BAB III
ANALISIS DAN PERANCANGAN

3.1 Analisis Sistem

Dalam analisis sistem dilakukan penguraian dari suatu sistem yang utuh ke dalam bagian-bagian komponennya dengan maksud untuk mengidentifikasikan dan mengevaluasi permasalahan-permasalahan sehingga ditemukan kelemahan-kelemahannya, kesempatan-kesempatan, hambatan-hambatan yang terjadi dan kebutuhan-kebutuhan yang diharapkan sehingga dapat diusulkan perbaikannya.

3.1.1 Analisis Masalah

Berdasarkan pengamatan, pembelajaran integral masih menjadi hal yang cukup sulit untuk dipahami. Apalagi jika pencarian hasil integral menggunakan metode analitik yang cukup panjang dan rumit. Oleh karena itu, perlu dibangun suatu sistem yang dapat mempermudah pembelajaran dan pencarian hasil integral dengan menggunakan metode lain (non-analitik) yang lebih mudah dipahami.

3.1.2 Gambaran Umum Sistem

Perangkat lunak yang akan dibangun terlebih dahulu dipaparkan di bagian ini, sehingga dapat dianalisis masukan (*input*), keluaran (*output*) dan proses apa saja yang terlibat di dalamnya.

Ketika pertama kali masuk ke dalam sistem, maka akan ditampilkan menu pilihan untuk user. Di bagian ini user harus meng-input-kan fungsi, batasan fungsi (batas atas dan batas bawah), memilih banyaknya dimensi integral (integral lipat satu atau lipat dua), dan memilih jenis metode *Monte Carlo* yang akan digunakan. Sistem akan menampilkan langkah-langkah perhitungan fungsi integral yang diinputkan oleh user. User dapat memilih untuk menyimpan hasil latihannya ke dalam bentuk file. User juga dapat menambahkan fungsi yang akan disimpan di dalam *memory* yang kapasitas penyimpanannya terbatas.

3.1.3 Analisis Kebutuhan Sistem Non Fungsional

Analisis non-fungsional merupakan analisis yang dibutuhkan untuk menentukan spesifikasi kebutuhan sistem. Spesifikasi ini juga meliputi elemen atau komponen-komponen apa saja yang dibutuhkan untuk sistem yang akan dibangun sampai dengan sistem tersebut diimplementasikan. Analisis kebutuhan ini juga menentukan spesifikasi masukan yang diperlukan sistem, keluaran yang akan dihasilkan sistem dan proses yang dibutuhkan untuk mengolah masukan sehingga menghasilkan suatu keluaran yang diinginkan.

Pada analisis kebutuhan sistem non fungsional ini dijelaskan mengenai input, output, perangkat keras (*hardware)*, perangkat lunak (*software)*, dan pengguna (*user*).

3.1.3.1 Analisis Input

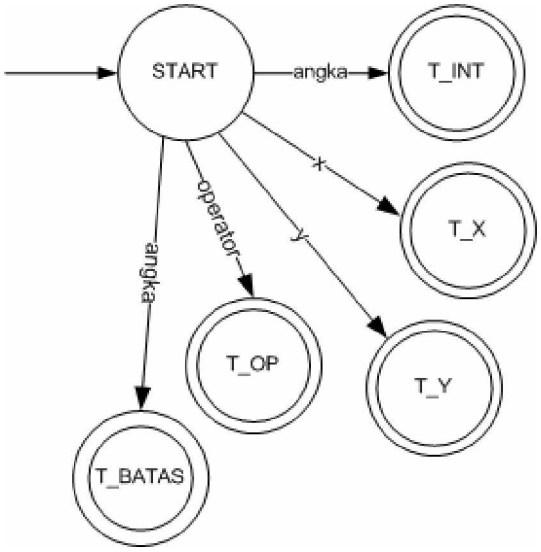
Sebuah sistem harus memiliki masukkan agar menghasilkan sesuatu. Masukkan (*input*) dari user bagi aplikasi ini adalah fungsi aljabar rasional dalam bentuk ekspresi matematika, titik sampel, batasan fungsi, pilihan dimensi integral, nilai pembagian interval untuk metode MISER *Monte Carlo*, dan fungsi hampiran p(x) untuk metode VEGAS *Monte Carlo*.

Bentuk input memiliki aturan tertentu agar fungsi yang akan dimasukkan nanti dapat diterima oleh sistem. Aturan bentuk input adalah sebagai berikut :

1. Fungsi integral yang dimasukkan adalah fungsi aljabar rasional yang didalamnya hanya mengandung operator ^, *, /, +, -.
2. Fungsi aljabar yang dimasukkan maksimal mengandung dua variabel.
3. Batas maksimal input fungsi aljabar adalah 15 digit.
4. Batas atas dan batas bawah fungsi bertipe data integer, dengan batas atas harus selalu lebih besar dari batas bawah dan inputan berupa bilangan bulat positif dari 0-99.

3.1.3.1.1 Analisis Leksikal

Fungsi integral yang dimasukkan harus mengikuti prosedur scan agar fungsi yang dimasukkan nantinya dapat dibaca oleh sistem. Input fungsi memiliki himpunan *token* seperti t_INT, t_X, t_Y, t_OP, t_BATAS. *Token* t_INT dan t_BATAS mengandung digit 0..9, *token* t_X mengandung variabel x, *token* t_Y mengandung variabel y, *token* t_OP mengandung operator /, *, -, + dan ^. Berikut adalah *state diagram* atau diagram keadaan dari prosedur scan.



Gambar 3.1 State Diagram

3.1.3.1.2 Analisis Sintaksis

Analisis sintaksis merupakan kelanjutan dari analisis leksikal yang mengubah prosedur *scan* ke dalam bentuk *token* beserta *parsing*-nya. Di bagian ini akan dibahas contoh bentuk masukan dan mencari bentuk BNF (*Backus Naur Form*) dan diagram sintaksnya.

3.1.3.1.2.1 Token

Contoh bentuk input : (2*x*y)+(x^2) -(x^(1/2)).

Dari contoh diatas dapat dianalisa sintaks :

V = { <exp>, <operand>, <operator>, <delimeter>, <batas> }

T = {t_INT, t_X, t_Y, t_BATAS, t_PLUS, t_MIN, t_MUL, t_DIV, t_SQR, t_LPARENT, t_RPARENT}

S = <exp>

P =

<exp> ::= <operand> <operator> <operand> | <operand> <operator> <operand> <operator> <operand> | <operand> <operator> <delimeter> <operand> <operator> <operand> <delimeter> | <delimeter> <operand> <operator> <operand> <delimeter> <operator> <operand> | <operand> <operator> <delimeter> <operand> <operator> <delimeter> <operand> <operator> <operand> <delimeter> <delimeter>

<delimeter> ::= t_LPARENT | t_RPARENT

<operator> ::= t_PLUS | t_MIN | t_MUL | t_DIV | t_SQR

<operand> ::= t_INT | t_X | t_Y

<batas> ::= t_BATAS

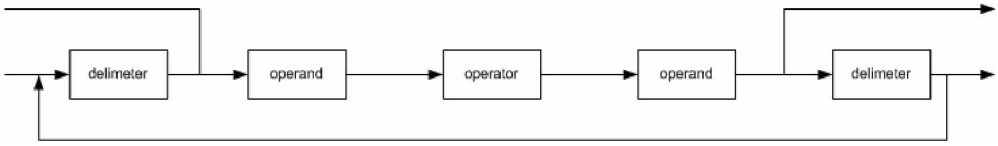
t_INT ::= 0..9

t_BATAS ::= 0..99

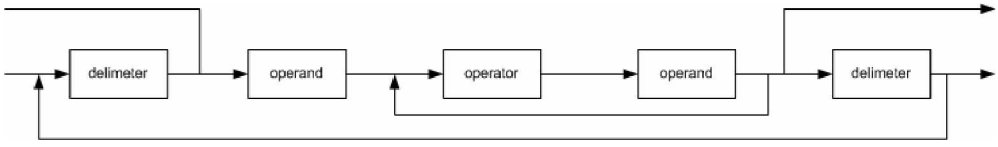
t_X ::= x

t_Y ::= y

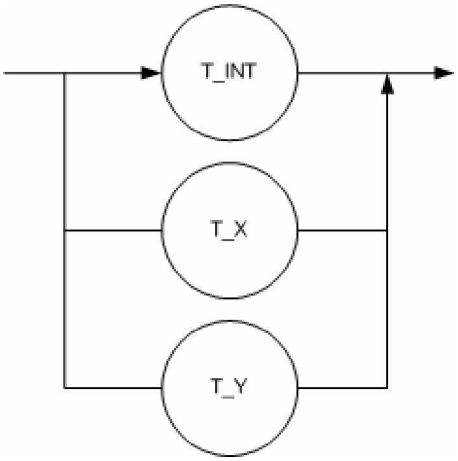
Dari bentuk BNF diatas maka dapat dibentuk diagram sintaksnya sebagai berikut :



