarkan atau dinotasikan dalam simbol-simbol yang cukup spesifik. Objek tersebut nantinya digunakan dalam membangun perangkat lunak berbagai aplikasi dalam bidang rekayasa dan sains.

9.5 Kelebihan Program Berorientasi Objek

Pemrograman berorientasi objek memliki beberapa keuntungan seperti :

1. Mudah dalam melakukan mintenance, program lebih mudah dibaca dan dipahami, dan pemrograman berorientasi objek mengontrol kerumitan program hanya dengan mengijinkan rincian yang dibutuhkan untuk programmer.
2. Data dan fungsi dibungkus dalam kelas-kelas atau objek-objek
3. Efektif digunakan untuk menyelesaikan masalah besar, karena OOP terdiri dari kelas-kelas yang memisahkan setiap code program menjadi kelompok-kelompok kecil, sesuai dengan fungsinya.
4. Fungsi-fungsi algoritma yang terbagi menjadi beberapa kelas dapat memudahkan kita dalam memahami program, begitu juga ketika ada bug pada program, kita bisa lebih mudah menemukan penyebab errornya dibanding ketika menggunakan prosedural programming.
5. Reusable, objek dan class dapat digunakan berkali-kali, sehingga dapat menghemat ruang memori. Sekali objek dibangun maka selanjunya dapat dipakai kembali pada aplikasi yang lain oleh programer yang lain. Dengan demikian objek reusing dapat meringkas waktu pembangunan suatu software dan meningkatkan produktifitasnya.

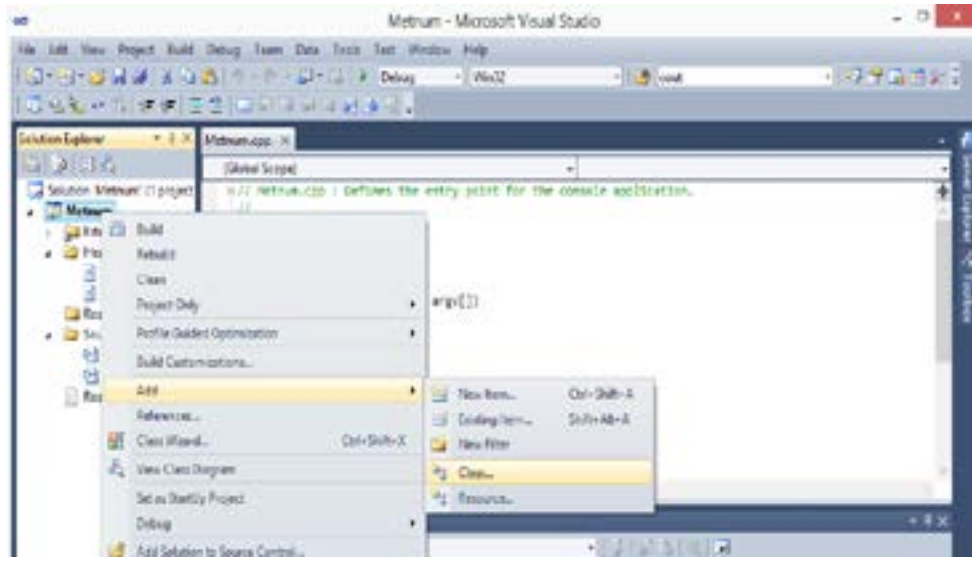
Permodelan yang baik tergantung dari kebutuhan dan dari sudut pandang masing-masing. Hal penting yang menjadi perhatian adalah perangkat lunak yang dibangun harus memenuhi kebutuhan pemakai, tepat waktu dan sesuai anggaran, serta mudah digunakan, dimengerti dan dipelihara. Sehingga pemrograman berorientasi objek sangat cocok sekali digunakan dalam pembuatan software yang rumit dan kompleks karena memberikan berbagai kemudahan kepada pemrogram seperti yang telah disebutkan diatas.

