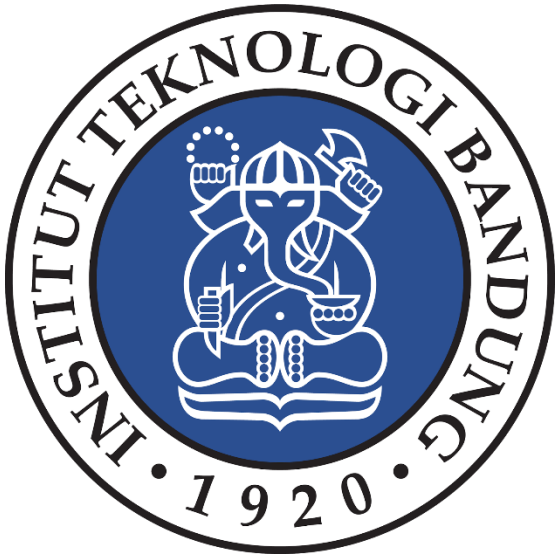
Tugas Besar EL2008 Pemecahan Masalah dengan C

Minimisasi Logika dengan Metode Tabular (Quine-McCluskey)



Oleh :

1. Jefferson Grizzlie (13220013)
2. Muhammad Mikhail Ghrilman (13220021)
3. Bostang Palaguna (13220055)
4. Muhammad Daffa Rasyid (13220059)

Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Jl. Ganesha No. 10 Bandung, Indonesia, 40132
2022

Daftar Isi

Abstrak

Fungsi boolean atau fungsi logika adalah fungsi matematis yang argumen dan hasilnya hanya memiliki dua kemungkinan nilai yaitu nol atau satu (benar atau salah). Dalam sistem digital, fungsi boolean diimplementasikan menggunakan gerbang logika. Variabel fungsi boolean dan komplemennya disebut dengan literal. Minterm adalah perkalian dari semua literal.[1] Sum of product (SOP) adalah bentuk ekspresi logika yang merupakan penjumlahan dari berbagai minterm. Fungsi logika yang dinyatakan sebagai penjumlahan dari minterm dikatakan masih dalam bentuk kanonikal.[2] Bentuk ini biasanya membutuhkan jumlah gerbang logika yang relatif banyak dan bisa diminimisasi. Minimisasi logika dapat dilakukan dengan berbagai cara seperti manipulasi aljabar boolean, karnaugh map, dan tabular (Quiene Mc-Cluskey). Tetapi pada tugas besar kali ini, kami menggunakan metode tabular karena metode ini memerlukan banyak iterasi/pengulangan sehingga sangat cocok untuk dikerjakan oleh komputer. Terdapat suatu kondisi dalam fungsi logika yang disebut dengan don't care. Don't care adalah input minterm yang outputnya tidak penting. Artinya ketika mendesain sistem digital, kita bisa mengasumsikan input don't care tidak pernah terjadi dan outputnya bisa dipilih secara sembarang. Don't care seringkali memberikan hasil minimisasi logika yang lebih sederhana. Dalam tugas besar kali ini, kami akan mengimplementasikan suatu program dalam bahasa pemrograman C yang dapat menerima input fungsi logika dalam bentuk SOP dan menampilkan ke layar hasil minimisasinya menggunakan metode tabular.

Kata kunci : minterm, sum of product, don't care, metode tabular (Quiene mc-cluskey)

Analisis Permasalahan

Pemilihan Metode Minimisasi Logika

Seperti yang disampaikan pada abstrak, terdapat beberapa metode yang bisa digunakan untuk melakukan minimisasi logika. Ketika mengambil mata kuliah Sistem Digital (EL2002), pernah disampaikan bahwa metode karnaugh map adalah metode yang sangat cocok untuk manusia tetapi sulit diimplementasikan pada mesin. Di sisi lain, metode tabular (Quiene mc-cluskey) adalah metode yang menjemukan bagian manusia karena tahapan iterasinya yang banyak dan lebih cocok dilakukan oleh mesin. Setelah melakukan riset terkait implementasi program minimisasi logika juga banyak diimplementasikan menggunakan metode tabular.

Oleh karena itu, kami memutuskan untuk mengembangkan program yang dapat meminimisasi fungsi logika menggunakan metode tabular. Dalam pelaksanaannya, kami tidak mengembangkan kode dari nol (*re-inventing the wheel*), tetapi kami mengembangkan fitur serta meningkatkan reliabilitas program dari referensi utama[3] yang kami gunakan.

Dasar Teori

Pada bagian ini akan disampaikan penjelasan tentang minimisasi logika menggunakan metode tabular.

Misalkan kita memilki sebuah fungsi logika untuk di-minimisasi sebagai berikut:

$$f(A,B,C,D) = \sum m(1,3,7,12,13,14,15)$$

atau dalam bentuk kanonikal:

Minterm ke-	Representasi biner	Term dalam SOP
1	0001	abcD
3	0011	abCD
7	0111	Abcd
12	1100	ABcd
13	1101	ABcD
14	1110	ABCd
15	1111	ABCD

$$f(A,B,C,D) = abcD + abCD + Abcd + ABcd + ABCd + ABCD$$

catatan : variabel dengan huruf kecil menunjukkan komplemen dari variabel huruf besar. Sedangkan variabel huruf besar bernilai logic 1.

Tahapan metode tabular secara garis besar dibagi menjadi dua, yaitu mencari prime implicant kemudian menemukan minimum cover.

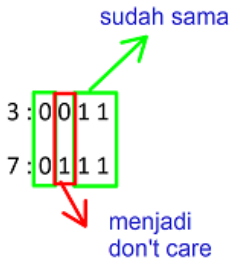
Langkah Pertama : Membuat L1 dengan mengurutkan term berdasarkan banyak angka 1
Kita akan kelompokkan minterm berdasarkan banyaknya angka 1 yang mereka punya dalam representasi binernya. Kita akan bagi menjadi grup yang memiliki satu angka 1, dua angka 1, sampai dengan empat angka 1.

$$L_1$$

Banyak angka 1	Minterm ke-	Representasi biner
1	1	0001
2	3	0011
2	12	1100
3	7	0111
3	13	1101
3	14	1110
4	15	1111

Kita mengelompokkan term berdasarkan banyaknya angka 1 yang mereka punya sehingga ketika mencari pasangan minterm yang hanya berbeda satu term, kita cukup membandingkan minterm-minterm pada grup yang bersebelahan. Misalkan minterm-minterm pada grup dengan banyak angka 1 sebanyak 1 cukup dibandingkan dengan minterm pada grup dengan banyak angka 1 sebanyak 2. Tidak perlu membandingkannya dengan minterm pada grup dengan banyak angka 1 sebanyak tiga ataupun empat.

Langkah kedua : kelompokkan minterm pada L1 yang hanya berbeda satu term
Contoh minterm yang dipasangkan (yang hanya berbeda satu term) adalah minterm ke-1 dan minterm ke-3.



Dengan menemukan semua pasangan minterm pada L1 (ingat kita hanya membandingkan minterm-minterm pada grup yang bersebelahan saja), kita akan peroleh L2 sebagai berikut:

L_2

Pasangan minterm	Representasi biner
1,3	00-1
3,7	0-11
12,13	110-
12,14	11-0
7,15	-111
13,15	11-1
14,15	111-

catatan:
1. - menandakan don't care
2. minterm yang bisa dipasangkan grup ke-n dan grup ke-$n+1$ pada L1 akan bergabung menjadi pasangan minterm dengan grup ke-n pada L2

perhatikan bahwa semua minterm pada L1 telah ter-*cover* oleh pasangan minterm pada L2, sehingga kita tidak lagi membutuhkan L1.

Langkah ketiga : kelompokkan pasangan minterm pada L2 yang hanya berbeda satu term
Kita lakukan cara yang sama untuk membentuk L3 dari L2 seperti membentuk L2 dari L1. Kita pasangkan 'pasangan minterm' pada L2 yang hanya berbeda satu term. Kita peroleh L3 sebagai berikut:

L_3

Pasangan ‘pasangan minterm’	Representasi biner
12,13 ; 14,15	11--
12 14 ; 13, 15	11--

perhatikan bahwa (12,13;14,15) memiliki representasi biner yang sama dengan (12,14;13,15) sehingga kita cukup mengambil salah satu. Misalkan kita ambil (12,13;14,15) sebagai prime implicant.

Langkah keempat : Bentuk prime implicant dari L2 dan L3
(12,13,14,15) pada L3 telah men-*cover* pasangan minterm (12,13), (12,14), (13,15), dan (14,15) pada L2. pasangan minterm yang belum tercover pada L2 adalah (1,3), (3,7), dan (7,15). Maka dari L2 dan L3, kita punya prime implicant P1, P2, P3, dan P4 sebagai berikut:

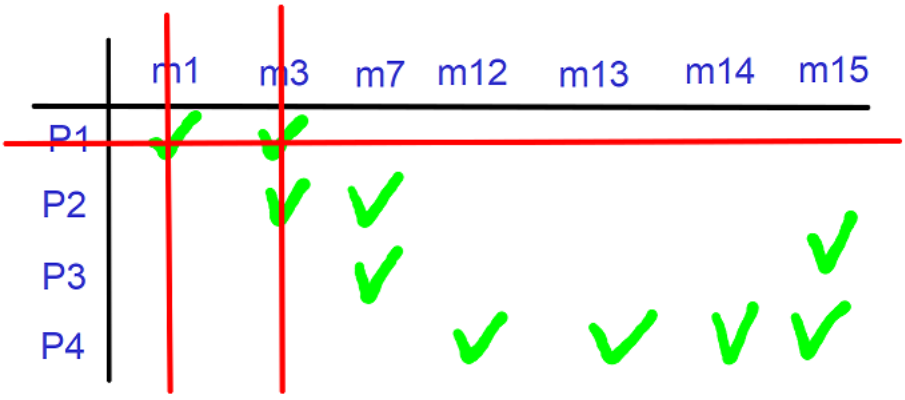
Pasangan minterm	
1,3	→ P1
3,7	→ P2
12,13	✓
12,14	✓
7,15	→ P3
13,15	✓
14,15	✓

P4 : (12,13,14,15)

Langkah kelima : Cari minium cover dari prime implicant yang telah dibentuk

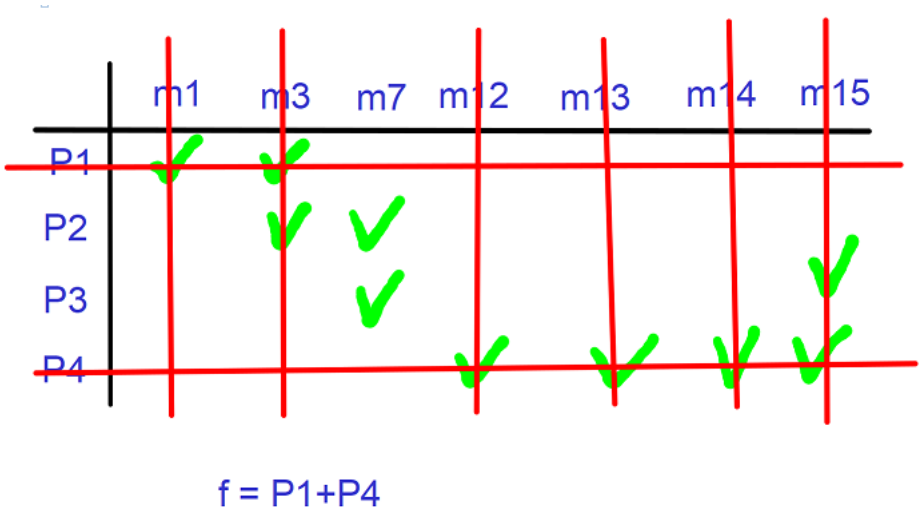
Kita buat sebuah tabel dengan baris (vertikal) berisi prime implicant P1 sampai dengan P4 lalu kolom (horizontal) berisi minterm ke-1 sampai dengan minterm ke 15 (sesuai fungsi logika).

sekarang kita harus memilih prime implicant se-sedikit mungkin sehingga semua minterm tercover. Perhatikan bahwa agar m1 tercover, mau tidak mau P1 harus dipilih. Selanjutnya, semua yang tercover oleh P1 bisa kita hiraukan.

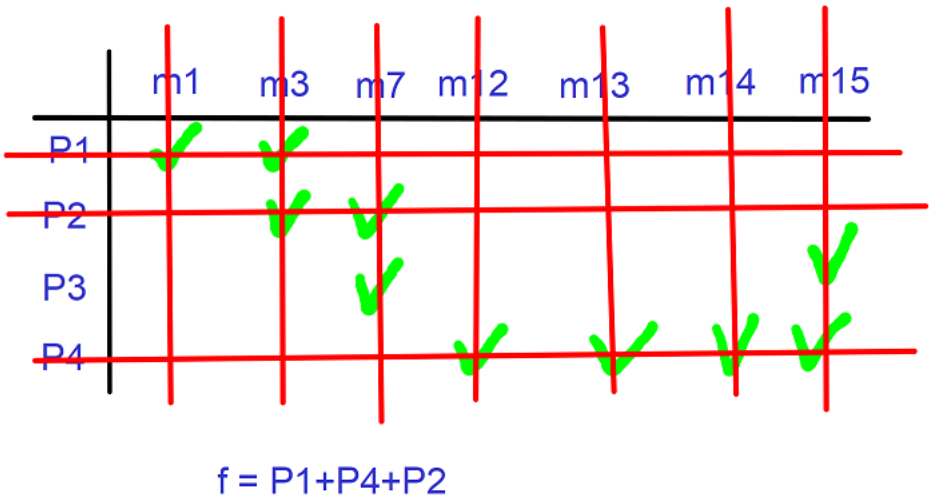


f = P1

Selanjutnya, agar m12 dan m13 tercover, kita harus memilih P4. Maka kita peroleh tabel minimum cover yang lebih sederhana sebagai berikut:



terakhir, untuk men-*cover* m7, kita bisa bebas memilih P2 ataupun P3. Kedua pilihan sama baiknya. Misalkan untuk sekarang kita pilih P2. Maka semua minterm telah tercover dan kita peroleh fungsi logika yang telah di-minimisasi sebagai berikut:



$$f(A,B,C,D) = P1 + P4 + P2 = 00\text{-}1 + 0\text{-}11 + 11\text{--} = \boxed{abD + aCD + AB}$$

Don’t Care

don't care adalah term yang boleh di-cover ataupun tidak. Ia membantu kita dalam memberikan ekspresi logika hasil minimisasi yang lebih sederhana. Sebagai contoh permasalahan, misalkan kita diminta untuk meminimisasi fungsi logika:

$$f(A,B,C,D) = \sum m(7,13,14,15) + \sum d(5,11)$$

sama seperti sebelumnya, kita bentuk L1, L2, dan L3 dengan cara yang sama.

L_1

Banyak angka 1	Minterm ke-	Representasi biner
2	5	0101
3	7	0111
3	11	1011
3	13	1101
3	14	1110
4	15	1111

L_2

Pasangan minterm	Representasi biner
5,7	01-1
5,13	-101
7,15	-111
11,15	1-11
13,15	11-1
14,15	111-

L_3

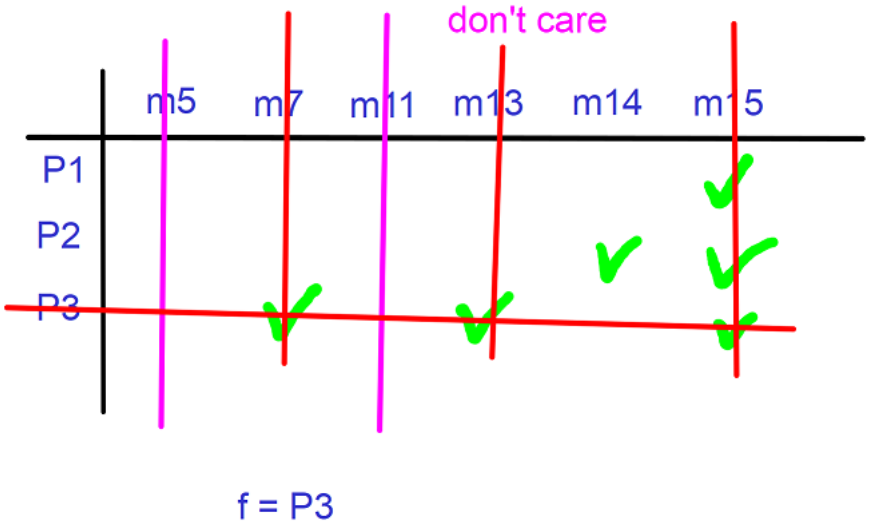
Pasangan ‘pasangan minterm’	Representasi biner
5,7 ; 13,15	-1-1
5,13 ; 7,15	-1-1

semua minterm pada L1 telah ter-cover oleh pasangan minterm pada L2 sehingga L1 tidak akan kita gunakan. Pasangan 'pasangan minterm' pada L3 (5,7,13,15) telah mencover pasangan minterm (5,7), (5,13), (7,15), dan (13,15) pada L2. Sedangkan yang belum tercover adalah pasangan minterm (11,15) dan (14,15).