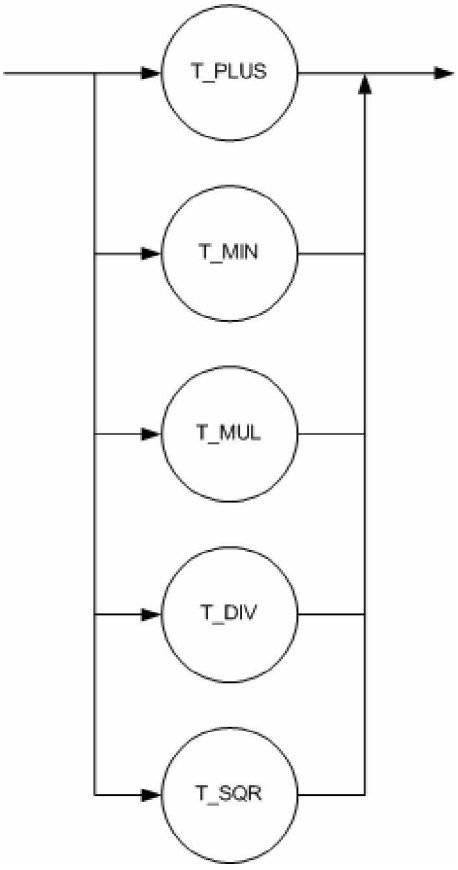
Gambar 3.2 Diagram Sintaks Exp Lipat1



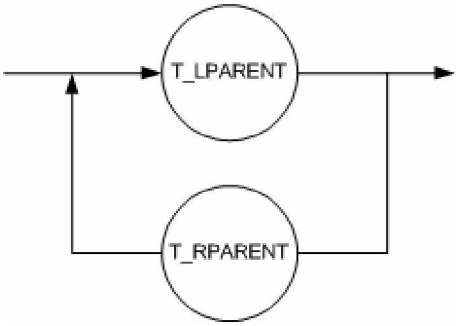
Gambar 3.3 Diagram Sintaks Exp Lipat2



Gambar 3.4 Diagram Sintaks Operand



Gambar 3.5 Diagram Sintaks Operator



Gambar 3.6 Diagram Sintaks Delimeter

3.1.3.1.2.2 Parsing

Ekspresi aritmatika yang digunakan di dalam perangkat lunak adalah ekspresi aritmatika dalam bentuk notasi *infix*. Notasi *infix* merupakan cara penulisan ekspresi aritmatika, dimana operator diletakkan di tengah-tengah atau di antara operand-operand. Operator merupakan fungsi pangkat, kali, bagi, tambah dan kurang, sedangkan operand adalah angka atau variabel. Contoh notasi *infix*: (a + b) / 2.

Misalkan, fungsi *x* yang akan diintegralkan adalah sebagai berikut:

$$\frac{\sqrt{1+x^{1/3}}}{x^{2/3}}$$

Maka, ekspresi aritmatika dari fungsi *x* di atas dapat ditulis dalam bentuk notasi *infix* sebagai berikut:

((1 + *x*^(1/3)) ^ (1/2)) / (*x*^(2/3))

Sistem yang akan dibangun menggunakan metode *Leftmost Derivation* untuk menurunkan (*parsing*) input ekspresi aritmatika. Berdasarkan metode *Leftmost Derivation*, ekspresi aritmatika akan diperiksa dan diturunkan dari

sebelah kiri ke sebelah kanan. Ekspresi aritmatika yang di-*input* oleh *user* harus dipecah menjadi sub-sub ekspresi aritmatika yang paling sederhana (terdiri atas 1 operator dan 2 operand), supaya perangkat lunak dapat menyelesaikan ekspresi aritmatika tersebut. Pemecahan ekspresi aritmatika ke bentuk sub-sub ekspresi aritmatika harus mengikuti urutan atau prioritas operasi. Operasi dengan prioritas yang lebih tinggi harus dikerjakan terlebih dahulu. Berikut adalah prioritas pengerjaan operasi dari yang tertinggi sampai prioritas yang paling rendah:

1. Operasi perpangkatan.

   Operasi perpangkatan adalah operasi yang memiliki prioritas tertinggi dan akan dikerjakan terlebih dahulu. Operasi pangkat dilambangkan dengan tanda ^.

2. Operasi perkalian dan pembagian.

   Kedua operasi ini memiliki prioritas tertinggi setelah operasi perpangkatan. Operasi perkalian dilambangkan dengan tanda *****, sedangkan operasi pembagian dilambangkan dengan tanda /.

3. Operasi penambahan dan pengurangan.

   Kedua operasi ini merupakan operasi yang paling akhir dikerjakan. Operasi penambahan dilambangkan dengan tanda +, sedangkan operasi pengurangan dilambangkan dengan tanda -.

Jika terdapat operasi yang memiliki prioritas yang sama, maka operasi yang terletak di sebelah kiri akan dikerjakan terlebih dahulu. Kemudian, bila ditemukan *delimiter*, berupa tanda kurung ‘()‘, maka sub ekspresi aritmatika yang berada di dalam *delimiter* ini harus dikerjakan terlebih dahulu. Sub ekspresi

aritmatika akan disimpan dalam bentuk *E*[*n*] = <*operand*> <*operator*> <*operand*>.

Secara umum, cara kerja algoritma untuk memeriksa validasi struktur ekspresi aritmatika (apakah benar atau tidak) dan sekaligus memecah ekspresi aritmatika menjadi sub-sub ekspresi adalah sebagai berikut:

1. Periksa apakah fungsi integral masih kosong. Bila ya, maka munculkan pesan kesalahan dan keluar dari algoritma.
2. Periksa apakah jumlah tanda kurung buka '(' sama dengan tanda kurung tutup ')'. Bila tidak sama, maka munculkan pesan kesalahan dan keluar dari algoritma.
3. Carilah tanda kurung tutup yang pertama ')', dimulai dari ekspresi aritmatika yang paling kiri. Kemudian, cari tanda kurung buka '(' yang ada di depan tanda kurung tutup tersebut. Sub ekspresi yang berada di antara kedua tanda kurung tersebut di-*passing* ke langkah ke-4. Bila tidak terdapat tanda kurung, maka *passing* semua ekspresi aritmatika ke langkah ke-4.
4. Bentuklah sub ekspresi baru (dalam bentuk *E*[*n*]) dari sub ekspresi yang dihasilkan pada langkah-3, dengan mengikuti prosedur berikut:
    a. Cari operator pangkat ('^') di dalam ekspresi aritmatika. Apabila ada, maka konversi sub ekspresi aritmatika ke bentuk *E*[*n*] = *a* ^ *b*. Apabila tidak ditemukan operasi pangkat, maka lanjutkan ke poin 4b. Periksa juga bahwa operasi pangkat harus diapit oleh 2 operand atau 2 angka. Bila tidak, maka munculkan pesan kesalahan dan keluar dari algoritma.

b. Cari operator kali (‘*’) atau operator bagi (‘/’) di dalam ekspresi aritmatika. Apabila ditemukan operasi tersebut, maka konversi sub ekspresi aritmatika ke bentuk $E[n] = a * b$ atau $E[n] = a / b$. Apabila tidak ditemukan operasi kali atau operasi bagi, maka lanjutkan ke poin 4c. Periksa juga bahwa operasi harus diapit oleh 2 operand atau 2 angka. Bila tidak, maka munculkan pesan kesalahan dan keluar dari algoritma.

c. Cari operator tambah (‘+’) atau operator kurang (‘-’) di dalam ekspresi aritmatika. Apabila ditemukan operasi tersebut, maka konversi sub ekspresi aritmatika ke bentuk $E[n] = a + b$ atau $E[n] = a – b$. Periksa juga bahwa operasi harus diapit oleh 2 operand atau 2 angka. Bila tidak, maka munculkan pesan kesalahan dan keluar dari algoritma.

5. Ulangi proses di atas hingga ekspresi aritmatika selesai dikonversi ke dalam bentuk $E[n]$.

Agar lebih jelas, coba perhatikan contoh berikut ini. Misalkan ekspresi aritmatika yang akan dipecah adalah **((1 + x^(1/3)) ^ (1/2)) / (x^(2/3))**, maka langkah-langkah pemecahan ekspresi aritmatika ke bentuk sub ekspresi adalah sebagai berikut:

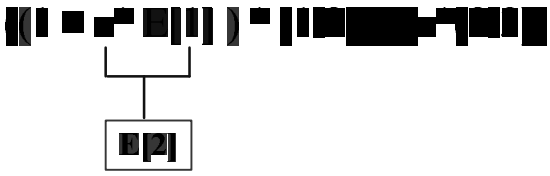
<u>Langkah-1:</u>

Bentuk sub ekspresi baru, E[1] = 1 / 3.



Langkah-2:

Bentuk sub ekspresi baru, E[2] = x ^ E[1].



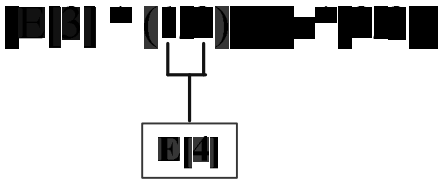
Langkah-3:

Bentuk sub ekspresi baru, E[3] = 1 + E[2].



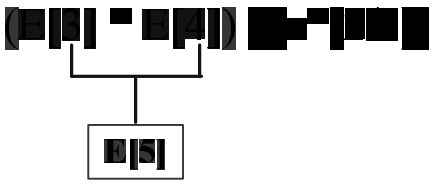
Langkah-4:

Bentuk sub ekspresi baru, E[4] = 1 / 2.



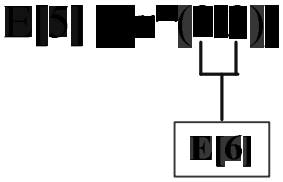
Langkah-5:

Bentuk sub ekspresi baru, E[5] = E[3] ^ E[4].



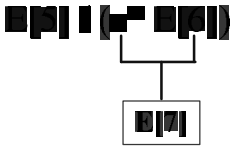
Langkah-6:

Bentuk sub ekspresi baru, E[6] = 2 / 3.



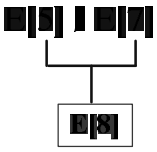
Langkah-7:

Bentuk sub ekspresi baru, E[7] = x ^ E[6].

