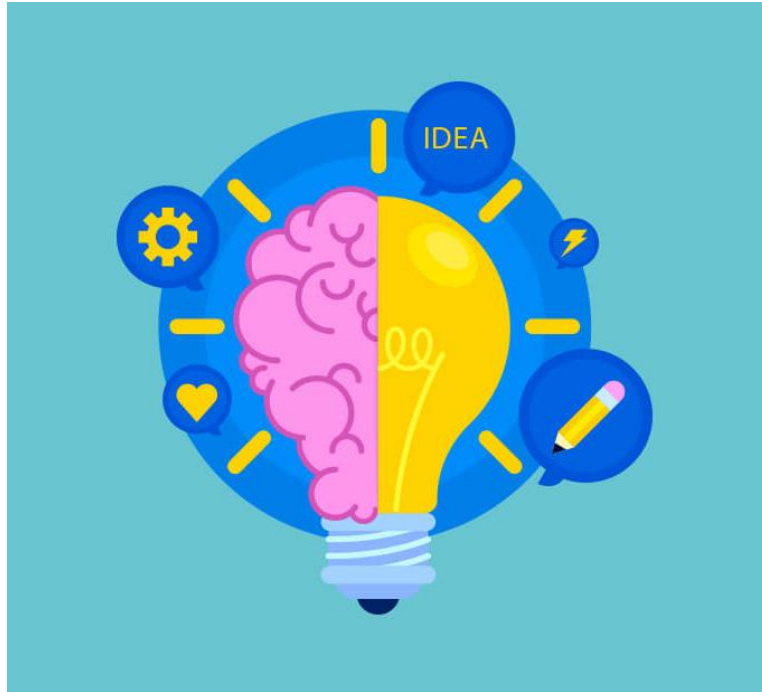


# **MODUL LOGIKA DAN ALGORITMA**



**Penyusun : Dany Pratmanto M.Kom**

**Sistem Informasi  
STMIK Nusa Mandiri  
Jakarta**

## DAFTAR ISI

DAFTAR ISI.....	i
-----------------	---

### BAB I ALGORITMA DAN *FLOWCHART*

1.1 Algoritma .....	1
1.1.1 Ciri Algoritma.....	1
1.1.2 Variabel dan Nilai .....	1
1.1.3 Struktur Dasar Algoritma.....	1
1.2 <i>Flowchart</i> .....	3
1.2.1 Jenis <i>Flowchart</i> .....	3
1.2.2 Simbol <i>Flowchart</i> .....	4
1.3 Pengujian.....	5
1.3.1 Verifikasi.....	5
1.3.2 Validasi .....	5
1.4 <i>Microsoft Visio</i> 2013.....	5
1.4.1 <i>User Interface Microsoft Visio</i> 2013.....	7
1.5 Pseudocode .....	9

### BAB II *VISUAL BASIC* 6.0

2.1 Bahasa Pemrograman.....	10
2.2 <i>Visual Basic</i> 6.0 .....	11
2.2.1 Jendela IDE .....	11
2.2.2 Toolbox .....	13
2.2.3 <i>Event</i> pada <i>Form</i> .....	14
2.2.4 Kotak Pesan (MsgBox).....	15
2.3 Pengenalan Tipe Data, Variabel, Konstanta, dan Operator .....	16
2.3.1 Tipe Data.....	16
2.3.2 Variabel.....	17

2.3.3 Konstanta .....	17
2.3.4 Operator .....	19
2.4 Prosedur .....	20
2.4.1 Prosedur <i>Sub</i> .....	20
2.4.2 Prosedur <i>Function</i> .....	21
2.5 <i>Record</i> dan <i>Array</i> .....	21
2.5.1 <i>Array</i> 1 Dimensi .....	22
2.5.2 <i>Array</i> 2 Dimensi .....	23
2.6 Tahap-tahap dalam Pemrograman .....	23
2.7 Pernyataan dalam <i>Visual Basic</i> .....	24
2.7.1 Struktur Kontrol Keputusan .....	24
2.7.2 Struktur Kontrol Pengulangan .....	27
2.7.3 Contoh Penggunaan Struktur Kontrol Keputusan .....	30
2.7.4 Contoh Penggunaan Struktur Kontrol Pengulangan .....	31

## DAFTAR PUSTAKA

# BAB I

## ALGORITMA DAN *FLOWCHART*

### 1.1 Algoritma

Algoritma adalah teknik penyusunan langkah-langkah penyelesaian masalah dalam bentuk kalimat dengan jumlah kata yang terbatas tetapi tersusun secara logis dan sistematis (Suarga, 2003:1). Algoritma mempunyai awal dan akhir, serta harus berhenti setelah mengerjakan serangkaian tugas. Setiap langkah harus didefinisikan secara tepat sehingga tidak memiliki arti ganda (*not ambiguous*).

#### 1.1.1 Ciri Algoritma

Menurut Sitorus (2015), algoritma memiliki kriteria atau ciri sebagai berikut :

##### 1) *Input*

Suatu algoritma harus memiliki nol atau lebih *input*. Artinya, suatu algoritma itu dimungkinkan tidak memiliki *input* secara langsung dari pengguna tetapi dapat juga memiliki beberapa masukan. Algoritma yang tidak memiliki *input* secara langsung dari pengguna, maka semua data dapat diinisialisasikan atau dibangkitkan dalam algoritma.

##### 2) *Output*

Suatu algoritma harus memiliki satu atau lebih *output*. Suatu algoritma yang tidak memiliki *output* adalah algoritma yang sia-sia karena algoritma dibuat untuk menghasilkan sesuatu yang diinginkan, yaitu hasil berupa keluaran.

##### 3) *Finiteness*

Algoritma harus terjamin dapat berhenti setelah melakukan sejumlah langkah proses.

##### 4) *Definiteness*

Setiap baris algoritma harus pasti dan tidak menimbulkan makna ganda (*ambiguous*), sehingga memberikan *output* yang sesuai dengan yang diharapkan oleh pengguna.

#### 1.1.2 Variabel dan Nilai

Nilai adalah suatu yang berupa huruf atau angka, dari suatu data yang memiliki satuan dan akan menjadi bahan baku untuk proses selanjutnya. Variabel adalah suatu wadah yang menampung nilai-nilai tertentu yang memiliki karakteristik atau tujuan yang sama (Rachmat, 2010:106).

#### 1.1.3 Struktur Dasar Algoritma

Algoritma memiliki tiga jenis struktur dasar, yaitu (Tim Dosen Alprog TI-UB, 2016):

1) Struktur Urut (*Sequence*)

Suatu struktur program dimana setiap baris program akan dikerjakan secara urut dari atas ke bawah sesuai dengan urutan penulisannya. Instruksi baris program 2 akan dikerjakan jika instruksi baris program 1 telah selesai dikerjakan. Contoh: Tahap-tahap instalasi Windows.

2) Struktur Pemilihan (*Selection*)

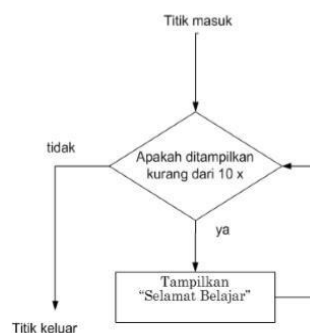
Instruksi untuk menentukan manakah yang akan dieksekusi diantara dua kemungkinan *sequences*. Keputusan berdasarkan kondisi *true/false*. Instruksi akan dikerjakan sesuai dengan benar atau salahnya suatu kondisi. Contoh stuktur pemilihan ditunjukkan pada gambar 1.1.

```
If NilaiMurid >= 60 Then
    Console.WriteLine("LOLOS")
Else
    Console.WriteLine("GAGAL")
End If
```

Gambar 1.1 Contoh *selection*

3) Struktur Pengulangan (*Repetition*)

Disebut juga dengan *iteration* atau *loop*. Pada struktur ini, langkah yang dijalankan terjadi selama beberapa kali sampai tercapai suatu kondisi tertentu. Sebelum melakukan pengulangan, dilakukan pengujian terhadap kondisi. Contohnya adalah ketika ingin menampilkan 10 tulisan “Selamat Belajar”. Bisa dituliskan dengan struktur *sequential*, berarti memberikan 10 instruksi untuk menampilkan 10 tulisan “Selamat Belajar”. Penulisan instruksi secara *sequential* memang praktis, namun tidak cocok untuk pengulangan dengan jumlah besar. Oleh karena itu, penggunaan strukur *repetition* membuat lebih praktis. Untuk lebih jelasnya dapat dilihat pada gambar 1.2.



Gambar 1.2 Contoh *Repetition*

## 1.2 Flowchart

*Flowchart* adalah untaian simbol gambar (*chart*) yang menunjukkan aliran (*flow*) dari proses terhadap data (Suarga, 2003:6). *Flowchart* membantu analis dan *programmer* untuk memecahkan masalah ke dalam segmen-segmen yang lebih kecil dan membantu dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* digunakan untuk menggambarkan prosedur sistem, baik sistem berjalan ataupun sistem yang akan diusulkan. *Flowchart* dapat digunakan sebagai alat bantu komunikasi dan untuk dokumentasi. Menurut Rachmat (2010), pada saat akan menggambar suatu *flowchart*, analis sistem atau *programmer* dapat mengikuti pedoman sebagai berikut:

- a. *Flowchart* sebaiknya digambar dari atas ke bawah dan mulai dari bagian kiri dari suatu halaman.
- b. Kegiatan di dalam *flowchart* harus dijelaskan dengan jelas.
- c. Harus ditunjukkan dari mana kegiatan akan dimulai dan di mana akan berakhirnya.
- d. Masing-masing kegiatan dalam *flowchart* sebaiknya menggunakan suatu kata yang mewakili suatu pekerjaan
- e. Masing-masing kegiatan di dalam *flowchart* harus di dalam urutan yang tepat.
- f. Kegiatan yang terpotong dan akan di tempat lain harus ditunjukkan dengan jelas menggunakan simbol penghubung.