Maka kita punya prime implicant : P1 (11,15), P2 (14,15), dan P3 (5,7,13,15).

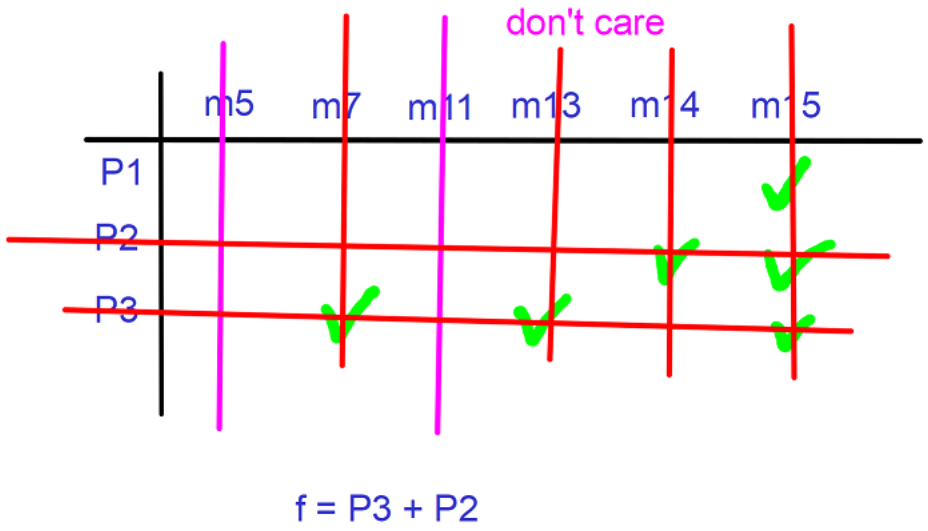
Sekarang yang membedakan don't care dengan minterm biasa adalah pada saat pencarian minimum cover, don't care tidak perlu masuk sebagai kolom (horizontal). Berikut adalah tabel minimum covernya:

P3 harus dipilih agar m7 dan m13 tercover.



f = P3

selanjutnya agar m14 tercover, kita pilih P2. Maka kita peroleh fungsi logika yang telah diminimisasi:



$$f(A,B,C,D) = P3 + P2 = \text{-1-1 + 111-} = \boxed{BD + ABC}$$

Misalkan kita memilki sebuah fungsi logika:

$$f(A,B,C,D) = \sum M(1,2,5,6,9,10,14)$$

Bentuk kanonikal dari fungsi logika di atas:

Maxterm ke-	Representasi biner	Term dalam POS
1	0001	A+B+C+d
2	0010	A+B+c+D
5	0101	A+b+C+d
6	0110	A+b+c+D
9	1001	a+B+C+d
10	1010	a+B+c+D
14	1110	a+b+c+D

$$f(A,B,C,D) = (A+B+C+d)(A+B+c+D)(A+b+C+d)(A+b+c+D)(a+B+C+d)(a+B+c+D)(a+b+c+D)$$

Dimana variabel dengan huruf kecil menunjukkan komplemen dari variabel huruf besar. Sedangkan variabel huruf besar bernilai logic 0.

Langkah-langkah minimisasi:

L_1

Banyak angka 1	Maxterm ke-	Representasi biner
1	1	0001
1	2	0010
2	5	0101
2	6	0110

2	9	1001
2	10	1010
3	14	1110

L_2

Pasangan maxterm	Representasi biner
1,5	0-01
1,9	-001
2,6	0-10
2,10	-010
6,14	-110
10,14	1-10

L_3

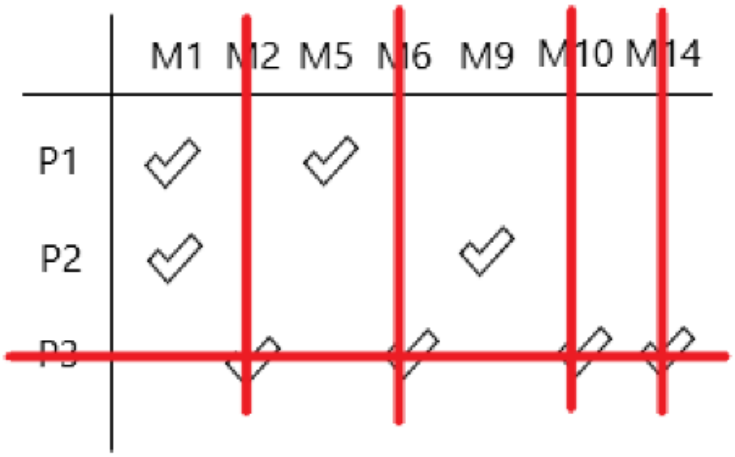
Pasangan ‘pasangan maxterm’	Representasi biner
2,6; 10,14	--10
2,10 ; 6,14	--10

Semua maxterm pada L1 sudah tercover oleh pasangan maxterm pada L2, sedangkan pasangan maxterm (2,6), (2,10), (6,14), dan (10,14) telah dicover oleh pasangan ‘pasangan maxterm’ (2,6,10,14) pada L3.

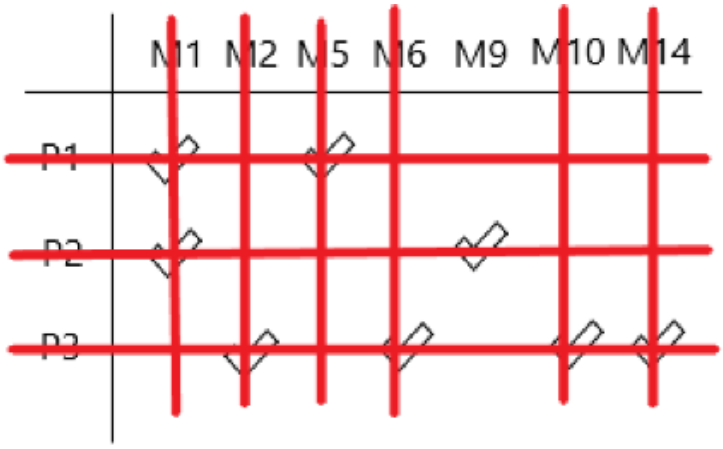
Maka prime implicant dari hasil minimisasi di atas: P1 (1,5); P2 (1,9); P3 (2,6,10,14)

Berdasarkan prime implicant yang didapat, maka dicari minimum cover:

P3 dipilih agar M2, M6, M10, dan M14 tercover



P2 dan P1 dipilih agar M1, M5, dan M9 tercover



Maka fungsi logika yang telah diminimisasi:

$f(A,B,C,D) = P3.P2.P1 = (\text{--}10)(\text{-}001)(0\text{-}01) = (c+D)(B+C+d)(A+C+d)$

Penjelasan Program

Struktur Data yang digunakan

Untuk menyimpan informasi suatu minterm, kita membuat tipe data bentukan berupa `Node` yang akan membentuk suatu linked-list.

```
struct Node                    // menyimpan informasi tentang minterm seperti
banyak pasangan dan banyak pasangan yang terbentuk
{
    struct Node* next;      // pointer ke node selanjutnya pada linked-list
    int hasPaired;          // kondisi pemasangan
    int numberOfOnes;       // banyaknya '1' pada minterm
    struct vector paired;   // struct vector untuk menyatakan minterm
berpasangan lain
    int group;              // menyatakan nomr grup sesuai jumlah angka satu
pada minterm
    int binary[ukuranBit]; // menyimpan nilai biner dari minterm
```

```
    int numberOfPairs;    // berapa banyak pasangan yang telah terbentuk
(contoh: 4 menyatakan 2 double atau 1 quad)
};
```

Kita juga memiliki tipe data bentukan `vector` yang berisi array untuk menyimpan informasi minterm mana yang telah berpasangan (nilai = 1) atau belum (nilai = 0).

```
struct vector            // menyimpan list semua minterm yang telah
dikelompokkan
{
    int paired[limit];
};
```

Untuk menyimpan informasi tentang tabel prime implicant, kita membentuk suatu tipe data bentukan `implicantsTable` dan sebuah variabel bertipe data tersebut bernama `Tabel`.

```
struct implicantsTable // Tabel prime implicant
{
    int arr[limit][ukuranBit]; // array referensi sebelum mencetak fungsi
logika yang telah diminimalisasi ke layar
    int brr[limit][limit]; // array untuk mencari minimum cover
    int top; // banyak implikan pada tabel implikan
    int mintermCounter[limit]; // banyak minterm pada tabel implikan
} Tabel;
```

Daftar Variabel dan Fungsi/Prosedur

Daftar Variabel

Berikut adalah variabel yang digunakan di dalam prosedur `logicMinimization()`:

Variabel	Deskripsi
`n_iteration : integer`	pencacah berapa banyak iterasi yang sudah terjadi
`minterms : array[0..limit-1] of integer`	array yang menyimpan minterm berapa saja yang terdapat pada fungsi logika
`dontCares : array[0..limit-1] of integer`	array yang menyimpan term berapa saja yang menjadi don't care pada fungsi logika
`modeInput : character`	mode input yang berasal dari input user : input secara manual atau melalui file eksternal
`filename : string`	string nama file eksternal sebagai input
`Tabel : implicantsTable`	Variabel dengan tipe data bentukan implicantsTable yang berfungsi sebagai tabel prime implicant pada metode minimisasi logika dengan tabular
`head : pointer to node`	minterm (node) pertama yang disimpan dalam linked list
`head2 : pointer to node`	minterm (node) pertama yang disimpan dalam linked list untuk merepresentasikan pasangan minterm
`maxGroup : integer`	menyatakan banyaknya minterm paling besar dalam satu grup pada iterasi sebelumnya
`newMaxGroup : integer`	menyatakan banyaknya minterm paling besar dalam satu grup pada iterasi sekarang

`kondisiDontCare : boolean`	kondisi yang menyatakan bahwa terdapat don't care dalam fungsi logika(jika bernilai true)
`banyakDontCare : integer`	banyak don't care pada fungsi logika.
`banyakMinterm : integer`	banyak minterm pada fungsi logika.

Daftar Prosedur

Berikut adalah prosedur yang berhubungan dengan algoritma tabular (digunakan pada prosedur `logicMinimization()`):

Fungsi / Prosedur	Deskripsi
`int logicMinimization(char modeInput, int counter)`	Algoritma utama untuk melakukan algoritma I.S modeInput operasi telah terdefinisi F.S dilakukan minimisasi logika dengan mode input operasi tertentu modeInput : m --> user memasukkan input secara manual modeInput : f --> menerima input dari file eksternal
`void add(int n)`	membuat linked-list untuk menyimpan minterm I.S. nilai minterm yang ingin ditambahkan ke linked list terdefinisi F.S. sebuah node dibuat dan ditambahkan ke linked-list
`node* buatNode(int n)`	membuat node untuk menyimpan data minterm
`void pair(int* n_iteration)`	melakukan pemasangan node sekaligus mencetaknya dalam tabel L1, L2, dan L3 I.S. linked list telah terdefinisi F.S. L1, L2, dan L3 tercetak (setelah pemanggilan rekursif terakhir selesai)
`void display()`	menampilkan minterm dan pasangannya dan nilai biner pada setiap pass (L1, L2, L3, dst.) I.S. node-node pada linked list siap untuk dicetak F.S. sebuah baris pada L1, L2, dan L3 telah ditampilkan
`void tampilkanTabel()`	menampilkan prime implicant table I.S. Tabel bertipe data implicant table telah terisi F.S. elemen pada Tabel ditampilkan dengan notasi minterm
`node* createNodePair(node* p,node* q)`	membuat node baru menggunakan Node yang sudah ada I.S. node p dan q terdefinsi F.S. sebuah node r telah dibuat dari node p dan q
`void binaryFill(node *p,node *q,node *r)`	mengisi nilai biner pada r dengan p dan q. Jika kedua bit sama, maka akan dipertahankan. Jika berbeda, akan bernilai -1 I.S. p,q, dan r terdefinisi F.S. nilai biner r telah diperbaharui
`void inisiasiTabel()`	menginisiasi seluruh elemen table menjadi -1 I.S. sebuah variabel bertipe data implicantsTable bernama Tabel telah terdefinisi F.S. seluruh elemen Tabel di-inisiasi dengan nilai -1
`int ifPairingPossible(node *a,node *b)`	memeriksa jika apakah hanya ada perbedaan satu bit antara p dan q (sehingga mungkin dipasangkan)
`int cekDontCare(int i)`	memeriksa apakah suatu minterm don't care atau bukan nilai balikan : 0 : bukan don't care ; 1 : don't care

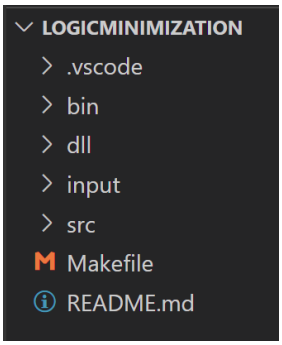
`void addPair(node *p,node *q)`	membuat linked list untuk menyimpan pasangan minterm I.S. dua buah node yang ingin dipasangkan terdefinisi F.S. sepasang node ditambahkan ke linked list
`void tambahKeTabel()`	membuat tabel untuk mencari minimum cover I.S. prime implicant telah ditemukan F.S. tabel untuk mencari minimum cover telah dibuat
`void analisisTabel()`	menganalisis tabel minimum cover yang telah dibentuk dan mencetak fungsi logika yang telah diminimisasi ke layar I.S. tabel minimum cover sudah terdefinisi F.S. fungsi logika yang telah diminimisasi dicetak ke layar
`void konversiBinerKeNotasiMinterm(int n)`	mengonversi dan mencetak biner ke notasi variabel I.S. implicantsTable bernama Tabel telah terdefinisi F.S. isi Tabel dalam bentuk variabel A,B,C,... telah dicetak ke layar
`int findMaxInTable(int *row)`	mencari prime implikan yang memiliki minterm paling banyak tidak digunakan pada suatu waktu dan mengembalikan banyak mintermnya
`int banyakImplikan(int n,int *temp)`	mengembalikan berapa banyak implikan yang ada suatu minterm tertentu
`void hapusMintermDariTabel(int)`	menghapus semua minterm pada suatu baris dan kolom implikan I.S. - F.S. seluruh minterm yang dicover prime implicant dihapus dari tabel

Berikut adalah prosedur bentukan lain yang tidak berhubungan dengan algoritma tabular

Fungsi / Prosedur	Deskripsi
`void validasi_file(char filename[])`	memvalidasi nama dan format file eksternal I.S. - F.S. nama file ada dan format file benar (.txt)
`void help()`	menampilkan opsi aks kepada user I.S - F.S pilihan opsi ditampilkan ke layar
`void cetakBumperOpening()`	menampilkan ASCII art 'Logic minimization' ke layar I.S array of string bumper_opening terdefinisi F.S ASCII art selamat datang ditampilkan ke layar
`void cetakBumperClosing()`	menampilkan ASCII art 'terimakasih' ke layar I.S array of string bumper_closing terdefinisi F.S ASCII art penutup ditampilkan ke layar
`void cetakPengarang()`	menampilkan daftar anggota kelompok ke layar I.S array of string pengarang terdefinisi F.S daftar pengarang program tercetak ke layar
`void cetakKredit()`	menampilkan daftar referensi ke layar I.S array of string referensi terdefinisi F.S daftar referensi program tercetak ke layar
`void opening()`	rutin pembuka program I.S - F.S bumper opening, pengarang, dan pilihan opsi ditampilkan ke layar
`void closing()`	rutin penutup program I.S -

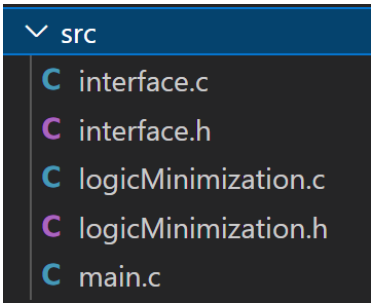
	F.S bumper closing, kredit ditampilkan ke lay

Struktur File
Berikut adalah pembagian folder pada program kami:



Folder `bin` berisi file binary executable hasil kompilasi sourcecode yang terdapat pada folder `src`. `Input` adalah folder khusus tempat meletakkan file input jika input fungsi logika ingin dilakukan melalui file eksternal. File `dll` berisi file lain-lain seperti gambar flowchart dan gambar-gambar untuk penjelasan di README.md github.

