

**LAPORAN TUGAS BESAR**  
**EKSPLORASI ALGORITMA MINIMISASI LOGIC**  
**EL2008 - PEMECAHAN MASALAH DENGAN C**



**Disusun oleh:**

Kelompok	7
Atadila Belva Ganya	18320015
Diandra Ramadhani Putri Naja	18320023
Reina Puteri Ramadhani	18320039

**SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA**  
**INSTITUT TEKNOLOGI BANDUNG**  
**TAHUN AKADEMIK 2021/2022**

# DAFTAR ISI

<b>DAFTAR ISI</b>	<b>1</b>
<b>PENDAHULUAN</b>	<b>2</b>
<b>DASAR TEORI</b>	<b>3</b>
Sinyal Digital dan Biner	3
Minimisasi Rangkaian Digital Kombinasional	3
<b>ANALISIS ALGORITMA</b>	<b>5</b>
Struktur Data dan Variabel Global	8
tambahMinterm(int) Function	10
buatNode(int) Function	10
gabungMinterm() Function	11
print() Function	11
printTabel() Function	12
buatNodePair(node*, node*) Function	12
isiBiner(node*, node*, node*) Function	13
initTable() Function	14
ifPairingPossible(node*, node*) Function	14
ifMintermPresentInImplicant(int, int) Function	14
tambahPair(node*, node*) Function	15
tambahTabel() Function	15
analisisTabel() Function	15
konversiBiner(int) Function	16
cariMax(int*) Function	16
jumlahImplicants(int, int*) Function	16
hapusMinterm(int) Function	17
<b>TEST CASE</b>	<b>18</b>
<b>KESIMPULAN</b>	<b>24</b>
<b>PEMBAGIAN TUGAS</b>	<b>26</b>
<b>REFERENSI</b>	<b>27</b>

## PENDAHULUAN

Minimisasi *logic boolean algebra* merupakan salah satu metode yang digunakan untuk memperkecil ukuran sebuah rangkaian digital. Karena hubungannya yang saling berbanding lurus, penyederhanaan ekspresi fungsional dapat membuat ukuran rangkaian digital yang merepresentasikannya juga mengecil. Dengan begitu, akan didapat sistem digital dengan rangkaian yang lebih optimal, dalam hal kecepatan dan konsumsi *power*-nya.

Secara umum, terdapat dua buah metode yang dapat digunakan untuk melakukan minimisasi *logic boolean algebra*, yakni metode Karnaugh Map (K-map) dan Quine-McCluskey (metode tabular). Metode karnaugh map pertama kali diusulkan oleh Edward Veitch pada tahun 1952 dan kemudian dimodifikasi oleh Karnaugh. Metode ini memanfaatkan array dua dimensi sebagai representasi grafis dari *truth table* fungsi-fungsi *logic* yang setiap kotaknya mewakili minterm dalam bentuk *boolean*. Metode Quine-McCluskey (metode tabular) dikembangkan oleh W.V. Quine dan Edward J. McCluskey pada tahun 1956. Secara garis besar, metode tabular melibatkan dua buah tahapan inti, yakni menemukan seluruh *prime implicants* pada sebuah fungsi, kemudian memanfaatkannya untuk mencari *essential prime implicants* dalam fungsi terkait. *Prime implicants* merupakan sekelompok minterm yang direpresentasikan dalam bentuk *boolean aljabar* paling sederhana, sehingga didapat setidaknya satu *prime implicant* untuk setiap minterm. Sementara, *essential prime implicant* merupakan minterm yang hanya memiliki satu *prime implicants*. Kemudian, penjumlahan dari seluruh *essential prime implicant* yang didapat, merupakan *boolean expression* yang paling sederhana.

Terdapat keterbatasan dalam penggunaan metode karnaugh map, yakni dalam hal jumlah variabel yang dapat ditinjau pada ekspresi fungsional terkait dan implementasinya dalam sebuah program komputer. Metode karnaugh map membatasi penggunaannya dengan *input* minimal dua dan maksimal lima variabel untuk setiap fungsinya. Pengimplementasian metode karnaugh map dalam sebuah program komputer juga terbatas dikarenakan penggunaannya yang lebih banyak menerapkan kemampuan visual manusia untuk melakukan minimisasi *logic boolean algebra*. Menjawab keterbatasan-keterbatasan tersebut, metode tabular membolehkan pengguna untuk memberikan *input* jumlah variabel lebih dari lima, hingga tak terbatas. Selain itu, tahapan utama yang diterapkan pada metode tabular tidak banyak memanfaatkan kemampuan visual manusia, sehingga lebih mudah untuk diterapkan dalam sebuah program komputer. Dengan alasan-alasan tersebut, Kami memilih metode tabular sebagai metode yang akan diterapkan dalam program yang telah disusun.

# DASAR TEORI

## A. Sinyal Digital dan Biner

Sinyal digital atau sinyal diskrit merupakan sinyal yang pada waktu tertentu dapat merepresentasikan satu set berhingga dari beberapa kemungkinan value. Sistem digital merupakan sistem yang terdiri atas sirkuit digital yang menjadi koneksi antar komponen digital di dalamnya. Input yang diterima oleh sebuah sistem digital akan berupa input digital, sementara output yang dihasilkan akan berupa output digital. Input serta output digital dari sistem digital dapat diekspresikan melalui sinyal-sinyal dengan value biner (binary).

Biner (binary) merupakan representasi dari sinyal digital, yang terdiri atas satu dari dua kemungkinan value, yakni 1 (on) atau 0 (off). Sebuah sinyal biner tunggal dikenal dengan sebutan bit (binary digit). Angka biner merupakan angka yang menggunakan sistem angka basis dua dan dapat diilustrasikan sebagai berikut.

$$\begin{array}{cccc} 1 & 0 & 1 & \\ 2^3 & 2^2 & 2^1 & 2^0 \end{array}$$

Sebuah sirkuit digital (rangkaian digital) yang menyusun sistem digital hanya mengenal dua simbol, yakni "1" untuk "on" dan "0" untuk "off". Jika diinginkan representasi nilai lebih dari 1, maka akan dibutuhkan digit lainnya,  $2^1$  hingga  $2^n$ . Sebagai contoh, biner "10" pada sistem angka basis dua, disebut sebagai "satu nol, basis dua", bermakna 1 digit ber-value  $2^1$  dan 0 digit ber-value  $2^0$ . Biner "10" mengekspresikan nilai  $1 * 2^1 + 0 * 2^0 = 2$  pada sistem angka basis sepuluh. Contoh lainnya, untuk biner "101", dapat disebut sebagai "satu nol satu, basis dua", mengekspresikan nilai  $1 * 2^2 + 0 * 2^1 + 1 * 2^0 = 5$  pada sistem angka basis sepuluh.

## B. Minimisasi Rangkaian Digital Kombinasional

Dalam mengonstruksi rangkaian digital, optimalisasi ukuran, kecepatan, serta konsumsi *power* dapat dilakukan agar rangkaian dapat berjalan secara efektif dan benar. Salah satu cara untuk meningkatkan kecepatan dan konsumsi *power* adalah dengan mereduksi ukuran rangkaian. Ukuran rangkaian digital memiliki hubungan yang berbanding lurus dengan kompleksitas atau ukuran ekspresi fungsionalnya. Hal ini menjadi satu-satunya keterkaitan antara keduanya. Dengan memanfaatkan teorema aljabar boolean, sebuah ekspresi fungsional dapat disederhanakan. Dengan begitu, ukuran dari rangkaian digital yang direpresentasikan oleh ekspresi fungsional terkait dapat diperkecil.

Terdapat dua buah metode yang dapat diterapkan untuk menyederhanakan sebuah ekspresi fungsional *Boolean*, yakni Karnaugh map (K-map) dan Quine-McCluskey (metode tabular).

### a. Karnaugh Map (K-Map)

Karnaugh map merupakan metode pemanfaatan array dua dimensi sebagai representasi grafis dari *truth table* fungsi-fungsi *logic* yang setiap kotaknya mewakili

sebuah minterm dalam fungsi *Boolean*, yang kemudian minterm-minterm tersebut akan dikelompokkan. Metode ini pertama kali diusulkan oleh Edward Veitch pada tahun 1952 dan kemudian dimodifikasi oleh Karnaugh. Metode karnaugh map memiliki keterbatasan dalam hal jumlah variabel yang dapat ditinjau pada ekspresi fungsional terkait, yakni minimal dua variabel dan maksimal lima variabel. Dengan dimisalkan terdapat  $n$  buah variabel yang sedang ditinjau, sebuah K-map akan memiliki total  $2^n$  kotak yang menyimpan data-data minterm dalam fungsi *Boolean*. Tak hanya itu, metode karnaugh map memiliki keterbatasan lain dalam implementasinya yang terbilang tidak mudah untuk diterapkan pada sebuah program komputer.

b. Quine-McCluskey (Metode Tabular)

Menjawab keterbatasan-keterbatasan dari karnaugh map, Quine-McCluskey (metode tabular) hadir dengan syarat jumlah variabel dapat melebihi lima (tidak terbatas) dan lebih mudah untuk diimplementasikan dalam program komputer. Secara garis besar, metode tabular melibatkan dua buah tahapan inti, yakni menemukan seluruh *prime implicants* pada sebuah fungsi, kemudian memanfaatkannya untuk mencari *essential prime implicants* dalam fungsi terkait (fungsi-fungsi tersebut disebut minterm). Secara lebih terperinci, metode tabular memanfaatkan empat buah tahapan dalam melaksanakan minimisasi *logic* menggunakan metode tabular, yakni iterasi 1, iterasi 2, iterasi 3, dan pembuatan *tabel prime implicants*, *essential prime implicants*, serta hasil akhir. Penjelasan lebih lanjut terkait tahapan-tahapan tersebut akan dibahas pada bagian Analisis Algoritma.