### 1.2.1 Jenis Flowchart

Menurut Darsono (2001), terdapat beberapa jenis *flowchart*, diantaranya:

- a. *Flowchart* Sistem (*System Flowchart*)

*Flowchart* Sistem merupakan bagan yang menunjukkan alur kerja atau apa yang sedang dikerjakan di dalam sistem secara keseluruhan dan menjelaskan urutan dari prosedur-prosedur yang ada di dalam sistem. *Flowchart* ini merupakan deskripsi secara grafik dari urutan prosedur-prosedur yang terkombinasi yang membentuk suatu sistem.

- b. *Flowchart* Dokumen (*Paperwork Flowchart*)

*Flowchart* Dokumen menelusuri alur dari data yang ditulis melalui sistem. Kegunaan utamanya adalah untuk menelusuri alur *form* dan laporan sistem dari satu bagian ke bagian lain, termasuk bagaimana alur *form* dan laporan diproses, dicatat, dan disimpan.

- c. *Flowchart* Program (*Program Flowchart*)

*Flowchart* Program merupakan keterangan yang lebih rinci tentang bagaimana setiap langkah program atau prosedur sesungguhnya dilaksanakan. *Flowchart* ini menunjukkan setiap

langkah program atau prosedur dalam urutan yang tepat saat terjadi. *Flowchart* Program dihasilkan dari pengembangan lebih lanjut dari *Flowchart* Sistem.















d. *Flowchart* Proses (*Process Flowchart*)

*Flowchart* Proses merupakan teknik penggambaran rekayasa industrial yang memecah dan menganalisis langkah-langkah selanjutnya dalam suatu prosedur atau sistem.

### 1.2.2 Simbol *Flowchart*

Tabel 1.1 di bawah ini merupakan simbol-simbol *flowchart* beserta fungsinya:

Tabel 1.1 Simbol-simbol *Flowchart*

Simbol	Nama	Fungsi
	<i>Terminator</i>	Permulaan dan akhir program
	<i>Preparation</i>	Persiapan dan pemberian harga awal
	<i>Proses</i>	Mempresentasikan proses dalam <i>flowchart</i>
	<i>Input/Output</i>	Proses <i>Input/Output</i> data
	<i>Decision</i>	Keputusan dalam program
	<i>Document</i>	Input/output dalam format data yang dicetak
	<i>On page connector</i>	Penghubung dalam satu halaman
	<i>Off page connector</i>	Penghubung beda halaman
	<i>Display</i>	<i>Output</i> yang ditampilkan di layar/monitor
	<i>Manual operation</i>	Pekerjaan/operasi secara manual
	<i>Magnetic Drum (Database)</i>	Input/output menggunakan disk magnetik (media penyimpanan database)
	<i>Manual Input</i>	Input yang dimasukan secara manual menggunakan <i>keyboard</i>
	<i>Predefined Process</i>	Rincian operasi berada di tempat lain. (untuk pemanggilan fungsi atau prosedur)
	Garis Alir	Menunjukan arah aliran dari <i>flowchart</i>

Sumber : Darsono (2001 : 9)

### **1.3 Pengujian**

Proses pengujian (*testing*) adalah suatu proses menjalankan program atau sistem yang bertujuan untuk menemukan kesalahan/kegagalan pada perangkat lunak/aplikasi. Menurut Black (2002), *Testing* juga dapat diartikan sebagai kegiatan yang berfokus untuk melakukan evaluasi terhadap kemampuan sebuah program atau sistem dan memastikan bahwa program atau sistem tersebut sudah memenuhi kebutuhan. Tujuan dari tahapan ini adalah membuktikan keunggulan penggunaan sistem baru dibandingkan dengan sistem lama. Pengujian yang dilakukan meliputi uji verifikasi dan uji validasi.

#### **1.3.1 Verifikasi**

Menurut Law dan Kelton (1991), verifikasi adalah suatu proses untuk memeriksa kesesuaian jalannya program komputer simulasi dengan yang diinginkan dengan cara melakukan pemeriksaan program komputer, selain itu verifikasi dapat diartikan sebagai proses penerjemahan model simulasi konseptual kedalam bahasa pemrograman secara benar. Verifikasi merupakan proses pemeriksaan apakah logika operasional model (program komputer) sesuai dengan logika diagram alur, ada pula pengecekan mengenai ada atau tidaknya kesalahan dalam program.

#### **1.3.2 Validasi**

Menurut Loftus (1970), validasi diartikan sebagai suatu tindakan pembuktian dengan cara yang sesuai bahwa tiap bahan, proses, prosedur, kegiatan, sistem, perlengkapan atau mekanisme yang digunakan dalam produksi dan pengawasan akan senantiasa mencapai hasil yang diinginkan. Validasi merupakan proses penentuan apakah model, sebagai konseptualisasi atau abstraksi, merupakan representasi akurat dari sistem nyata.

### **1.4 Microsoft Visio 2013**

*Microsoft Visio* 2013 adalah suatu alat yang digunakan untuk membuat diagram resmi dan serbaguna yang berfungsi untuk menyederhanakan informasi kompleks. *Microsoft Visio* menyediakan alat-alat standar yang memudahkan pengguna untuk membuat gambar atau diagram menggunakan bentuk-bentuk dasar. Berikut merupakan cara penggunaan *Microsoft Visio* 2013 dari menu *Start*:

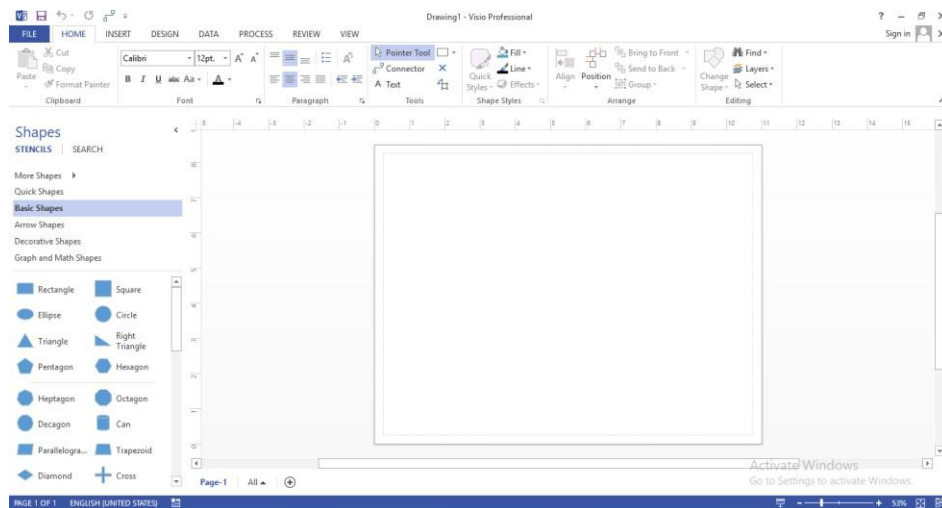


1. Klik tombol *start*, klik All Programs, klik Microsoft Office 2013, kemudian Microsoft Visio 2013 hingga layar menampilkan halaman awal untuk Microsoft Visio seperti pada gambar 1.3.



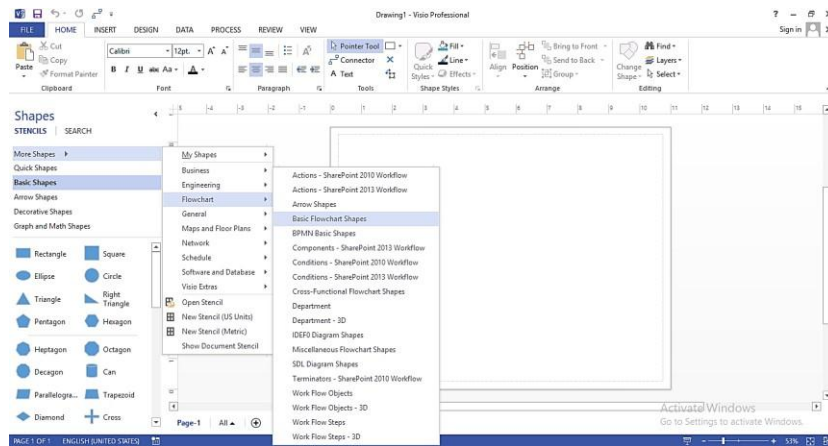
Gambar 1.3 Tampilan Awal *Microsoft Visio* 2013

2. Selanjutnya akan muncul beberapa pilihan *template*, seperti yang ditunjukkan pada Gambar 1.3. Jenis *template* yang akan digunakan adalah *Basic Diagram*. Dengan *template* ini, dapat dibuat diagram menggunakan *shapes* dasar yang ditunjukkan pada Gambar 1.4.