Untuk sourcecode sendiri, kami tidak mengimplementasikan semua dalam satu file.C. Namun kami membagi ke dalam 3 buah/pasang sourcecode .C dan file header .h :

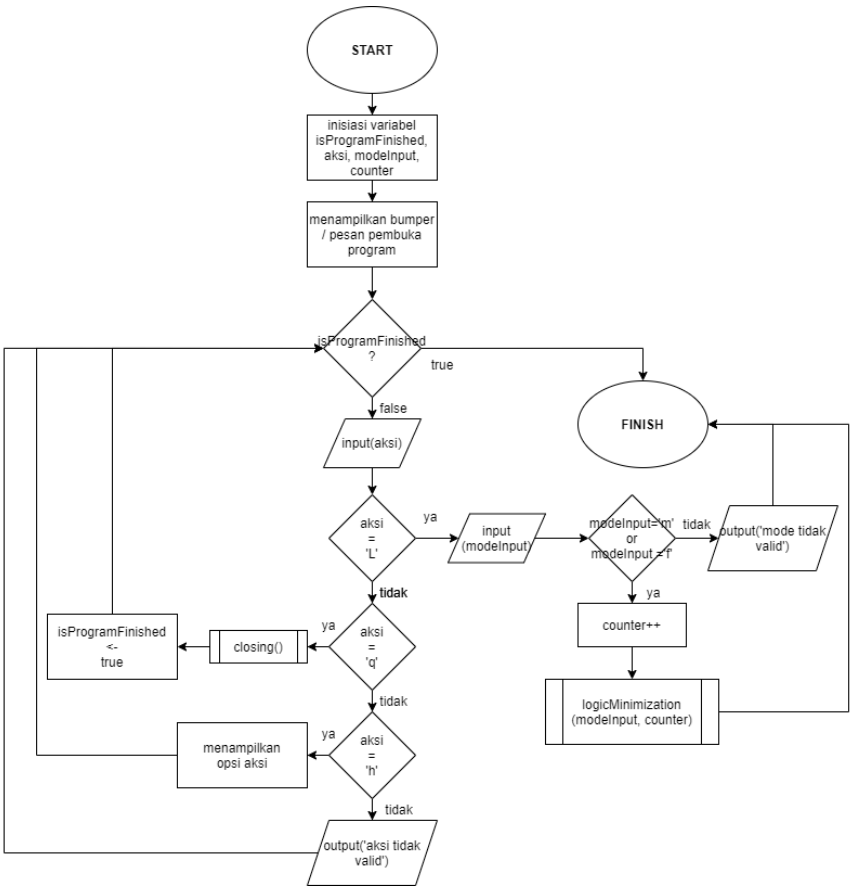


logicMinimization.h adalah tempat deklarasi variabel, konstanta, tipe bentukan, serta prototipe fungsi /prosedur yang berhubungan dengan algoritma minimisasi logika dengan tabular. Sedangkan implementasi untuk masing-masing prosedur/fungsi bentukan dapat ditemukan pada logicMinimization.c.

interface.h dan interface.c adalah tempat deklarasi variabel dan deklarasi+realisasi fungsi/prosedur untuk fungsi/prosedur yang tidak berhubungan dengan algoritma utama yang berupa periferal / interaksi program dengan user seperti pesan selamat datang, pesan penutup, menampilkan opsi aksi kepada user, dan sebagainya.

Sedangkan main.c adalah sebuah sourcecode yang mengintegrasikan semua sourcecode lain (driver code).

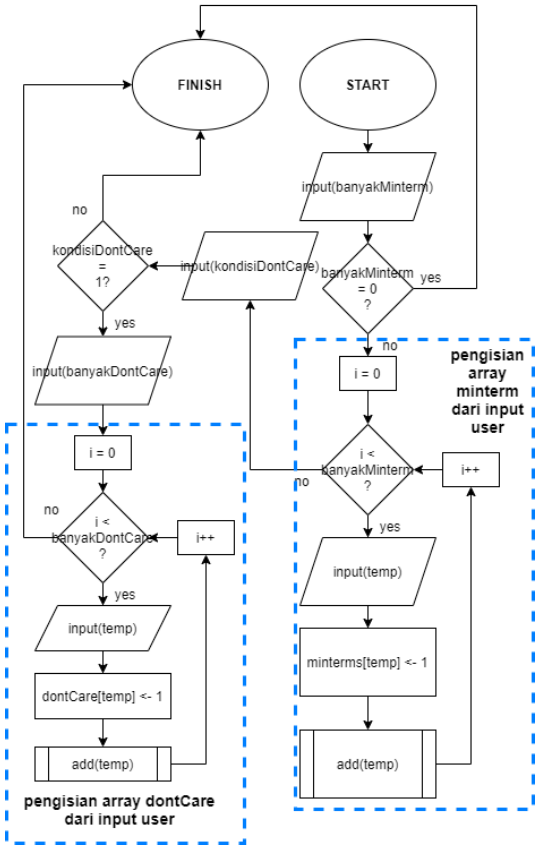
Alur Program
Berikut adalah flowchart untuk `main()`:



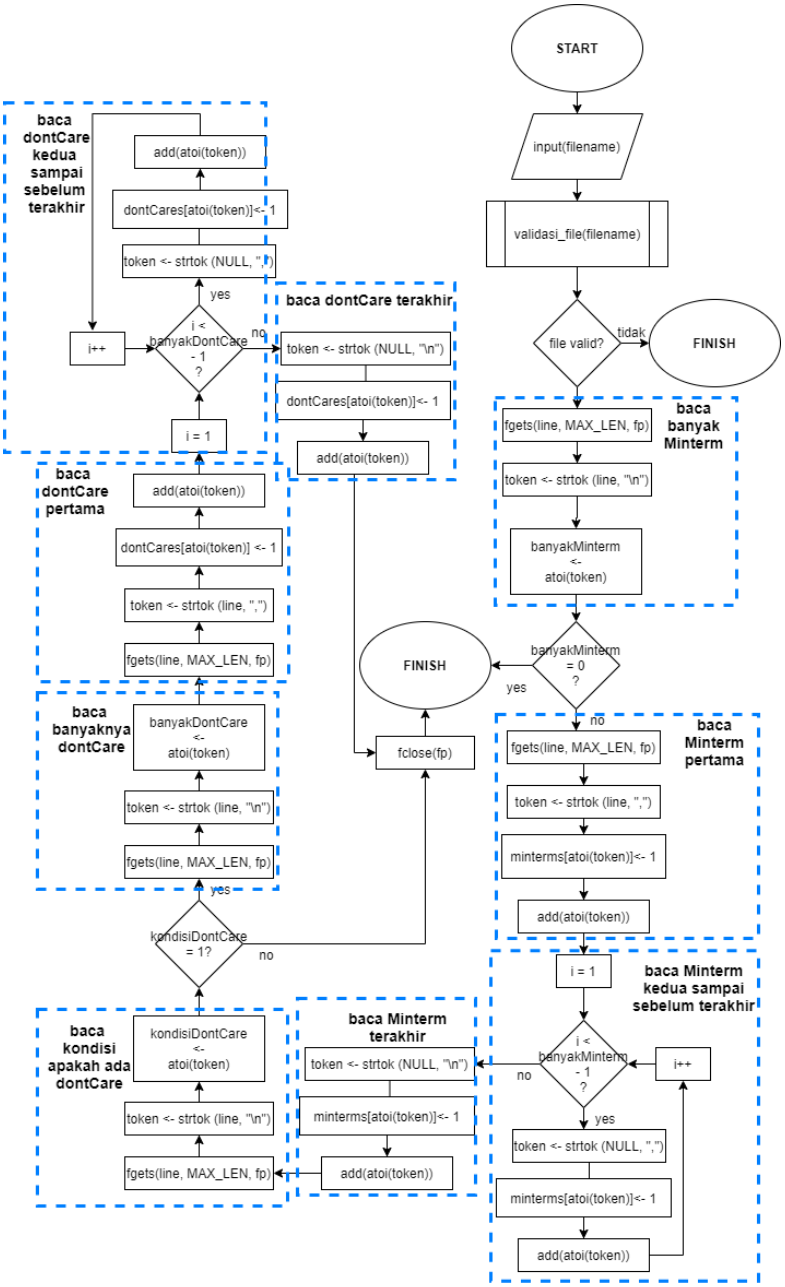
Seperti program pada umumnya, alur program bisa kita bagi menjadi 3 tahap yaitu input, proses, dan output.

Input

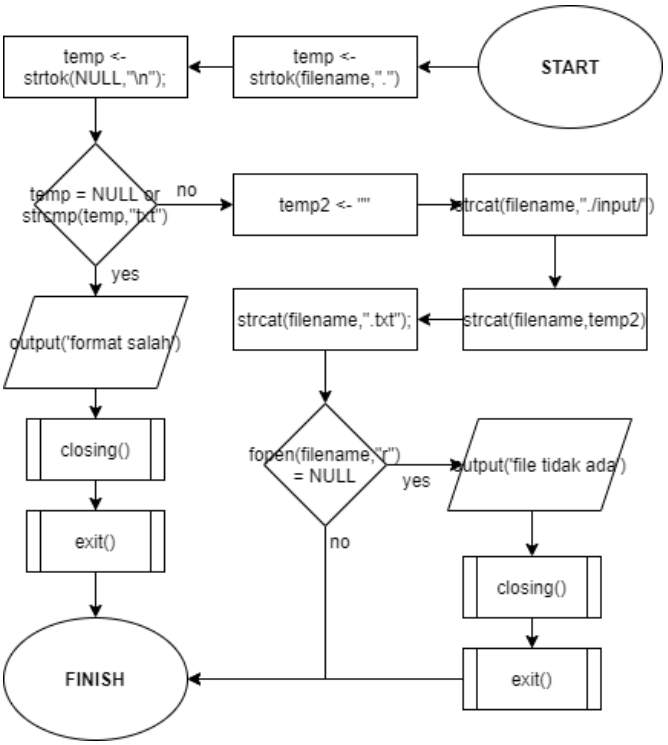
Di tahap input, Input dapat dilakukan dengan dua cara, yaitu input menggunakan file eksternal atau user secara manual dapat mengetikkan input melalui keyboard. Hal ini bergantung terhadap nilai variabel modeInput yang dimasukkan oleh user. Bila modeInput bernilai sama dengan ‘m’, maka input dilakukan secara manual. Berikut adalah flowchartnya:



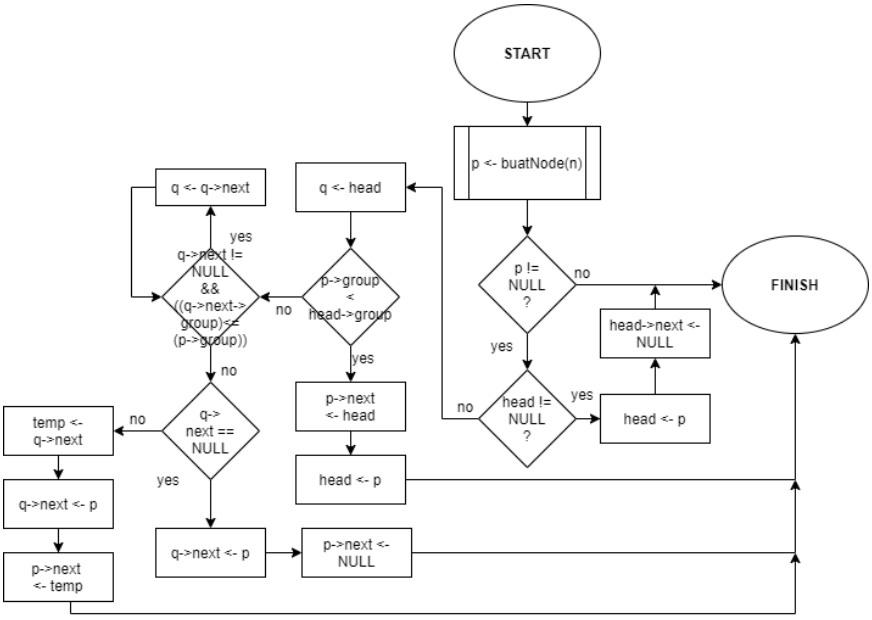
Bila modeInput bernilai sama dengan ‘f’, maka input akan diterima dari file eksternal. Berikut adalah flowchartnya:



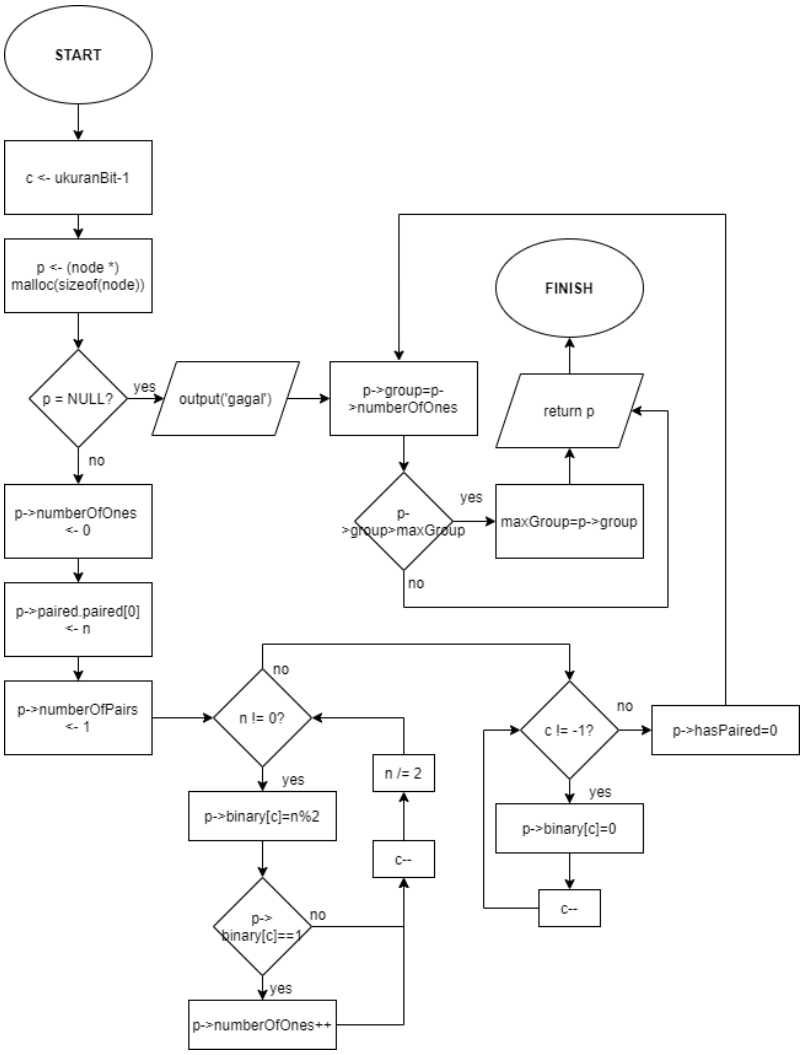
Jika input diterima dari file, terlihat bahwa terdapat proses validasi input. Berikut adalah prosedur untuk hal tersebut:



Saat input diterima melalui user secara manual ataupun melalui file, terdapat pembuatan node pada linked-list yang dibuat dengan pemanggilan prosedur add(). Berikut adalah flowchart untuk prosedur add():



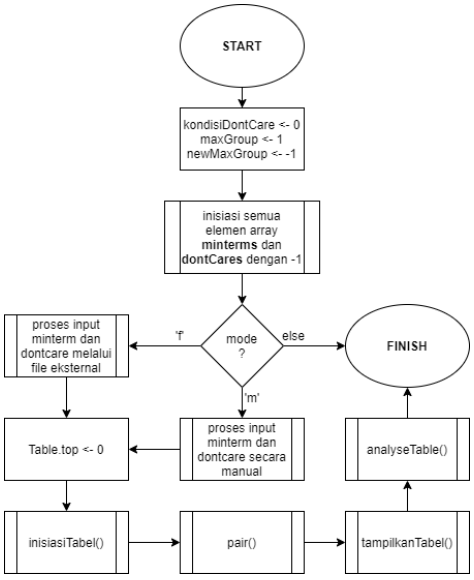
Prosedur add() memerlukan pemanggilan fungsi yang mengembalikan pointer to node bernama buatNode(). Berikut adalah flowchart untuk fungsi buatNode():



Proses dan Output

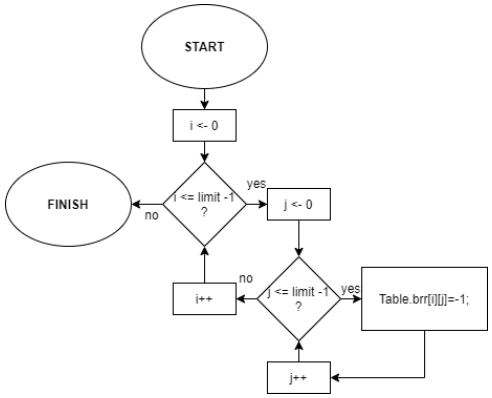
Setelah input berhasil diterima dan disimpan dalam variabel, kemudian kita masuk ke bagian inti dari program, yaitu algoritma minimisasi fungsi logika.