## ANALISIS ALGORITMA

Program yang disusun dalam Laporan Tugas Besar EL2008 ini merupakan bentuk penyelesaian dari masalah minimisasi *logic boolean algebra*. Mula-mula, program akan meminta *input* berupa banyaknya minterm yang akan di minimisasi (dari 1 s.d. 16 berbentuk integer). Selanjutnya, program akan meminta bentuk desimal (dari 0 s.d. 15 berbentuk integer) dari minterm-minterm terkait sesuai jumlah yang telah di-*input*-kan. Berikut merupakan tabel representasi minterm dalam bentuk desimal dan biner yang dapat di-*input*-kan pada program.

**Tabel 2.1.** Tabel minterm

AB \ CD	00	01	10	11
00	0 abcd	4 aBcd	8 Abcd	12 ABcd
01	1 abcD	5 aBcD	9 AbcD	13 ABcD
10	2 abCd	6 aBCd	10 AbCd	14 ABCd
11	3 abCD	7 aBCD	11 AbCD	15 ABCD

Nilai desimal pada tabel di atas diperoleh dari konversi biner ke desimal. Variabel yang digunakan untuk melambangkan nilai 0 adalah huruf kecil, sedangkan penggunaan huruf kapital melambangkan nilai biner 1 pada minterm. Perlu dipahami bahwa setiap desimal pada program akan direpresentasikan dalam 4 bit, sehingga banyaknya minterm yang dapat di minimisasi maksimal 16 minterm dengan *range* desimal minterm dari 0 s.d. 15.

Program akan melakukan minimisasi *logic expression* menggunakan metode tabular. Dalam minimisasi *logic expression* menggunakan metode ini, terdapat beberapa tahap yang dilakukan, yaitu sebagai berikut :

a. Iterasi 1

Pada iterasi 1, mula-mula, dilakukan konversi desimal minterm menjadi binernya, sesuai dengan yang telah dituliskan pada Tabel 2.1. Kemudian, dari bentuk biner minterm tersebut, dilakukan pengurutan terhadap minterm-minterm yang ada, dari jumlah bit berdigit “1” terkecil hingga terbesar, dengan setiap minterm yang memiliki jumlah bit berdigit “1” yang sama, dikelompokkan.

b. Iterasi 2

Pada iterasi 2, dilakukan perbandingan antara setiap elemen (minterm) pada kelompok minterm yang memiliki jumlah bit berdigit “1” paling sedikit dengan setiap elemen pada kelompok minterm berikutnya (kelompok minterm yang memiliki jumlah bit berdigit “1”

paling sedikit kedua). Perbandingan antara dua elemen tersebut akan menghasilkan komparasi elemen dengan hanya satu perbedaan digit bit yang dimiliki. Hasil komparasi antara kedua kelompok minterm pertama akan membentuk *kelompok-satu-iterasi-dua*. Selanjutnya, proses perbandingan akan kembali dilakukan, antara kelompok minterm dengan jumlah bit berdigit “1” paling sedikit kedua dengan kelompok minterm berikutnya. Hal ini akan menghasilkan komparasi untuk *kelompok-dua-iterasi-dua*. Proses perbandingan akan kerap dilanjutkan hingga dihasilkan kelompok komparasi antara kelompok minterm yang memiliki jumlah bit berdigit “1” terbanyak dengan yang sebelumnya. Dalam pengaplikasian dan pendokumentasiannya pada tabel iterasi 2, dituliskan digit bit yang sama akan ditulis kembali, sementara digit bit yang berbeda akan dituliskan sebagai *underscore* ( \_ ) atau *strip* (–).

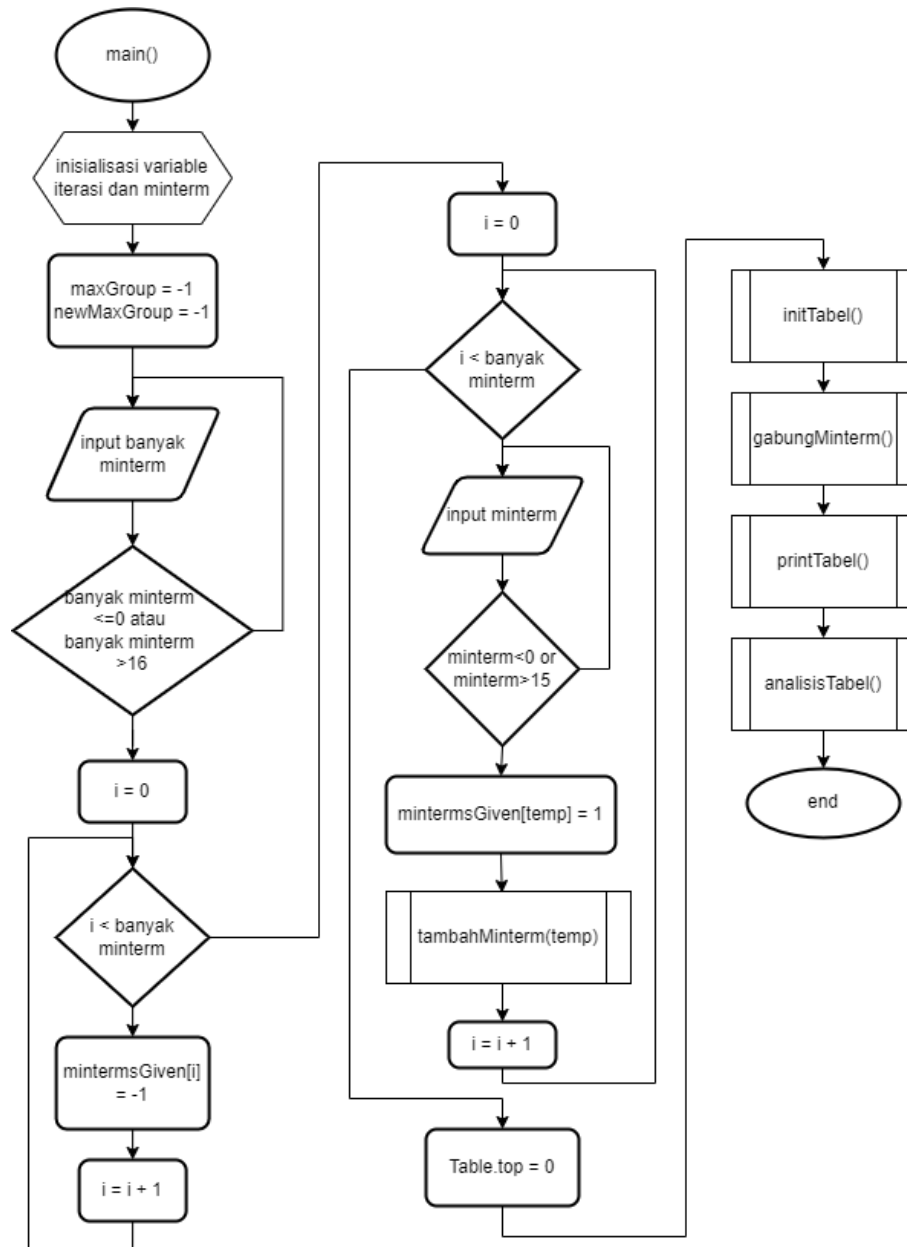
c. Iterasi 3

Proses iterasi 3 merupakan proses iterasi terakhir untuk fungsi bervariasi 4 (desimal yang terdiri atas 4 bit dalam konversi binernya). Pada iterasi 3, perbandingan antara setiap elemen pada *kelompok-satu-iterasi-dua* dengan setiap elemen pada *kelompok-dua-iterasi-dua* akan dilakukan. Sama seperti proses perbandingan pada iterasi 2, hasil komparasi yang didapat berupa sekelompok minterm dengan perbedaan pada salah satu digit bit-nya. Setelah proses perbandingan pertama selesai, setiap elemen pada *kelompok-dua-iterasi-dua* akan kembali dibandingkan dengan setiap elemen pada *kelompok-tiga-iterasi-dua*. Proses perbandingan ini akan kerap dilakukan hingga setiap elemen pada kelompok terakhir iterasi dua dibandingkan dengan setiap elemen pada kelompok sebelumnya. Jika didapati adanya perbedaan pada dua digit bit atau lebih, maka kedua *kelompok-minterm-iterasi-dua* tersebut tidak akan memenuhi dan didokumentasikan pada tabel iterasi 3. Jika syarat proses perbandingan dipenuhi, maka dilakukan dokumentasi terhadap elemen-elemen (minterm) yang terlibat dalam proses komparasi tersebut, dengan menuliskan kembali seluruh digit bit yang sama dan menuliskan digit bit yang berbeda dengan *underscore* ( \_ ) atau *strip* (–).