Sebelum diperkenalkan pemrograman berorientasi objek (OOP), perangkat lunak dibangun dengan menggunakan pendekatan pemrograman struktural. Namun, pendekatan ini memiliki banyak kelemahan seperti biaya pengembangan tinggi, produktivitas yang rendah, kualitas perangkat lunak tidak dapat diatur, dan risiko untuk pindah ke teknologi baru [X.Cai et al, 2002] . OOP datang untuk menyelesaikan masalah ini dengan pengembangan alami dari pemrograman terstruktur. Ciri utama dari pemrograman berorientasi objek adalah pengenalan objek untuk menggantikan fungsi mana obyek memungkinkan kita untuk merangkum data dan fungsi. Hal ini memungkinkan fungsi untuk berperilaku berbeda tergantung pada parameter yang disediakan dan memungkinkan operator untuk mengambil perangkat lunak yang lebih efisien. Menggunakan kembali objek dapat sangat mengurangi waktu pengembangan software dan meningkatkan produktivitas.

Sebagai dasar pemograman berorientasi objek akan dijelaskan bagaimana membuat kelas matrik sederhana dari awal sampai cara penggunaanya. Pada akhir bab ini juga akan diperlihatkan konsep membuat semua aplikasi numerik berorientasi objek.

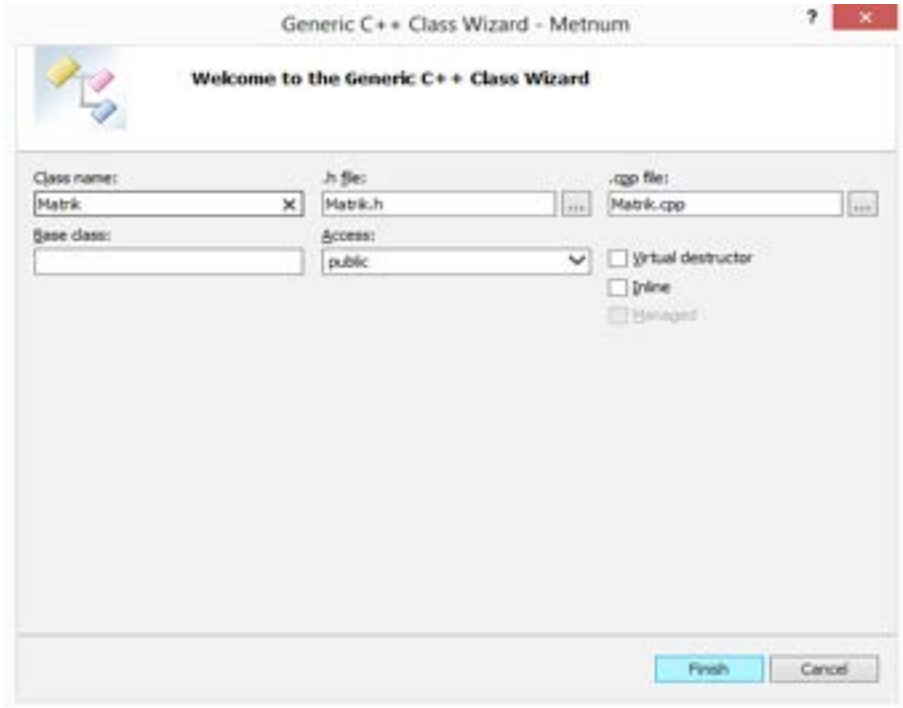
9.6 Merancang Kelas Matrik

Untuk merancang kelas matrik, buat project baru dengan nama Metnum. Selanjutnya kelas matrik dapat dibuat dengan klik kanan pada project Metnum dari aplikasi yang akan dibangun seperti gambar 7.3 berikut:

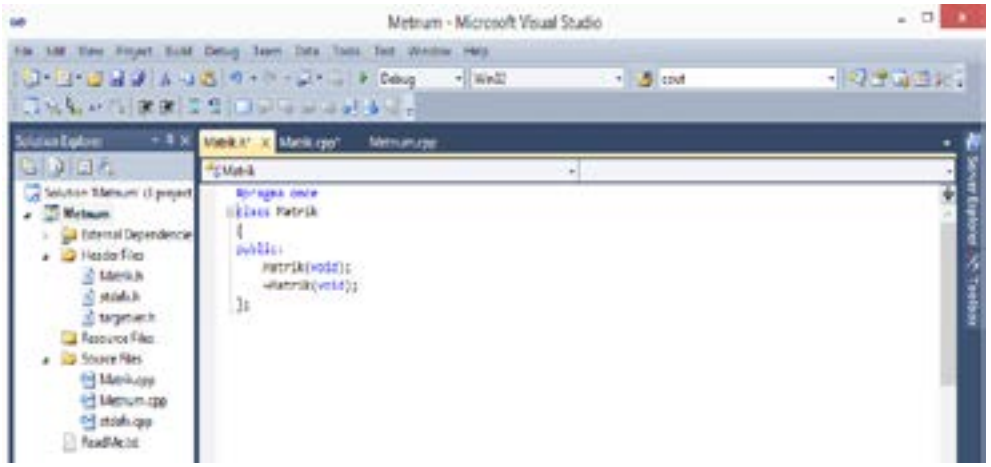


Gambar 9.3 Memulai kelas matrik

Selanjutnya ketik nama kelas matrik sebagai file Matrik.h dan Matrik.cpp seperti tampilan gambar 7.4, selanjutnya klik finish. Maka akan muncul dua file baru yaitu Matrik.h dan Matrik.cpp dibawah project Metnum dan dua program editor seperti gambar 7.5



Gambar 9.4 Membuat file .h dan .cpp



Gambar 9.5 Editor code kelas matrik

Selanjutnya ketikan kode program berikut dalam file Matrik.h:

```
#pragma once
class Matrik
{
private:
   int i,k,j,l;
public:
   Matrik(void);
   ~Matrik(void);
   void Matrik::invm(float a[100][100],int n);
   void Matrik::add(float a[100][100], float b[100][100],int n);

};
```

Kelas matrik ini terdiri dari empat member yang berstatus private atau hanya dikenal dalam kelas matrik dan tiga method yaitu invers matrik atau void invm. Semua member dan method yang ada pada kelas matrik harus dideklarasikan dalam file header .h tersebut. Kelas matrik merupakan kelas utama yang mempunyai hak akses publik, sehingga dari kelas manapun dapat mengakses kelas Matrik. Pada kelas ini terdapat konstruktor, destruktor dan kelas invermatrik yaitu Matrik(void), ~Matrik(void) dan void Matrik::invm(float a[100][100],int n). Konstruktor ini digunakan untuk mengalokasikan ruang untuk suatu objek dan untuk memberikan nilai awal. Sedangkan destruktor merupakan suatu fungsi anggota yang dijalankan secara otomatis manakala suatu objek akan terbebas dari memori karena ruang lingkup keberadaannya telah menyelesaikan tugasnya. Destruktor harus mempunyai nama yang sama dengan kelas dan diawali karakter tilde(~).

Kemudia ketik kode program berikut dalam file Matrik.cpp:

```
#include "StdAfx.h"
#include "Matrik.h"

Matrik::Matrik(void)
{
}

Matrik::~Matrik(void)
{
}

void Matrik::invm(float a[100][100],int n)
{
```

```
    for (i=1;i<=n;i++) {
    a[i][i] = 1.0/a[i][i];
    for (j=1;j<=n;j++) {
      if (j!=i){
        a[j][i] = a[j][i] * a[i][i];
        for (k=1;k<=n;k++) {
          if (k!=i){
            a[j][k] = a[j][k] - a[j][i]*a[i][k];
            if (j==n)
                  a[i][k] = -a[i][i]* a[i][k];
            }
          }
        }
      }
    }
    k = n-1;
    for (l = 1;l<=k;l++)
      a[n][l] = -a[n][n]*a[n][l];
}
```

