LAPORAN

TUGAS BESAR MULTI LINKED LIST

Disusun untuk memenuhi tugas

Mata Kuliah: Struktur Data
Dosen Pengampu: AULIA KHAMAS HEIKHMAKHTIAR, B.Sc., M.Eng., Ph.D.

Oleh:

MUHAMMAD RAFI ANDEO PRAJA (1301200278)
RICARDO HAMONANGAN(1301204201)

KELAS IF-44-01
JURUSAN S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM

ALGORITMA MULTI LINKED LIST

1. Deskripsi Multi Linked List

Multi Linked List yang kami gunakan adalah multi linked list dari single linked list, tanpa last, dan memiliki relasi 1-N tipe 2 (yaitu tipe yang mana node child tidak menggunakan linked list). Kami menggunakan tema benua dan negara yang dimana benua sebagai node parent dan negara sebagai node child, kami menggunakan tema tersebut karena nama - nama benua dan negara akan menjadi suatu tema yang mudah untuk dikembangkan di Multi Linked List. Kami memiliki Abstract Data Type dan Algoritma prosedur dan fungsi yang berjumlah 21 fungsi atau prosedur. Secara ringkas, prosedur yang kami miliki dimulai dari insertion parent sampai ke deletion child, dan untuk fungsi yang kami miliki dimulai dari searching sampai counting. Multi Linked List ini masih sangat jauh dari kata sempurna, jadi kami berharap bahwa untuk kedepannya kami dapat belajar dan menambah wawasan kami dalam menggunakan struktur data Multi Linked List ini dan struktur data lainnya.

2. Abstract Data Type

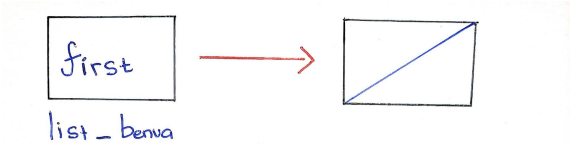
```
Type benua <
        nama_benua          : string
        jumlah_negara       : integer
>
Type negara <
        nama_negara         : string
        jumlah_kota         : integer
>
Type adr_benua      : pointer to elm_benua
Type adr_negara     : pointer to elm_negara
Type elm_benua <
        info            : benua
        next            : adr_benua
        next_negara     : adr_negara
>
Type elm_negara <
        info            : negara
        next            : adr_negara
>
Type Multi_Linked_List <
        first           : adr_benua >
```

Procedure create_list(**input/output** list_benua: Multi_Linked_List)
Procedure new_elm_benua(**input** info: benua, **input/output** b: adr_benua)
Procedure new_elm_negara(**input** info: negara, **input/output** n: adr_negara)
Procedure show_data_benua(**input** list_benua: Multi_Linked_List)
Procedure delete_last_benua(**input/output** list_benua: Multi_Linked_List, **output** b: adr_benua)
Procedure delete_first_benua(**input/output** list_benua: Multi_Linked_List, **output** b: adr_benua)
Procedure insert_first_benua(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua)
Procedure insert_After_benua(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua)
Procedure insert_last_benua(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua)
Procedure insert_new_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, n: adr_negara)
Function search_benua(list_benua: Multi_Linked_List, **input** nama_benua: string) → boolean
Function search_negara_of_x(list_benua: Multi_Linked_List, **input** nama_negara: string, nama_benua: string) → boolean
Procedure show_data_negara_of_x(**input** list_benua: Multi_Linked_List, nama_benua: string)
Function count_jumlah_negara(list_benua: Multi_Linked_List, b: adr_benua, n: adr_negara) → integer
Function sum_jumlah_kota(list_benua: Multi_Linked_List, b: adr_benua, n: adr_negara) → integer
Procedure insert_relation(**input/output** list_benua: Multi_Linked_List, b: adr_benua, n: adr_negara)
Procedure delete_first_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, n: adr_negara)
Procedure delete_last_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, **output** n: adr_negara)
Procedure delete_after_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, prec: adr_negara, **output** n: adr_negara)
Procedure delete_relation(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, n: adr_negara)
Procedure delete_negara_of_x(**input/output** list_benua: Multi_Linked_List, b: adr_benua, n: adr_negara, **input** nama_benua: string, nama_negara: string)

2. Algoritma

Procedure create_list(**input/output** list_benua: Multi_Linked_List)
Algoritma
    first(list_benua) ← NULL
Endprocedure

Visual Algoritma:

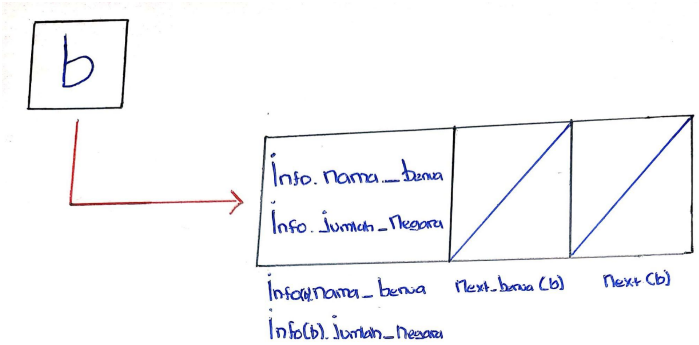


Deskripsi Algoritma:

Procedure create_list membuat suatu Multi Linked List dengan cara membuat pointer benua first(list_benua) menjadi NULL, procedure create_list menjadi prosedur pertama yang harus dipanggil ketika kita ingin membuat suatu struktur data.

Procedure new_elm_benua(**input** info: benua, **input/output** b: adr_benua)
Algoritma
    alokasi(b)
    next(b) ← NULL
    next_negara(b) ← NULL
    info(b).nama_benua ← info.nama_benua
    info(b).jumlah_negara ← info.jumlah_negara
Endprocedure

Visual Algoritma:

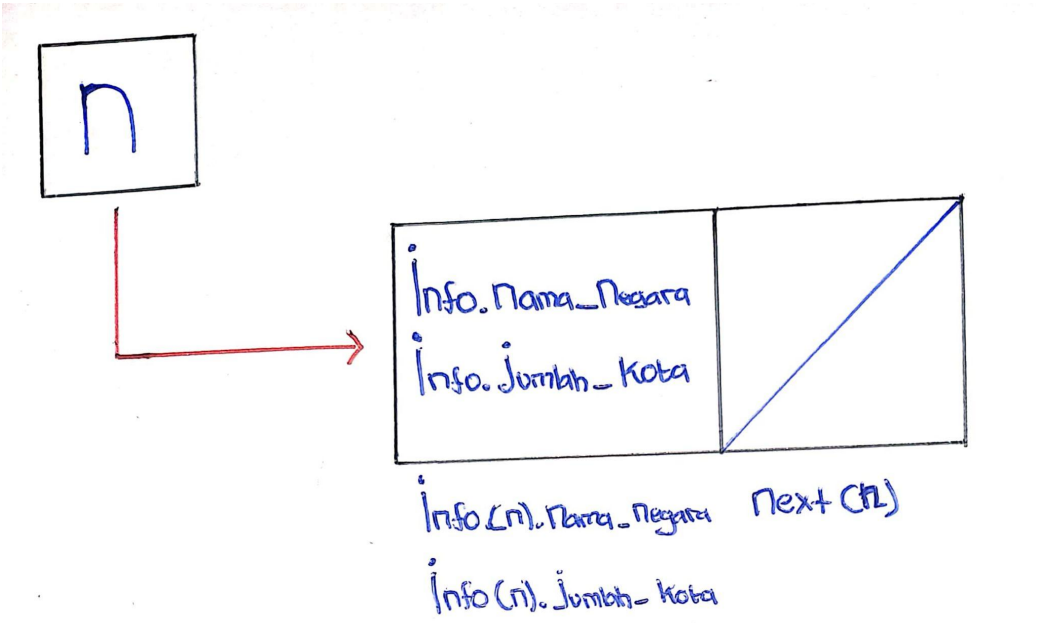


Deskripsi Algoritma:

Procedure new_elm_benua membuat isi dari Multi Linked List dengan mengalokasikan node b dan menetapkan next(b) menjadi NULL, next_negara(b) menjadi NULL, info(b).nama_benua menjadi info.nama_benua yang didapat sebagai inputan user dan dimasukkan menjadi parameter prosedur, dan juga info(b).jumlah_negara menjadi info.jumlah_negara yang didapat dari variabel input bertipe benua.