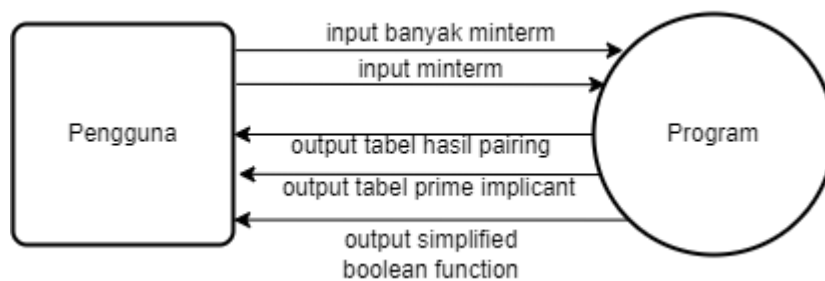
d. Tabel *Prime Implicants*, *Essential Prime Implicants*, dan Hasil Akhir

Tabel *Prime Implicants* didapat dengan melakukan *listing* terhadap fungsi-fungsi aljabar paling sederhana dari setiap minterm yang sebelumnya telah di-*input*-kan. Sebuah tabel *prime implicants* dapat dituliskan seperti pada Tabel 3.3, 3.6, dan 3.10. Baris pertama akan merepresentasikan minterm-minterm yang pada awal program di-*input*-kan, secara terurut mulai dari yang terkecil hingga terbesar (dalam desimal). Sementara, kolom pertama akan merepresentasikan bentuk aljabar dari digit-digit bit sekelompok minterm yang paling sederhana (*prime implicants*), baik hasil dari proses iterasi 1, 2, ataupun 3. Pengurutannya dilakukan dengan memposisikan hasil komparasi terkecil hingga terbesar dari proses iterasi 3, 2, dan 1 pada baris teratas hingga terakhir. Selanjutnya, dilakukan penandaan (*checklist*) terhadap kotak dengan posisi kolom (minterm) dan baris (*prime implicants*) yang saling berkaitan. Proses penandaan akan dilakukan hingga baris terakhir pada tabel *prime implicants*. Kemudian, dengan melakukan pengamatan terhadap ada atau tidaknya lebih dari satu *checklist* pada setiap kolom (minterm), *essential prime implicants* dapat ditentukan. Jika dikaitkan dengan definisinya yang telah dijabarkan pada bagian Pendahuluan, pada tabel *prime implicants*, *essential prime implicants* direpresentasikan oleh minterms yang hanya memiliki satu buah *checklist* pada kolomnya. Hal ini akan membuat bentuk aljabar pada kolom pertama (*prime implicants*) dengan minterm *essential prime implicants*, menjadi jawaban atau hasil akhir penyederhanaan *logic boolean algebra* menggunakan metode tabular.

Dari tahapan penyelesaian menggunakan Tabular Method, implementasi program utama yang dibuat adalah sebagai berikut :

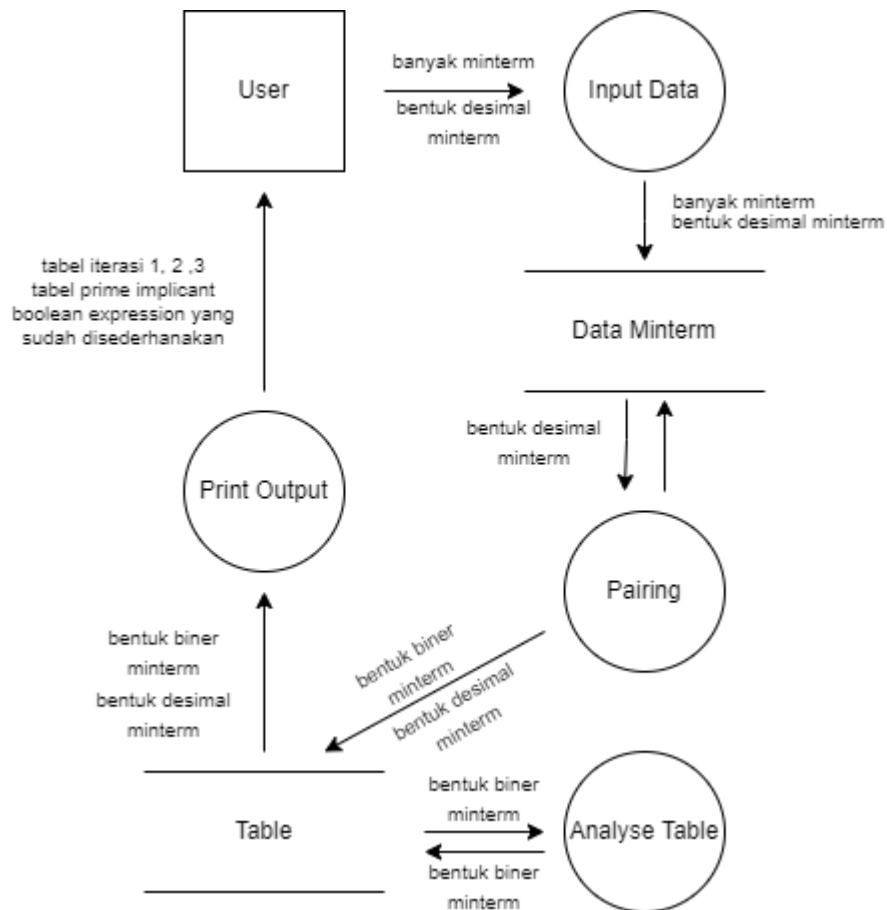


**Gambar 2.1.** Flowchart main function



**Gambar 2.2.** DFD level 0





**Gambar 2.3.** DFD level 1

Pada main function, program akan melakukan inisialisasi untuk beberapa variabel. Kemudian program akan meminta input berupa banyaknya minterm yang akan di minimisasi. Input yang diberikan pengguna harus berjumlah lebih dari 0 dan kurang dari sama dengan 16. Apabila banyak minterm yang dimasukkan tidak di rentang 0 - 16, program akan melakukan looping dengan menampilkan pesan kesalahan dan meminta pengguna untuk memasukkan ulang banyak minterm. Selanjutnya, setelah input banyak minterm yang dimasukkan pengguna benar, program akan meminta minterm yang akan di minimisasi sesuai dengan jumlah yang telah diberikan sebelumnya. Pada input ini, pengguna diminta untuk memasukkan input minterm dari rentang 0 - 15. Apabila pengguna memberikan angka yang tidak sesuai dengan batasan input, program akan kembali menampilkan pesan kesalahan dan meminta pengguna memasukkan ulang minterm yang diinginkan.

Setiap minterm akan dibuat linked list baru dengan struktur data Node untuk menyimpan informasi dari setiap minterm menggunakan fungsi tambahMinterm(). Kemudian, setelah input berhasil dilakukan, program akan memanggil fungsi initTable() untuk menginisialisasi tabel prime implicant dengan mengisi setiap sel dalam tabel -1 untuk menandakan sel tersebut kosong. Berikutnya akan dipanggil fungsi gabungMinterm() untuk melakukan pengelompokkan. Lalu tabel prime implicant akan ditampilkan pada output dengan memanggil fungsi printTabel(). Terakhir, tabel prime implicant akan dianalisis menggunakan fungsi analisisTabel() dan menampilkan hasil akhir berupa minterm yang telah disederhanakan.

#### **A. Struktur Data dan Variabel Global**

- struct vector

Struktur data vektor digunakan untuk menyimpan list minterm yang berpasangan. Variabel dalam struktur data ini adalah :

- int paired[limit]
- struct Node

Struktur data Node digunakan untuk menyimpan informasi dari minterm seperti pasangan minterm. Variabel dalam struktur data ini adalah :

**Tabel 2.2.** Variabel dalam struktur data Node

struct Node* next	struktur data yang menghubungkan ke node berikutnya pada linked list
int hasPaired	integer yang menyimpan nilai 1 atau 0 untuk menunjukkan tahapan pengelompokkan minterm
int numberOfOnes	integer yang menyimpan banyaknya nilai biner 1 pada minterm
struct vector paired	struktur data yang menunjukkan pasangan minterm lain
int group	integer yang menunjukkan kelompok dari minterm sesuai dengan banyaknya biner 1 pada minterm
int binary[bitsSize]	array yang menyimpan nilai biner hingga terbentuk pasangan
int numberOfPairs	integer yang menyimpan banyak pasangan yang terbentuk

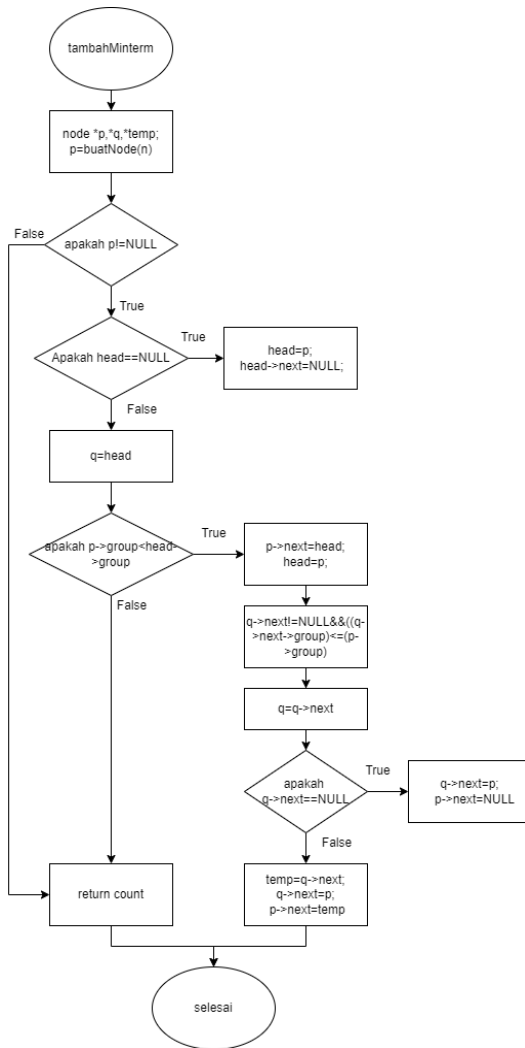
- struct implicantsTable

Struktur data implicantsTable digunakan untuk menyimpan tabel prime implicant. Variabel dalam struktur data ini adalah :

**Tabel 2.3.** Variabel dalam struktur data implicantsTable

int arr[limit][bitsSize]	tabel dalam setiap iterasi
int brr[limit][limit]	tabel untuk prime implicant
int top	menunjukkan banyak prime implicant yang ada
int mintermCounter[limit]	menyimpan nilai minterm pada prime implicant tertentu