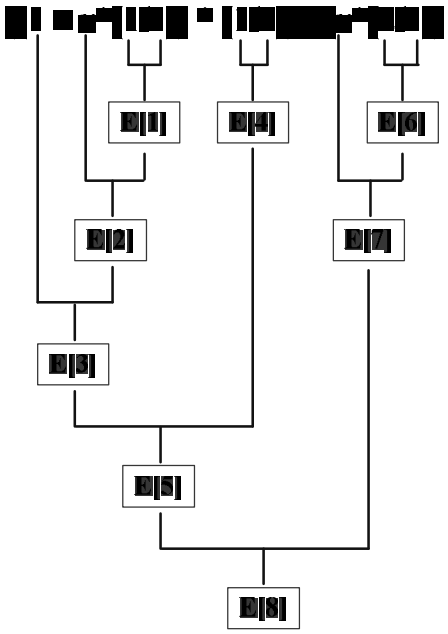


Langkah-8:

Bentuk sub ekspresi baru, E[8] = E[5] / E[7].



Pada langkah ke-8, ekspresi aritmatika sudah selesai dipecah menjadi sub-sub ekspresi dalam bentuk *E*[*n*]. Secara umum, proses pemecahan ekspresi aritmatika di atas dapat juga digambarkan sekaligus sebagai berikut:



Dengan demikian, hasil pemecahan ekspresi aritmatika **((1 + x^(1/3)) ^ (1/2)) / (x^(2/3))** ke bentuk sub ekspresi aritmatika adalah sebagai berikut:

1. E[1] = 1 / 3.
2. E[2] = x ^ E[1].
3. E[3] = 1 + E[2].
4. E[4] = 1 / 2.
5. E[5] = E[3] ^ E[4].
6. E[6] = 2 / 3.
7. E[7] = x ^ E[6].
8. E[8] = E[5] / E[7].

Misalkan, nilai x = 1, maka hasil perhitungan dari ekspresi aritmatika **((1 + x^(1/3)) ^ (1/2)) / (x^(2/3))**, dengan menggunakan sub ekspresi aritmatika adalah sebagai berikut:

1. E[1] = 1 / 3 = 0.33333
2. E[2] = x ^ E[1] = 1 ^ 0.33333 = 1
3. E[3] = 1 + E[2] = 1 + 1 = 2
4. E[4] = 1 / 2 = 0.5
5. E[5] = E[3] ^ E[4] = 2 ^ 0.5 = 1.4142
6. E[6] = 2 / 3 = 0.66667
7. E[7] = x ^ E[6] = 1 ^ 0.66667 = 1
8. E[8] = E[5] / E[7] = 1.4142 / 1 = 1.4142

Ini berarti, fungsi **((1 + *x*^(1/3)) ^ (1/2)) / (*x*^(2/3))** akan menghasilkan nilai 1.4142, untuk *x* = 1.

3.1.3.1.2 Analisis Semantik

Analisis semantik bertugas untuk mengubah hasil parsing ke dalam kode antara atau *intermediate code* agar masukan yang dimasukkan user dapat dibaca oleh mesin. Terdapat dua macam kode antara, yaitu Notasi Postfix dan N-*Tuple*. Sistem ini akan menggunakan notasi *Quadruple* dengan format

<*operator*> <*operand*> <*operand*> <*hasil*>

Hasil adalah *temporary variable* yang bisa ditempatkan pada memory atau register. Contoh instruksi :

((1 + *x*^(1/3)) ^ (1/2)) / (*x*^(2/3))

Bentuk kode antaranya adalah :

1. /, 1, 3, E[1]
2. ^, x, E[1], E[2]
3. +, 1, E[2], E[3]
4. /, 1, 2, E[4]
5. ^, E[3], E[4], E[5]
6. /, 2, 3, E[6]
7. ^, x, E[6], E[7]
8. /, E[5], E[7], E[8]

Pada kode antara di atas dapat dijelaskan, kode antara nomor satu adalah membagi 1 dengan tiga dan menyimpannya di E[1]. Kode antara nomor dua adalah memangkatkan variabel x dengan hasil dari kode antara nomor satu yang tersimpan di E[1], dan seterusnya.

Selanjutnya adalah melakukan pembangkitan kode (*code generation*) dengan mengubah kode antara di atas ke dalam bahasa mesin.

Contoh : Kita akan mengubah kode antara 1 dan 2 ke dalam bahasa mesin.

```
LDA   1
DIV   3
STO   E[1]
LDA    x
SQR   E[1]
STO   E[2]
```

3.1.3.2 Analisis Output

Keluaran yang dihasilkan oleh aplikasi ini bagi user adalah hasil pencarian integral dan langkah-langkah penyelesian dalam bentuk file.

3.1.3.3 Analisis Kebutuhan Perangkat Keras

Agar aplikasi dapat berjalan dengan baik, maka dibutuhkan perangkat keras yang sesuai dengan kebutuhan aplikasi. Adapun perangkat keras yang dibutuhkan oleh pengguna (*user*) agar dapat menjalankan aplikasi ini adalah sebagai berikut :

Tabel 3.1 Spesifikasi Perangkat Keras

Spesifikasi
Prosesor 1.7 MHz
Monitor SVGA 14“
Memori 256 MB
Harddisk 40 GB
Papan ketik (*keyboard*)
Tetikus (*Mouse*)

3.1.3.4 Analisis Kebutuhan Perangkat Lunak

Sistem dibangun pada Sistem Operasi Windows XP Service Pack 2. Sedangkan tools yang dipakai untuk membangun aplikasi adalah Microsoft Visual Basic 6.0.

3.1.3.5 Analisis Pengguna (*User*)

Pengguna dari aplikasi ini adalah *public*, oleh karena itu aplikasi ini menggunakan antarmuka berbasis *Graphic User Interface* (GUI) yang dirancang

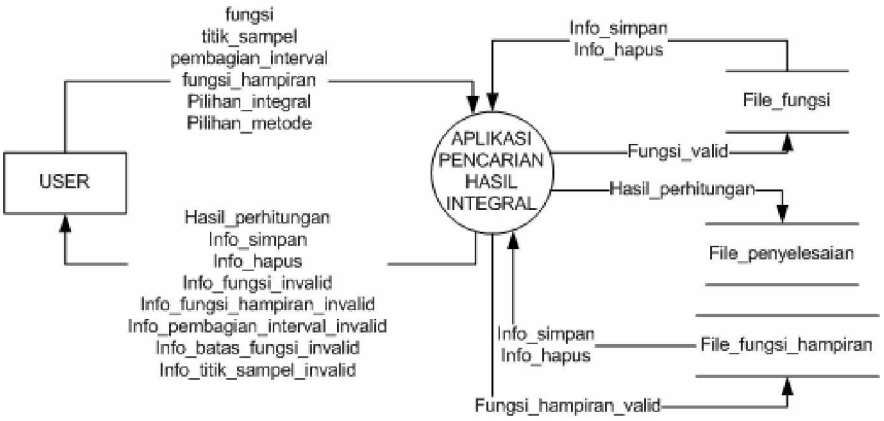
secara sederhana, menarik dan interaktif agar pengguna dapat mengoperasikannya dengan mudah (*user friendly*).

3.2 Analisis Kebutuhan Fungsional

Perancangan sistem merupakan suatu proses yang mengaplikasikan berbagai teknik dan prinsip yang bertujuan mendefinisikan kebutuhan suatu sistem sedemikian detail sehingga mendekati keadaan fisik yang sebenarnya dan untuk menghasilkan sesuatu yang baru atau memperbaharui sistem yang ada untuk meningkatkan efektifitas kerja yang sesuai dengan teknologi dan fasilitas yang tersedia. Dimana suatu perancangan sistem informasi merupakan suatu langkah awal dalam pembuatan suatu sistem sehingga perlu diperhatikan kebutuhan dari sistem atau *software* yang diharapkan.

3.2.2 Diagram Konteks Sistem

Berikut adalah *context diagram* atau diagram konteks dari sistem yang akan dibangun.



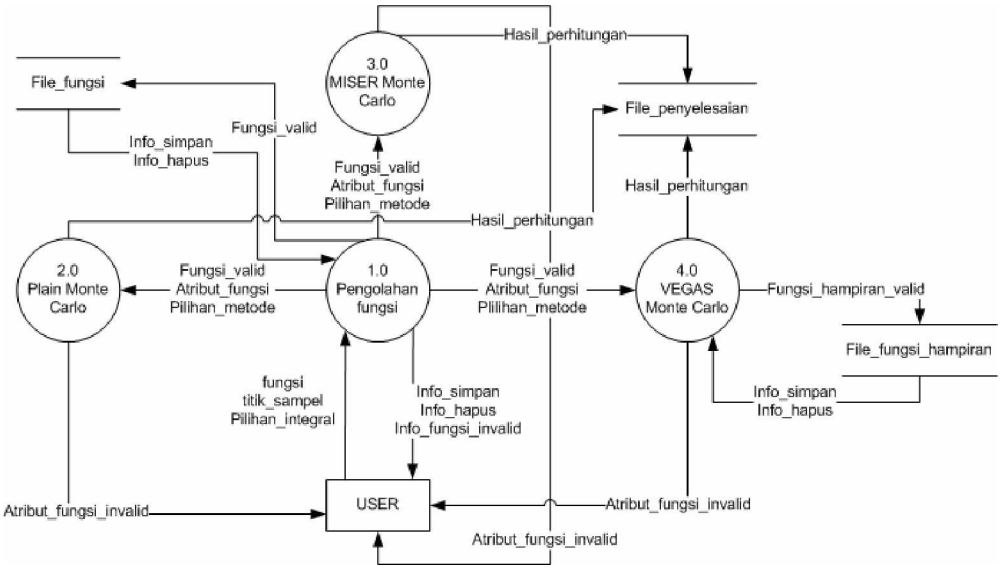
Gambar 3.7 Diagram Konteks Sistem

3.2.3 Data Flow Diagram (DFD)

Data flow diagram merupakan model dari sistem untuk menggambarkan pembagian sistem ke model yang lebih kecil. Berikut ini adalah gambar d*ata flow diagram* untuk aplikasi pencarian hasil integral multidimensi menggunakan metode *Monte Carlo*.