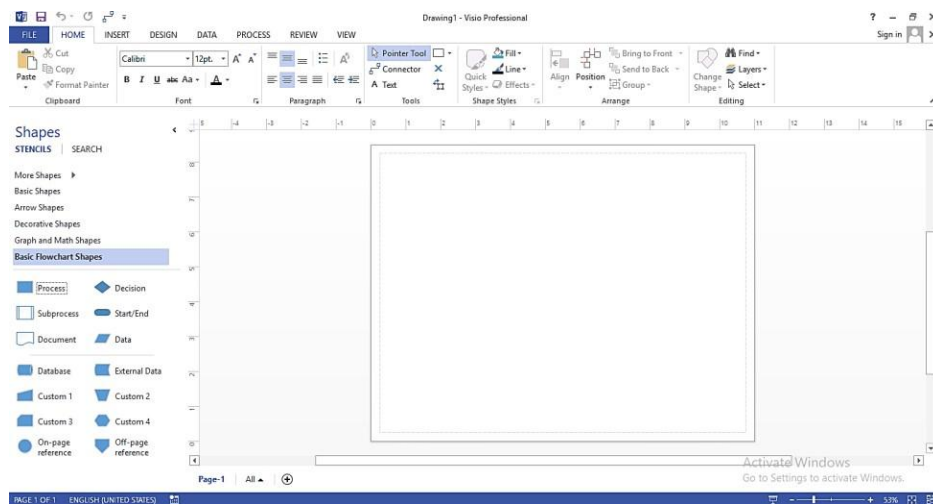
Gambar 1.4 Tampilan *Microsoft Visio* 2013 dalam menggunakan *template Basic Diagram*

3. Untuk memunculkan simbol-simbol *flowchart*, langkah-langkahnya adalah sebagai berikut: Pada segmen *Shapes*, klik *more shapes*, kemudian pilih *Flowchart*, lalu klik *Basic Flowchart Shapes*.



Gambar 1.5 Tampilan *Microsoft Visio 2013* dalam memilih *Basic Flowchart Shapes*

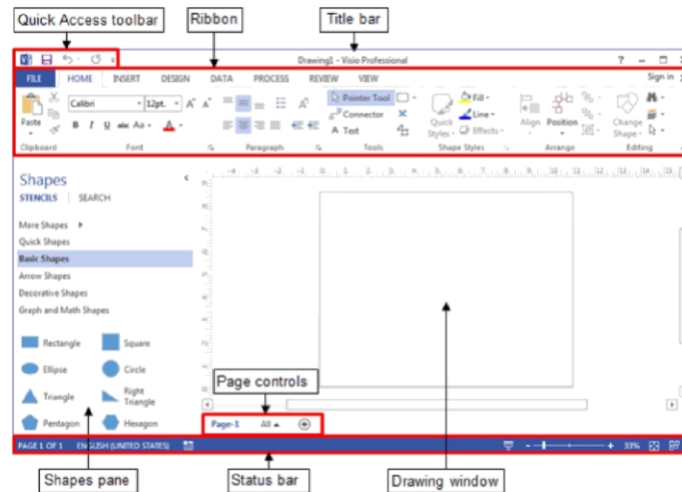
4. Maka akan muncul simbol-simbol *flowchart* yang kemudian dapat digunakan untuk membuat *flowchart* dalam *Microsoft Visio 2013*, seperti pada Gambar 1.6.



Gambar 1.6 *Basic Flowchart Shapes* dalam *Microsoft Visio 2013*

### 1.4.1 User Interface *Microsoft Visio 2013*

*Microsoft Visio 2013* didesain untuk membantu pengguna untuk menemukan alat yang dibutuhkan untuk menyelesaikan pekerjaan dengan cepat dengan cepat. Pada gambar 1.7 terdapat *user interface* beserta fungsi *toolbar Microsoft Visio 2013*.



Gambar 1.7 User Interface Microsoft Visio 2013

Bagian-bagian dari *Microsoft Visio* 2013 diterangkan dalam tabel 1.2.

Tabel 1.2 Bagian-bagian pada *user interface Microsoft Visio* 2013

Nama	Deskripsi
<i>Title Bar</i>	Menampilkan nama dari program atau <i>project</i> yang sedang dikerjakan.
<i>Quick Access toolbar</i>	Menyediakan suatu fasilitas untuk <i>one-click access</i> untuk perintah-perintah yang sering digunakan.
<i>Ribbon</i>	Terdiri dari kumpulan dari tab-tab dan setiap tab nya terdiri dari beberapa grup dari suatu perintah.
<i>Shapes pane</i>	Terdiri dari satu atau lebih <i>stencils</i> , setiap <i>stencil</i> direpresentasikan dengan <i>header bar</i> yang berisi nama dari <i>stencil</i> . <i>Stencil</i> merupakan kumpulan dari bentuk-bentuk yang akan digunakan.
<i>Drawing Window</i>	Bagian atas dan kiri dibatasi oleh <i>ruler</i> . <i>Page controls</i> berada di bagian bawah dan digunakan untuk mengubah tiap halaman dan menambah halaman baru.
<i>Status Bar</i>	Digunakan untuk mengubah <i>view</i> dan menyesuaikan <i>zoom level</i> yang diinginkan.

Sumber : *California State University* (2013:6)

## 1.5 Pseudocode

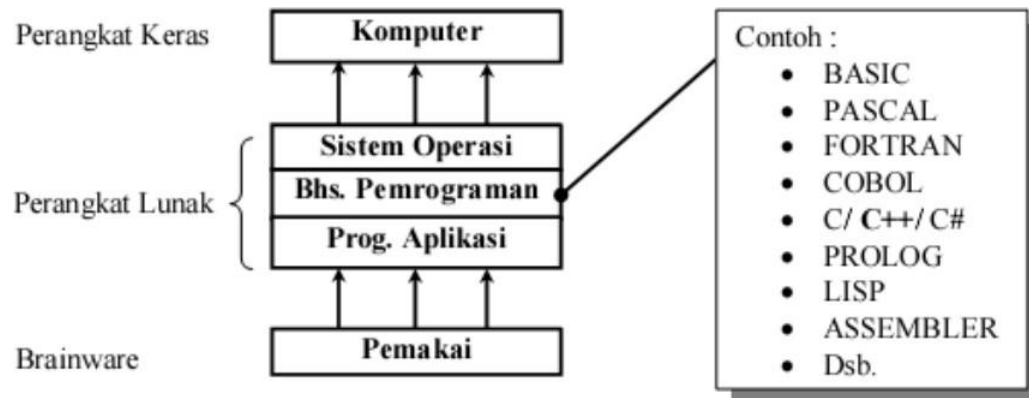
Algoritma yang disajikan bisa dengan tulisan yaitu dengan struktur Bahasa tertentu (misalnya Bahasa Indonesia atau Bahasa Inggris) dan *pseudocode*. *Pseudocode* adalah kode yang mirip dengan kode pemrograman yang sebenarnya. *Pseudocode* ditulis berdasarkan Bahasa pemrograman tertentu misalnya Pascal, C, atau Python, sehingga lebih tepat digunakan untuk menggambarkan algoritma yang akan dikomunikasikan kepada pemrogram (Utami, 2005:23). Berbeda dengan *flowchart* yang menjelaskan alur logika berpikir dengan simbol, *pseudocode* menjelaskan alur logika dengan menggunakan bahasa yang dituliskan dalam standar Bahasa Inggris untuk mendekatkan dengan perintah-perintah yang terdapat pada bahasa pemrograman.

Menurut Rachmat (2010), pada dasarnya struktur pseudocode hanya dibagi atas 7 perintah, yaitu:

1. *Sequence*, perintah ini dilakukan tahap demi tahap, tidak boleh melompat-lompat.
2. Perintah Percabangan, menggunakan *keyword* IF dan perintah dijalankan jika dan hanya kondisi terpenuhi (bernilai TRUE). Jika tidak, maka solusi alternatif dengan perintah ELSE yang akan dikerjakan.
3. *While*, perintah perulangan yang akan mengulang satu atau beberapa perintah sekaligus selama perintah masih dalam kondisi benar (TRUE). Jika kondisi sudah bernilai salah (FALSE) maka perintah tidak akan diulang lagi dan akan dilanjutkan ke perintah di bawah perulangan.
4. *Invoking Procedure*, *procedure* adalah kumpulan program yang diberi nama unik. Perintah ini menggunakan kata kunci CALL untuk memanggil kumpulan program tersebut untuk mempersingkat dan mempermudah penggunaan program-program tertentu.
5. *Repeat-Until*, perintah perulangan dimana minimal dilakukan sekali perulangan, kemudian diperiksa kondisinya apakah masih bernilai benar atau tidak. Perulangan akan berhenti saat kondisi bernilai FALSE.
6. *For*, sama seperti perintah *while*, selama kondisi terpenuhi perintah akan dijalankan. Perintah *for* memiliki karakteristik khusus, yaitu programmer dapat menentukan batas perulangan yang akan dilakukan
7. *Case*, adalah perintah percabangan dengan banyak kondisi. Apabila semua kondisi tidak ada yang terpenuhi maka bagain OTHERS yang akan dijadikan sebagai solusi.

## 2.1 Bahasa Pemrograman

Sebuah sistem komputer terdiri dari *hardware* atau perangkat keras, *software* atau perangkat lunak, dan *brainware*. Hubungan antara ketiga sistem ini dijelaskan pada gambar 2.1.



Gambar 2.1 Bagan Sistem Komputer  
Sumber: Rahman (2016)

Menurut Rachmat (2010 : 35), program harus ditulis dalam suatu bahasa yang dimengerti oleh komputer yaitu dalam Bahasa pemrograman. Bahasa merupakan suatu sistem untuk berkomunikasi, sedangkan bahasa pemrograman merupakan teknik komunikasi standar untuk mengekspresikan instruksi kepada komputer. Bahasa pemrograman dibedakan menjadi:

- Bahasa tingkat rendah (*low level language*) : bahasa yang berorientasi ke mesin. Contoh: bahasa *assembly*.
- Bahasa tingkat tinggi (*high level language*) : bahasa yang berorientasi ke manusia (seperti bahasa inggris). Contoh: bahasa Pascal, bahasa C, dll.