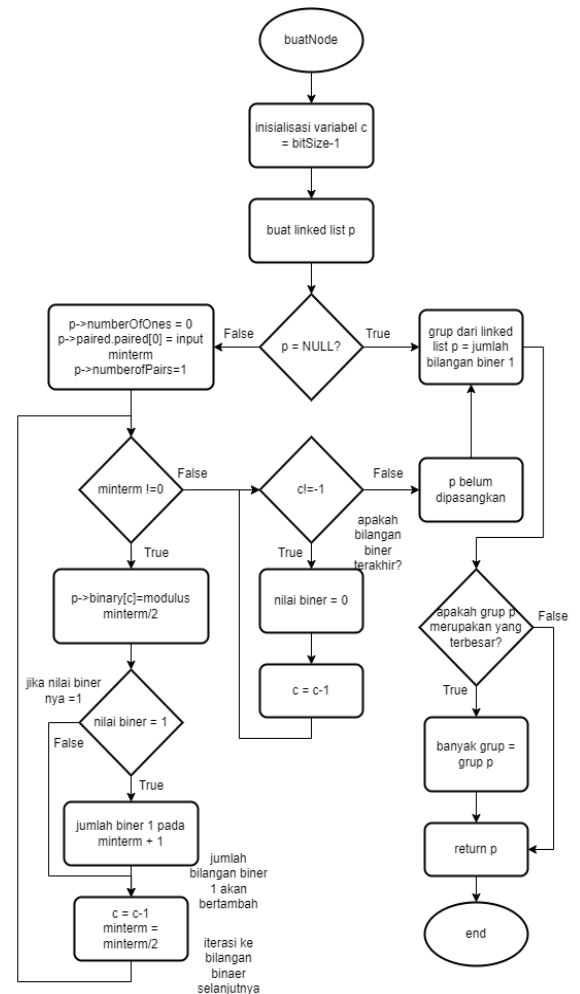
## B. tambahMinterm(int) Function



**Gambar 2.4.** Flowchart fungsi tambahMinterm(int)

Fungsi ini digunakan untuk membuat linked list baru untuk menyimpan minterms yang diberikan. Struktur data yang digunakan adalah Node, yang akan menyimpan beberapa informasi mengenai minterm tersebut. Informasi yang tersimpan dalam setiap minterm dapat dilihat pada bagian sebelumnya terkait struktur data Node. Dalam membuat node baru fungsi ini akan memanggil fungsi buatNode() dan kemudian akan ditambahkan beberapa informasi ke dalam node tersebut. Parameter dari fungsi ini adalah sebuah integer yang merupakan minterm dalam angka desimal 0 - 15 yang diberikan sebelumnya oleh pengguna pada bagian main function.

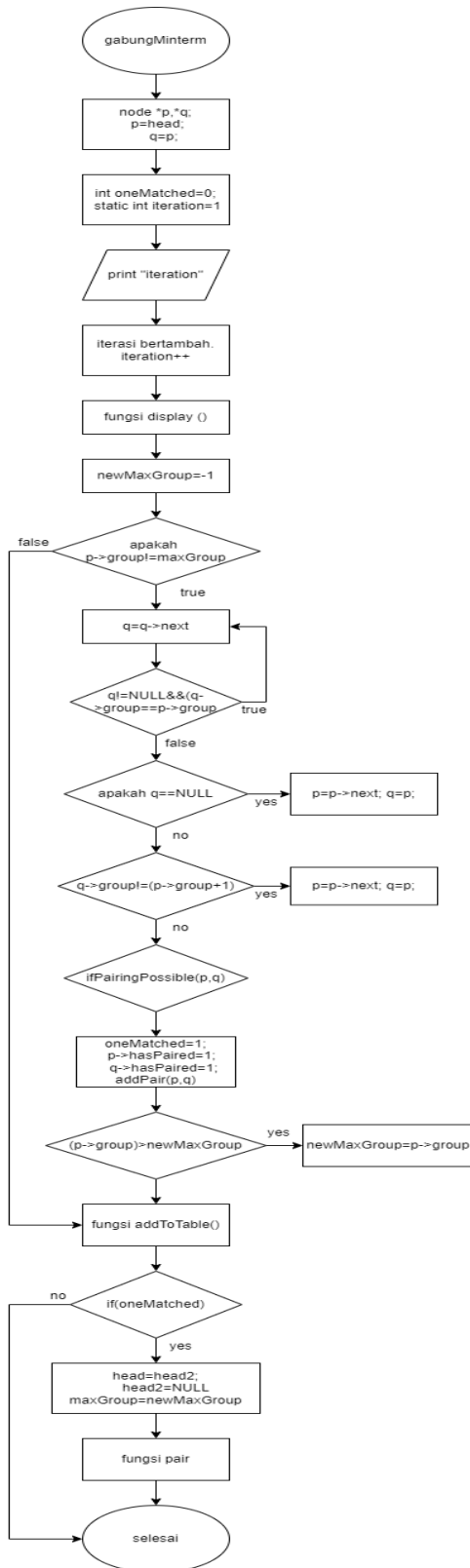
## C. buatNode(int) Function



**Gambar 2.5.** Flowchart fungsi buatNode(int)

Fungsi ini digunakan untuk membuat linked list baru pada setiap minterm untuk menyimpan informasi dari minterm tersebut. Dalam fungsi ini, beberapa informasi pada minterm yang ditambahkan adalah banyaknya biner 1 dalam minterm, menentukan array biner pada minterm, pengelompokkan minterm, dan kelompok minterm. Parameter yang digunakan dalam fungsi ini adalah int yang merupakan minterm yang diberikan. Kemudian fungsi ini akan me-return atau mengembalikan sebuah linked list baru untuk minterm tersebut.

#### D. gabungMinterm() Function

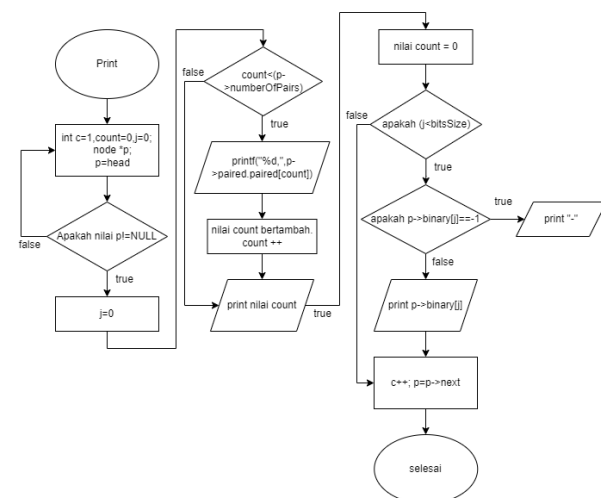


**Gambar 2.6.** Flowchart fungsi gabungMinterm()

Fungsi ini digunakan untuk menampilkan minterm dan minterm pasangannya serta nilai biner dalam setiap iterasi yang telah dilakukan.

Fungsi ini adalah fungsi yang digunakan untuk melakukan pairing atau pengelompokkan pada minterm. Fungsi ini melakukan pairing atau pengelompokkan beberapa kali sampai proses pengelompokkan tidak lagi dapat dilakukan. Fungsi ini akan memanggil fungsi tambahPair() untuk membuat node baru yang merupakan gabungan dari kedua node sebelumnya, kemudian akan memanggil fungsi tambahTabel() untuk menambahkan informasi baru ke dalam tabel. Selanjutnya fungsi akan mengecek apakah terdapat setidaknya satu pasangan yang telah dibuat. Apabila terdapat setidaknya satu pasangan minterm dibuat, fungsi gabungMinterm() akan melakukan rekursi. Parameter dari fungsi ini adalah dua node yang akan digabungkan.

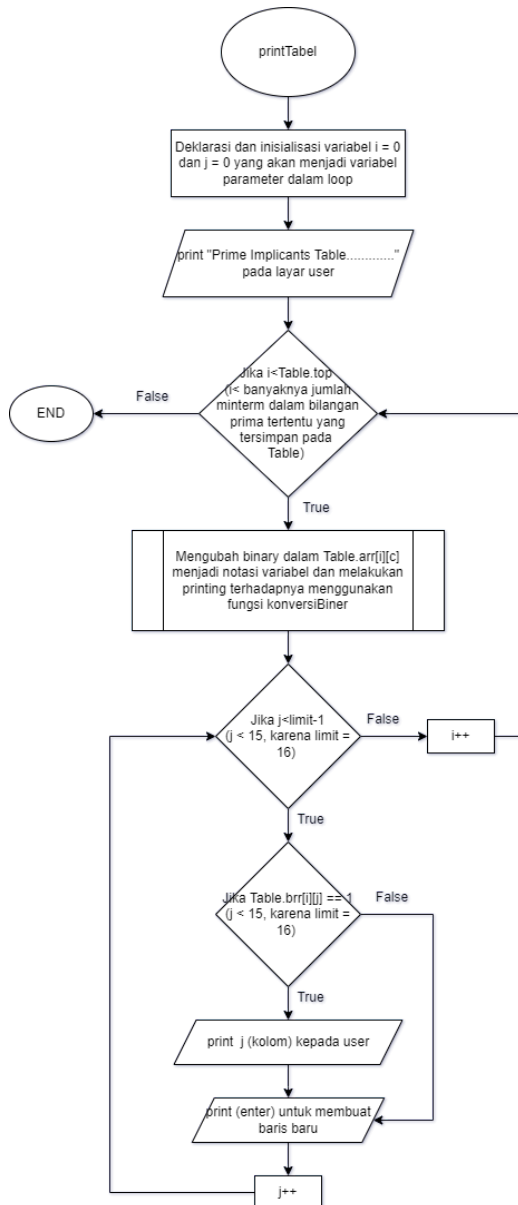
#### E. print() Function



**Gambar 2.7.** Flowchart fungsi print()

Fungsi digunakan untuk melakukan *printing* terhadap minterm dan pasangannya, serta nilai-nilai biner dalam setiap iterasi yang dilakukan. Pada fungsi ini, tidak dilakukan pemanggilan terhadap fungsi-fungsi lain di luar main() program.