Pada fungsi main, proses ini dilakukan dengan pemanggilan prosedur logicMinimization(). Berikut adalah flowchart untuk prosedur logicMinimization():

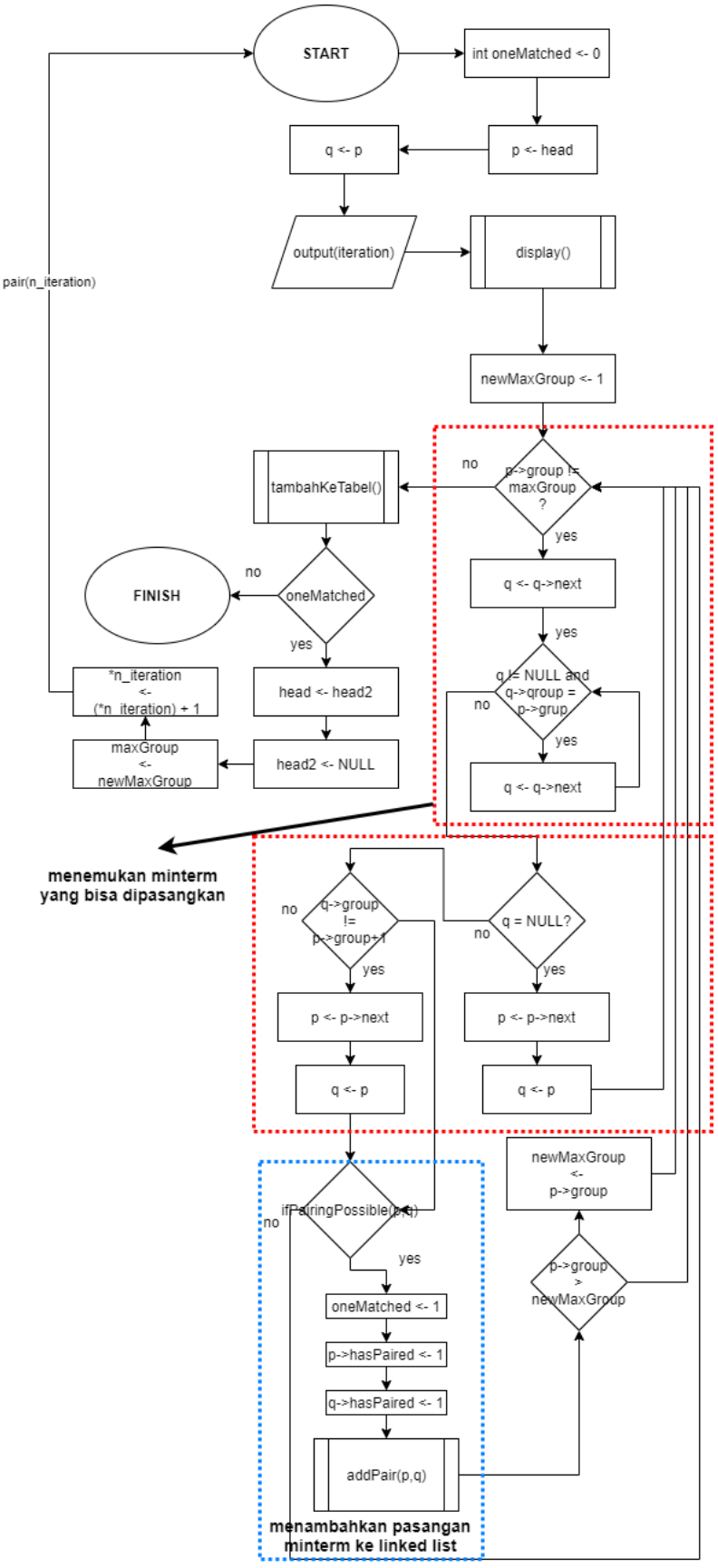


Terlihat bahwa secara garis besar, prosedur logicMinimization() terdiri atas ... bagian utama. Yaitu menerima input, proses pengisian dan penampilan tabel minterm (L1, L2, dst.) + tabel prime implicant pada pair() dan tampilkanTabel(), dan proses pencarian serta penampilan ekspresi fungsi logika yang telah diminimisasi pada prosedur analisisTabel().

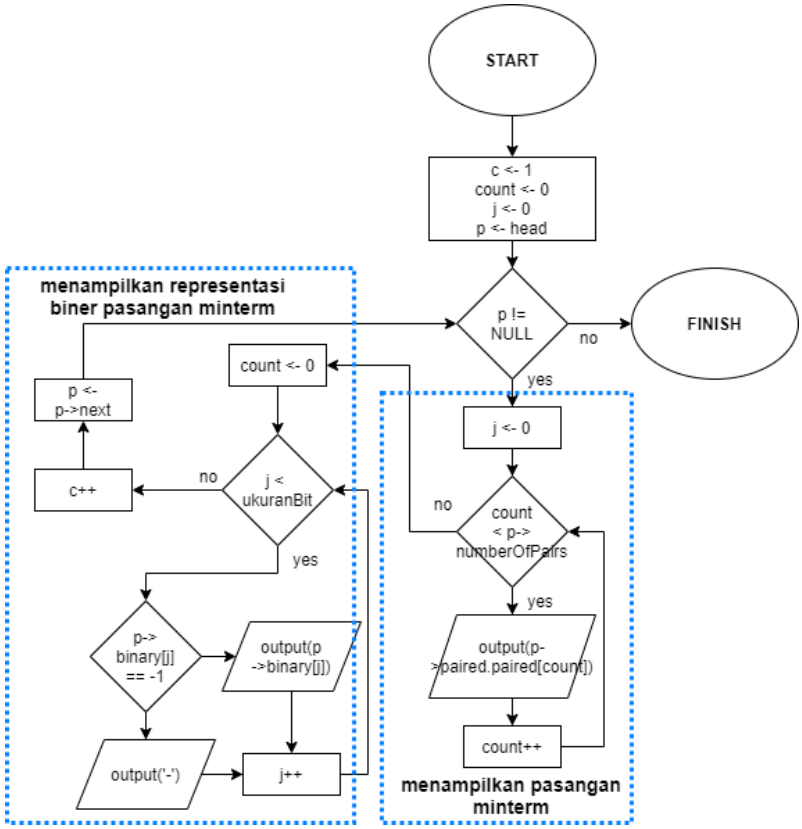
Berikut adalah flowchart untuk prosedur inisiasiTabel():



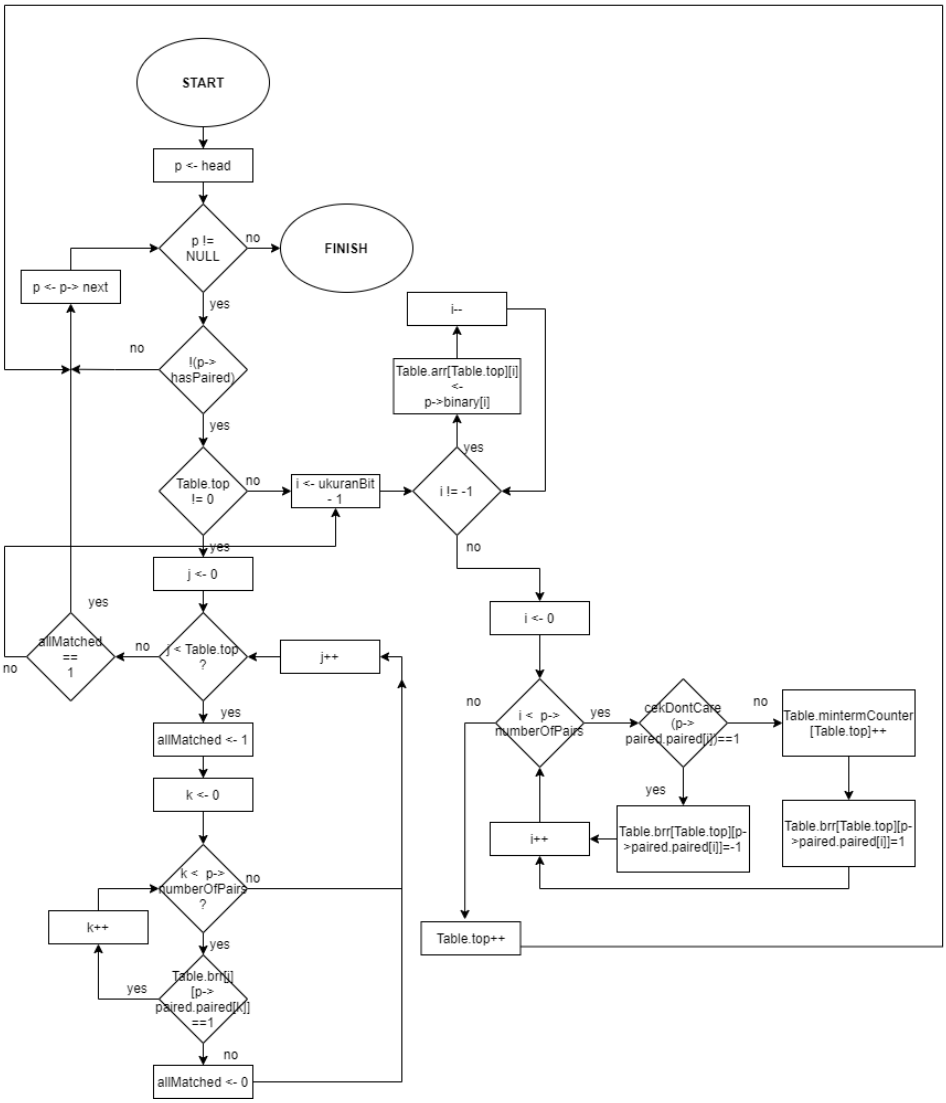
Berikut adalah flowchart untuk prosedur pair():


START
int oneMatched <- 0
p <- head
q <- p
output(iteration)
display()
pair(n_iteration)
newMaxGroup <- 1
no
p->group != maxGroup ?
yes
tambahKeTabel()
q <- q->next
yes
q != NULL and q->group = p->grup
no
yes
q <- q->next
no
oneMatched
yes
FINISH
*n_iteration <- (*n_iteration) + 1
head <- head2
maxGroup <- newMaxGroup
head2 <- NULL
menemukan minterm yang bisa dipasangkan
no
q->group != p->group+1
yes
p <- p->next
q <- p
no
q = NULL?
yes
p <- p->next
q <- p
newMaxGroup <- p->group
p->group > newMaxGroup
no
ifPairingPossible(p,q)
yes
oneMatched <- 1
p->hasPaired <- 1
q->hasPaired <- 1
addPair(p,q)
menambahkan pasangan minterm ke linked list

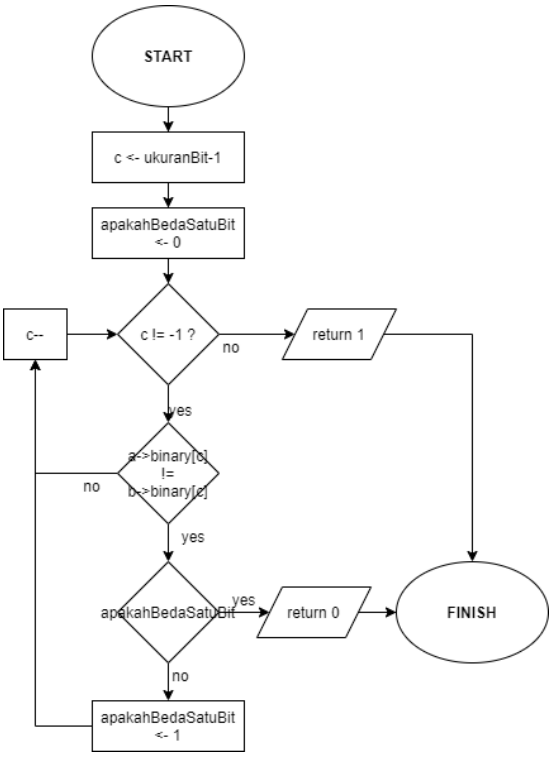
Pada prosedur pair(), terdapat pemanggilan prosedur display() yang menampilkan tabel minimisasi minterm L1, L2, dan seterusnya. Berikut adalah flowchart untuk prosedur display():



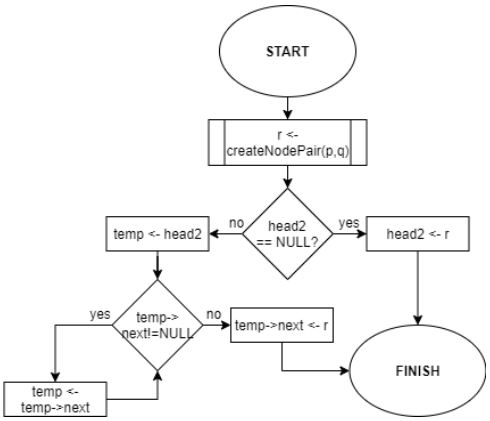
Pada prosedur pair(), juga terdapat pemanggilan prosedur tambahKeTabel(). Berikut adalah flowchart untuk prosedur tersebut:



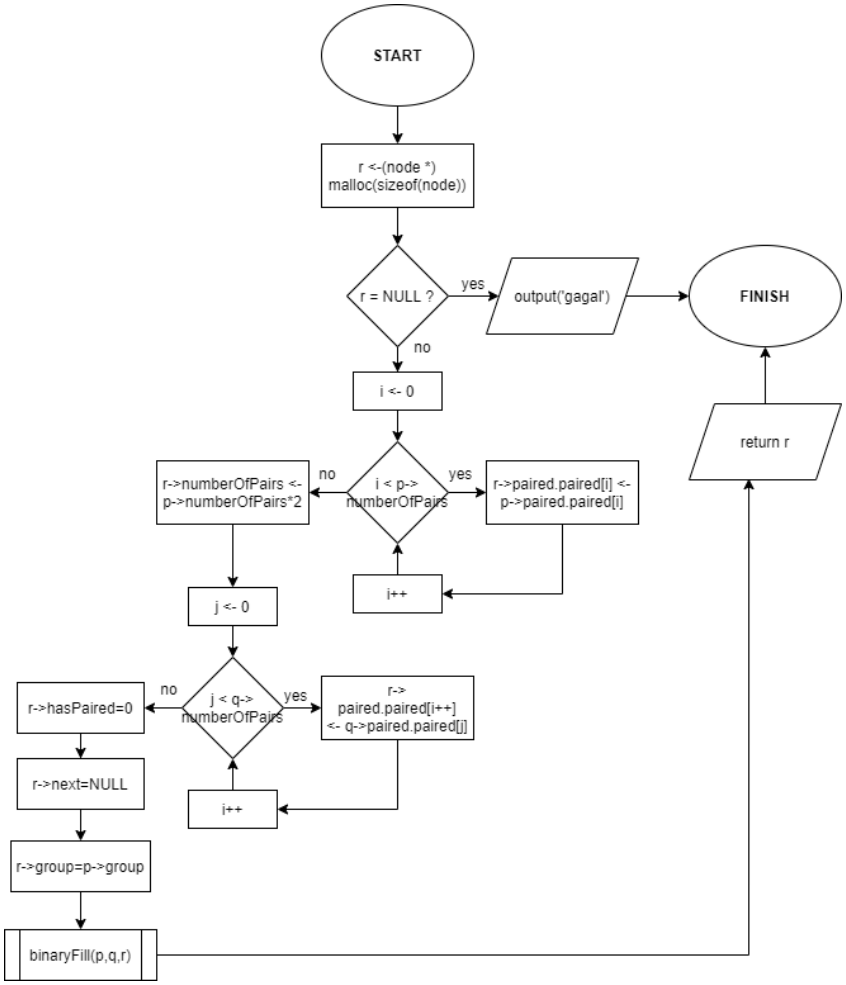
Pada prosedur pair() terdapat conditional statement yang memanggil fungsi ifPairingPossible(). Berikut adalah flowchart untuk ifPairingPossible():



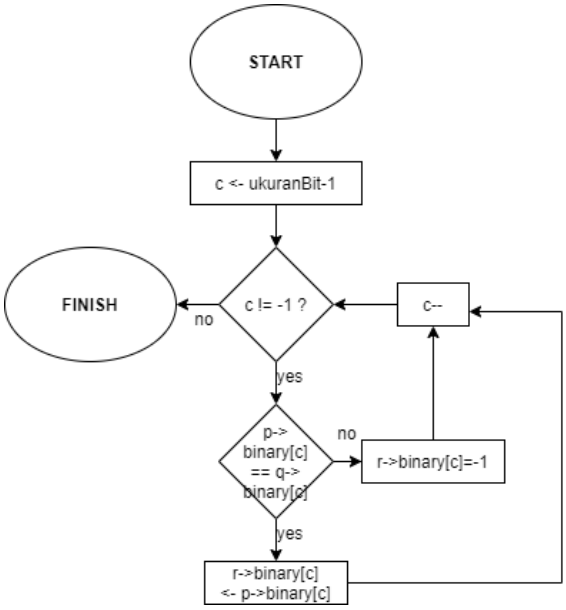
Pada prosedur pair(), juga terdapat pemanggilan prosedur addPair(). Berikut adalah flowchart untuk prosedur tersebut:



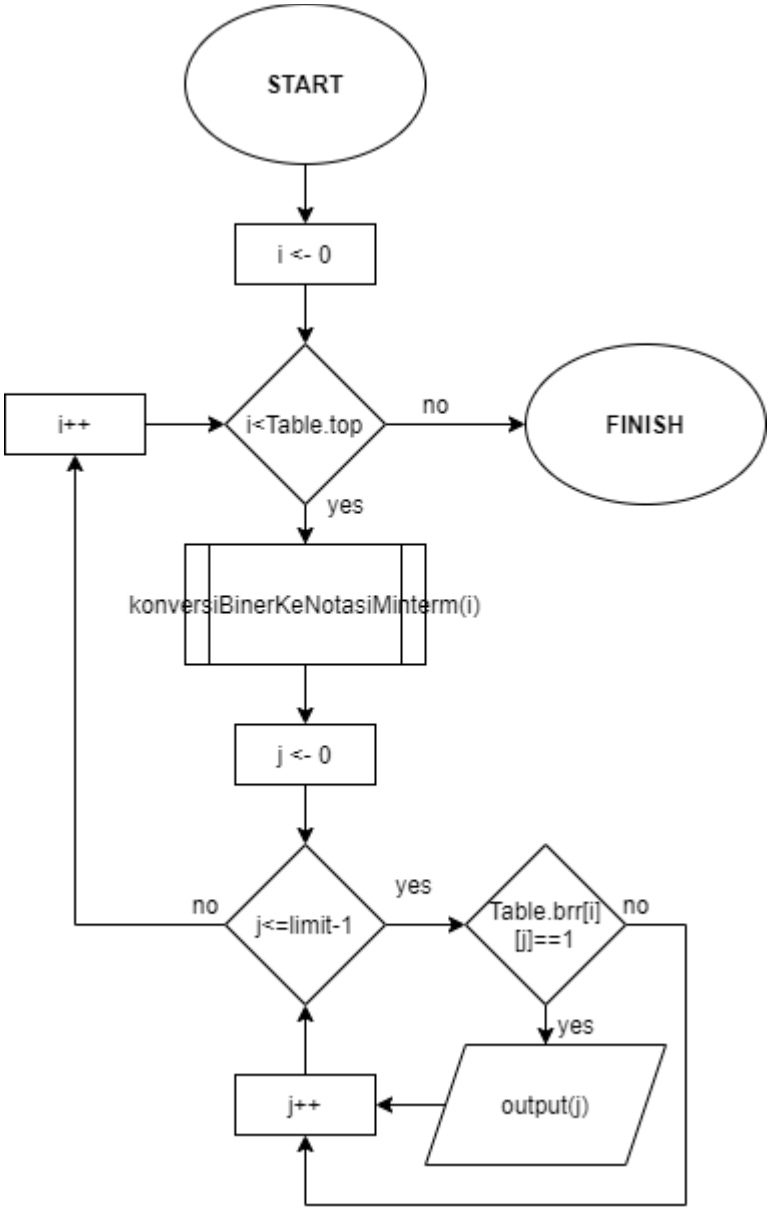
Pada prosedur addPair() terlihat terdapat pemanggilan prosedur createNodePair(). Berikut adalah flowchartnya:



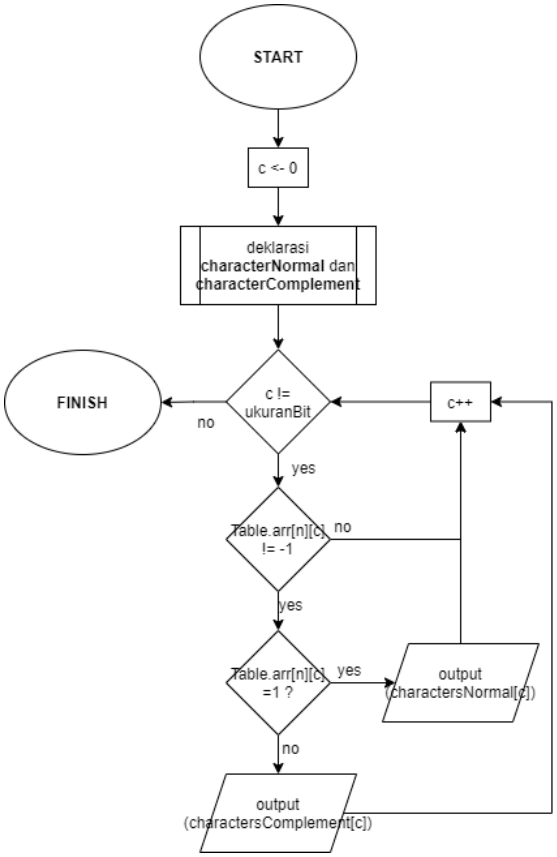
Pada prosedur createNodePair(), terdapat pemanggilan prosedur binaryFill(). Berikut adalah flowchartnya:



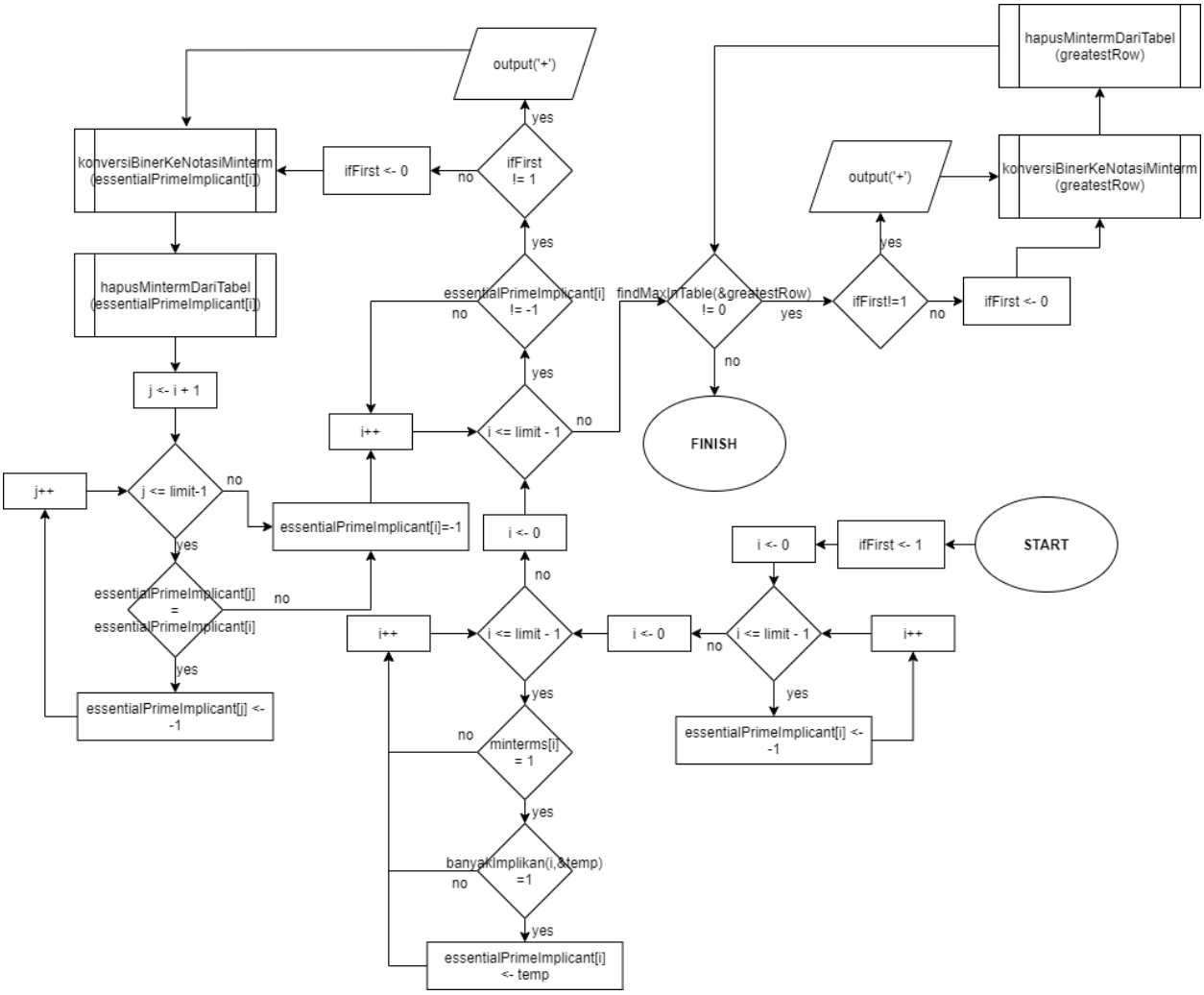
Kemudian setelah pemanggilan prosedur pair() pada logicMinimization(), selanjutnya terdapat pemanggilan prosedur tampilkanTabel(). Berikut adalah flowchart untuk prosedur tampilkanTabel():



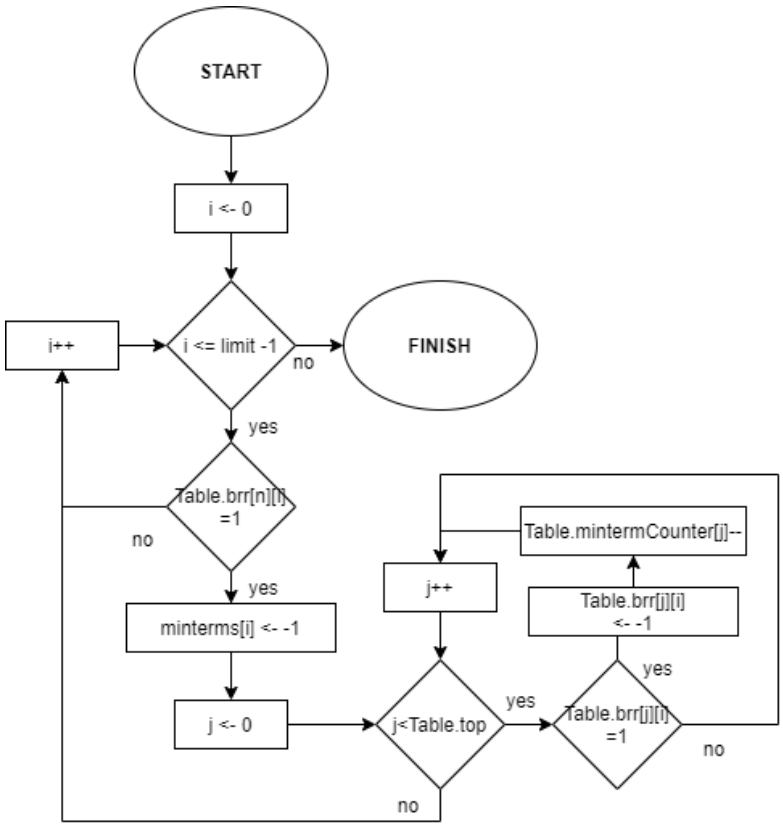
Terlihat bahwa terdapat pemanggilan prosedur konversiBinerKeNotasiMinterm() di sana. Berikut adalah flowchartnya:



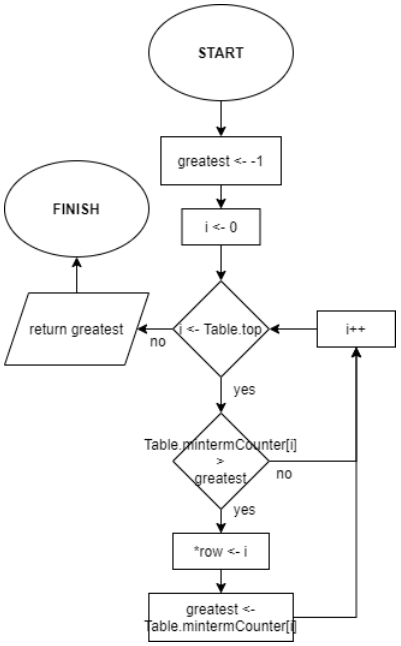
Terakhir pada logicMinimization() terdapat pemanggilan prosedur analisisTabel() untuk mencari dan menampilkan bentuk fungsi logika yang telah diminimisasi (dalam notasi variabel). Berikut adalah flowchart untuk prosedur analisisTabel():



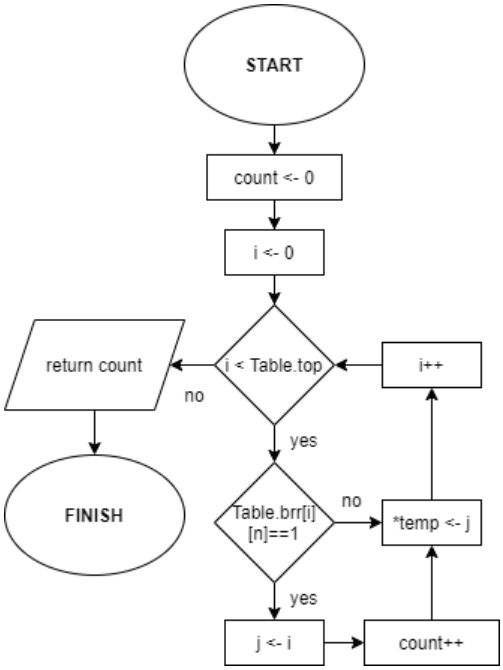
Di dalam analisisTabel() terlihat terdapat beberapa pemanggilan prosedur. Yang pertama adalah konversiBinerKeNotasiMinterm() yang sebelumnya telah ditampilkan flowchartnya. Kemudian terdapat pemanggilan prosedur hapusMintermDariTabel(). Berikut adalah flowchartnya:



Pada analisisTabel() juga terdapat conditional statement yang memanggil fungsi findMaxInTable(). Berikut adalah flowchartnya:

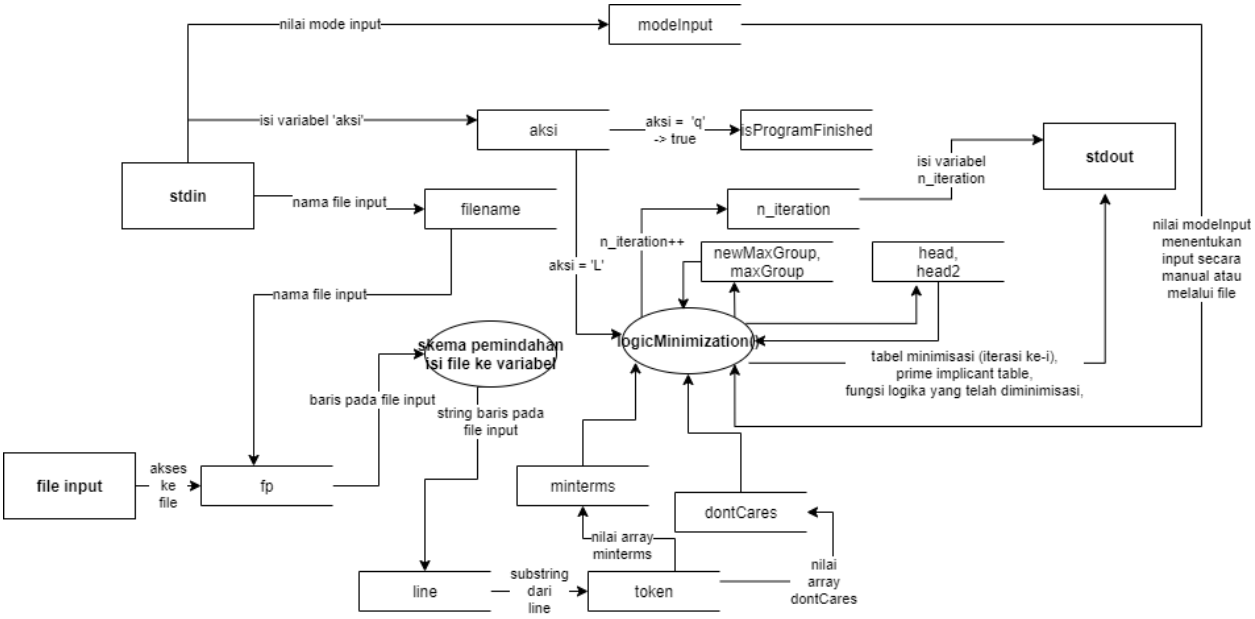


Terakhir, pada analisisTabel() juga terdapat conditional statement yang memanggil fungsi banyakImplikan(). Berikut adalah flowchartnya:

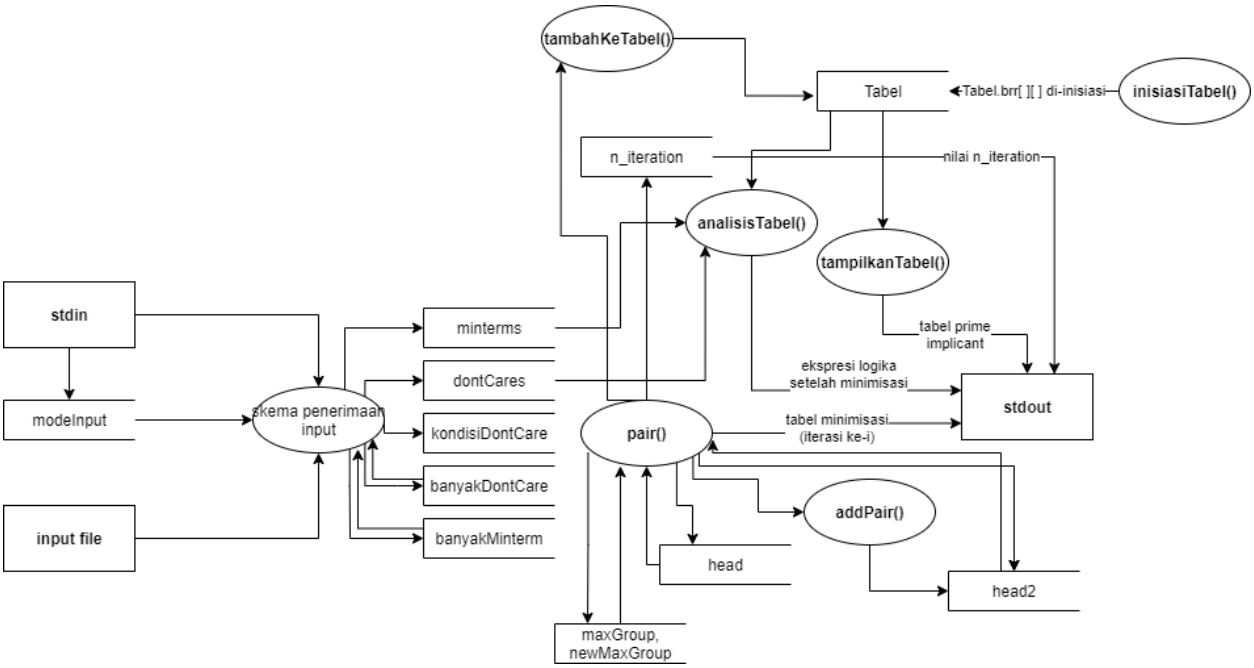


Data Flow Diagram

Berikut adalah data flow diagram level 0 (dilihat dari fungsi main()):



Berikut adalah data flow diagram yang lebih detail (yang terjadi di dalam prosedur logicMinimization()):



Pengujian Program

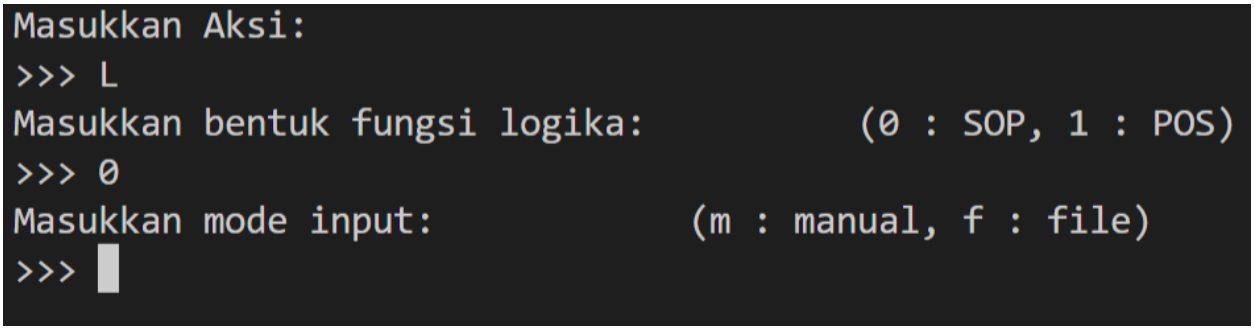
- Contoh 1 SOP : (tidak menggunakan don't care)

$$f(A,B,C,D) = \sum m(1,3,7,12,13,14,15)$$

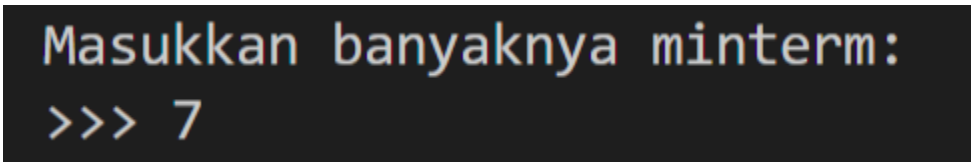
a.) Menggunakan Sistem Input Manual

```
Selamat datang di program:

 _                _          __  __ _      _                 _   _
| |              (_)        |  \/  (_)    (_)               | | (_)
| | ___   __ _ _  ___      | \  / |_ _ __  _ _ __ ___   __ _| |_ _  ___  _ __
| |/ _ \ / _` | |/ __|     | |\/| | | '_ \| | '_ ` _ \ / _` | __| |/ _ \| '_ \
| | (_) | (_| | | (__      | |  | | | | | | | | | | | | (_| | |_| | (_) | | | |
|_|\___/ \__, |_|\___|     |_|  |_|_|_| |_|_|_| |_| |_|\__,_|\__|_|\___/|_| |_|
          __/ |
         |___/
oleh:
1. Jefferson Grizzlie (13220013)
2. Muhammad Mikhail Ghrilman (13220021)
3. Bostang Palaguna (13220055)
4. Muhammad Daffa Rasyid (13220059)

--- Opsi Aksi ---
L : Melakukan minimisasi logika (algoritma utama)
q : keluar dari program
h : bantuan

Masukkan Aksi:
>>>
```

Langkah 1 : Ketik ‘L’ pada inputan Masukkan Aksi dan kemudian masukkan ‘0’ pada inputan bentuk fungsi logika untuk memilih SOP kemudian masukkan ‘m’ pada mode input untuk memilih sistem manual.



Langkah 2 : Masukkan jumlah minterm sesuai pada soal di atas yaitu 7.



Langkah 3 : Masukkan elemen-elemen minterm sesuai pada soal di atas.

```
Masukkan banyaknya minterm:
>>> 7
Masukkan minterm ke-1:
>>> 1
Masukkan minterm ke-2:
>>> 3
Masukkan minterm ke-3:
>>> 7
Masukkan minterm ke-4:
>>> 12
Masukkan minterm ke-5:
>>> 13
Masukkan minterm ke-6:
>>> 14
Masukkan minterm ke-7:
>>> 15
```

Langkah 4 : Input 0 yang berarti tidak ada don’t care sesuai soal di atas.

```
Apakah ada don't care? (1/0)
>>> 0
```

Langkah 5 : Setelah hasil keluar, input q untuk keluar dari program

```
Iterasi ke-1:
1    0001
3    0011
12    1100
7    0111
13    1101
14    1110
15    1111
Iterasi ke-2:
1,3    00-1
3,7    0-11
12,13    110-
12,14    11-0
7,15    -111
13,15    11-1
14,15    111-
Iterasi ke-3:
12,13,14,15    11--
12,14,13,15    11--
Tabel Prime Implicant:
abD    1      3
aCD    3      7
BCD    7      15
AB    12      13      14      15
Fungsi Logika setelah minimisasi:
         abD + AB + aCD
```

```
Masukkan Aksi:
>>> q
```

```
 _                 _                                 _ _                    __
| |               (_)                               (_) |                  _  \ \
| |_ ___ _ __ _ _ __ ___   __ _ | | ____ _ ___ _| |__ (_) | |
| __/ _ \ '__| | '_ ` _ \ / _` | | |/ / _` / __| | '_ \ | | |
| || __/ | | | | | | | (_| | |   < (_| \__ \ | | | | _ | | |
 \__\___|_| |_|_| |_| |_|\__,_| |_|\_\__,_|___/_|_| |_| (_) | | |
                                                          /_/
```

b.) Menggunakan Sistem Read File

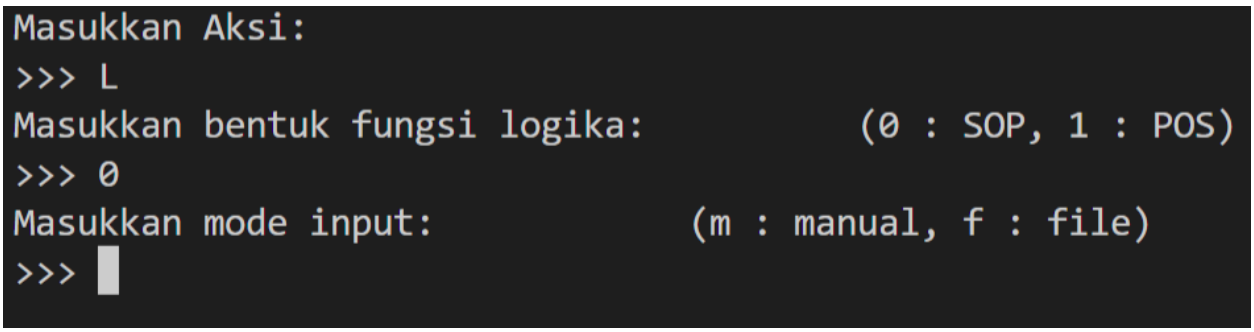
File eksternal :

```
input > ≡ input6.txt
    1   7
    2   1,3,7,12,13,14,15
    3   0
```

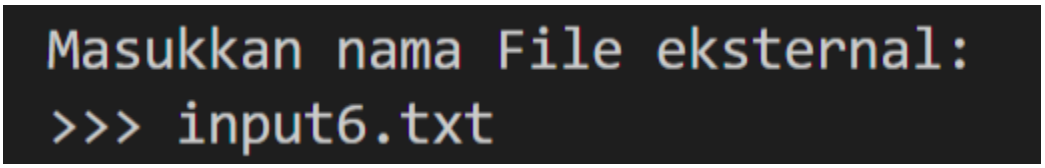
- Line 1 menyatakan banyaknya minterm
- Line 2 menyatakan elemen-elemen minterm
- Line 3 menyatakan ada atau tidaknya don’t care apabila 0 maka berhenti di line tersebut

```
Selamat datang di program:
 _                 _                 __ __ _                 _                 _   _
| |               (_)               |  \/  (_)               (_)               | | (_)
| | ___   __ _ _  ___  | \  / |_ _ __  _ _ __ ___  _ ______ _| |_ _  ___  _ __
| |/ _ \ / _` | |/ __| | |\/| | | '_ \| | '_ ` _ \| |_  / _` | __| |/ _ \| '_ \
| | (_) | (_| | | (__  | |  | | | | | | | | | | | | |/ / (_| | |_| | (_) | | | |
|_|\___/ \__, |_|\___| |_|  |_|_|_| |_|_|_| |_| |_|_/___\__,_|\__|_|\___/|_| |_|
          __/ |
         |___/
oleh:
1. Jefferson Grizzlie (13220013)
2. Muhammad Mikhail Ghrilman (13220021)
3. Bostang Palaguna (13220055)
4. Muhammad Daffa Rasyid (13220059)

--- Opsi Aksi ---
L : Melakukan minimisasi logika (algoritma utama)
q : keluar dari program
h : bantuan

Masukkan Aksi:
>>>
```

Langkah 1 : Ketik ‘L’ pada inputan Masukkan Aksi dan kemudian masukkan ‘0’ pada inputan bentuk fungsi logika untuk memilih SOP kemudian masukkan ‘f’ pada mode input untuk memilih sistem manual.



Langkah 2 : Masukkan nama file sesuai contoh di atas.