3.2.3.1 DFD Level 0

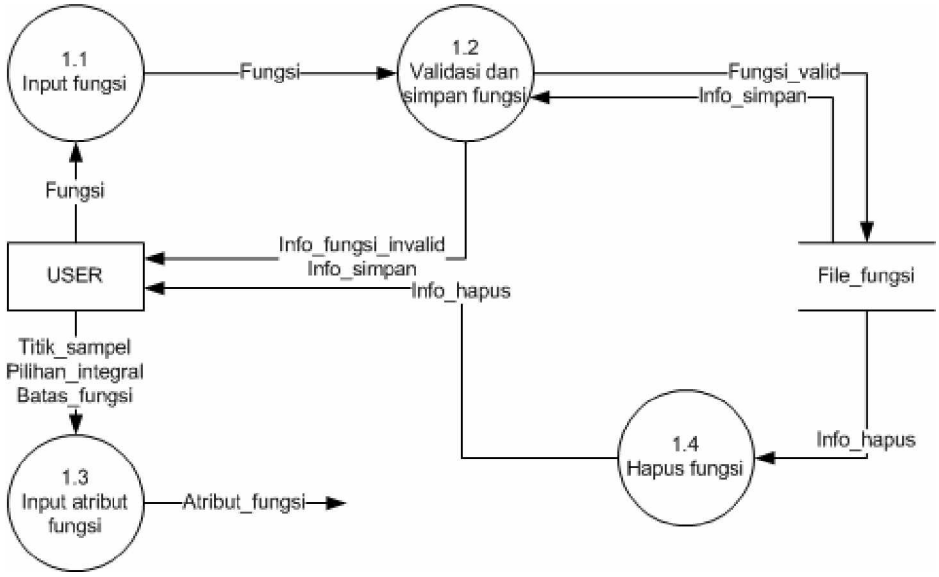
Berikut adalah DFD level 1 dari diagram konteks di atas. DFD level 1 ini menggambarkan proses yang lebih lengkap dari diagram konteks yang telah ada.



Gambar 3.8 DFD Level 0

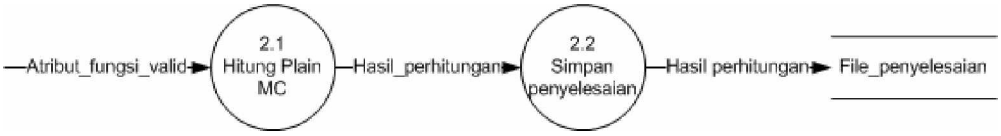
3.2.3.2 DFD Level 1

Berikut adalah DFD level 2 dari DFD level 1. DFD level 2 ini menggambarkan proses yang lebih lengkap yang terjadi pada DFD level 1 proses 1 (pengolahan fungsi).



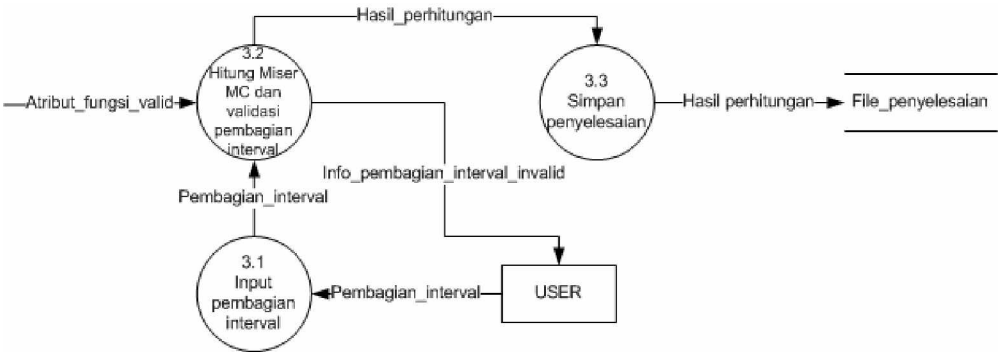
Gambar 3.9 DFD Level 1 Proses 1

DFD level 2 ini menggambarkan proses yang lebih lengkap yang terjadi pada DFD level 1 proses 2 (plain monte carlo).



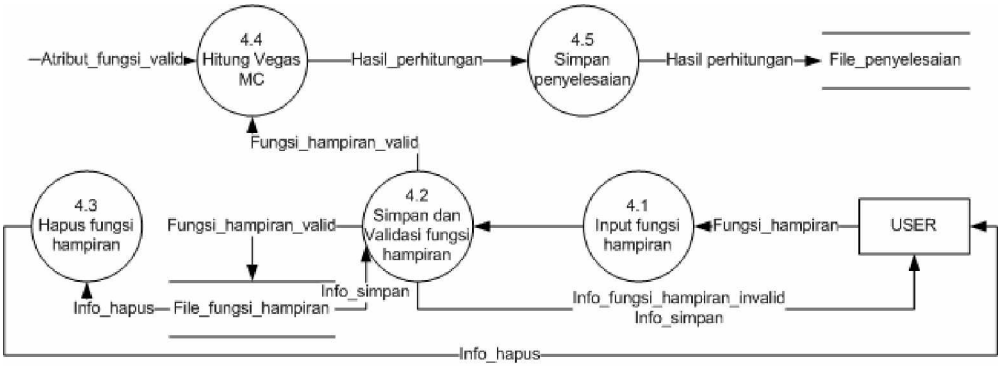
Gambar 3.10 DFD Level 1 Proses 2

DFD level 2 ini menggambarkan proses yang lebih lengkap yang terjadi pada DFD level 1 proses 2 (miser monte carlo).



Gambar 3.11 DFD Level 1 Proses 3

DFD level 2 ini menggambarkan proses yang lebih lengkap yang terjadi pada DFD level 1 proses 2 (vegas monte carlo).



Gambar 3.12 DFD Level 1 Proses 4

3.2.4 Spesifikasi Proses

Spesifikasi proses digunakan untuk menggambarkan proses model aliran yang terdapat pada DFD. Spesifikasi tersebut meliput :

Tabel 3.2 Spesifikasi Proses

No	Proses	Keterangan
1	No Proses	1.2
	Nama Proses	Validasi dan Simpan Fungsi
	Input	fungsi
	Output	fungsi_valid

	Logika Proses	Begin
		If input_fungsi valid and simpan_fungsi
		Then fungsi_valid and fungsi_tersimpan
		Else
		Notifikasi (fungsi salah)
		End

No	**Proses**	**Keterangan**
2	No Proses	1.4
	Nama Proses	Hapus fungsi
	Input	fungsi_valid
	Output	Info_hapus
	Logika Proses	Begin If Hapus_fungsi Then fungsi terhapus End
No	**Proses**	**Keterangan**
3	No Proses	2.1
	Nama Proses	Hitung Plain MC
	Input	fungsi_valid
	Output	Hasil_perhitungan
	Logika Proses	Sum = 0 For (i = 0 ; i <= n ; i = i+1) do Presum = f(xi) Sum = presum + sum Mean = sum / n End Delta = b-a Approx = Mean*Delta
No	**Proses**	**Keterangan**
4	No Proses	3.2
	Nama Proses	Hitung MISER MC
	Input	fungsi_valid, atribut_fungsi_valid,

		pembagian_interval_valid
	Output	Hasil_perhitungan

	Logika Proses	Begin
		If input_pembagian_interval valid
		Then pembagian_interval_valid
		Sum = 0
		Delta = b-a
		Disjoint = delta / 2
		Disjoint1 = disjoint - a
		Disjoint2 = b - disjoint
		For (i = 0 ; i <= n ; i = i+1) do
		Presum = f(xi)
		Sum = presum + sum
		Mean = sum / n
		End
		Approx1 = Mean*Disjoint1
		Approx2 = Mean*Disjoint2
		Approx = Approx1+Approx2
		Else
		Notifikasi (pembagian_interval salah)
		End

No	**Proses**	**Keterangan**
5	No Proses	4.2
	Nama Proses	Validasi dan simpan fungsi hampiran
	Input	fungsi_hampiran
	Output	Fungsi_hampiran_valid

	Logika Proses	Begin
		If input_fungsi_hampiran valid and simpan_fungsi
		Then fungsi_hampiran_valid and fungsi_tersimpan
		Else
		Notifikasi (fungsi salah)
		End

No	**Proses**	**Keterangan**
6	No Proses	4.4
	Nama Proses	Perhitungan VEGAS MC
	Input	fungsi_valid
	Output	Hasil_perhitungan
	Logika Proses	Sum = 0 For (i = 0 ; i <= n ; i = i+1) do Presum = f(xi) / p(xi) Sum = presum + sum Mean = sum / n End

		Delta = b-a Approx = Mean*Delta

3.2.5 Kamus Data

Kamus data atau *data dictionary* adalah katalog fakta tentang data dan kebutuhan-kebutuhan informasi dari suatu sistem informasi. Dengan menggunakan kamus data, dapat menggambarkan data yang mengalir di sistem dengan lengkap.

Kamus data dibuat berdasarkan arus data yang ada di diagram alir data (DFD). Arus data di diagram alir data sifatnya adalah global, hanya menunjukan nama arus data-datanya saja.