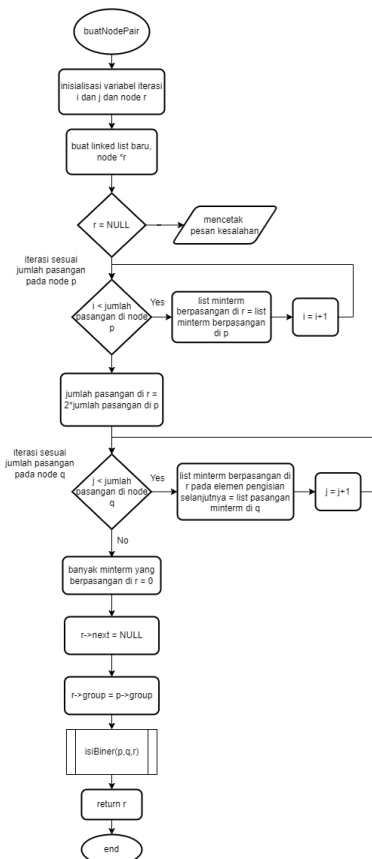
## F. printTabel() Function



**Gambar 2.8.** Flowchart fungsi printTabel()

Fungsi ini digunakan untuk menampilkan tabel prime implicant yang didapat dari informasi yang terdapat pada struktur data Table. Pada fungsi ini terdapat pemanggilan terhadap fungsi konversiBiner yang akan mengubah nilai biner ke bentuk notasi minterm.

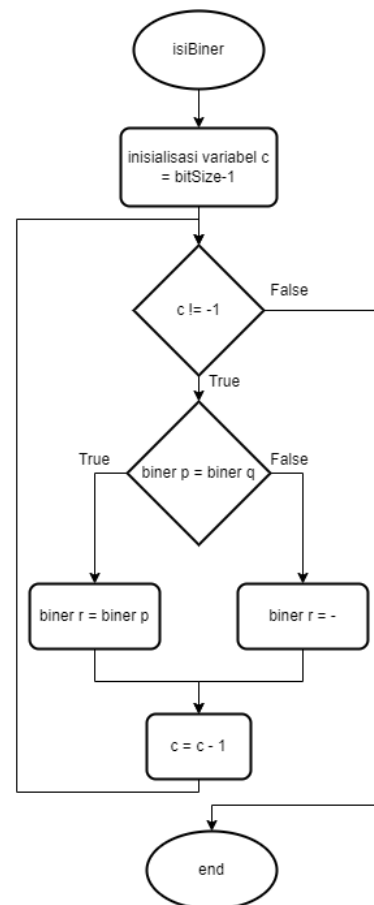
## G. buatNodePair(node\*, node\*) Function



**Gambar 2.9.** Flowchart Fungsi nodePair(node\*, node\*)

Fungsi ini digunakan untuk membuat linked list baru hasil pengelompokkan dua minterm sesuai dengan linked list yang diberikan pada parameter fungsi. Pada kedua minterm yang berpasangan akan digabungkan membentuk linked list baru hasil pairing keduanya. Informasi dari linked list minterm sebelumnya akan disalin ke dalam linked list yang baru. Pada fungsi ini akan dilakukan pemanggilan fungsi isiBiner untuk menyimpan nilai biner baru dalam linked list tersebut. Sesuai yang disebutkan sebelumnya,

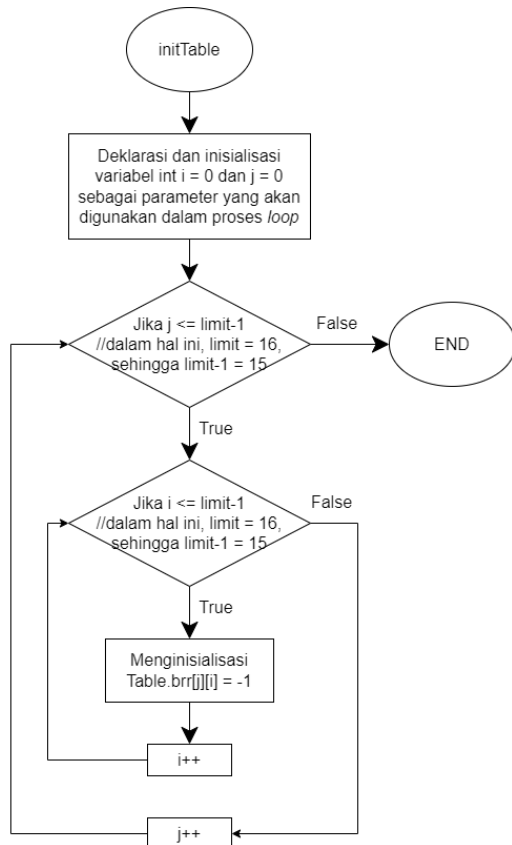
**H. isiBiner(node\*, node\*, node\*)**  
*Function*



**Gambar 2.10.** Flowchart fungsi isiBine

Fungsi ini digunakan untuk mengisi informasi nilai biner pada linked list baru hasil gabungan dari dua linked list minterm sebelumnya. Jika dalam kedua linked list yang digabungkan memiliki biner sama pada bits yang sama, maka akan menyimpan nilai sesuai dengan nilai sebelumnya. Namun jika nilai di bits yang sama dalam kedua linked list berbeda, maka akan menyimpan nilai -1. Parameter yang digunakan dalam fungsi ini adalah tiga linked list, dengan dua linked list merupakan linked list yang akan digabungkan dan satu linked list yang merupakan hasil penggabungan keduanya.

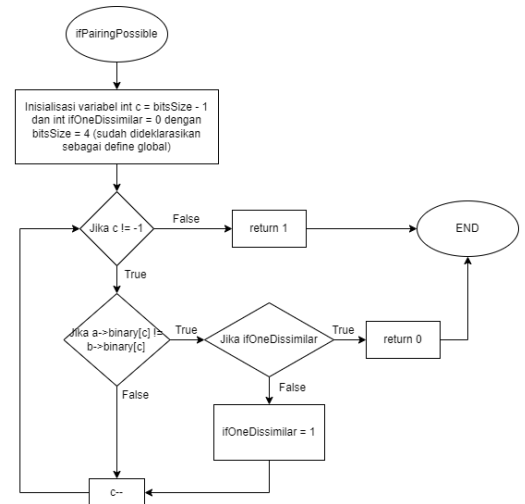
#### I. **initTable() Function**



**Gambar 2.11.** Flowchart fungsi initTable()

Fungsi ini digunakan untuk menginisialisasi struktur data implicantsTable pada Table.brr dengan menyimpan seluruh nilai awal dengan nilai -1 untuk menandakan bahwa sel pada tabel tersebut masih kosong.

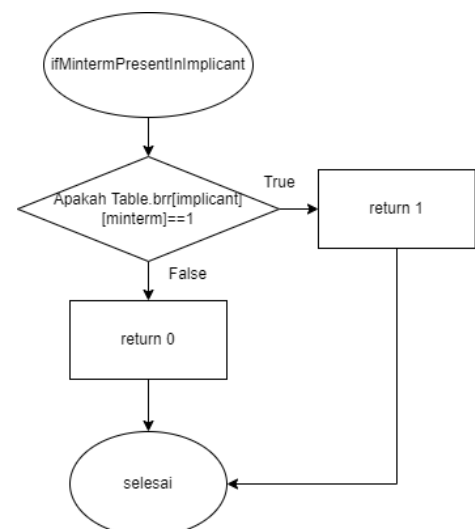
#### J. **ifPairingPossible(node\*, node\*) Function**



**Gambar 2.12.** Flowchart fungsi ifPairingPossible(node\*,node\*)

Fungsi ini digunakan untuk mengecek apakah terdapat perubahan atau perbedaan satu bit. Hal ini digunakan untuk menunjukkan apakah pairing atau pengelompokkan mungkin dilakukan dengan melihat adanya perbedaan untuk satu bit pada informasi nilai biner yang ada pada minterm. Apabila terdapat satu perbedaan, maka pairing atau pengelompokkannya dapat dilakukan. Parameter yang digunakan dalam fungsi ini adalah dua linked list dari minterm yang akan dicek apakah mungkin untuk dipasangkan.

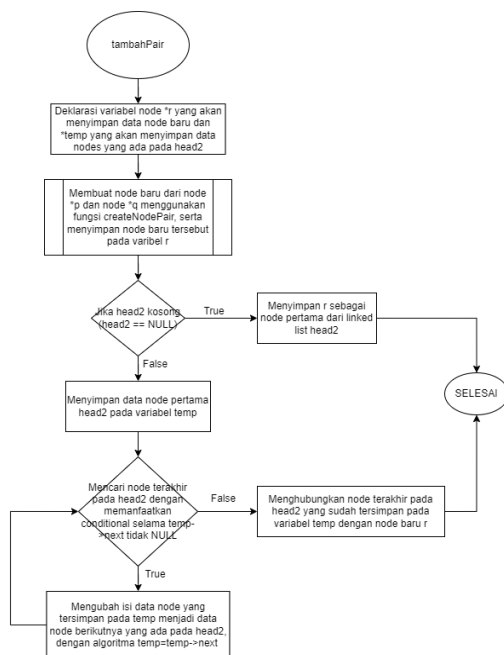
#### K. **ifMintermPresentInImplicant(int, int) Function**



**Gambar 2.13.** Flowchart fungsi `ifMintermPresentInImplicant(int, int)`

Fungsi ini akan melakukan pengecekan apakah minterm tertentu ada dalam di suatu baris implicant. Apabila minterm tersebut ada dalam baris implicant, fungsi akan me-return 1, lainnya, fungsi akan me-return 0. Parameter yang digunakan dalam fungsi ini adalah dua integer yang menunjukkan minterm dan baris implicant.