Program yang ditulis dalam bahasa pemrograman akan diterjemahkan ke dalam bahasa mesin (*biner digit*) dengan menggunakan penerjemah. Menurut Rachmat (2010 : 39), penerjemah terbagi menjadi 2, yaitu :

### a. Interpreter

Pada *Interpreter*, kode program tidak dianalisis secara keseluruhan, tapi per baris sehingga *error* yang ditampilkan ke *programmer* saat ada kesalahan adalah per baris. Jika pada baris tertentu terdapat *error*, maka *interpreter* akan langsung melaporkannya dan tidak akan melanjutkan pemrosesan sampai baris tersebut diperbaiki. Contoh bahasa pemrograman *interpreter* adalah bahasa *web*, *JavaScript*, *PHP*, *ASP classic*, *Visual Basic*, dan *Basic*.

### *b. Compiler*

Pada mesin kompilasi, kode program dikompilasi sehingga menghasilkan bentuk lain, yaitu *object code*. Kemudian *Linker* akan menggabungkan *object code* dengan *library* yang dibutuhkan sehingga menjadi *file* yang langsung bisa dieksekusi pada *processor* tertentu. Proses penerjemahan kode lambat, tetapi saat sudah menjadi *file executable*, proses kerjanya sangat cepat. Pada kompilasi kode program dianalisis secara keseluruhan kemudian *error* akan ditampilkan. Contoh kompilasi adalah bahasa *C*, *Pascal*, *Fortran*, dan *C++*.

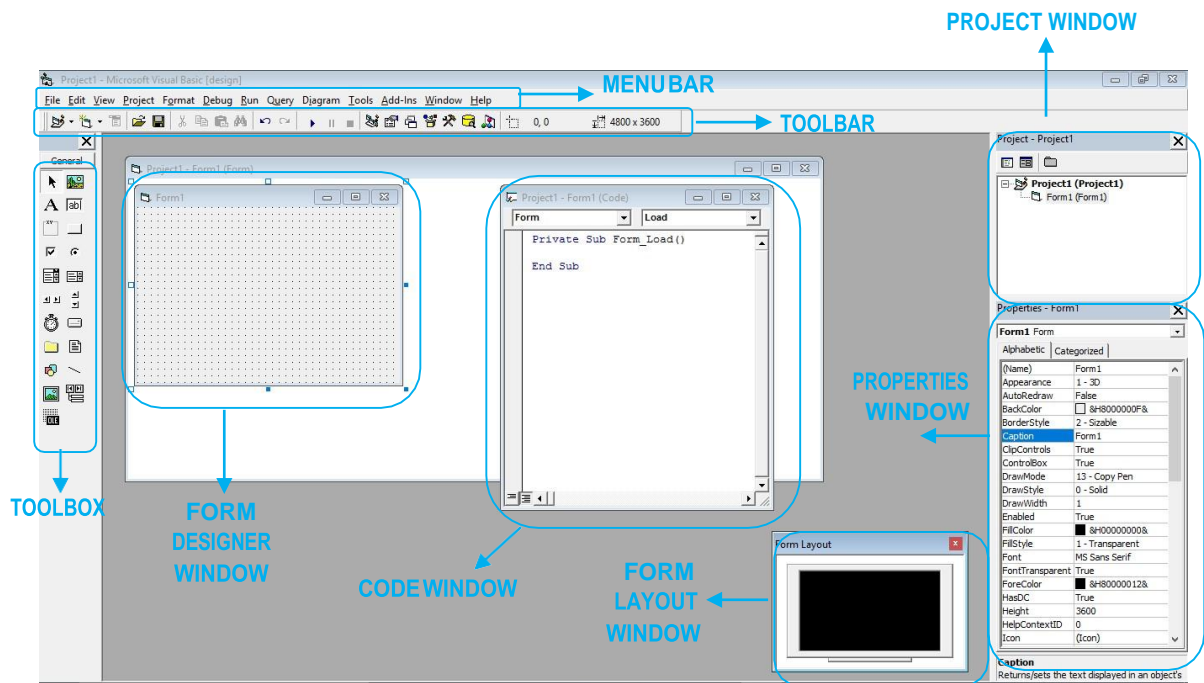
## **2.2 Visual Basic 6.0**

*Visual Basic 6.0* adalah salah satu bahasa pemrograman yang menggunakan prinsip *Object Oriented Programming* (OOP). *Object Oriented Programming* (OOP) adalah sebuah paradigma pemrograman berbasis objek, dimana dalam struktur *software* tersebut didasarkan kepada interaksi terhadap *object* dalam penyelesaian suatu proses/tugas (MADCOMS, 2008 : 2). Interaksi tersebut mengambil *form* dari pesan-pesan dan mengirimkannya kembali antar *object* tersebut. *Object* akan merespon pesan tersebut menjadi sebuah tindakan/*action* atau metode.

*Object Oriented Programming* terdiri dari *objects* yang berinteraksi satu sama lainnya untuk menyelesaikan sebuah tugas. Contoh, ketika mencetak sebuah halaman di *word processor*, inisialisasi tindakan adalah dengan mengklik tombol printer. Kemudian dilanjutkan dengan proses menunggu respon apakah *job* tersebut sukses atau gagal, sedangkan proses internal terjadi tanpa kita ketahui. Tentunya setelah menekan tombol *printer*, maka secara simultan *object* tombol tersebut berinteraksi dengan *object printer* untuk menyelesaikan *job* tersebut.

### **2.2.1 Jendela IDE**

*Integrated Development Environment* (IDE) adalah lingkungan/*software* tempat membangun suatu program. IDE memungkinkan *programmer* untuk membuat *user interface*, melakukan *coding*, *testing*, dan *debuging* serta mengkompilasi program sehingga menjadi program yang *executable* (Basuki, 2006 : 2). Salah satu contoh IDE adalah *Visual Basic for Application*. Berikut ini adalah gambar yang menunjukkan bagian-bagian dan nama-nama jendela yang dapat tampil pada *IDE Visual Basic*.



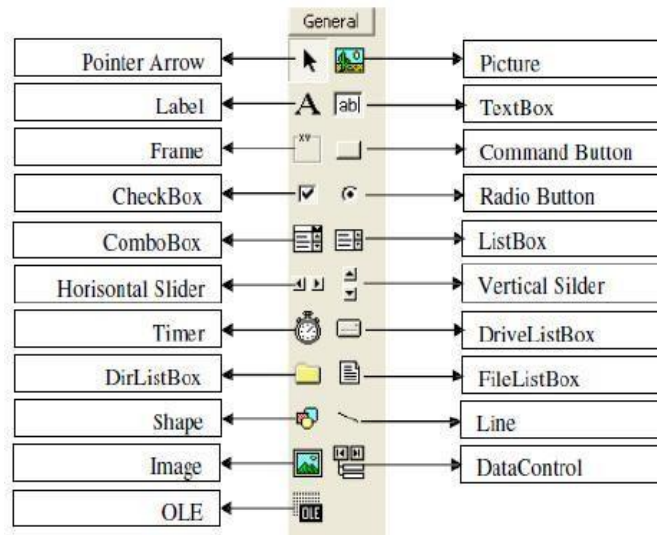
Gambar 2.2 IDE Visual Basic dengan jendela-jendela yang terbuka.  
Sumber: Paul (2013:1)

Fungsi dari jendela-jendela pada *IDE Visual Basic* adalah sebagai berikut:

- *Menu Bar*, digunakan untuk memilih tugas-tugas tertentu seperti menyimpan *project*, membuka *project*, dll
- *Toolbar*, digunakan untuk melakukan tugas-tugas tertentu dengan cepat.
- *Project window*, jendela ini berisi gambaran dari semua modul yang terdapat dalam aplikasi.
- *Form Designer window*, jendela ini merupakan tempat untuk merancang *user interface* dari aplikasi.
- *Toolbox*, jendela ini berisi komponen-komponen yang dapat digunakan untuk mengembangkan *user interface*.
- *Code window*, merupakan tempat untuk menulis *coding*.
- *Properties window*, merupakan daftar properti-properti *object* yang sedang terpilih.
- *Form Layout window*, akan menunjukkan bagaimana *form* bersangkutan ditampilkan ketika *runtime*.

## 2.2.2 Toolbox

*Toolbox* adalah kotak perangkat yang terdiri atas beberapa *class* objek yang digunakan dalam proses pembuatan aplikasi (Basuki, 2006 : 3). *Toolbox* berfungsi untuk memudahkan dan mempercepat kerja pengguna dalam pembuatan objek. Dari jendela ini dapat diambil komponen-komponen (*object*) yang akan ditanamkan pada *form* untuk membentuk *user interface*.



Gambar 2.3 *Toolbox Visual Basic 6.0*  
Sumber: Paul (2013:1)

Menurut Rachmat (2010 : 11), fungsi dari masing-masing kontrol tersebut adalah:

- *Pointer*, untuk mengaktifkan fungsi *pointer*.
- *Picture Box*, adalah kontrol yang digunakan untuk menampilkan *image*, dengan *format*: *BMP*, *DIB* (*bitmap*), *ICO* (*icon*), *CUR* (*cursor*), *WMF* (*metaFile*).
- *Label*, adalah kontrol yang digunakan untuk menampilkan teks yang tidak dapat diperbaiki oleh pemakai.
- *Text Box*, adalah kontrol yang mengandung string yang dapat diperbaiki oleh pemakai, dapat berupa satu baris tunggal, atau banyak baris.
- *Frame*, adalah kontrol yang digunakan sebagai kontainer bagi kontrol lainnya.
- *Command Button*, merupakan kontrol hampir ditemukan pada setiap *form*, dan digunakan untuk membangkitkan *event* proses tertentu ketika pemakai melakukan klik padanya.
- *Check Box* digunakan untuk pilihan yang isinya bernilai *yes/no* atau *true/false*.
- *Option Button* sering digunakan lebih dari satu sebagai pilihan terhadap beberapa *option* yang hanya dapat dipilih satu.