Kamus data dari aplikasi pencarian hasil integral adalah sebagai berikut :

Tabel 3.3 Kamus Data

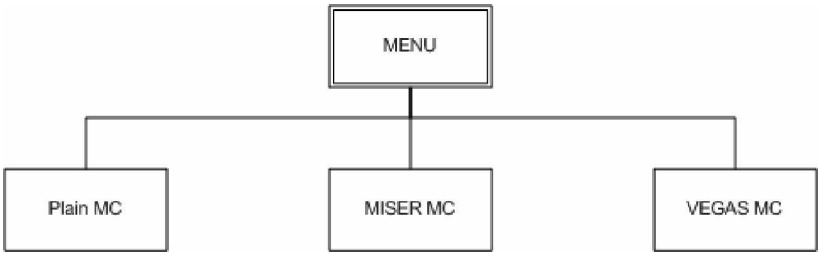
Nama	fungsi
Where used / how used	Proses 1.1 – Input fungsi (input) Proses 1.2 – Validasi dan simpan fungsi (input)
Deskripsi	Berisi data fungsi yang diinput oleh user
Struktur data	Fungsi
Fungsi	[x – y | 0 – 9 | + | - | * | / | ^]
Nama	Fungsi_valid
Where used / how used	Proses 1.2 – Validasi dan simpan fungsi (output)
Deskripsi	Berisi data fungsi yang diinput oleh user
Struktur data	Fungsi
Fungsi	[x – y | 0 – 9 | + | - | * | / | ^]
Nama	titik_sampel
Where used / how used	Proses 1.0 – Pengolahan fungsi (input)
Deskripsi	Berisi data titik sampel yang diinput oleh user
Struktur data	n
n	[0-9]

Nama	batasan_fungsi
Where used / how used	Proses 1.0 – Pengolahan fungsi (input)
Deskripsi	Berisi batasan fungsi yang diinput oleh user
Struktur data	x1+x2+y1+y2
x1 x2 y1 y2	[0-9] [0-9] [0-9] [0-9]
Nama	pembagian_interval
Where used / how used	Proses 3.1 – Input pembagian interval (input) Proses 3.2 – Validasi pembagian interval dan Hitung MISER MC (input)
Deskripsi	Berisi pembagian interval yang diinput oleh user
Struktur data	M
m	[0-9]
Nama	Pembagian_interval_valid
Where used / how used	Proses 3.2 – Validasi pembagian interval dan Hitung MISER MC (output)
Deskripsi	Berisi pembagian interval yang diinput oleh user
Struktur data	M
m	[0-9]
Nama	fungsi_hampiran
Where used / how used	Proses 4.1 – Input fungsi (input) Proses 4.2 – Validasi dan simpan fungsi (input)
Deskripsi	Berisi data fungsi hampiran yang diinput oleh user
Struktur data	Fungsi
Fungsi	[x – y | 0 – 9 | + | - | * | / | ^]
Nama	Fungsi_hampiran_valid
Where used / how used	Proses 4.2 – Validasi dan simpan fungsi (output) Proses 4.4 – Hitung Vegas MC (input)
Deskripsi	Berisi data fungsi hampiran yang diinput oleh user
Struktur data	Fungsi
Fungsi	[x – y | 0 – 9 | + | - | * | / | ^]
Nama	Info_simpan
Where used / how used	Proses 1.2 – Validasi dan Simpan fungsi (output) Proses 4.2 – Validasi dan Simpan fungsi hampiran (output)
Deskripsi	Berisi notifikasi simpan
Struktur data	

Nama	Info_hapus
Where used / how used	Proses 1.4 – Hapus fungsi (output) Proses 4.3 – Hapus fungsi (output)
Deskripsi	Berisi notifikasi hapus
Struktur data	
Nama	Hasil_perhitungan
Where used / how used	Proses 2.2 – Simpan Penyelesaian (input) Proses 3.3 – Simpan Penyelesaian (input) Proses 4.5 – Simpan Penyelesaian (input)
Deskripsi	Berisi nilai hampiran hasil perhitungan
Struktur data	Nilai_hampiran
Nilai_hampiran	[0-9]
Nama	Pilihan integral
Where used / how used	Proses 1.0 – Pengolahan fungsi (input)
Deskripsi	Berisi pilihan dimensi integral
Struktur data	
Nama	Info_fungsi_invalid
Where used / how used	Proses 1.2 – Analisis semantik (output)
Deskripsi	Berisi info data fungsi yang diinput oleh user salah
Struktur data	
Nama	Info_fungsi__hampiran_invalid
Where used / how used	Proses 4.2 – Analisis semantik (output)
Deskripsi	Berisi info data fungsi hampiran yang diinput oleh user salah
Struktur data	
Nama	Info_pembagian_interval_invalid
Where used / how used	Proses 3.2 – Pemeriksaan pembagian interval (output)
Deskripsi	Berisi info data pembagian interval yang diinput oleh user salah
Struktur data	
Nama	Atribut_fungsi_invalid
Where used / how used	Proses 2.0 – Plain MC (output) Proses 3.0 – Plain MC (output) Proses 4.0 – Plain MC (output)
Deskripsi	Berisi info data atribut fungsi yang diinput oleh user salah
Struktur data	titik_sampel_invalid, batas_fungsi_invalid

3.3 Perancangan Menu Aplikasi

Perancangan menu aplikasi digunakan untuk menjelaskan menu, submenu yang terdapat di dalam sistem, agar pengguna dapat lebih mudah dalam menggunakanya.



Gambar 3.13 Bagan Menu

Menu ini dapat diakses oleh user / pengguna. Menu ini berguna untuk mengelola data sistem. Masukkan dilakukan di dalam menu, namun ada beberapa masukan tambahan bagi tiap metode karena berbeda metode berbeda pula masukannya. Menu ini memiliki fasilitas :

1. Plain MC

   Di dalam submenu ini user tidak perlu memasukkan masukan tambahan, karena semua input yang diperlukan sudah diinputkan di bagian menu.

2. MISER MC

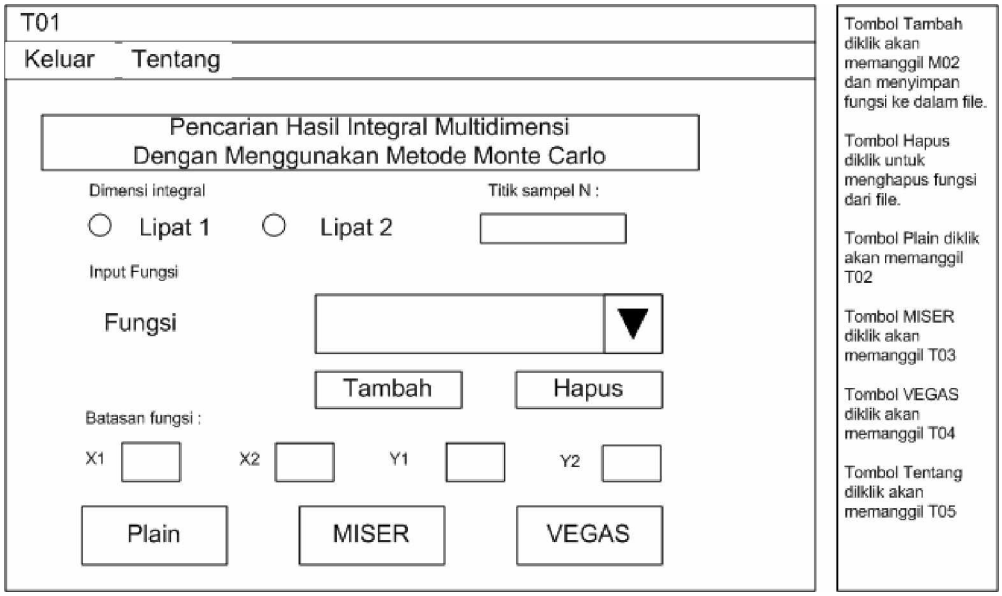
   Di dalam submenu ini user perlu memasukkan masukan tambahan berupa nilai pembagi interval fungsi.

3. VEGAS MC

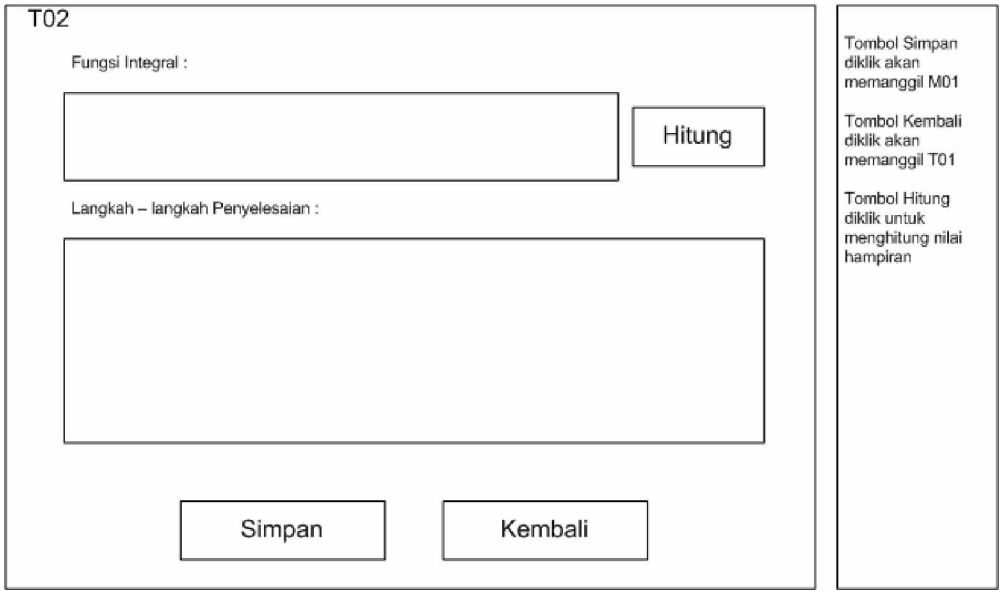
   Di dalam submenu ini user perlu memasukkan masukan tambahan berupa fungsi hampiran, karena metode Vegas *Monte Carlo* memerlukan masukan fungsi hampiran dalam perhitungannya.