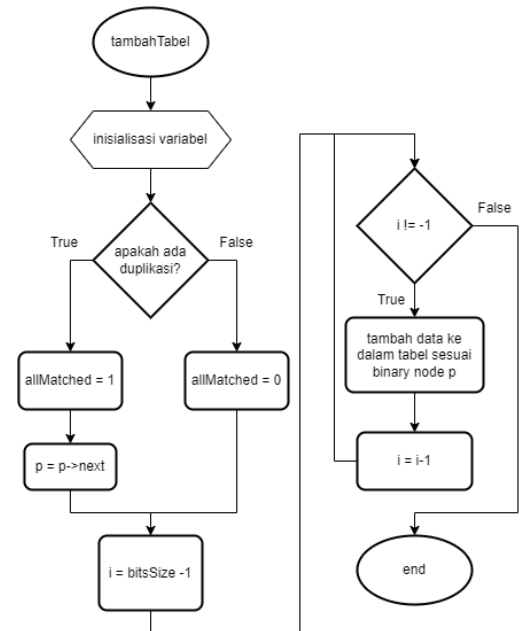
#### L. `tambahPair(node*, node*) Function`



**Gambar 2.14.** Flowchart fungsi `tambahPair(node*,node*)`

Fungsi ini digunakan untuk membuat linked list. Linked list yang dibuat akan digunakan untuk menyimpan minterms yang berpasangan. Parameter dari fungsi ini sendiri adalah dua linked list yang akan digabungkan ke dalam satu linked list baru. Dalam fungsi ini akan dilakukan pemanggilan fungsi `buatNodePair`.

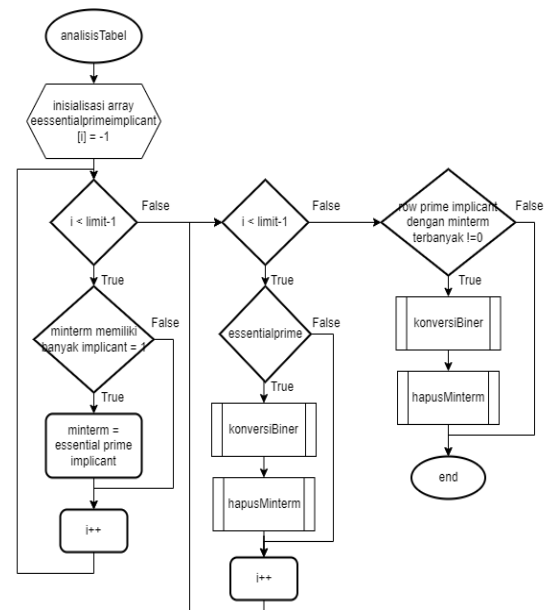
#### M. `tambahTabel() Function`



**Gambar 2.15.** Flowchart fungsi `tambahTabel()`

Fungsi ini akan menambahkan informasi yang diperoleh setelah proses iterasi dan pengelompokkan yang dilakukan pada minterm yang diberikan. Selanjutnya informasi yang diperoleh setelah tahapan tersebut akan dimasukkan ke dalam struktur data Table.

#### N. `analisisTabel() Function`



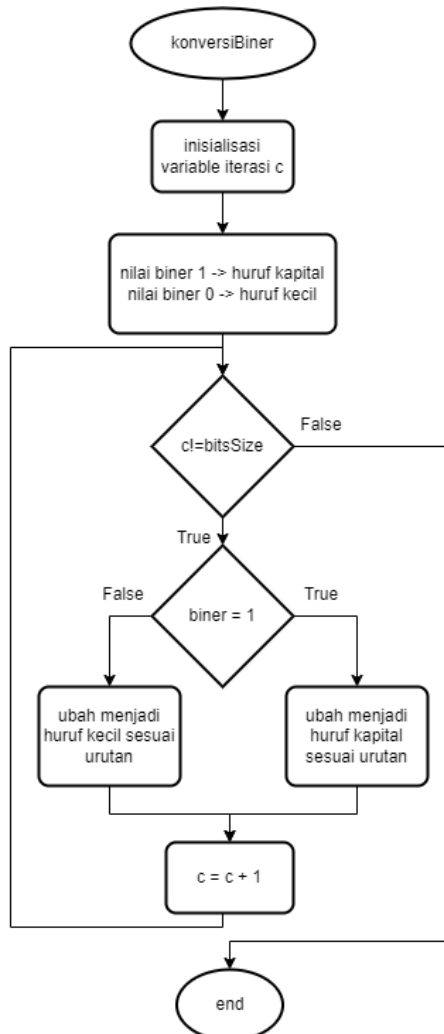
**Gambar 2.16.** Flowchart fungsi `analisisTabel()`

Fungsi ini digunakan untuk menganalisa tabel prime implicants untuk memperoleh essential prime



implicant yang kemudian akan menjadi bentuk sederhana dari boolean function yang diberikan oleh user. Essential prime implicant merupakan minterm yang hanya memiliki satu prime implicant.

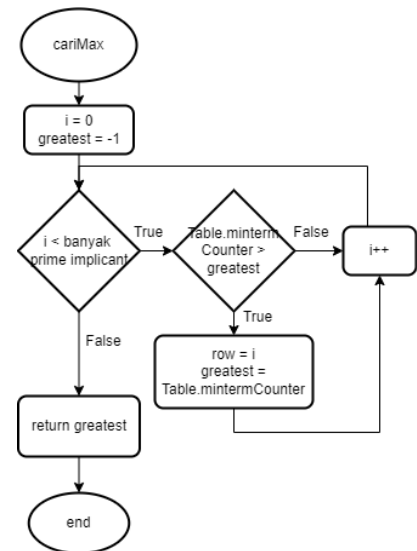
#### O. konversiBiner(int) Function



**Gambar 2.17.** Flowchart fungsi konversiBiner(int)

Fungsi ini digunakan untuk mengkonversi nilai biner ke dalam notasi variabelnya, kemudian menampilkannya sebagai output dari program. Variabel yang digunakan berurut dari  $2^0 - 2^1 - 2^2 - 2^3$  adalah A-B-C-D. Huruf kapital menandakan nilai biner 1 dan huruf kecil menandakan nilai biner 0. Parameter yang digunakan adalah sebuah integer. Fungsi ini digunakan dalam fungsi analisisTabel().

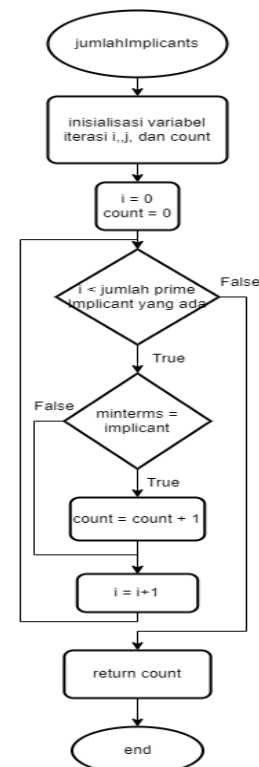
#### P. cariMax(int\*) Function



**Gambar 2.18.** Flowchart fungsi cariMax(int\*)

Fungsi ini digunakan untuk mencari prime implicant yang memiliki nilai minterm tertinggi yang tidak digunakan saat itu dan mengembalikan nilainya. Parameter yang digunakan adalah sebuah integer yang menandakan suatu baris dari prime implicant tertentu.

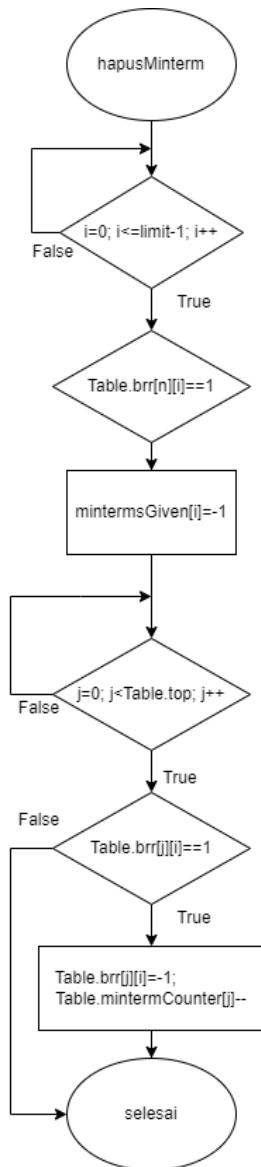
#### Q. jumlahImplicants(int, int\*) Function



**Gambar 2.19.** Flowchart fungsi jumlahImplicants(int, int\*)

Fungsi ini digunakan untuk menghitung banyaknya implicant yang ada di minterm tertentu.

#### R. hapusMinterm(int) *Function*



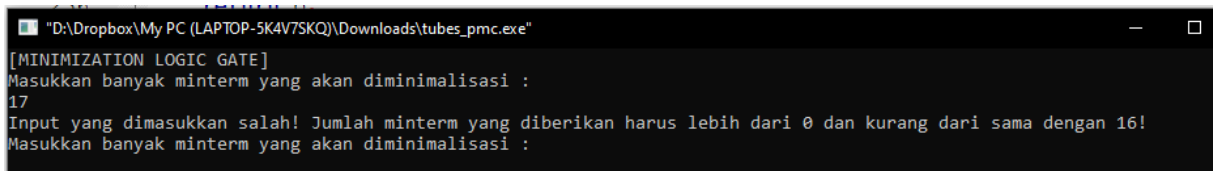
**Gambar 2.20.** Flowchart fungsi hapusMinterm(int)

Fungsi ini digunakan untuk menghapus minterm yang telah ditambahkan menjadi essential prime implicant dan kemudian dikonversi kedalam bentuk notasi variabel boolean dan dikeluarkan menjadi output hasil penyederhanaan.