Langkah 3 : Setelah hasil keluar, input q untuk keluar dari program

```
Iterasi ke-1:
1   0001
3   0011
12   1100
7   0111
13   1101
14   1110
15   1111
Iterasi ke-2:
1,3   00-1
3,7   0-11
12,13   110-
12,14   11-0
7,15   -111
13,15   11-1
14,15   111-
Iterasi ke-3:
12,13,14,15   11--
12,14,13,15   11--
Tabel Prime Implicant:
abD   1     3
aCD   3     7
BCD   7     15
AB   12     13     14     15
Fungsi Logika setelah minimisasi:
        abD + AB + aCD
```

```
Masukkan Aksi:
>>> q
```

- Contoh 2 : (menggunakan don’t care)

$$f(A,B,C,D) = \sum m(7,13,14,15) + \sum d(5,11)$$

a.) Menggunakan Sistem Input Manual

```
Selamat datang di program:
  _             _        __  __ _       _           _              _   _
 | |           (_)      |  \/  (_)     (_)         (_)            | | (_)
 | | ___   __ _ _  ___  | \  / |_ _ __  _ _ __ ___  _ ______ _ ___| |_ _  ___  _ __
 | |/ _ \ / _` | |/ __| | |\/| | | '_ \| | '_ ` _ \| |_  / _` |_  / __| |/ _ \| '_ \
 | | (_) | (_| | | (__  | |  | | | | | | | | | | | | |/ / (_| |/ /| |_| | (_) | | | |
 |_|\___/ \__, |_|\___| |_|  |_|_|_| |_|_|_| |_| |_|_/___\__,_/___|\__|_|\___/|_| |_|
           __/ |
          |___/
oleh:
1. Jefferson Grizzlie (13220013)
2. Muhammad Mikhail Ghrilman (13220021)
3. Bostang Palaguna (13220055)
4. Muhammad Daffa Rasyid (13220059)

--- Opsi Aksi ---
L : Melakukan minimisasi logika (algoritma utama)
q : keluar dari program
h : bantuan

Masukkan Aksi:
>>>
```

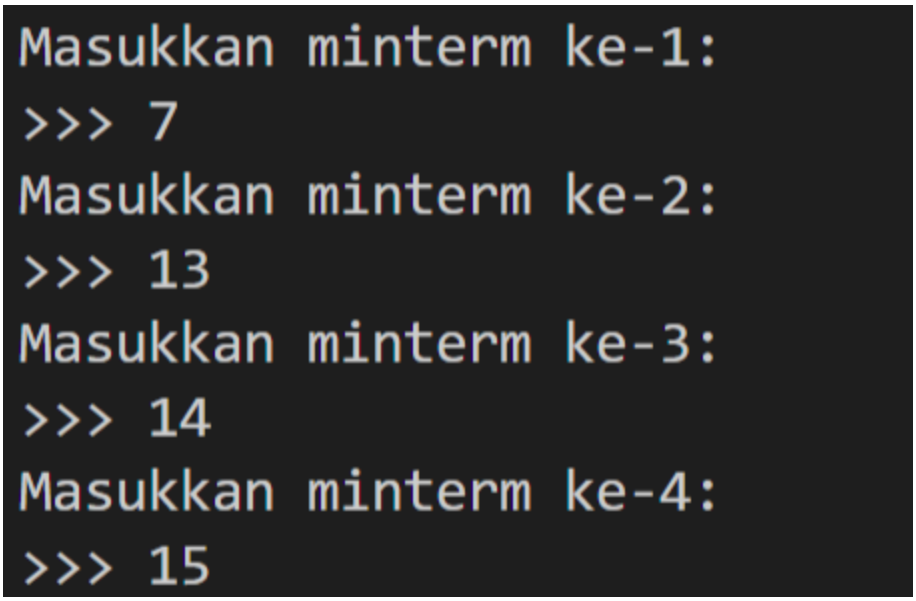
Langkah 1 : Ketik ‘L’ pada inputan Masukkan Aksi dan kemudian masukkan ‘0’ pada inputan bentuk fungsi logika untuk memilih SOP kemudian masukkan ‘m’ pada mode input untuk memilih sistem manual.

```
Masukkan Aksi:
>>> L
Masukkan bentuk fungsi logika:          (0 : SOP, 1 : POS)
>>> 0
Masukkan mode input:            (m : manual, f : file)
>>>
```

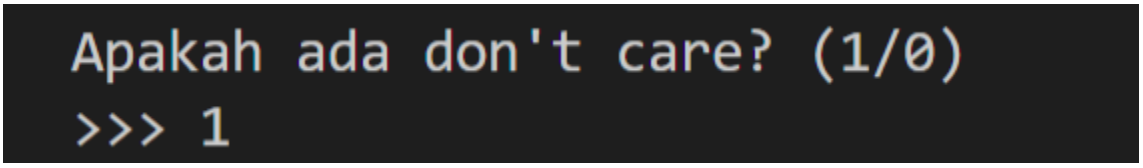
Langkah 2 : Masukkan jumlah minterm sesuai pada soal di atas yaitu 4.

```
Masukkan banyaknya minterm:
>>> 4
```

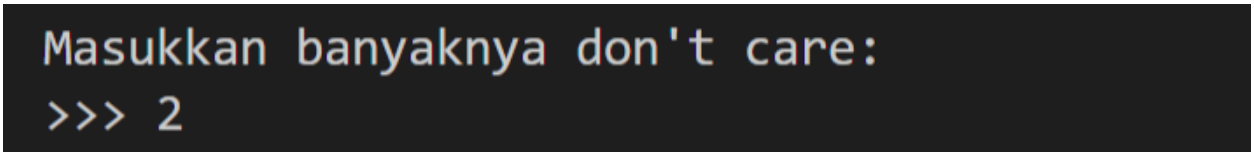
Langkah 3 : Masukkan elemen-elemen minterm sesuai pada soal di atas.



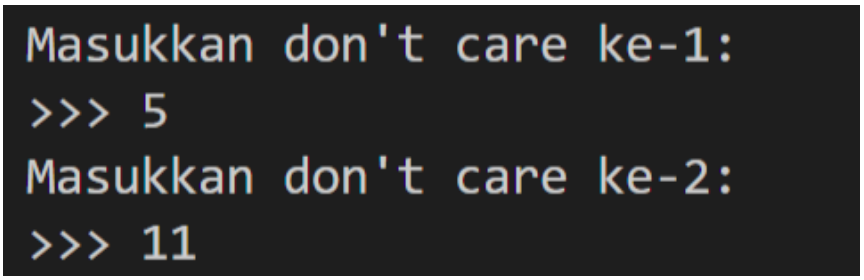
Langkah 4 : Input 1 yang berarti ada don’t care sesuai soal di atas.



Langkah 5 : Masukkan jumlah don’t care sesuai soal di atas yaitu 2.



Langkah 6 : Masukkan elemen-elemen don’t care sesuai soal di atas.

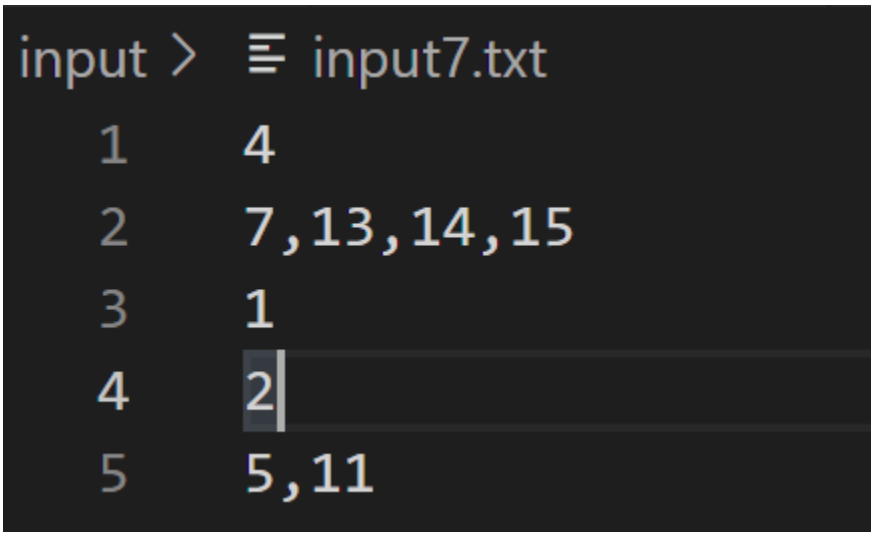


Langkah 7 : Setelah input keluar, input q untuk keluar dari program.

```
Masukkan nama File eksternal:
>>> input7.txt
Iterasi ke-1:
5   0101
7   0111
13   1101
14   1110
11   1011
15   1111
Iterasi ke-2:
5,7   01-1
5,13   -101
7,15   -111
13,15   11-1
14,15   111-
11,15   1-11
Iterasi ke-3:
5,7,13,15   -1-1
5,13,7,15   -1-1
Tabel Prime Implicant:
ABC   14     15
ACD   15
BD   7     13     15
BD   7     13     15
Fungsi Logika setelah minimisasi:
        ABC + BD
```

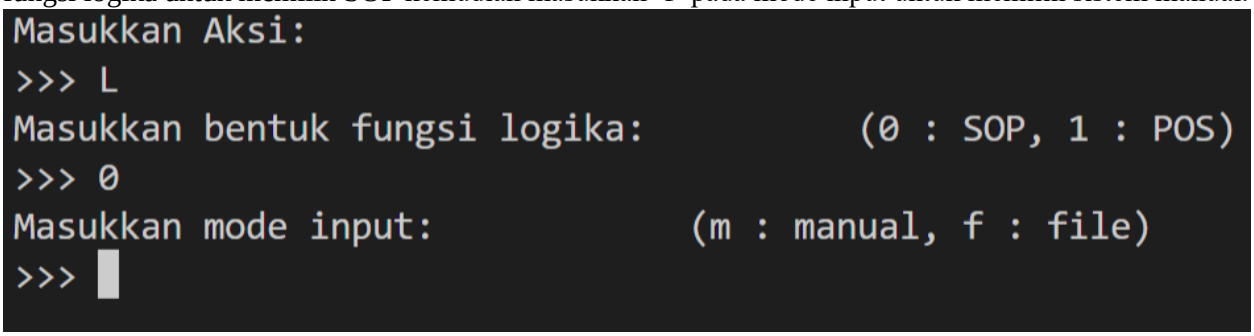
b.) Menggunakan Sistem Read File

File eksternal :

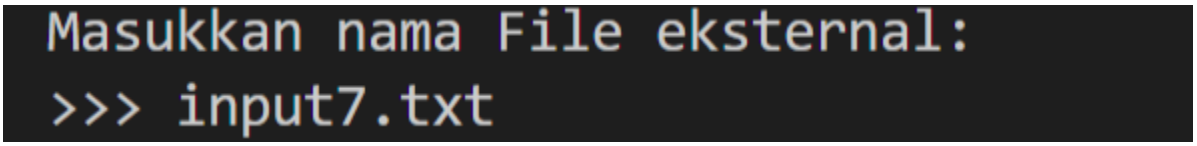


- Line 1 menyatakan banyaknya minterm
- Line 2 menyatakan elemen-elemen minterm
- Line 3 menyatakan ada atau tidaknya don’t care apabila 1 maka lanjut ke line selanjutnya
- Line 4 menyatakan jumlah don’t care
- Line 5 menyatakan elemen-elemen don’t care.

Langkah 1 : Ketik ‘L’ pada inputan Masukkan Aksi dan kemudian masukkan ‘0’ pada inputan bentuk fungsi logika untuk memilih SOP kemudian masukkan ‘f’ pada mode input untuk memilih sistem manual.



Langkah 2 : Masukkan nama file sesuai contoh di atas.



Langkah 3 : Setelah hasil keluar, input q untuk keluar dari program

```
Masukkan nama File eksternal:
>>> input7.txt
Iterasi ke-1:
5   0101
7   0111
13   1101
14   1110
11   1011
15   1111
Iterasi ke-2:
5,7   01-1
5,13   -101
7,15   -111
13,15   11-1
14,15   111-
11,15   1-11
Iterasi ke-3:
5,7,13,15   -1-1
5,13,7,15   -1-1
Tabel Prime Implicant:
ABC   14     15
ACD   15
BD   7     13     15
BD   7     13     15
Fungsi Logika setelah minimisasi:
        ABC + BD
```

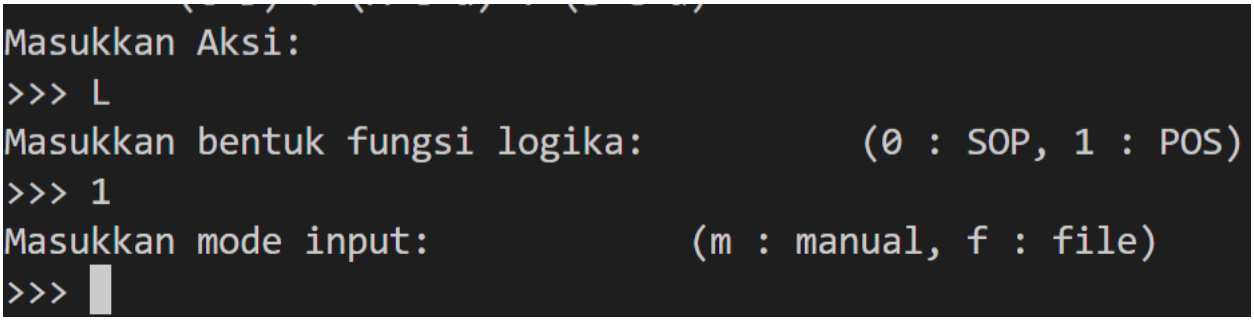
```
Masukkan Aksi:
>>> q
```

- Contoh 3 POS : (tidak menggunakan don’t care)

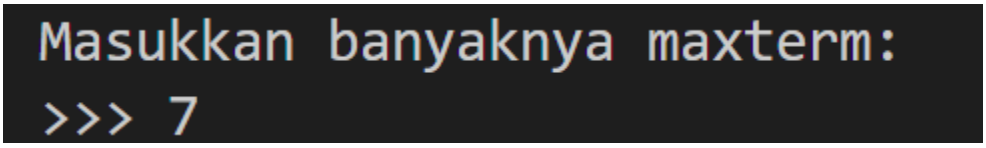
$$f(A,B,C,D) = \sum M(1,2,5,6,9,10,14)$$

a.) Menggunakan Sistem Input Manual

Langkah 1 : Ketik ‘L’ pada inputan Masukkan Aksi dan kemudian masukkan ‘1’ pada inputan bentuk fungsi logika untuk memilih POS kemudian masukkan ‘m’ pada mode input untuk memilih sistem manual.



Langkah 2 : Masukkan jumlah maxterm sesuai pada soal di atas yaitu 7.



Langkah 3 : Masukkan elemen-elemen minterm sesuai pada soal di atas.

```
Masukkan maxterm ke-1:
>>> 1
Masukkan maxterm ke-2:
>>> 2
Masukkan maxterm ke-3:
>>> 5
Masukkan maxterm ke-4:
>>> 6
Masukkan maxterm ke-5:
>>> 9
Masukkan maxterm ke-6:
>>> 10
Masukkan maxterm ke-7:
>>> 14
```

Langkah 4 : Input ‘0’ yang berarti tidak ada don’t care sesuai soal di atas.

```
Apakah ada don't care? (1/0)
>>> 0
```

Langkah 5 : Setelah hasil keluar, input q untuk keluar dari program

```
Iterasi ke-1:
1    0001
2    0010
5    0101
6    0110
9    1001
10    1010
14    1110
Iterasi ke-2:
1,5    0-01
1,9    -001
2,6    0-10
2,10    -010
6,14    -110
10,14    1-10
Iterasi ke-3:
2,6,10,14    --10
2,10,6,14    --10
Tabel Prime Implicant:
(A+C+d)    1      5
(B+C+d)    1      9
(c+D)    2      6      10      14
Fungsi Logika setelah minimisasi:
         (c+D) . (A+C+d) . (B+C+d)
```

```
Masukkan Aksi:
>>> q
```

```
 _                                                                 _ _                 __
| |                (_)                                           (_) |               _  \ \
| |_ ___ _ __ _ _ __ ___   __ _  | |____ _ ___ _| |__      (_)  | |
| __/ _ \ '__| | '_ ` _ \ / _` | | |/ / _` / __| | '_ \             | |
| ||  __/ |  | | | | | | | (_| | |   < (_| \__ \ | | | |     _   | |
 \__\___|_|  |_|_| |_| |_|\__,_| |_|\_\__,_|___/_|_| |_|    (_)  | |
                                                                  /_/
```

b.) Menggunakan Sistem Read File

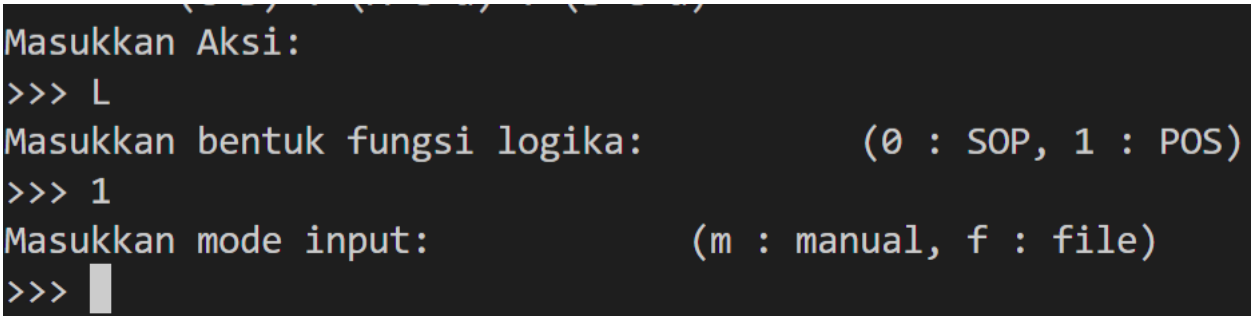
File eksternal :

```
input > ≡ input9.txt
    1   7
    2   1,2,5,6,9,10,14
    3   0
```