- *Combo Box* merupakan kombinasi dari *TextBox* dan suatu *ListBox* dimana pemasukkan data dapat dilakukan dengan pengetikkan maupun pemilihan.
- *List Box* mengandung sejumlah *item*, dan *user* dapat memilih lebih dari satu (bergantung pada property *MultiSelect*).
- *HScrollBar* dan *VScrollBar* digunakan untuk membentuk *scrollbar* berdiri sendiri.
- *Timer* digunakan untuk proses *background* yang diaktifkan berdasarkan interval waktu tertentu. Merupakan kontrol *non-visual*.
- *Drive List Box*, *Dir List Box*, dan *File List Box* sering digunakan untuk membentuk dialog *box* yang berkaitan dengan *File*.
- *Shape* dan *Line* digunakan untuk menampilkan bentuk seperti garis, persegi, bulatan, oval.
- *Image* berfungsi menyerupai *picture box*, tetapi tidak dapat digunakan sebagai kontainer bagi kontrol lainnya. Sesuatu yang perlu diketahui bahwa *control image* menggunakan *resource* yang lebih kecil dibandingkan dengan *PictureBox*.
- *Data* digunakan untuk *data binding*
- *OLE* digunakan sebagai tempat untuk program eksternal yang ada pada *windows* seperti *spread sheet* yang dihasilkan oleh *Microsoft Excel*. Dengan menggunakan *control* tersebut kita dapat menampilkan program lain pada sebuah aplikasi.

### 2.2.3 Event pada Form

*Event* adalah kejadian yang terjadi pada suatu objek (Priyono, 2007:13). *Event* berfungsi sebagai prosedur yang akan dikerjakan ketika *event* tersebut dikenakan pada objek yang dipilih. Pada tabel 2.1 akan dijelaskan beberapa *event* yang dimiliki oleh objek *form*.

Tabel 2.1 *Event* pada *Visual Basic 6.0*

<b>Event</b>	<b>Keterangan</b>
<i>Activate</i>	Kejadian saat <i>form</i> menjadi jendela aktif
<i>Click</i>	Kejadian saat pengguna program mengklik <i>form</i>
<i>DbClick</i>	Kejadian saat pengguna program melakukan klik ganda pada <i>form</i>
<i>Deactivate</i>	Kejadian saat <i>form</i> lain menjadi jendela aktif
<i>DragDrop</i>	Kejadian saat proses drag melewati <i>form</i> selesai
<i>DragOver</i>	Kejadian saat proses drag melewati <i>form</i>
<i>GotFokus</i>	Kejadian saat <i>form</i> mendapatkan fokus
<i>KeyDown</i>	Kejadian saat pengguna mengetikkan karakter pada <i>form</i>

<i>KeyPress</i>	Kejadian saat pengguna mengetikkan karakter pada objek <i>form</i>
<i>KeyUp</i>	Kejadian saat pengguna melepaskan tombol keyboard
<i>Load</i>	Kejadian saat <i>form</i> diaktifkan dan belum tampak di layar monitor
<i>LostFocus</i>	Kejadian saat <i>form</i> kehilangan fokus
<i>MouseDown</i>	Kejadian saat pengguna menekan mouse melewati <i>form</i>
<i>MouseMove</i>	Kejadian saat pengguna menggerakkan mouse pada <i>form</i>
<i>MouseUp</i>	Kejadian ketika pengguna melepaskan penekanan mouse pada <i>form</i>
<i>Resize</i>	Kejadian saat pengguna program mengubah ukuran <i>form</i>
<i>Unload</i>	Kejadian saat <i>form</i> dinonaktifkan

Sumber: MADCOMS (2008:26)

#### 2.2.4 Kotak Pesan (*MsgBox*)

*MsgBox* adalah fasilitas internal dari *Visual Basic 6.0* yang berupa *pop up* kotak pesan. *MsgBox* digunakan untuk menampilkan kotak pesan untuk *user* (MADCOMS, 2008:133). Berikut ini adalah aturan penulisan untuk membuat kotak pesan:

*MsgBox(prompt [, buttons][, title][, helpfile, context])*

Tabel 2.2 Keterangan *MsgBox*

Istilah	Keterangan
<i>Prompt</i>	Ekspresi string untuk menampilkan pesan di dalam kotak dialog. Panjang maksimum teks 1024 karakter
<i>Button</i>	Menentukan tipe tombol yang ditampilkan dan bentuk ikon yang digunakan
<i>Title (optional)</i>	Ekspresi string sebagai judul kotak pesan
<i>Helpfile (optional)</i>	Ekspresi string yang menunjukan <i>file Help</i> yang digunakan
<i>Context (optional)</i>	Ekspresi numerik berisi nomor konteks <i>Help</i>

Sumber : MADCOMS (2008:133)

Tabel 2.3 menjelaskan argumen *Button* yang digunakan untuk menampilkan tipe tombol dan ikon pada kotak pesan.

Tabel 2.3 Keterangan jenis-jenis argument *button*

Konstanta	Keterangan
vbOkOnly	Hanya menampilkan tombol OK
vbOKCancel	Menampilkan tombol OK dan <i>Cancel</i>
vbAbortRetryIgnore	Menampilkan tombol <i>Abort</i> , <i>Retry</i> , dan <i>Ignore</i>
vbYesNoCancel	Menampilkan tombol <i>Yes</i> , <i>No</i> , dan <i>Cancel</i>
vbYesNo	Menampilkan tombol <i>Yes</i> dan <i>No</i>
vbRetryCancel	Menampilkan tombol <i>Retry</i> dan <i>Cancel</i>
vbCritical	Menampilkan ikon <i>critical message</i>
vbQuestion	Menampilkan ikon <i>Warning Query</i>
vbExclamation	Menampilkan ikon <i>Information Message</i>

Sumber : MADCOMS (2008:134)

## 2.3 Pengenalan Tipe Data, Variabel, Konstanta dan Operator

Dalam pemrograman komputer terdapat beberapa istilah yang perlu dipahami, diantaranya yaitu tipe data, variabel, konstanta dan operator.

### 2.3.1 Tipe Data

Setiap variabel dan konstanta dalam Visual Basic 6.0 memiliki tipe data. Tipe data menentukan nilai-nilai yang dapat ditampung oleh variabel dan konstanta (Ramadhan, 2004 : 15). Tipe-tipe data ditunjukkan pada tabel 2.4.

Tabel 2.4 Tipe-tipe Data

Tipe Data	Ukuran	Batas Nilai
<b><i>Boolean</i></b>	2 bytes	<i>True</i> (Benar) atau <i>False</i> (Salah)
<b><i>Integer</i></b>	2 bytes	-32,768 s/d 32,767
<b><i>Long</i></b>	4 bytes	-2,147,483,648 s/d 2,147,483,647
<b><i>Double</i></b>	8 bytes	-1.79769313486231e308 s/d -4.940656455841247e-324 (negatif) 4.940656455841247e-324 s/d 1.79769313486231e308 (positif)
<b><i>Currency</i></b>	8 bytes	-922,337,203,685,477,5808 s/d 922,337,203,685,477.5807

<b>Date</b>	8 bytes	1 Jan 100 s/d 31 Des 9999
<b>Object</b>	4 bytes	Beberapa referensi objek
<b>String</b>	1 bytes per karakter	0 s/d 2,000,000,000 huruf
<b>Variant</b>	16 bytes + 1 bytes per karakter	<i>Null, Error</i> , dan seluruh tipe data lain

Sumber : MADCOMS (2008:92)

### 2.3.2 Variabel

Variabel merupakan suatu wadah atau tempat penampung data yang akan digunakan untuk proses berikutnya (MADCOMS, 2008:85). Variabel sering digunakan untuk efisiensi dan penghematan penulisan perintah. Pemakaian variable dapat bersifat khusus dan umum. Bersifat umum jika variabel tersebut dapat digunakan untuk semua prosedur pada jendela modul aktif, dan bersifat khusus jika hanya digunakan pada prosedur tempat memasang variabel tersebut.

Nama variabel dapat berupa satu karakter atau lebih. Menurut Rachmat (2010 : 107), Untuk memberi nama variabel terdapat beberapa syarat yang harus dipenuhi, antara lain:

1. Nama variabel harus diawali dengan huruf abjad, contoh: Bayar2, Brg.
2. Nama variabel tidak boleh menggunakan spasi, spasi dapat digantikan dengan garis bawah/*underscore* (  )
3. Nama variabel tidak boleh sama dengan nama metode, *property*, atau nama prosedur.
4. Nama variabel sebaiknya tidak terlalu panjang, maksimal 40 karakter. Variabel dalam suatu program harus dikenalkan atau dideklarasikan terlebih dahulu dengan bentuk berikut: **Dim variabel As tipe\_variabel**

Contoh:

‘mendeklarasikan variabel barang dengan tipe data string

**Dim** barang **As** String

### 2.3.3 Konstanta

Konstanta memiliki fungsi yang hampir sama dengan variabel. Hanya saja nilai dari konstanta tetap atau tidak dapat diubah-ubah dan nilainya harus sudah dikenalkan sejak awal (MADCOMS, 2008:89). Dalam *Visual Basic* 6.0 terdapat juga konstanta intrinsik yaitu konstanta yang secara *default* sudah disediakan *Visual Basic* 6.0, seperti vbYes, vbNo, dan sebagainya. Untuk mendeklarasikan konstanta digunakan kata kunci “Const”. Berikut contoh penulisan konstanta:

Const Nilai = 10

Sama halnya dengan variabel, jika konstanta digunakan prosedur-prosedur lain dalam satu modul maka dapat dideklarasikan di awal modul dengan diawali kata '*Private*'. Tetapi jika digunakan dalam modul yang berbeda, maka diawali dengan kata '*Public*'.