## TEST CASE

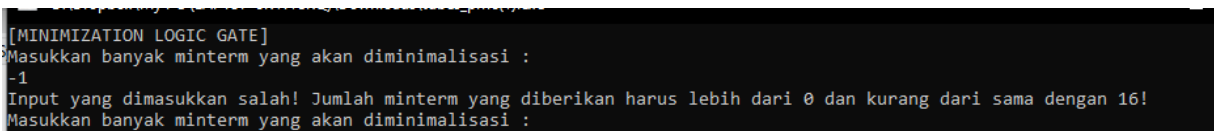
1. Nilai input minterm akan salah jika dimasukkan angka bukan diantara 1-16 dan akan lanjut ke input minterm yang diminimalisasi jika benar

Jika nilai input banyak minterm yang dimasukkan salah, maka akan ditampilkan keluaran/output “input yang dimasukkan salah! jumlah minterm yang diberikan harus lebih dari nol dan kurang dari sama dengan 16”. Selanjutnya, akan dilakukan iterasi memasukkan ulang banyak minterm yang akan diminimalisasi



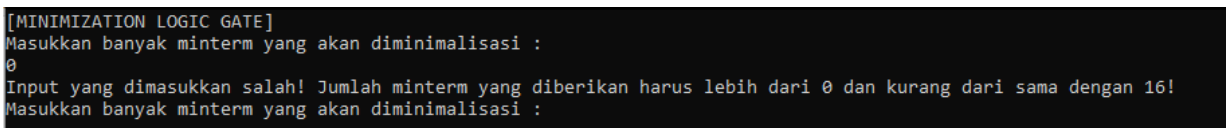
```
"D:\Dropbox\My PC (LAPTOP-5K4V7SKQ)\Downloads\tubes_pmc.exe"
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
17
Input yang dimasukkan salah! Jumlah minterm yang diberikan harus lebih dari 0 dan kurang dari sama dengan 16!
Masukkan banyak minterm yang akan diminimalisasi :
```

**Gambar 3.1** Test case 1, nilai input banyak minterm  $> 16$



```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
-1
Input yang dimasukkan salah! Jumlah minterm yang diberikan harus lebih dari 0 dan kurang dari sama dengan 16!
Masukkan banyak minterm yang akan diminimalisasi :
```

**Gambar 3.2** Test case 2, nilai input banyak minterm  $< 0$



```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
0
Input yang dimasukkan salah! Jumlah minterm yang diberikan harus lebih dari 0 dan kurang dari sama dengan 16!
Masukkan banyak minterm yang akan diminimalisasi :
```

**Gambar 3.3** Test case 3, nilai input banyak minterm  $= 0$

jika nilai input minterm yang dimasukkan benar, maka selanjutnya akan diminta masukkan minterm yang akan diminimalisasi



```
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
4
Masukkan minterm yang akan diminimalisasi :
```

**Gambar 3.4** Test case 4, nilai input minterm diantara 1-16

2. Nilai input minterm yang akan diminimalisasi akan salah jika dimasukkan angka bukan di antara 0-15

Jika nilai input minterm yang akan diminimalisasi  $< 0$  atau  $> 15$  maka akan menampilkan output yang memberitahu bahwa input yang diberikan salah, lalu akan dilakukan iterasi memasukkan ulang nilai input minterm yang akan diminimalisasi hingga benar dan jumlahnya sesuai dengan input banyak minterm yang akan diminimalisasi. Nilai input yang benar akan disimpan.

```

"D:\Dropbox\My PC (LAPTOP-5K4V7SKQ)\Downloads\tubes_pmc(1).exe"
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
4
Masukkan minterm yang akan diminimalisasi :
1
-1
Input yang diberikan salah, masukkan ulang :
2
14
17
Input yang diberikan salah, masukkan ulang :
3
[Iterasi - 1]
1 0001
2 0010
3 0011
14 1110
[Iterasi - 2]
1,3 00-1
2,3 001-
[Tabel Prime Implicants]
ABCD 14
abD 1 3
abC 2 3
Boolean Expression telah disederhanakan menjadi : abD + abC + ABCd
Process returned 1 (0x1) execution time : 33.708 s
Press any key to continue.

```

**Gambar 3.5** Test case 5, nilai input desimal tidak sesuai ketentuan (0 s.d. 15)

**3. Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan diminimalisasi benar, dengan total 2 kali iterasi**

Akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, tabel prime implicants, dan *boolean expression* yang sudah disederhanakan.

```

"D:\Dropbox\My PC (LAPTOP-5K4V7SKQ)\Downloads\tubes_pmc(1).exe"
[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
4
Masukkan minterm yang akan diminimalisasi :
1
2
4
6
[Iterasi - 1]
1 0001
2 0010
4 0100
6 0110
[Iterasi - 2]
2,6 0-10
4,6 01-0
[Tabel Prime Implicants]
abcD 1
aCd 2 6
aBd 4 6
Boolean Expression telah disederhanakan menjadi : abcD + aCd + aBd
Process returned 1 (0x1) execution time : 9.472 s
Press any key to continue.

```

**Gambar 3.6** Test case 6, dilakukan 2 kali iterasi hingga didapat *boolean expression* yang sudah disederhanakan

**4. Nilai input banyak minterm yang akan di minimalisasi dan input minterm yang akan diminimalisasi benar, dengan total 3 kali iterasi**

Akan ditampilkan keluaran/output berupa tabel iterasi-1, tabel iterasi-2, iterasi-3, tabel prime implicants, dan *boolean expression* yang sudah disederhanakan.

```

[MINIMIZATION LOGIC GATE]
Masukkan banyak minterm yang akan diminimalisasi :
9
Masukkan minterm yang akan diminimalisasi :
0
5
6
9
10
7
13
14
15
[Iterasi - 1]
0 0000
5 0101
6 0110
9 1001
10 1010
7 0111
13 1101
14 1110
15 1111

[Iterasi - 2]
5,7 01-1
5,13 -101
6,7 011-
6,14 -110
9,13 1-01
10,14 1-10
7,15 -111
13,15 11-1
14,15 111-

[Iterasi - 3]
5,7,13,15 -1-1
5,13,7,15 -1-1
6,7,14,15 -11-
6,14,7,15 -11-

[Tabel Prime Implicants]
abcd 0
AcD 9 13
ACd 10 14
BD 5 7 13 15
BC 6 7 14 15
Boolean Expression telah disederhanakan menjadi : abcd + BD + BC + AcD + ACd

```

**Gambar 3.7** Test case 7, dilakukan 3 kali iterasi hingga didapat *boolean expression* yang sudah disederhanakan

Sebagai pembuktian dari beberapa *test case* di atas, kami telah menyusun beberapa penyelesaian secara manual. Berikut adalah hasil dari penyelesaian manual yang telah kami susun:

1. Penyelesaian manual untuk *test case 5*

Dengan input desimal minterm adalah 1, 2, 14, 3.

a. Iterasi 1

**Tabel 3.1** Iterasi 1 untuk *test case 5* dengan penyelesaian metode tabular secara manual

Group	Minterm	Biner			
G0	1	0	0	0	1
	2	0	0	1	0
G1	3	0	0	1	1
G2	14	1	1	1	0

b. Iterasi 2

**Tabel 3.2** Iterasi 2 untuk *test case 5* dengan penyelesaian metode tabular secara manual

Group	Minterm	Biner			
G0	1, 3	0	0	—	1
	2, 3	0	0	1	—

c. Iterasi 3

Tidak ada kelompok lain pada tabel iterasi 2 selain G0, sehingga elemen-elemennya tidak dapat dibandingkan dengan kelompok hasil iterasi 2 lainnya. Maka, tabel iterasi 3 untuk kasus ini, kosong (tidak perlu dilakukan).

d. Tabel *prime implicants*

**Tabel 3.3** Tabel *prime implicants* untuk *test case 5* dengan penyelesaian metode tabular secara manual

<i>Prime Implicants</i> \ Minterm	1	2	3	14
abD	✓		✓	
abC		✓	✓	
ABCd				✓

Pada tabel tersebut, akan didapat:

- *prime implicants* : abD, abC, ABCd
- *essential prime implicants* : abD, abC, ABCd

Dari tahapan serta tabel di atas, *boolean expression* yang sudah disederhanakan untuk *test case 5* adalah  $abD + abC + ABCd$ , sesuai dengan *boolean expression* yang didapat dari program.

## 2. Penyelesaian manual untuk *test case 6*

Dengan input desimal minterm adalah 1, 2, 4, 6.

### a. Iterasi 1

**Tabel 3.4** Iterasi 1 untuk *test case 6* dengan penyelesaian metode tabular secara manual

Group	Minterm	Biner			
G0	1	0	0	0	1
	2	0	0	1	0
	4	0	1	0	0
G1	6	0	1	1	0

### b. Iterasi 2

**Tabel 3.5** Iterasi 2 untuk *test case 6* dengan penyelesaian metode tabular secara manual

Group	Minterm	Biner			
G0	2, 6	0	–	1	0
	4, 6	0	1	–	0

### c. Iterasi 3

Tidak ada kelompok lain pada tabel iterasi 2 selain G0, sehingga elemen-elemennya tidak dapat dibandingkan dengan kelompok hasil iterasi 2 lainnya. Maka, tabel iterasi 3 untuk kasus ini, kosong (tidak perlu dilakukan).

### d. Tabel *prime implicants*

**Tabel 3.6** Tabel *prime implicants* untuk *test case 6* dengan penyelesaian metode tabular secara manual

<i>Prime Implicants</i> \ Minterm	1	2	4	6
aCd		✓		✓
aBd			✓	✓
abcD	✓			

Pada tabel tersebut, akan didapat:

- *prime implicants* : aCd, aBd, abcD
- *essential prime implicants* : abcD, aCd, aBd

Dari tahapan serta tabel di atas, *boolean expression* yang sudah disederhanakan untuk *test case 6* adalah  $abcD + aCd + aBd$ , sesuai dengan *boolean expression* yang didapat dari program.

### 3. Penyelesaian manual untuk *test case 7*

Dengan input desimal minterm adalah 0, 5, 6, 9, 10, 7, 13, 14, 15.