Kode program invers matrik diketik pada file .cpp dengan void Matrik::invm(float a[100][100],int n). Pada baris tersebut menggunakan operator skop (::) untuk mendefinisikan atau membuat implementasi fungsi – fungsi yang terdapat dalam kelas Matrik. Ini berarti bahwa pengguna dapat mempunyai fungsi yang sama pada kelas yang berbeda dan C++ akan memperlakukannya terpisah. Selain itu pada file .cpp juga ditambahkan method untuk konstruktor dan destruktor. Konstruktor digunakan untuk mengalokasikan ruang suatu objek. Konstruktor yang dideklasrasikan tanpa parameter disebut konstruktor default. Destruktor merupakan suatu fungsi anggota yang dijalankan secara otomatis ketika suatu objek akan terbebas dari memori karena keberadaannya telah menyelesaikan tugas. Destruktor digunakan secara khusus manakala suatu objek menggunakan memori dinamis selama keberadaaannya dan kemudian melepaskan memori itu setelah tidak menggunakannya lagi.

Program perkalian matrik dapat ditambahkan pada kelas matrik sebagai suatu fungsi perkalian matrik. Prosedurnya adalah dengan menambahkan void baru perkalian seperti berikut:

```
#pragma once
class Matrik
{
private:
   int i,k,j,l;
   float c[100];
public:
   Matrik(void);
   ~Matrik(void);
   void Matrik::invm(float a[100][100],int n);
   void Matrik::perkalian(float a[100][100],float b[100],int n);
};
```

Void perkalian akan melakukan perkalian matrik $A_{i,j}$ dengan vektor b_i seperti berikut:

A . b = c

$$\begin{bmatrix} a_1 & a_2 & \dots & a_{1n} \\ a_2 & a_2 & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_n \end{bmatrix} \quad \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix} = \begin{bmatrix} c_1 \\ c_2 \\ \dots \\ c_n \end{bmatrix}$$

Cara pengoperasian perkalian matriks dengan vektor :

$$c_1 = a_{11} b_1 + a_{12} b_2 + \dots + a_{1n} b_n$$

$$= \sum_{j=1}^{n} a_{1j} . b_j$$

$$c_i = \sum_{j=1}^{n} a_{ij} . b_j$$

untuk : i = 1, 2, 3, ..., n;

$c_i = 0$

j = 1, 2, 3, ..., n;

$c_i = c_i + a_{ij} . b_j$

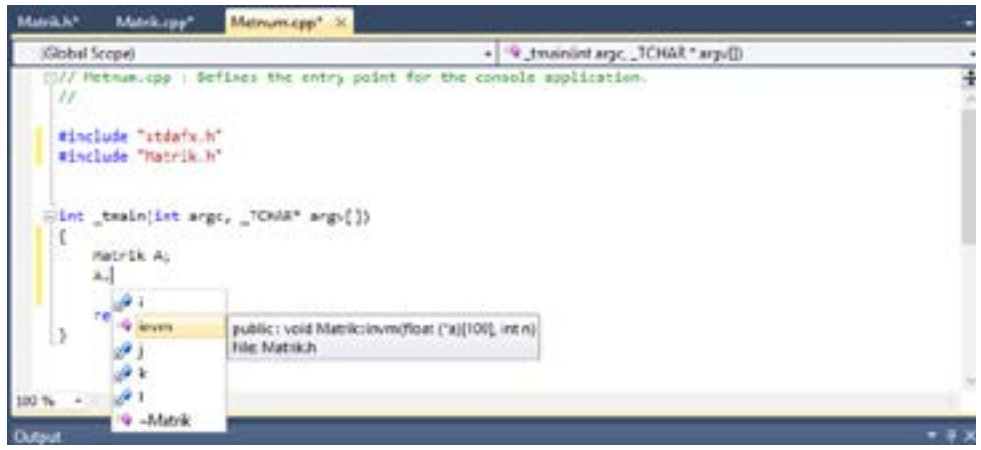
Sedangkan kode program perkalian diketikan pada file .cpp adalah:

```
void Matrik::perkalian(float a[100][100],float b[100],int n)
{
for(i=1;i<=n;i++) {
   c[i]=0;
   for(j=1;j<=n;j++)
       c[i]=x[i]+c[i][j]*b[j];
}
for(i=1;i<=n;i++)
   b[i]=c[i];
}
```