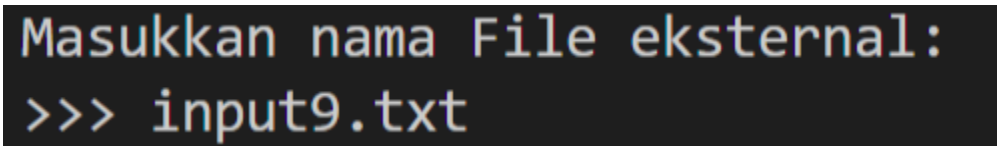
- Line 1 menyatakan banyaknya maxterm
- Line 2 menyatakan elemen-elemen maxterm
- Line 3 menyatakan ada atau tidaknya don’t care apabila 0 maka berhenti di line tersebut

```
Selamat datang di program:
 _                _          __  __  _          _                 _   _
| |              (_)        |  \/  (_)        (_)               | | (_)
| |  ___   __ _  _   ___    | \  / |_ _ __   _ _ __ ___   _ ______ _| |_ _  ___  _ __
| | / _ \ / _` | |/ __|  | |\/| | | '_ \| | '_ ` _ \| |_  / _` | __| |/ _ \| '_ \
| || (_) | (_| | | (__   | |  | | | | | | | | | | | | |/ / (_| | |_| | (_) | | | |
|_| \___/ \__, |_|\___|  |_|  |_|_|_| |_|_|_| |_| |_|_/___\__,_|\__|_|\___/|_| |_|
           __/ |
          |___/
oleh:
1. Jefferson Grizzlie (13220013)
2. Muhammad Mikhail Ghrilman (13220021)
3. Bostang Palaguna (13220055)
4. Muhammad Daffa Rasyid (13220059)

--- Opsi Aksi ---
L : Melakukan minimisasi logika (algoritma utama)
q : keluar dari program
h : bantuan

Masukkan Aksi:
>>>
```

Langkah 1 : Ketik ‘L’ pada inputan Masukkan Aksi dan kemudian masukkan ‘1’ pada inputan bentuk fungsi logika untuk memilih POS kemudian masukkan ‘f’ pada mode input untuk memilih sistem read file.



Langkah 2 : Masukkan nama file sesuai contoh di atas.



Langkah 3 : Setelah hasil keluar, input q untuk keluar dari program

```
Iterasi ke-1:
1    0001
2    0010
5    0101
6    0110
9    1001
10    1010
14    1110
Iterasi ke-2:
1,5    0-01
1,9    -001
2,6    0-10
2,10    -010
6,14    -110
10,14    1-10
Iterasi ke-3:
2,6,10,14    --10
2,10,6,14    --10
Tabel Prime Implicant:
(A+C+d)    1      5
(B+C+d)    1      9
(c+D)    2      6      10      14
Fungsi Logika setelah minimisasi:
         (c+D) . (A+C+d) . (B+C+d)
```

- Contoh 4 : (menggunakan don't care)

$$f(A,B,C,D) = \sum M(7,13,14,15) + \sum d(5,11)$$

a.) Menggunakan Sistem Input Manual

Langkah 1 : Ketik 'L' pada inputan Masukkan Aksi dan kemudian masukkan '1' pada inputan bentuk fungsi logika untuk memilih POS kemudian masukkan 'm' pada mode input untuk memilih sistem

manual.

```
Masukkan Aksi:
>>> L
Masukkan bentuk fungsi logika:          (0 : SOP, 1 : POS)
>>> 0
Masukkan mode input:             (m : manual, f : file)
>>>
```

Langkah 2 : Masukkan jumlah minterm sesuai pada soal di atas yaitu 4.

```
Masukkan banyaknya minterm:
>>> 4
```

Langkah 3 : Masukkan elemen-elemen minterm sesuai pada soal di atas.

```
Masukkan minterm ke-1:
>>> 7
Masukkan minterm ke-2:
>>> 13
Masukkan minterm ke-3:
>>> 14
Masukkan minterm ke-4:
>>> 15
```

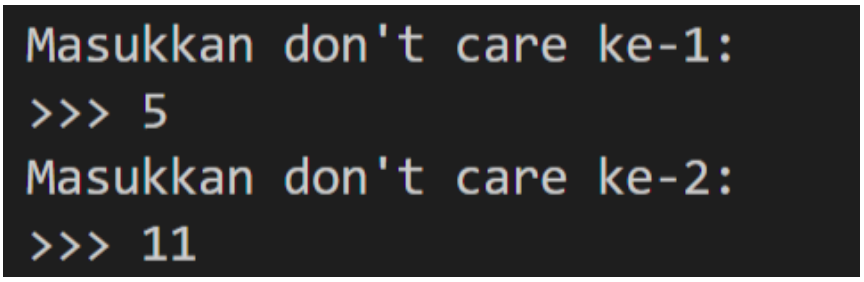
Langkah 4 : Input 1 yang berarti ada don’t care sesuai soal di atas.

```
 Apakah ada don't care? (1/0)
 >>> 1
```

Langkah 5 : Masukkan jumlah don’t care sesuai soal di atas yaitu 2.

```
Masukkan banyaknya don't care:
>>> 2
```

Langkah 6 : Masukkan elemen-elemen don’t care sesuai soal di atas.

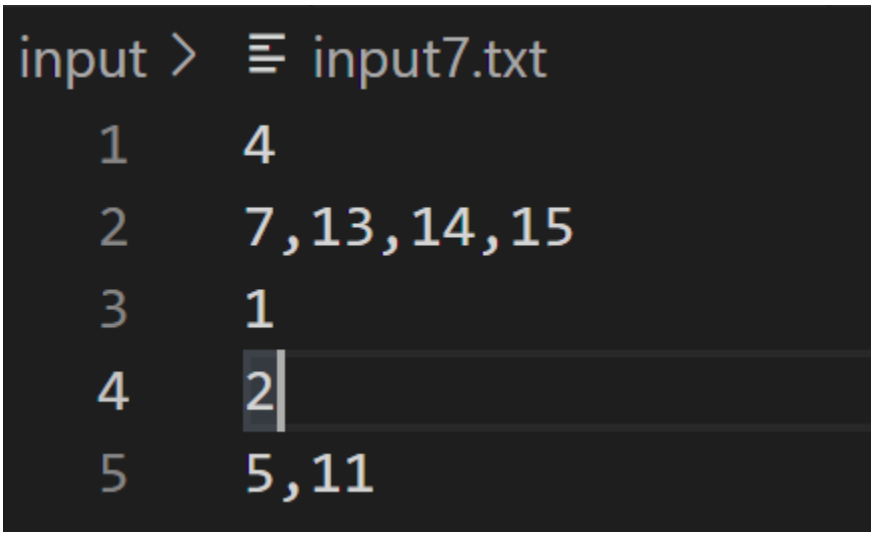


Langkah 7 : Setelah input keluar, input q untuk keluar dari program.

```
Masukkan nama File eksternal:
>>> input7.txt
Iterasi ke-1:
5   0101
7   0111
13   1101
14   1110
11   1011
15   1111
Iterasi ke-2:
5,7   01-1
5,13   -101
7,15   -111
13,15   11-1
14,15   111-
11,15   1-11
Iterasi ke-3:
5,7,13,15   -1-1
5,13,7,15   -1-1
Tabel Prime Implicant:
ABC   14     15
ACD   15
BD   7     13     15
BD   7     13     15
Fungsi Logika setelah minimisasi:
        ABC + BD
```

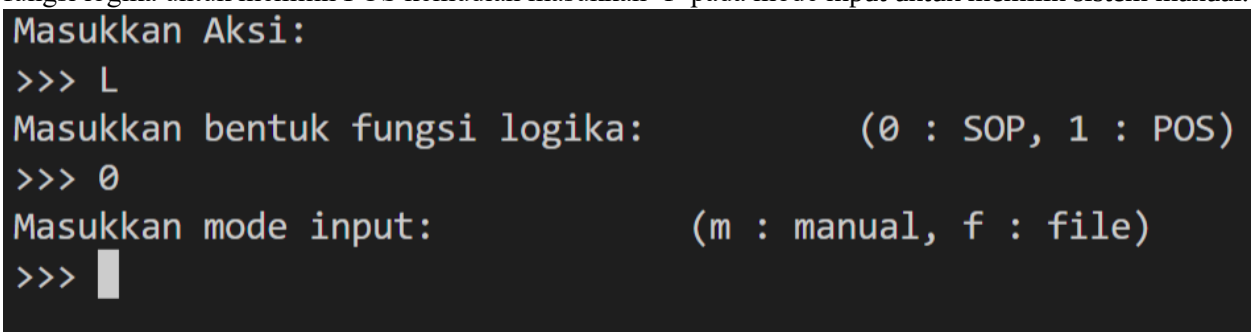
c.) Menggunakan Sistem Read File

File eksternal :

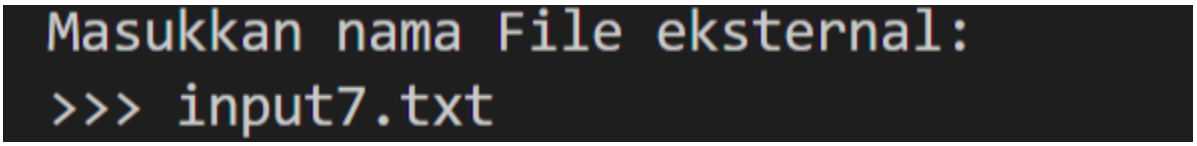


- Line 1 menyatakan banyaknya minterm
- Line 2 menyatakan elemen-elemen minterm
- Line 3 menyatakan ada atau tidaknya don’t care apabila 1 maka lanjut ke line selanjutnya
- Line 4 menyatakan jumlah don’t care
- Line 5 menyatakan elemen-elemen don’t care.

Langkah 1 : Ketik ‘L’ pada inputan Masukkan Aksi dan kemudian masukkan ‘0’ pada inputan bentuk fungsi logika untuk memilih POS kemudian masukkan ‘f’ pada mode input untuk memilih sistem manual.



Langkah 2 : Masukkan nama file sesuai contoh di atas.

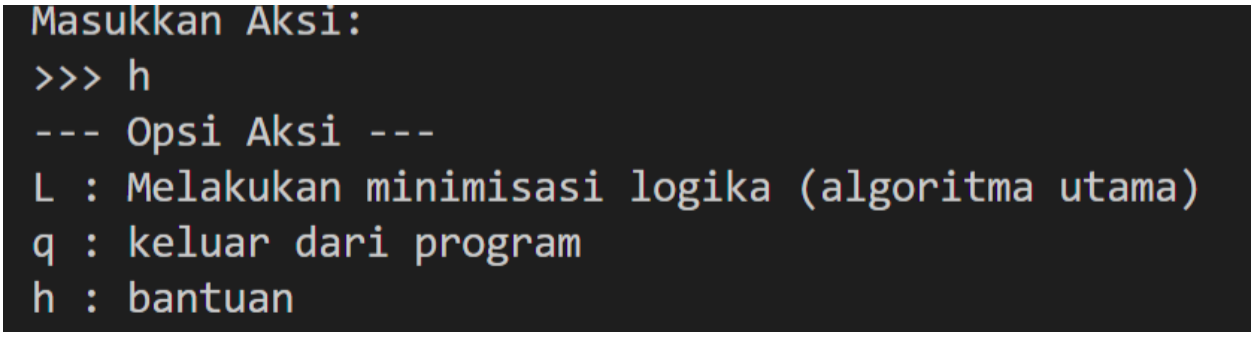


Langkah 3 : Setelah hasil keluar, input q untuk keluar dari program

```
Masukkan nama File eksternal:
>>> input7.txt
Iterasi ke-1:
5   0101
7   0111
13   1101
14   1110
11   1011
15   1111
Iterasi ke-2:
5,7   01-1
5,13   -101
7,15   -111
13,15   11-1
14,15   111-
11,15   1-11
Iterasi ke-3:
5,7,13,15   -1-1
5,13,7,15   -1-1
Tabel Prime Implicant:
ABC   14     15
ACD   15
BD   7     13     15
BD   7     13     15
Fungsi Logika setelah minimisasi:
        ABC + BD
```

```
Masukkan Aksi:
>>> q
```

For information : (input h pada aksi untuk menunjukkan opsi aksi)



Untuk pengujian program di atas, asumsi user memasukkan input yang valid.

Kekurangan Program

Kekurangan program yang ditemukan adalah ketika ingin mengganti jumlah variabel, nilai konstanta yang terdapat pada logicMinimization.h harus diganti terlebih dahulu

Lesson Learned

1. Pembuatan program tidak harus dimulai dari nol (re-inventing the wheel) melainkan bisa dilakukan dengan mengembangkan program yang telah tersedia di internet. Tetapi perlu dipastikan untuk menambahkan fitur dan tidak hanya mengganti nama variabelnya saja serta tidak lupa untuk mencantumkan referensi untuk menghindari plagiasi.

2. Penggunaan git/github untuk melakukan pengembangan program secara berkolaborasi sangatlah efektif. Dengan penggunaan git/github, histori dan detail terhadap perubahan yang dilakukan terhadap program dapat terlihat dengan jelas. Pada github secara spesifik juga terdapat fitur 'Issues' yang kami manfaatkan untuk pembagian tugas dan saling berbagi kendala. Pada github juga terdapat markdown yang kami manfaatkan untuk membuat dokumentasi/penjelasan terhadap metode yang kami gunakan.

3. Pengetahuan yang telah kita dapatkan dari mata kuliah yang dulu pernah diambil sangatlah bermanfaat sekarang dan akan terus terpakai. Dalam kasus ini, pengetahuan dari mata kuliah sistem digital (EL2002) terkait dengan fungsi logika dan minimisasinya menjadi tema utama dalam program ini. Selain itu, pengetahuan terkait skema-skema dasar pemrograman dari mata kuliah pengenalan komputasi (KU1102) dan dasar pemrograman (IF1210) juga sangat terpakai.

Kesimpulan

Akan dibandingkan hasil dari percobaan yaitu percobaan perhitungan secara eksak pada dasar teori dan percobaan pengujian program. Pada pengujian program bagian contoh 1 dengan tidak menggunakan don't care untuk sistem input manual dan baca file didapat hasil kedua fungsi logikanya yang sudah diminimisasi yaitu abD + AB + aCD. Hal tersebut sama dengan hasil perhitungan secara eksak pada bagian dasar teori contoh pertama untuk fungsi yang tidak menggunakan don't care. Kemudian untuk contoh 2 dengan menggunakan don't care untuk sistem input manual dan baca file didapat hasil kedua fungsi logikanya yang sudah diminimisasi yaitu ABC + BD. Hal tersebut sama dengan hasil perhitungan secara eksak pada bagian dasar teori contoh kedua untuk fungsi yang menggunakan don't care. Dapat disimpulkan hasil pengujian program berjalan dengan baik.

Pembagian Tugas dalam Kelompok

Anggota	Kontribusi
Jefferson Grizzlie (13220013)	1. Laporan - Membuat daftar isi
Muhammad Mikhail Ghrilman (13220021)	1. Laporan - Membuat pengujian program - Membuat kesimpulan 2. Presentasi - Membuat bahan fitur yang digunakan
Bostang Palaguna (13220055)	1. Source Code - Modularisasi program menjadi beberapa file - Mengerjakan interface - Fitur input dari file - Dokumentasi fungsi/prosedur, variabel 2. Laporan - Membuat rancangan flowchart - Membuat rancangan DFD
Muhammad Daffa Rasyid (13220059)	1. Laporan - Merapihkan format keseluruhan laporan 2. Presentasi - Membuat keseluruhan PPT di canva

Daftar Referensi

[1] Minterm and Maxterm - Boolean Algebra - DYclassroom | Have fun learning :-)

[2] Canonical and Standard Form - GeeksforGeeks

[3] McQuicksy.c | download free open source code (freesourcecode.net)