#### a. Iterasi 1

**Tabel 3.7** Iterasi 1 untuk *test case 7* dengan penyelesaian metode tabular secara manual

Group	Minterm	Biner			
G0	0	0	0	0	0
G1	5	0	1	0	1
	6	0	1	1	0
	9	1	0	0	1
	10	1	0	1	0
G2	7	0	1	1	1
	13	1	1	0	1
	14	1	1	1	0
G3	15	1	1	1	1

#### b. Iterasi 2

**Tabel 3.8** Iterasi 2 untuk *test case 7* dengan penyelesaian metode tabular secara manual

Group	Minterm	Biner			
G0	5, 7	0	1	—	1
	5, 13	—	1	0	1
	6, 7	0	1	1	—
	6, 14	—	1	1	0
	9, 13	1	—	0	1

	10, 14	1	–	1	0
G1	7, 15	–	1	1	1
	13, 15	1	1	–	1
	14, 15	1	1	1	–

c. Iterasi 3

**Tabel 3.9** Iterasi 3 untuk *test case 7* dengan penyelesaian metode tabular secara manual

Group	Minterm	Biner			
G0	5,7 - 13, 15	–	1	–	1
	6,14 - 7,15	–	1	1	–

d. Tabel *prime implicants*

**Tabel 3.10** Tabel *prime implicants* untuk *test case 7* dengan penyelesaian metode tabular secara manual

Minterm Prime Implicants	0	5	6	7	9	10	13	14	15
BD		✓		✓					
BC			✓	✓				✓	✓
AcD					✓		✓		
ACd						✓		✓	
abcd	✓								

Pada tabel tersebut, akan didapat:

- *prime implicants* : BD, BC, AcD, ACd, abcd
- *essential prime implicants* : abcd, BD, BC, AcD, ACd

Dari tahapan serta tabel di atas, *boolean expression* yang sudah disederhanakan untuk *test case 7* adalah  $abcd + BD + BC + AcD + ACd$ , sesuai dengan *boolean expression* yang didapat dari program.



## KESIMPULAN

Minimisasi *logic boolean algebra* diperlukan untuk memperkecil ukuran sebuah rangkaian, sehingga kecepatan akan meningkat dan konsumsi *power* rangkaian menjadi semakin optimal. Terdapat dua buah metode yang dapat digunakan untuk melakukan minimisasi *logic boolean algebra*, yakni metode Karnaugh Map (K-Map) dan metode Quine-McCluskey (Metode Tabular). Melalui metode tabular, keterbatasan metode karnaugh map terkait jumlah variabel yang dapat diinputkan (minimal dua dan maksimal lima variabel), dapat teratasi, karena sifat metode tabular yang dapat menerima variabel dalam jumlah tak terbatas. Selain itu, dikarenakan metode karnaugh map lebih banyak memanfaatkan visual manusia daripada metode tabular, metode karnaugh map akan lebih sulit untuk diimplementasikan dalam sebuah kode program. Karena alasan-alasan tersebut, kami pun memilih metode tabular untuk dibahas dan diimplementasikan dalam *Tugas Besar Eksplorasi Algoritma Minimisasi Logic EL2008 - Pemecahan Masalah dengan C* untuk Kelompok 7.

Tujuan utama yang ingin dicapai dalam metode tabular adalah untuk menemukan seluruh *prime implicants* kemudian memanfaatkannya untuk mencari *essential prime implicants* dari fungsi-fungsi yang di-input-kan (minterm). Dalam perjalanannya demi mencapai tujuan utama tersebut, dilakukan setidaknya empat buah tahapan, yakni iterasi 1, iterasi 2, iterasi 3, dan pembuatan *tabel prime implicants*, *essential prime implicants*, serta hasil akhir. Pada kode program yang telah disusun, *input* akan berupa banyaknya minterm yang akan diminimisasi (dalam *range* 1 s.d. 16 berbentuk integer) dan bentuk desimal dari minterm-minterm tersebut, (dalam *range* 0 s.d. 15 berbentuk integer) sesuai jumlah banyaknya minterm yang telah di-input-kan. Pada akhir keberjalanan, *output* yang akan disajikan kepada pengguna adalah tabel hasil iterasi 1, iterasi 2, iterasi 3 (jika diperlukan), *tabel prime implicants*, serta *boolean expression* yang telah disederhanakan. Program ini memanfaatkan 17 fungsi lain di luar *main()* function, yakni *tambahMinterm(int)*, *buatNode(int)*, *gabungMinterm()*, *print()*, *printTabel()*, *buatNodePair(node\*, node\*)*, *isiBiner(node\*, node\*, node\*)*, *initTable()*, *ifPairingPossible(node\*, node\*)*, *ifMintermPresentInImplicant(int, int)*, *tambahPair(node\*, node\*)*, *tambahTabel()*, *analisisTabel()*, *konversiBiner(int)*, *cariMax(int\*)*, *jumlahImplicants(int, int)*, dan *hapusMinterm(int)*.

Kelebihan dari program ini adalah tersedianya tampilan bentuk tabel dari hasil iterasi 1, 2, 3, dan *prime implicants*, sehingga memungkinkan pengguna menganalisis setiap tahapan dari metode tabular. Sementara, kekurangan dari program ini terletak pada keterbatasan input yang dapat dimasukkan. Jumlah banyaknya minterm yang dapat dimasukkan terbatas dalam *range* 1 s.d. 16 dan bentuk desimal dari minterm terkait yang dapat dimasukkan terbatas dalam *range* 0 s.d. 15. Adapun, *lesson learned* yang kami dapatkan melalui penyusunan tugas besar ini adalah:

- Pada mata kuliah sebelumnya, kami lebih familiar dengan penggunaan Karnaugh Map dalam melakukan minimisasi *logic boolean algebra*. Namun untuk tugas besar kali ini, kami menggunakan Tabular Method. Hal ini dikarenakan, penggunaan Tabular Method dapat digunakan untuk variabel yang lebih banyak sedangkan Karnaugh Map lebih terbatas. Melalui hal tersebut, kami memperoleh ilmu baru terkait metode lain yang dapat dilakukan untuk praktik minimisasi *logic boolean algebra*.
- Pada mata kuliah sebelumnya, kami lebih menitikberatkan pada analisis manual minimisasi *logic boolean algebra*, sementara pada tugas besar ini, kami melakukan

analisis dalam implementasinya di sebuah program komputer. Sehingga, kami kembali menggali ilmu terkait penggambaran program dalam bentuk flowchart dan DFD, bahasa C dan fungsi-fungsi di dalamnya, serta diperlukan analisis kembali terhadap beberapa contoh implementasinya yang telah ada di internet.

- Dengan adanya keterbatasan variabel yang terjadi pada program yang disusun, kami merasa program masih dapat dikembangkan agar variabel-variabel yang dapat di-*input*-kan lebih dari 4.

## PEMBAGIAN TUGAS

### a. Pembagian tugas untuk code dan flowchart

Bagian	Penyusun	Penguji
Struktur data dan variabel global	Reina Puteri R. / 18320039	Reina Puteri R. / 18320039
tambahMinterm	Reina Puteri R. / 18320039	Reina Puteri R. / 18320039
buatNode	Reina Puteri R. / 18320039	Reina Puteri R. / 18320039
gabungMinterm	Reina Puteri R. / 18320039	Reina Puteri R. / 18320039
print	Reina Puteri R. / 18320039	Reina Puteri R. / 18320039
printTabel	Reina Puteri R. / 18320039	Reina Puteri R. / 18320039
buatNodePair	Atadila Belva G. / 18320015	Atadila Belva G. / 18320015
isiBiner	Atadila Belva G. / 18320015	Atadila Belva G. / 18320015
initTabel	Atadila Belva G. / 18320015	Atadila Belva G. / 18320015
ifPairingPossible	Atadila Belva G. / 18320015	Atadila Belva G. / 18320015
ifMintermPresentInImplicant	Atadila Belva G. / 18320015	Atadila Belva G. / 18320015
tambahPair	Atadila Belva G. / 18320015	Atadila Belva G. / 18320015
tambahTabel	Diandra R. P. N. / 18320023	Diandra R. P. N. / 18320023
analisisTabel	Diandra R. P. N. / 18320023	Diandra R. P. N. / 18320023
konversiBiner	Diandra R. P. N. / 18320023	Diandra R. P. N. / 18320023
cariMax	Diandra R. P. N. / 18320023	Diandra R. P. N. / 18320023
jumlahImplicant	Diandra R. P. N. / 18320023	Diandra R. P. N. / 18320023
hapusMinterm	Diandra R. P. N. / 18320023	Diandra R. P. N. / 18320023

### b. Pembagian tugas untuk laporan

Bagian	Penyusun
Pendahuluan dan Teori Dasar	Atadila Belva G. / 18320015
Analisis Algoritma	Diandra R. P. N. / 18320023
Test Case	Reina Puteri R. / 18320039

## REFERENSI

- Vahid, F. 2010. *Digital Design with RTL Design, VHDL, and Verilog, 2nd Edition*. California: John Wiley and Sons Publishers.
- Hwang, E. O. 2005. *Digital Logic and Microprocessor Design With VHDL*. California: Brooks / Cole.
- freesourcecode.net. *Tabular method of minimization of boolean functions in c*. Diakses pada April 15, 2022, dari [http://freesourcecode.net/cprojects/102643/Tabular-method-of-minimization-of-boolean-functions-in-c#.YpUM\\_ahBzIU](http://freesourcecode.net/cprojects/102643/Tabular-method-of-minimization-of-boolean-functions-in-c#.YpUM_ahBzIU).
- Banerji, Sourangsu. (2014). Computer Simulation Codes for the Quine-McCluskey Method of Logic Minimization. *Other Computer Science (cs.OH)*. doi: <https://doi.org/10.48550/arXiv.1404.3349>.