Contoh:

Private Const Bayar = "Tunai"

Public Const Bayar = "Tunai"

Konstanta adalah suatu tempat untuk menampung data yang nilainya selalu tetap dan tidak pernah berubah.

### 1. Aturan Penamaan Konstanta

- a. Harus diawali huruf.
- b. Boleh terdiri dari huruf, angka dan garis bawah.
- c. Maksimal 255 karakter.
- d. Tidak boleh menggunakan *reserved word*.

### 2. Deklarasi Konstanta

Bentuk umum deklarasi konstanta adalah sebagai berikut:

[Public | Private] Const nama\_konstanta [AS tipe\_konstanta] = eksplisit

Contoh:

Public Const POTONGAN = 0.05

Const NAMAPERSEH = "PT ABC"

### 3. Ruang Lingkup Konstanta

Ruang lingkup konstanta adalah ruang atau daerah dimana konstanta yang dibuat dikenali (dapat dipakai) biasanya juga berhubungan dengan umur (waktu hidup) konstanta. Ruang lingkup paling dalam adalah lingkup *procedure* dengan perintah Dim, *Private* atau *Static*.

Contoh:

Private Sub CmdTambah\_Click()

Dim Alamat as String \* 30

Static No as Integer

End Sub

### 2.3.4 Operator

Operator adalah tanda atau simbol yang digunakan untuk mengoperasikan data-data (MADCOMS, 2008:97). Simbol-simbol yang digunakan dapat berupa karakter maupun kata.

#### 1) Operator Aritmatika

Dalam menggunakan fungsi yang berkaitan dengan operasi aritmatika, maka harus mengenal terlebih dahulu beberapa operator aritmatika. Operator aritmatika terdiri dari: + (penjumlahan), - (pengurangan), \* (perkalian), / (pembagian), ^ (perpangkatan), dan & (penggabungan). Lebih jelasnya dapat dilihat pada tabel 2.5 berikut:

Tabel 2.5 Operator Aritmatika

Operator	Simbol	Contoh
Penjumlahan	+	15 + 5 → Hasilnya 20
Pengurangan	-	15 - 5 → Hasilnya 10
Perkalian	*	10 * 5 → Hasilnya 50
Pembagian	/	15/5 → Hasilnya 3
Perpangkatan	^	5 ^ 3 → Hasilnya 125
Penggabungan	&	“15” & “5” → Hasilnya 155

Sumber : MADCOMS (2008:98)

#### 2) Operator Pembandingan

Operator pembandingan adalah operator yang digunakan untuk membandingkan antara dua nilai (MADCOMS, 2008:106). Jika menggunakan operator pembandingan untuk membandingkan dua nilai, maka akan menghasilkan nilai logika benar (*true*) atau salah (*false*). Pada tabel 2.6, jenis-jenis operator pembandingan yang dapat digunakan adalah sebagai berikut:

Tabel 2.6 Operator Pembandingan

Operator	Simbol	Contoh	Operator	Simbol	Contoh
Sama dengan	=	A = B	Lebih kecil sama dengan	≤	A ≤ B
Lebih kecil	<	A < B	Lebih besar sama dengan	≥	A ≥ B
Lebih besar	>	A > B	Tidak sama dengan	≠	A ≠ B

Sumber: MADCOMS (2008:106)

#### 3) Operator Logika

Operator logika merupakan operator yang menghasilkan nilai logika (*true* atau *false*) (MADCOMS, 2008:104). Operator logika ini juga sering disebut dengan operator *boolean*.

Operator logika terdiri dari *AND*, *OR*, *NOT*, dan *XOR*. Lebih jelasnya dapat dilihat pada tabel 2.7 berikut ini:

Tabel 2.7 Operator Logika

Operator	Keterangan	Contoh
<b>AND</b>	Menghasilkan nilai <i>true</i> dari dua pernyataan yang bernilai <i>true</i> .	A AND B
<b>NOT</b>	Menyatakan kondisi pernyangkalan dari suatu pernyataan.	NOT A
<b>OR</b>	Menghasilkan nilai <i>true</i> dari dua pernyataan yang bernilai <i>true</i> semua atau salah satunya bernilai <i>true</i> .	A OR B
<b>XOR</b>	menghasilkan nilai <i>true</i> jika salah satu operand bernilai <i>true</i> tetapi bukan keduanya.	-

Sumber : MADCOMS (2008:104)

## 2.4 Prosedur

Prosedur (*procedure*) adalah bagian dari suatu program yang disusun secara terpisah untuk melakukan suatu tugas khusus atau fungsi tertentu (MADCOMS, 2008:178). Dalam *Visual Basic* 6.0, ada dua macam prosedur yang dikenal yaitu prosedur *Sub* dan prosedur *Function*. Perbedaan paling mendasar dari kedua prosedur ini yaitu bahwa prosedur *Function* menghasilkan nilai balik (*return value*), sedangkan prosedur *Sub* tidak. Nilai balik yang dihasilkan akan dikembalikan ke program utama (Kurniawan, 2005 : 74).

### 2.4.1 Prosedur Sub

Prosedur *Sub* adalah prosedur yang tidak menghasilkan nilai balik. Prosedur ini dapat menerima data untuk diolah, tetapi hasil olahan tersebut tidak dapat dikembalikan pada program utama yang memanggil prosedur *Sub* (Kurniawan, 2005 : 74).

Dalam penulisan Prosedur *Sub* dibuka dengan pernyataan *Sub* dan diakhiri *End Sub*. Prosedur *Sub* dapat menggunakan *argument*, seperti *Constant*, *Variable*, atau *Expression* yang dapat dijalankan dengan memanggil prosedur. Bentuk penulisan Prosedur Sub:

Private/Public sub

*name* ( )

*Statements*

End Sub

Dalam penulisan Prosedur *Sub* terdapat beberapa bagian dalam pernyataannya yaitu:

Tabel 2.8 Prosedur *Sub*

Bagian	Keterangan
<i>Public</i>	Bersifat opsional, pernyataan Prosedur Sub dapat berlaku pada semua prosedur lain dalam semua modul
<i>Private</i>	Bersifat opsional, pernyataan Prosedur Sub hanya dapat berlaku pada prosedur lain dalam modul dimana prosedur tersebut sudah dikenalkan
<i>Name</i>	Bersifat wajib/mutlak, nama lain dari sub mengikuti standar ketentuan nama variable bernilai <i>true</i> .
<i>Statements</i>	Bersifat opsional, berisi beberapa <i>Group</i> dari pernyataan akan dijadikan dalam Sub

Sumber : MADCOMS (2008:178)

#### 2.4.2 Prosedur *Function*

Prosedur *Function* adalah prosedur yang menghasilkan nilai balik. Nilai balik ini akan dikembalikan pada program yang memanggil prosedur *Function*. Prosedur *Function* inilah yang umumnya disebut fungsi (Kurniawan, 2005 : 77).

Prosedur *Function* dapat membawa argument seperti *constant*, *variable*, atau *expression* yang dijalankan dengan memanggil prosedur tersebut. Jika prosedur *function* tidak mempunyai *argument*, pernyataan *function* harus memasukan tempat kosong dalam tanda kurung. Nilai yang dihasilkan oleh sebuah fungsi dengan menempatkan sebuah nilai pada nama dalam satu atau lebih pernyataan dari prosedur tersebut. Bentuk penulisan Prosedur *Function*:

Private/ Public Function name ()

Statements

End Function

#### 2.5 *Record dan Array*

*Record* adalah struktur data yang tersusun atas elemen-elemen yang jumlahnya tertentu dan tipe data elemennya dapat berbeda-beda (Sismoro, 2004 : 59). *Array* adalah suatu tipe data terstruktur yang berupa sejumlah data sejenis (bertipe data sama) yang jumlahnya sama dan diberi suatu nama tertentu (Basuki, 2006 : 60). Variabel *array* dapat dibuat jika dalam pemrograman membutuhkan lebih dari satu variabel dengan nama sama dan dibedakan dengan nomor. Sebagai contoh variabel array x menampung nilai-nilai bilangan bulat {3, 6, 8, 7, 5, 1} berarti indeks untuk



variabel *x* ini adalah 1 sampai dengan 6, dan ditulis sebagai *x*(1), *x*(2), *x*(3), *x*(4), *x*(5) dan *x*(6). Array terbagi menjadi dua, yaitu array 1 dimensi dan multi dimensi.

*Record* dan *array* merupakan struktur data yang panjangnya tertentu. Namun demikian, terdapat perbedaan di antara keduanya, yakni:

- 1) Elemen *record* bersifat heterogen, yaitu campuran beberapa tipe data. Sedangkan elemen *array* bersifat homogen.
- 2) Elemen *record* diidentifikasi dengan simbol/*identifier*, sedangkan elemen *array* diidentifikasi dengan indeks (Sismoro, 2004 : 60).

### 2.5.1 Array 1 dimensi

Array 1 dimensi adalah variabel yang digunakan untuk menampung beberapa nilai hanya pada satu dimensi (Basuki, 2006 : 60). Sebagai contoh, seperti menempatkan nilai pada baris-baris di sebuah kolom pada *worksheet excel*. Sebuah *array* dideklarasikan dengan cara yang sama dengan variabel, yaitu menggunakan perintah *Dim*, kemudian diikuti dengan tanda kurung dan jumlah.