3.4 Perancangan Antarmuka

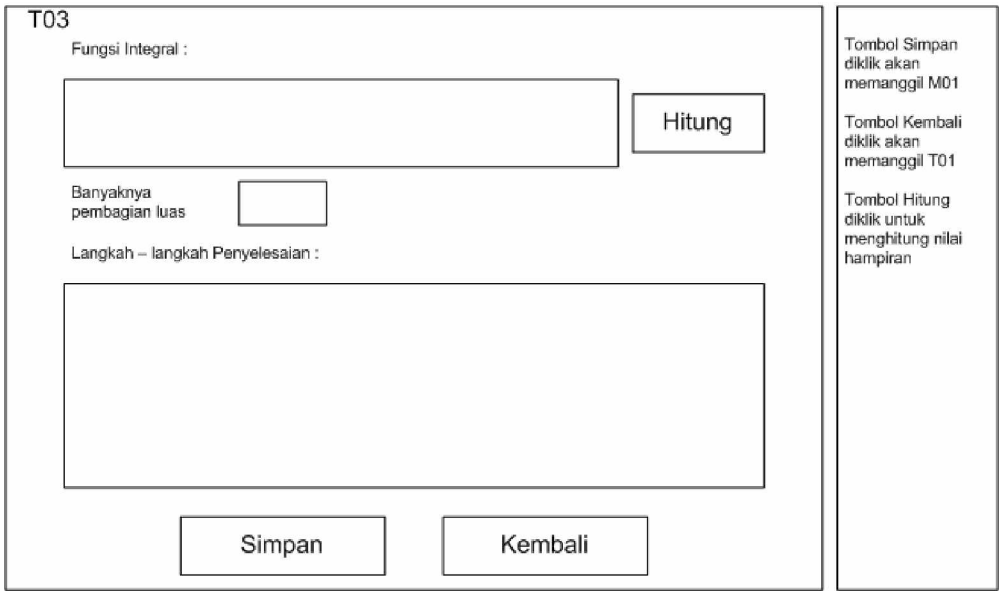
Antarmuka pemakai merupakan media komunikasi antara pemakai dengan sistem komputer. Hasil perancangan antarmuka pemakai akan memudahkan proses implementasi nanti. Adapun antarmuka pemakai perangkat lunak yang akan dirancang ditampilkan sebagai berikut :



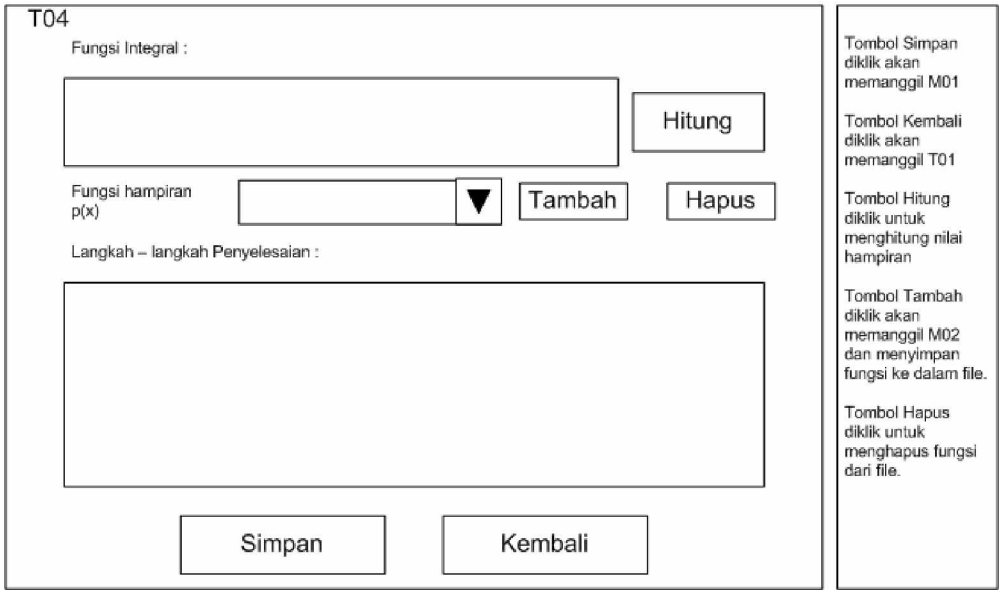
Gambar 3.14 Rancangan Antarmuka Form Utama



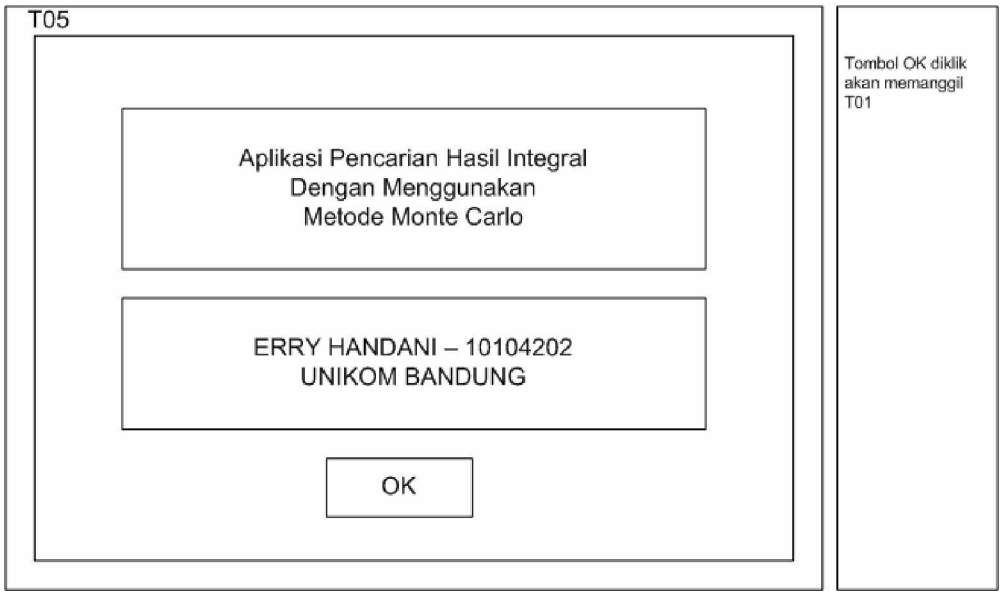
Gambar 3.15 Rancangan Antarmuka Form Plain MC



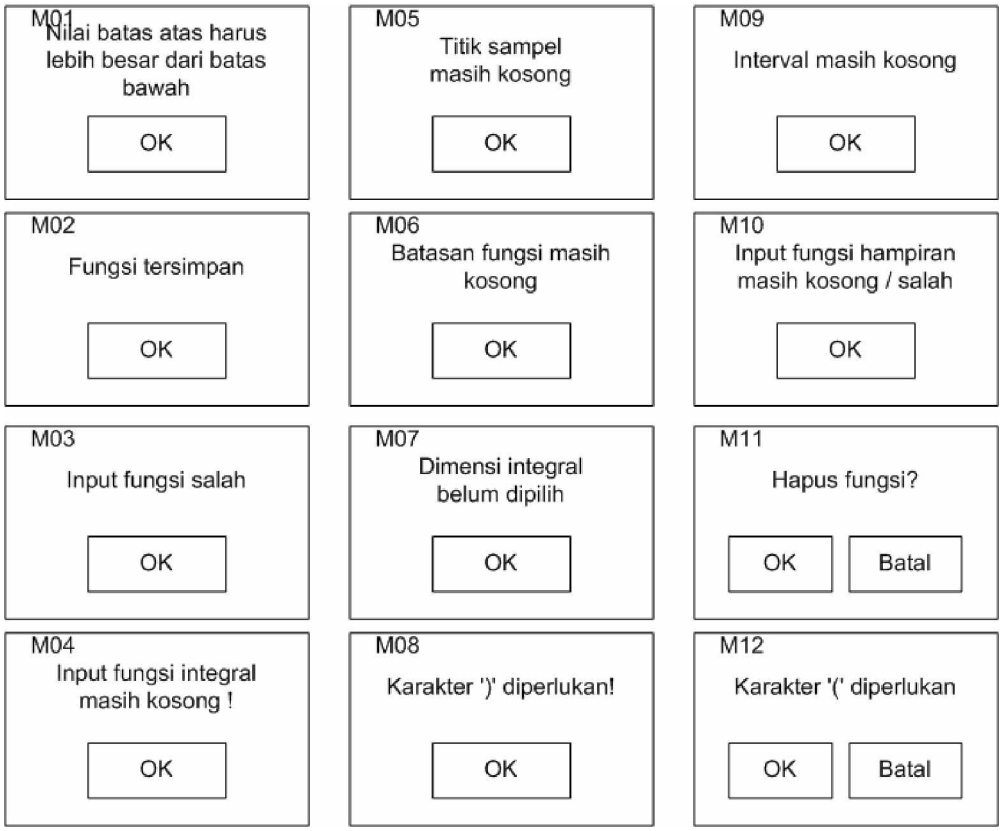
Gambar 3.16 Rancangan Antarmuka Form MISER MC