Hasil perkalian matrik kembali ditampung dalam vektor b[i].
Dengan cara yang sama fungsi operasi matrik yang lain dapat ditambahkan dalam kelas matrik, seperti fungsi transpos, kurang, bagi, tambah dal lain-lain.

9.7 Penggunaan Kelas Matrik

Setelah kelas matrik dibangun, maka fungsi invers matrik sekarang dapat dipanggil dalam program utama dengan memasukkan header file Matrik.h. Perintah untuk memanggil fungsi invers matrik dapat menggunakan operator dot (.) seperti diperlihatkan pada gambar 7.6. Semua member dan fungsi yang ada pada kelas matrik akan muncul dan dapat digunakan pada program utama sesuai hak akses dari member dan fungsi yang dideklarasikan pada header file dari kelas matrik.



Gambar 9.6 Kelas matrik dapat dipanggil menggunakan operator dot (.)

Kelas matrik dapat digunakan dengan mendeklarasikan variabel atau objek matrik A dan B sebagai kelas matrik. Operator dot (.) dan panah (->) dapat digunakan untuk memanggil fungsi invers dalam kelas matrik seperti kode program di berikut:

```
Include "Matrik.h"
Int main()
{
Matrik A;
Matrik *B = new Matri();
. . .
A.invm(a,nb);
B->invm(a,nb);
. . .
return 0;
}
```


Contoh 9.1.
Buat program utama untuk menghitung invers matrik A berikut menggunakan kelas matrik.

$$A=\begin{bmatrix} 1 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 3 \end{bmatrix}$$

a. Menggunakan operator dot (.)
b. Menggunakan operator panah (->)

Jawab:

a. Menggunakan operator dot (.)

```
#include "stdafx.h"
#include "Matrik.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    float a[100][100];
    int i,j,n=3;
    Matrik A;

    a[1][1] =  1; a[1][2] = -1; a[1][3] = -1;
    a[2][1] = -1; a[2][2] =  2; a[2][3] =  0;
    a[3][1] = -1; a[3][2] =  0; a[3][3] =  3;

    A.invm(a,3);

    for(i=1;i<=n;i++)
          for(j=1;j<=n;j++)
                cout<<" Invers A["<<i<<"]["<<j<<"] = "<<a[i]
    [j]<<endl;
    return 0;
}
```

Hasil invers matrik adalah seperti diperlihatkan pada gambar 7.7

Gambar 9.7 Hasil invers matrik menggunakan operator dot (.)

a. Menggunakan operator panah (->)

```
#include "stdafx.h"
#include "Matrik.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    float a[100][100];
    int i,j,n=3;
    Matrik *A= new Matrik();

    a[1][1] =  1; a[1][2] = -1; a[1][3] = -1;
    a[2][1] = -1; a[2][2] =  2; a[2][3] =  0;
    a[3][1] = -1; a[3][2] =  0; a[3][3] =  3;

    A->invm(a,3);

    for(i=1;i<=n;i++)
          for(j=1;j<=n;j++)
                cout<<" Invers A["<<i<<"]["<<j<<"] = "<<a[i]
    [j]<<endl;
    return 0;
}
```


Hasil invers matrik adalah seperti diperlihatkan pada gambar 7.8

Gambar 9.8 Hasil invers matrik dengan operator panah (->)

Jadi hasil running program keduanya sama.