*Dim nama\_array(jumlah\_elemen) As tipe\_data.*

Sebagai contoh untuk membuat *array* bilangan bulat yang dapat menampung 10 bilangan dapat dituliskan dengan:

*Dim as bilangan(10) integer*

Bila jumlah *array* sudah ditentukan 10 maka tidak boleh menggunakan data lebih dari 10, karena 10 menyatakan jumlah maksimum dari data yang akan ditampung adalah suatu *array*. Sebagai contoh:

Bilangan(4) = 100: Perintah ini benar

Bilangan(12) = 8: Perintah ini salah karena batasnya sudah ditentukan 10.

Pada *array* dengan tipe numerik, nilai batas bawah 0. Untuk menghindari kesalahan, bisa mendeklarasikan nilai batas bawah secara manual dengan *format*:

*Dim nama\_array(batas\_bawah To batas\_atas) As tipe\_data*

*Visual Basic* juga mengenal *array* dinamis, dimana jumlahnya bisa tak terbatas. Untuk mendefinisikan *array* dinamis dapat dilakukan dengan mendefinisikan *array* tanpa menuliskan jumlah maksimum arraynya sebagai berikut:

*Dim variabel() as tipe\_data*

### 2.5.2 Array 2 Dimensi

*Array* dua dimensi adalah variabel yang bisa menampung beberapa nilai yang memiliki 2 dimensi (Basuki, 2006 : 66). Apabila *array* 1 dimensi dapat diilustrasikan seperti mengisi nilai pada baris-baris di sebuah kolom tertentu, maka *array* 2 dimensi dapat diilustrasikan seperti mengisi nilai pada baris dan kolom tertentu pada *worksheet*. Pada ilustrasi tersebut elemen yang digunakan ada 2, yaitu elemen baris dan elemen kolom.

## 2.6 Tahap-tahap dalam Pemograman

Menurut Rachmat (2010 : 41), berikut adalah tahap-tahap dalam membuat program:

### 1) Mendefinisikan masalah

Menurut hukum Murphy (oleh Henry Ledgard), “Semakin cepat menulis program, akan semakin lama kita dapat menyelesaikannya”. Hal tersebut berlaku untuk permasalahan yang kompleks. Tentukan masalahnya, apa saja yang harus dipecahkan dengan menggunakan komputer, dan apa *inputan* serta *outputnya*.

### 2) Menemukan Solusi

Setelah masalah didefinisikan, maka langkah berikutnya adalah menentukan solusi. Jika masalah terlalu kompleks, maka ada baiknya masalah tersebut dipecah menjadi modul-modul kecil agar lebih mudah diselesaikan.

Contohnya masalah invers matriks, maka kita dapat membagi menjadi beberapa modul:

- Meminta masukkan berupa matriks bujur sangkar
- Mencari invers matriks
- Menampilkan hasil kepada pengguna

Dengan penggunaan modul tersebut, program utama akan menjadi lebih singkat dan mudah dilihat.

### 3) Memilih Algoritma

Pilihlah algoritma yang benar-benar sesuai dan efisien untuk permasalahan tersebut.

### 4) Menulis Program

Pilihlah bahasa yang mudah dipelajari, mudah digunakan, dan lebih baik lagi jika sudah dikuasai, memiliki tingkat kompatibilitas tinggi dengan perangkat keras dan *platform* lainnya.

## 5) Menguji program

Setelah program jadi, silahkan uji program tersebut dengan segala macam kemungkinan yang ada, termasuk *error-handling*nya sehingga program tersebut akan benar-benar handal dan layak digunakan.

## 6) Menulis dokumentasi

Menulis dokumentasi sangat penting agar pada suatu saat jika kita akan melakukan perubahan atau membaca *source code* yang sudah kita tulis dapat kita ingat-ingat lagi dan kita akan mudah membacanya. Caranya adalah dengan menuliskan komentar komentar kecil tentang apa maksud kode tersebut, untuk apa, variabel apa saja yang digunakan, untuk apa, dan parameter-parameter yang ada pada suatu prosedur dan fungsi.

## 7) Merawat program

Program yang sudah jadi perlu dirawat untuk mencegah munculnya *bug* yang sebelumnya tidak terdeteksi. Atau mungkin juga pengguna membutuhkan fasilitas baru yang dulu tidak ada.

## 2.7 Pernyataan dalam Visual Basic

Pernyataan dalam suatu program aplikasi digunakan untuk mengendalikan alur eksekusi suatu program aplikasi. Dalam *Visual Basic* terdapat beberapa jenis pernyataan yang digunakan untuk mengoperasikan proses perputaran alur program (*loop*), memilih pernyataan lain untuk dijalankan atau memindah kontrol. Pernyataan dalam *Visual Basic* dapat dikelompokkan menjadi dua, yaitu struktur kontrol perulangan dan struktur kontrol keputusan.

### 2.7.1 Struktur Kontrol Keputusan

Struktur kontrol keputusan adalah pernyataan yang mengijinkan *user* untuk memilih dan mengeksekusi blok kode dan mengabaikan blok kode yang lain (MADCOMS, 2008:115).

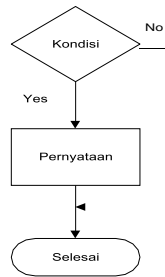
#### 1) If... Then

Pernyataan percabangan bersyarat ini digunakan untuk menjalankan suatu pilihan pernyataan atau blok pernyataan jika suatu kondisi terpenuhi dan sebaliknya (MADCOMS, 2008:115).

Bentuk penulisan:

If <kondisi> Then <pernyataan>

Bentuk *flowchart*:



Gambar 2.4 *Flowchart If ... Then ...*

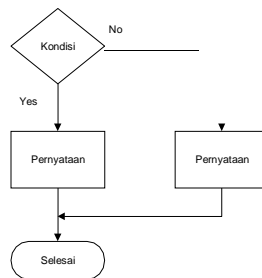
## 2) *If... Then... Else*

Pernyataan ini akan menjalankan sebuah blok pernyataan jika kondisi bernilai *true* dan blok pernyataan lain bernilai *false* (MADCOMS, 2008:117).

Bentuk penulisan:

```
If <kondisi> Then<Pernyataan_Jika_Kondisi_Benar>
Else
<Pernyataan_Jika_Kondisi_Salah>
End if
```

Bentuk *flowchart*:



Gambar 2.5 *Flowchart If... Then... Else*

## 3) *If... Then... ElseIf*

Kondisi yang merupakan suatu ekspresi logika akan diuji perintah *If* yang pertama ada di sebelah kirinya. Jika kondisi benar maka blok perintah yang terletak di bawahnya akan dikerjakan sampai menerima perintah *ElseIf* atau *Else* lalu melompat ke pernyataan *End If*.

Bentuk penulisan:

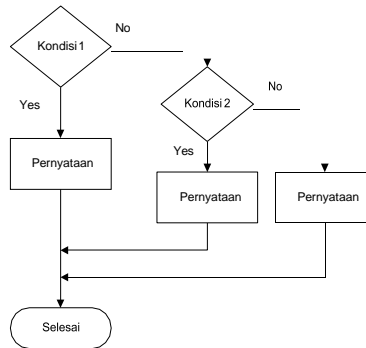
```
If <kondisi_1> Then
    <Pernyataan_1>
ElseIf <Kondisi_2> Then
```

<Pernyataan\_2>

...

```
ElseIf<Kondisi_n> Then  
<Pernyataan> Else  
<Pernyataan>  
End If
```

Bentuk *flowchart*:



Gambar 2.6 Flowchart If... Then... ElseIf

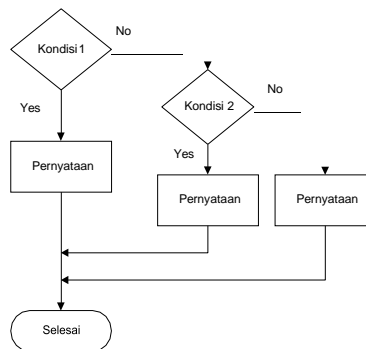
#### 4) Select Case

*Select case* akan menguji ekspresi tunggal yang akan dievaluasi hanya sekali pada bagian atas dari struktur keputusan, kemudian hasilnya dibandingkan dengan beberapa nilai (MADCOMS, 2008:118). Jika ada salah satu *case* yang cocok, maka program akan dijalankan.

Bentuk penulisan:

```
Select Case <Variabel Penguji>  
Case <Kondisi_1>  
    <Pernyataan_1>  
Case  
    <Kondisi_2>  
    <Pernyataan_2>  
Case  
    <Kondisi_n>  
    <Pernyataan_n>  
End Select
```

Bentuk *flowchart*:



Gambar 2.7 *Flowchart Select Case*

## 2.7.2 Struktur Kontrol Pengulangan

Struktur kontrol pengulangan digunakan untuk melakukan pengulangan kode program. Hal ini dilakukan untuk lebih mempersingkat perintah (MADCOMS, 2008:119). Berikut merupakan jenis dari struktur kontrol perulangan.

### 1) *For... Next*

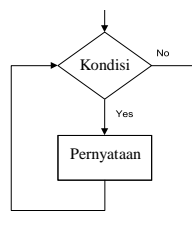
Pernyataan ini digunakan untuk melakukan pengulangan suatu blok program dengan dibatasi nilai awal dan akhir (MADCOMS, 2008:120). Variabelnya bernilai numerik dan mempunyai nilai yang berurutan. Nilai penghitung dari variabelnya dapat bertambah atau berkurang pada setiap pengulangan yang dilakukan dengan menambahkan kata *step*.