Gambar 3.17 Rancangan Antarmuka Form VEGAS MC

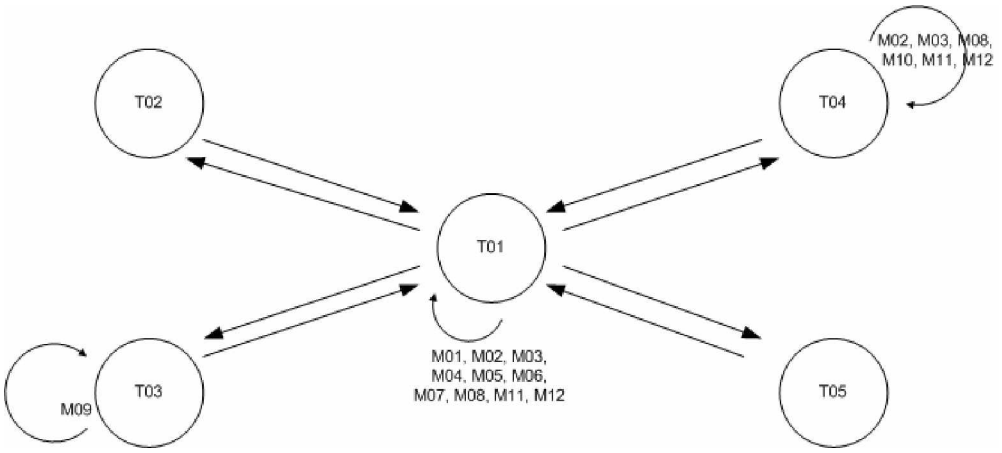


Gambar 3.18 Rancangan Antarmuka Form Tentang



Gambar 3.19 Rancangan Antarmuka Pesan

3.5 Jaringan Semantik



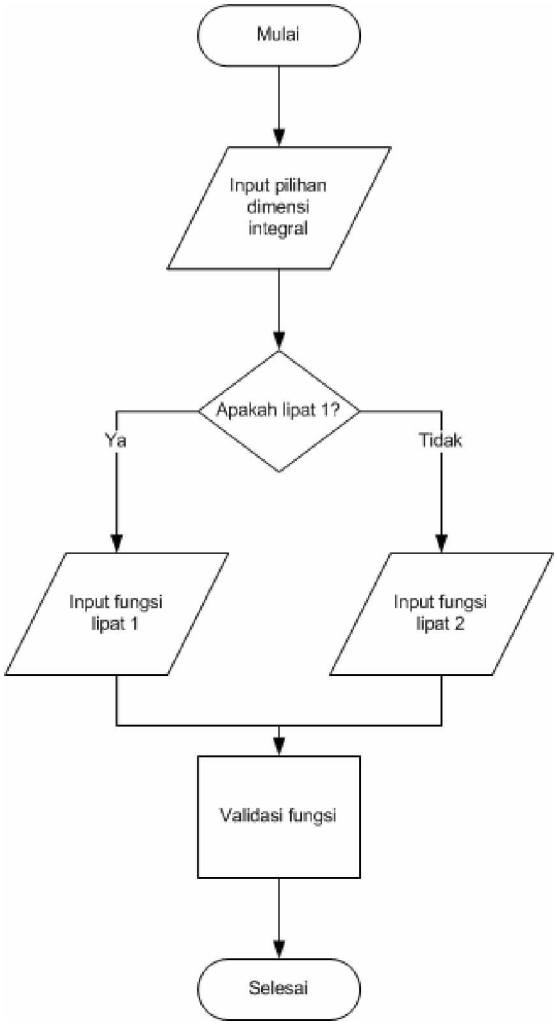
Gambar 3.20 Jaringan Semantik Sistem

3.6 Perancangan Prosedural

Di bagian ini dibahas mengenai perancangan prosedural, yaitu berupa tahapan-tahapan yang akan dijalani oleh user.

3.6.1 Prosedur Pilih Dimensi Integral

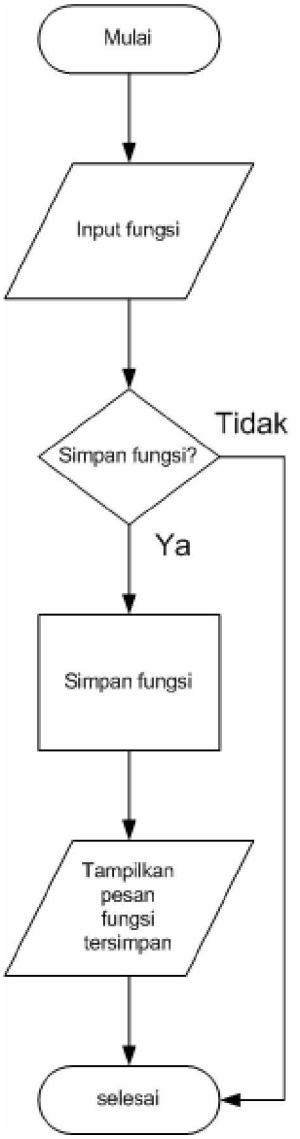
Prosedur pilih dimensi integral adalah prosedur yang akan dilalui oleh user pertama kali di dalam sistem.



Gambar 3.21 Diagram Alir Pilih Dimensi

3.6.2 Prosedur Tambah Fungsi

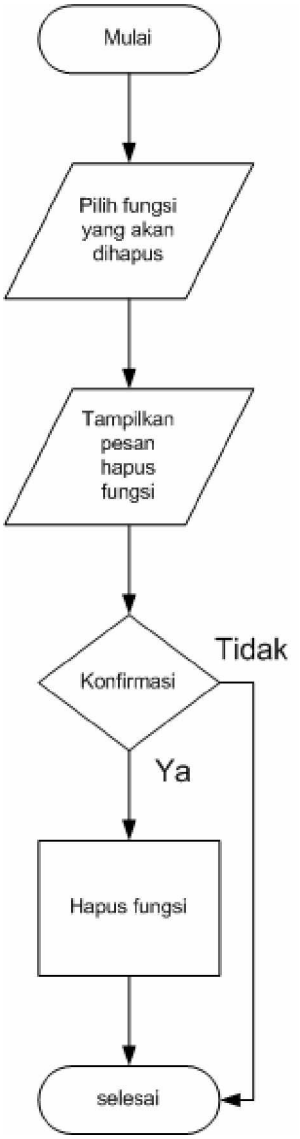
Prosedur tambah fungsi adalah prosedur yang dilakukan ketika menambah data fungsi ke dalam sistem.



Gambar 3.22 Diagram Alir Tambah Fungsi

3.6.3 Prosedur Hapus Fungsi

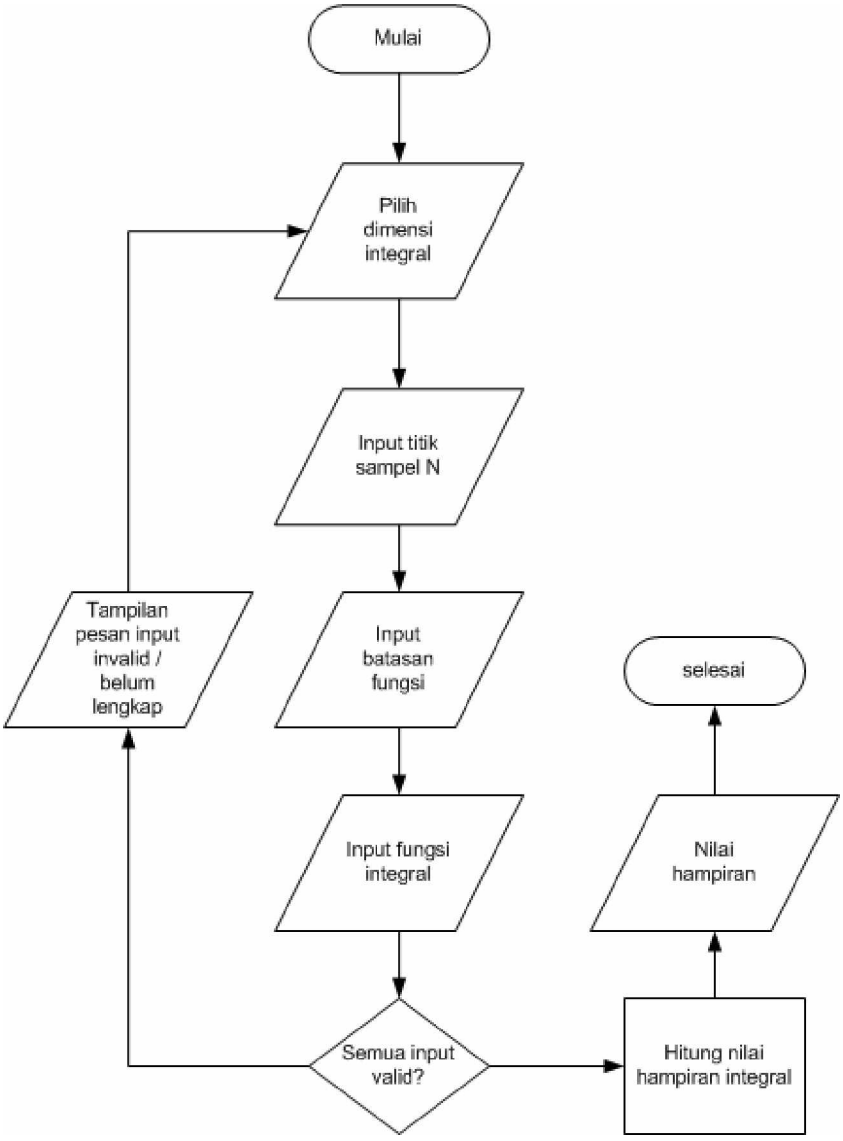
Prosedur hapus fungsi adalah prosedur yang dilakukan ketika menghapus data fungsi dari sistem.