Contoh 9.2

Buat program utama untuk menghitung vektor x dari sistem SPL **A.x = b** menggunakan kelas matrik, dengan nilai matrik:

$$A=\begin{bmatrix} 1 & -1 & -1 \\ -1 & 2 & 0 \\ -1 & 0 & 3 \end{bmatrix}$$

dan vektor b = $[-1, 3, -1]^T$

Jawab:

```
#include "stdafx.h"
#include "Matrik.h"
#include <iostream>
using namespace std;

int _tmain(int argc, _TCHAR* argv[])
{
    float a[100][100];
    float b[100];
    int i,j,n=3;
    Matrik *A= new Matrik();

    a[1][1] =  1; a[1][2] = -1; a[1][3] = -1;
    a[2][1] = -1; a[2][2] =  2; a[2][3] =  0;
    a[3][1] = -1; a[3][2] =  0; a[3][3] =  3;
    b[1]=-1;b[2]=3,b[3]=-1;
```

```
    A->invm(a,3);

    for(i=1;i<=n;i++)
          for(j=1;j<=n;j++)
                cout<<" Invers A["<<i<<"]["<<j<<"] = "<<a[i]
    [j]<<endl;

    A->perkalian(a,b,3);
    for(i=1;i<=n;i++)
          cout<<" Hasil x["<<i<<"] = "<<b[i]<<endl;
    return 0;
}
```

Hasil penyelesain x diperoleh setelah program dijalankan adalah sebagai berikut:

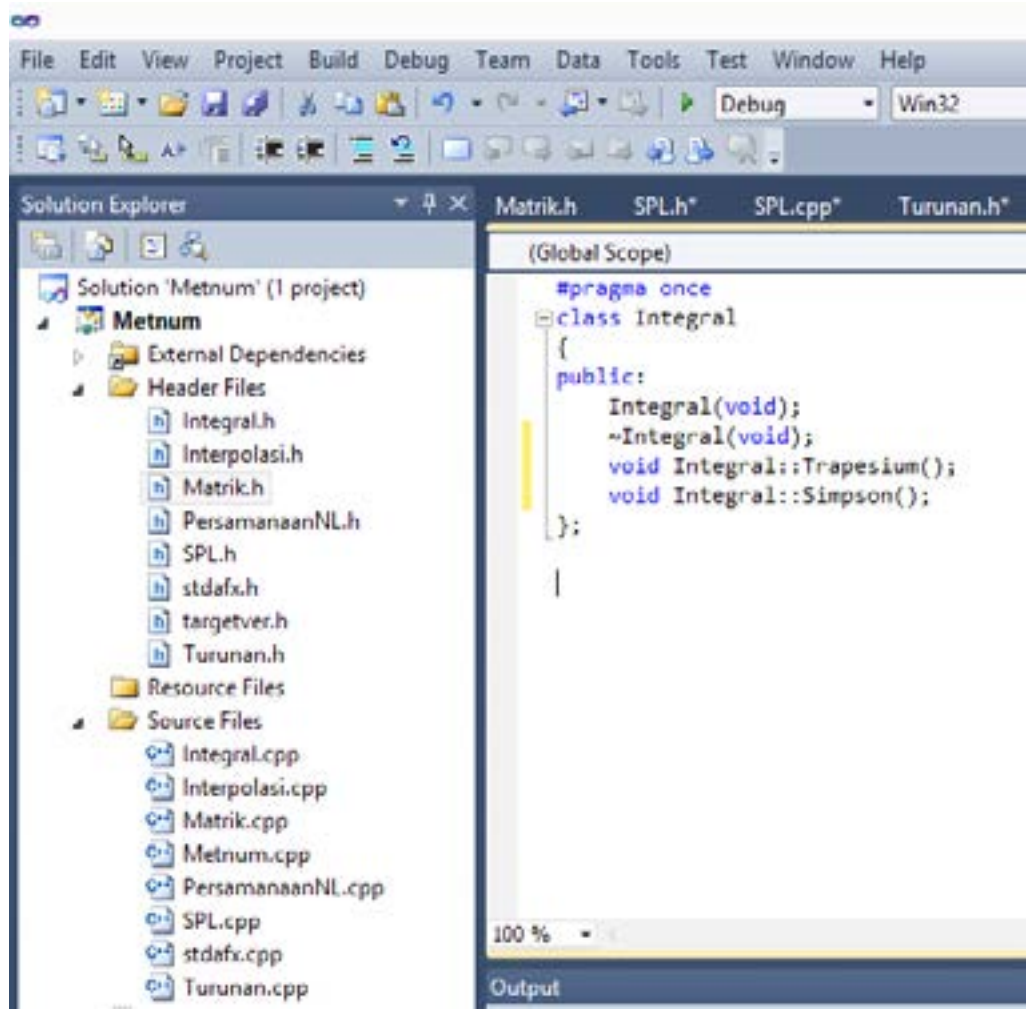
Gambar 9.9 Hasil penyelesaian SPL

Dekan cara yang sama fungsi operasi matrik yang lain dapat ditambahkan sebagai fungsi pada kelas matrik.

Aplikasi-aplikasi yang lain dapat dibuat sebagai kelas pada project Metnum mengunakan prosedur yang sama dengan membangun kelas matrik. Kelas-kelas tersebut adalah:

1. Kelas PersamaanNL
2. Kelas Interpolasi
3. Kelas Integral
4. Kelas Turunan
5. Kelas SPL

Setelah kelas-kelas tersebut ditambahkan, maka file .h dan .cpp akan muncul pada solution explore seperti pada gambar 7.10 berikut:



Gambar 9.10 Kelas Aplikasi Metnum

Program Create, Read, Update, Delete menggunakan fstream pada C++

```
#include <iostream>
#include <fstream>
#include <cstdlib>
#include <string>
#include <iomanip>
#include <direct.h>
```

PEMROGRAMAN C++ DASAR & BERORIENTASI OBJEK

Buku ini diperuntukan bagi para mahasiswa dan pengguna komputer yang ingin mempelajari pemograman komputer lebih cepat dan lebih baik. Dengan bahasan yang sederhana, dan disertai contoh aplikasi, pembaca dapat memahami isi bacaan yang sederhana dan langsung mempraktekkan tutorial yang diberikan. Sehingga diharapkan buku ini dapat dijadikan solusi bagi mahasiswa dan pengguna komputer yang ingin belajar pemograman C++.

Dengan program persoalan yang rumit dibidang rekayasa dapat diselesaikan dengan mudah menggunakan komputer. Permasalahan yang penting dalam komputasi adalah menemukan algoritma yang tepat dan akurat dalam memecahkan persoalan dan pembuatan program komputer yang effisien. Buku ini terdiri dari sebelas bab seperti berikut:

Bab 1.Pendahuluan
Bab 2. Dasar Pemograman C++
Bab 3. PerintahMasukan dan Keluaran
Bab 4. Perulangan dan Percabangan
Bab 5. Fungsi dan Prosedur
Bab 6. Variabel Array dan Aplikasinya
Bab 7. Pointer
Bab 8. Pemograman Arduino
Bab 9. PemogramanBerorientasiObjek
Bab 10. Kelas MatrikJarang
Bab 11.Aplikasi Windows Form

Contoh aplikasi program dibuat menggunakan pemograman Visual C++ atau Dev C++. Pada akhir bab diberikan soal latihan untuk menguji pemahaman pembaca terhadap materi yang disajikan untuk bab yang bersangkutan. Dasar pemograman Arduino dan teknik pemograman berorientasi objek dan penanganan matrik jarang juga diberikan dalam buku ini.

Indomedia Pustaka

TEKNIK
ISBN 978-623-7137-33-7
9 786237 137337
Harga Pulau Jawa: Rp 64.000,00