Bentuk penulisan:

```

For <variabel> = <nilai_awal> To <nilai_akhir> Step <nilai_penghitung>
  <Pernyataan>
Next <variabel>
  
```

Bentuk *flowchart*:



Gambar 2.8 *Flowchart For... Next*

### 2) *Do... Loop*

*Statement* ini mengulang blok statemen bila kondisi benar atau sampai kondisi ini menjadi benar. Bila tidak ada perintah keluar, proses pengulangan akan terus berlangsung.

### 3) *Do While... Loop*

Pengulangan *Do While... Loop* digunakan untuk menjalankan satu atau beberapa pernyataan yang akan diulang selama kondisi terpenuhi (MADCOMS, 2008:121).

Bentuk penulisan:

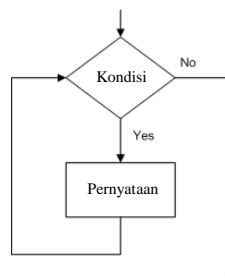
Do While <Kondisi>

<Pernyataan>

...

Loop

Bentuk *flowchart*:



Gambar 2.9 *Flowchart Do While... Loop*

### 4) *Do... Loop While*

*Statement* ini akan mengerjakan pernyataan dalam blok statemen ketika kondisi bernilai benar dan akan berhenti ketika kondisi sudah bernilai salah (MADCOMS, 2008:121).

Bentuk penulisan:

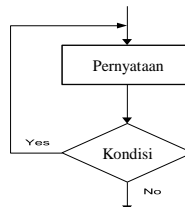
Do

...

Loop While <Kondisi>

<Pernyataan>

Bentuk *flowchart*:



Gambar 2.10 *Flowchart Do... Loop While*



### 5) *Do Until... Loop*

Pada pengulangan *Do Until... Loop*, pengulangan akan terus berjalan apabila kondisi bernilai salah dan akan berhenti jika kondisi bernilai benar (MADCOMS, 2008:121).

Bentuk penulisan:

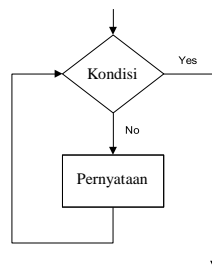
Do Until <Kondisi>

<Pernyataan>

...

Loop

Bentuk *flowchart*:



Gambar 2.11 *Flowchart Do Until... Loop*

### 6) *Do... Loop Until*

*Statement* ini akan mengerjakan pernyataan dalam blok statemen ketika kondisi bernilai salah dan akan berhenti jika kondisi bernilai benar (MADCOMS, 2008:121).

Bentuk Penulisan:

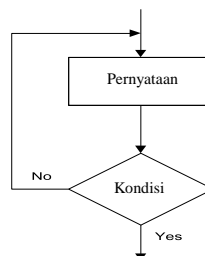
Do

...

Loop Until <Kondisi>

<Pernyataan>

Bentuk *flowchart*:



Gambar 2.12 *Flowchart Do... Loop Until*

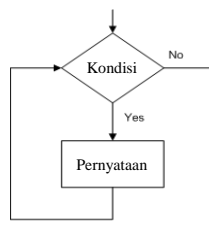
### 7) *While... Wend*

Statemen ini digunakan apabila kita ingin pernyataan pada program dijalankan selama kondisi ekspresi yang ditetapkan masih bernilai benar (MADCOMS, 2008:123).

Bentuk penulisan:

```
While <Kondisi>  
<Pernyataan>  
...  
Wend
```

Bentuk *flowchart*:



Gambar 2.13 *Flowchart While... Wend*

### 2.7.3 Contoh Penggunaan Struktur Kontrol Keputusan

Untuk membantu pengambilan keputusan mengenai alur program mana yang dipilih, dibutuhkan sesuatu yang disebut struktur kontrol (Kurniawan, 2005 : 49). Struktur Kontrol dalam *Visual Basic 6.0* antara lain *If ... Then ...*, *If ... Then ... Else*, *If ... Then ... ElseIf*, dan *Select Case*. Contoh *coding* dari penggunaan *If ... Then ... ElseIf* ditunjukkan pada gambar 2.14.

```
Sub main()  
    Dim Ujian As Integer = 85  
    Dim Skor As String  
    If Ujian >= 85 Then  
        Skor = "A"  
    ElseIf Ujian >= 75 Then  
        Skor = "B"  
    ElseIf Ujian >= 45 Then  
        Skor = "C"  
    Else  
        Skor = "D"  
    End If  
    Console.WriteLine ("Nilai Ujian Anda = " & Ujian)  
    Console.WriteLine ("Skor akhir = " & Skor)  
End Sub
```

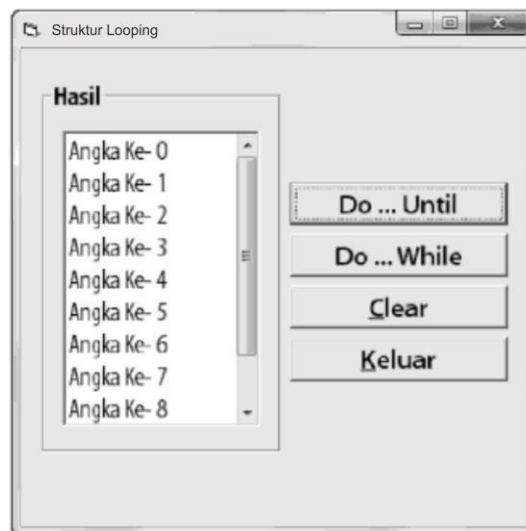
Gambar 2.14 Contoh Penggunaan *If... Then... ElseIf*

Pembacaan alur program pada Gambar 2.14 adalah sebagai berikut:

- Variabel Ujian diinisialisasi dengan nilai 75.
- Deklarasi Variabel skor.
- Jika nilai ujian lebih dari atau sama dengan 75, maka skor bernilai A. Jika tidak, alur program berikutnya dievaluasi lebih lanjut.
- Jika nilai ujian lebih dari atau sama dengan 60, maka skor bernilai B. Jika tidak, alur program berikutnya dievaluasi lebih lanjut.
- Jika nilai ujian lebih dari atau sama dengan 45, maka skor bernilai C. Jika tidak, skor bernilai D.
- Menampilkan *output* ke *console*.

#### 2.7.4 Contoh Penggunaan Struktur Kontrol Pengulangan

Menurut Aminudin (2016 : 97), *Visual Basic* mendukung beberapa versi statement *Do*. Pengulangan dengan menggunakan *While* mungkin yang paling populer digunakan dalam pemrograman *Visual Basic*. Seperti statement *If... Then, Do... While* juga membutuhkan ekspresi perbandingan untuk keluar dari *looping*. Gambar 2.15 menampilkan angka dengan *Do ... Loop*.



Gambar 2.15 Program menampilkan angka dengan *Do ... Loop*

Setelah membuat *interface* pada program sesuai dengan Gambar 2.15, ketik kode programnya seperti pada Gambar 2.16.

```
Private Sub Cmdclear_Click()  
    Lsthasil.Clear  
End Sub  
  
Private Sub Cmdexit_Click()  
End  
End Sub  
  
Private Sub Cmduntil_Click()  
    Lsthasil.Clear  
    i = 0  
    Do Until i > 10  
        Lsthasil.AddItem "Angka Ke-" & i  
        i = i + 1  
    Loop  
End Sub  
  
Private Sub Cmdwhile_Click()  
    Lsthasil.Clear  
    i = 0  
    Do While i <= 10  
        Lsthasil.AddItem "Angka Ke-" & i  
        i = i + 1  
    Loop  
End Sub
```

Gambar 2.16 *Coding* Program menampilkan angka dengan *Do ... Loop*

## DAFTAR PUSTAKA

- Aminudin, Nur. 2016. *Dasar Pemrograman Visual Basic*. Yogyakarta: Penerbit Andi.
- Basuki, Achmad. 2006. *Algoritma Pemrograman 2 Menggunakan Visual Basic 6.0*. Surabaya: ITS.
- Darsono, Max, dkk. 2001. *Belajar dan Pembelajaran*. Semarang: IKIP Semarang Press.
- Kurniawan, Yahya. 2005. *Belajar Sendiri: Otomatisasi Office 2003 dengan Visual Basic .NET 2005*. Jakarta: Gramedia.
- MADCOMS. 2008. *Microsoft Visual Basic 6.0 untuk Pemula*. Madiun: Penerbit Andi.
- Priyono, Christopher Henry. 2007. *Seri Penuntun Praktis: Siapa Bilang Pemrograman Itu Sulit*. Jakarta: Gramedia.
- Rachmat, Antonius. 2010. *Algoritma dan Pemrograman dengan Bahasa C*. Yogyakarta: Penerbit Andi.
- Rahman, Arif. 2016. *Modul Algoritma dan Pemrograman*. Malang: <http://arifindustri.lecture.ub.ac.id/lecturer-notes/lec-alprog>
- Ramadhan, Arief. 2004. *Seri Penuntun Praktis: Microsoft Visual Basic 6.0*. Jakarta: Gramedia.
- Sismoro, Heri dan Kusri Iskandar. 2004. *Struktur Data dan Pemrograman dengan Pascal*. Yogyakarta: Penerbit Andi.
- Sitorus, Lamhot. 2015. *Algoritma dan Pemrograman*. Yogyakarta: Penerbit Andi.
- Suarga. 2004. *Algoritma Pemrograman*. Makassar: Penerbit Andi.
- Tim Dosen Alprog. 2016. *Modul ajar tidak dipublikasikan: Algoritma dan Flowchart*. Malang: Teknik Industri Universitas Brawijaya
- Utami, Ema dan Sukrisno. 2005. *10 Langkah Belajar Logika dan Algoritma Menggunakan Bahasa C dan C++ di GNU/Linux*. Yogyakarta: Andi.