Gambar 3.23 Diagram Alir Hapus Fungsi

3.6.4 Prosedur Plain *Monte Carlo*

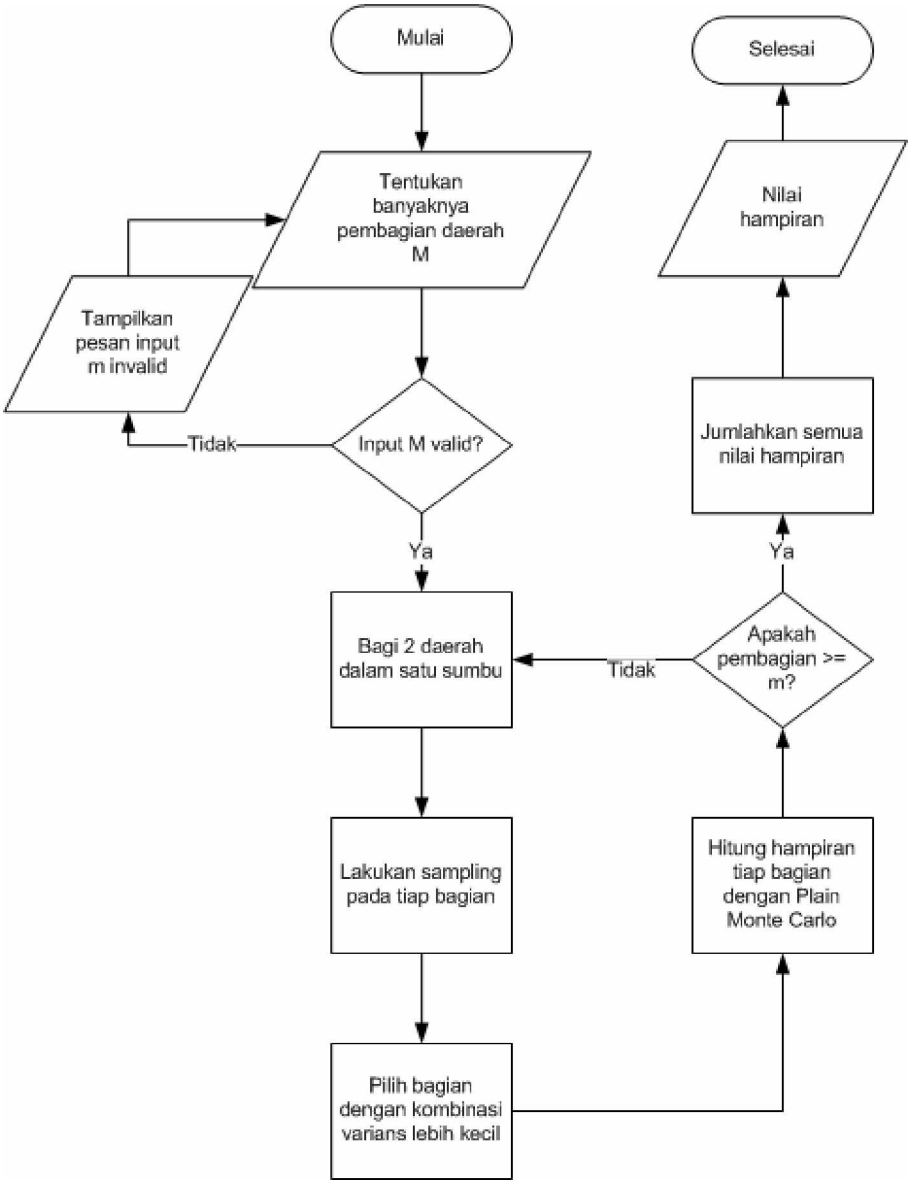
Prosedur plain Monte Carlo adalah prosedur yang terjadi di dalam sistem ketika user memilih metode ini sebagai metode perhitungan.



Gambar 3.24 Diagram Alir Plain *Monte Carlo*

3.6.5 Prosedur MISER *Monte Carlo*

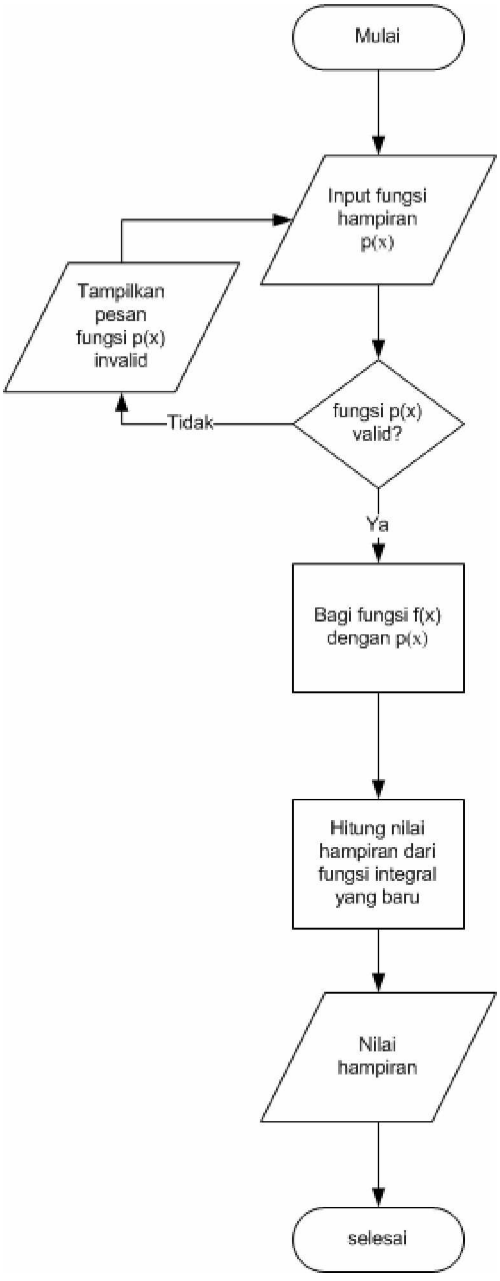
Prosedur MISER Monte Carlo adalah prosedur yang terjadi di dalam sistem ketika user memilih metode ini sebagai metode perhitungan.



Gambar 3.25 Diagram Alir MISER *Monte Carlo*

3.6.6 Prosedur VEGAS *Monte Carlo*

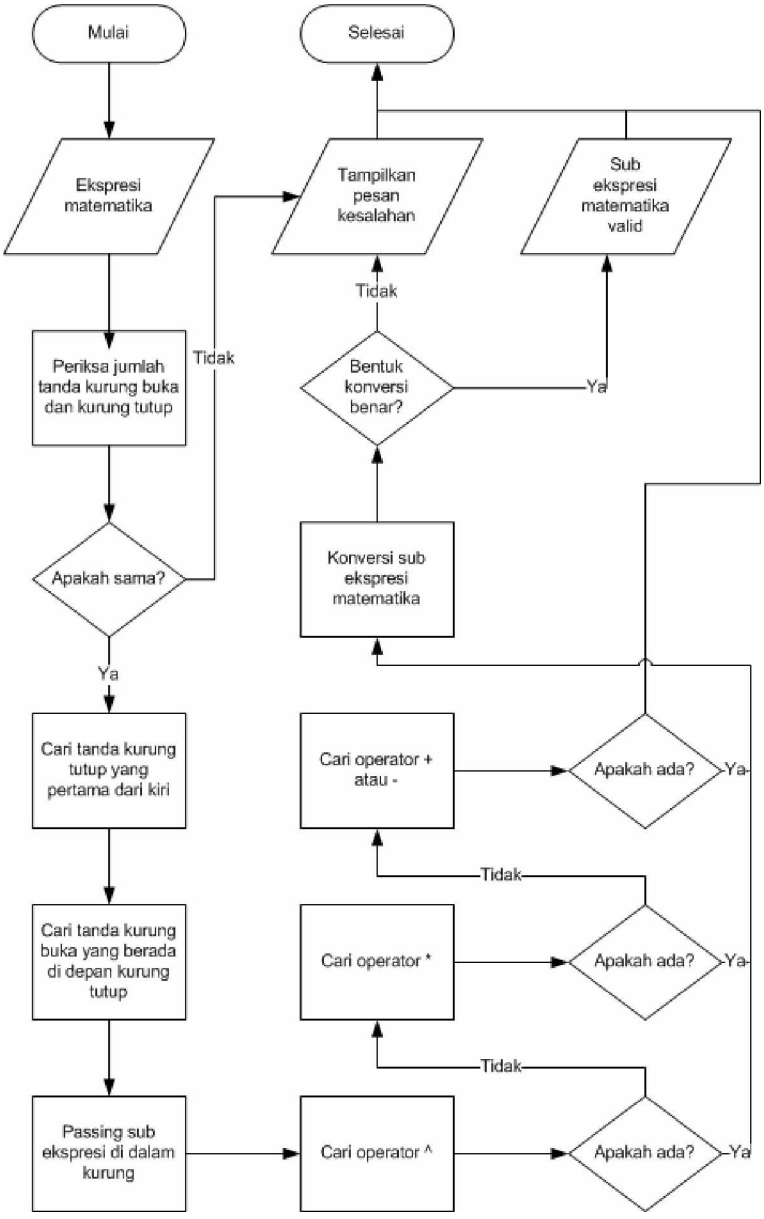
Prosedur VEGAS Monte Carlo adalah prosedur yang terjadi di dalam sistem ketika user memilih metode ini sebagai metode perhitungan.



Gambar 3.26 Diagram Alir VEGAS *Monte Carlo*

3.6.7 Prosedur Infix Algoritma *Parsing Left Most Derivation*

Prosedur parsing adalah prosedur yang dilakukan untuk memriksa validasi strukstur ekspresi matematika dan memecah ekspresi matematika menjadi sub-sub ekspresi.



Gambar 3.27 Diagram Alir *Parsing Left Most Derivation*