Procedure new_elm_negara(**input** info: negara, **input/output** n: adr_negara)
Algoritma
    alokasi(n)
    next(n) ← NULL
    info(n).nama_negara ← info.nama_negara
    info(n).jumlah_kota ← info.jumlah_kota
Endprocedure

Visual Algoritma:



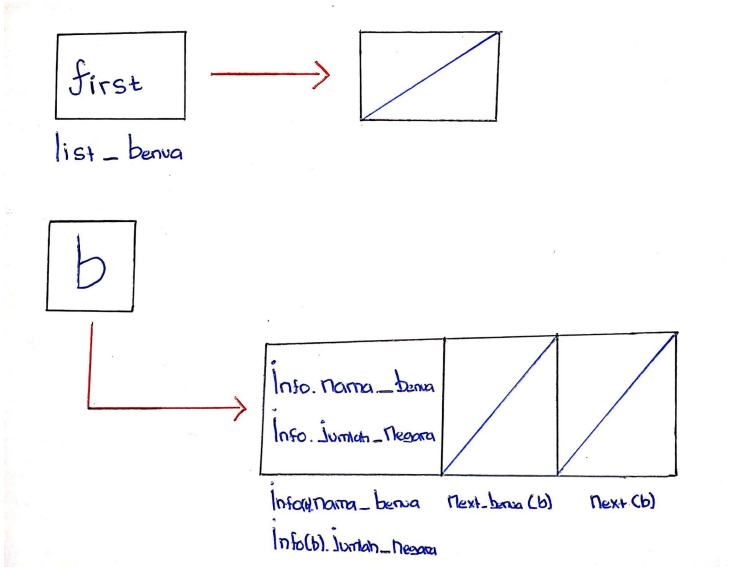
Deskripsi Algoritma:

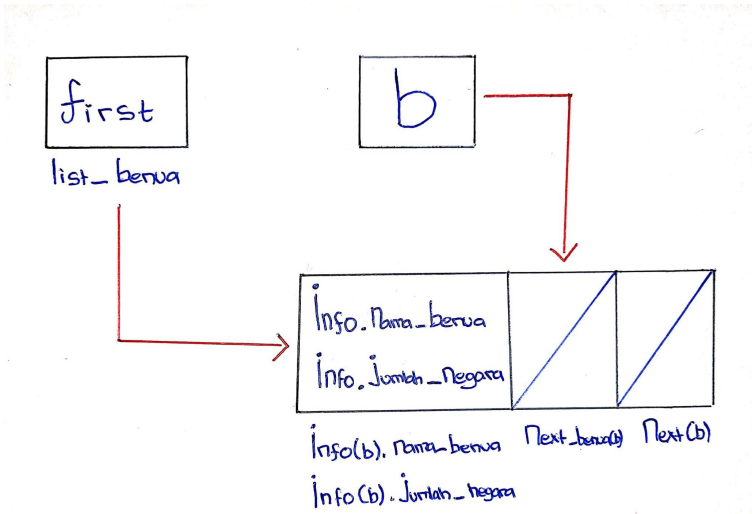
Procedure new_elm_negara sama seperti new_elm_benua, bedanya adalah prosedur new_elm_negara mengalokasikan node n dan menetapkan next(n) menjadi NULL, info(n).nama_negara menjadi info.nama_negara yang didapatkan dari variabel info bertipe negara, dan juga info.jumlah_kota didapat dari variabel info.

Procedure insert_first_benua(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua)
Algoritma
	If (first(list_benua) == NULL) then
		first(list_benua) ← b
	Else then
		next(b) ← first(list_benua)
		first(list_benua) ← b
	Endif
Endprocedure

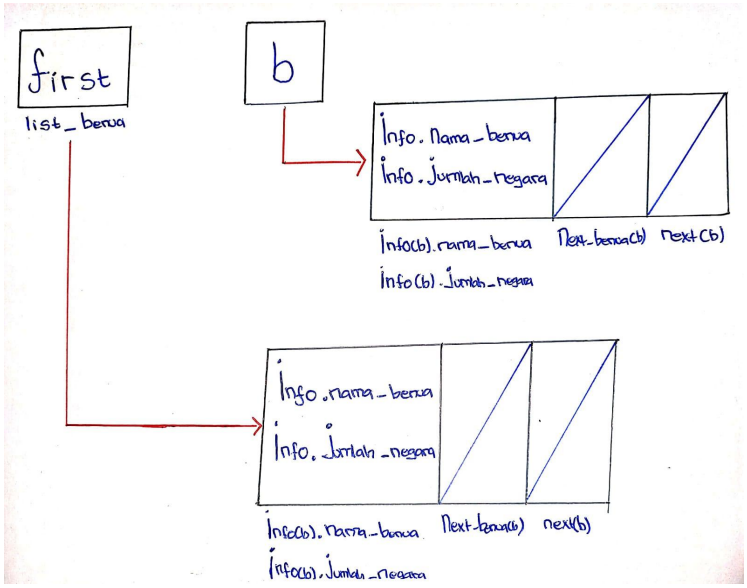
Visual Algoritma:

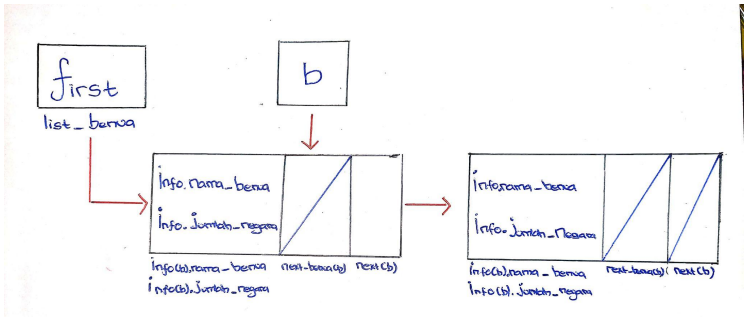
Kondisi If:





Kondisi Else:



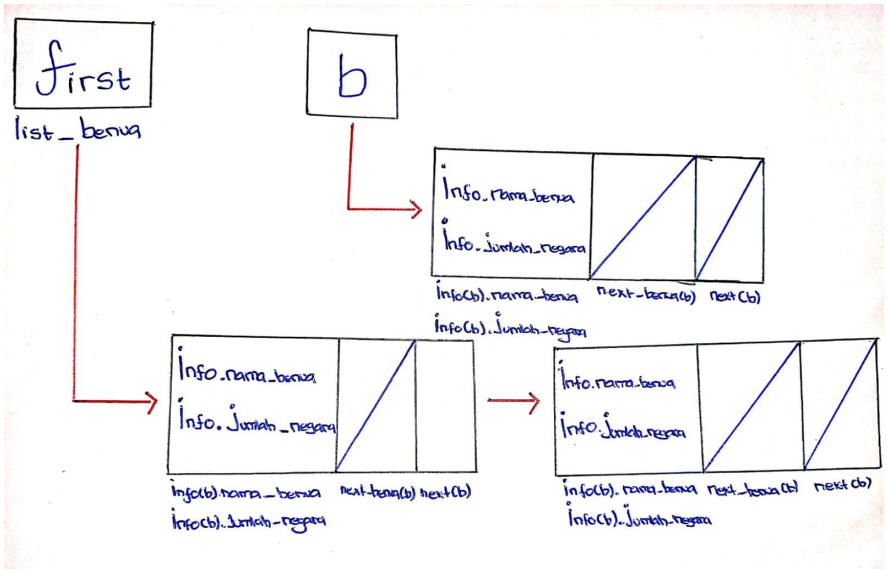


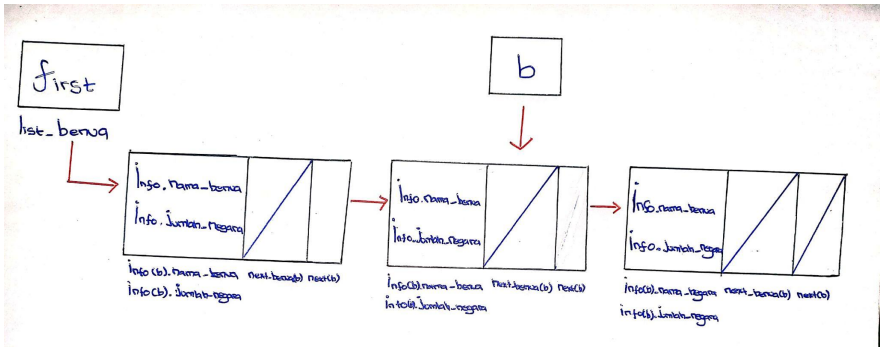
Deskripsi Algoritma:

Procedure insert_first_benua melakukan penambahan node baru ke Multi Linked List. Penambahan node yang dilakukan hanya ditempatkan di paling depan. Prosedur tersebut memiliki 2 kondisi, yang pertama jika kondisi list masih kosong, maka node b langsung ditambahkan ke Multi Linked List. Kondisi terakhir adalah kondisi dimana jika list terisi sekurang - kurangnya 1 node, maka node b akan ditambahkan di paling depan, tidak di belakang node atau di tengah node yang sudah ada.

Procedure insert_after_benua(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua)
Kamus
	q: adr_benua
Algoritma
	If (first(list_benua) == NULL) then
		first(list_benua) ← b
	Else then
		q ← first(list_benua)
		next(b) ← NULL
		next(b) ← next(q)
		next(q) ← b
	Endif
Endprocedure

Visual Algoritma:





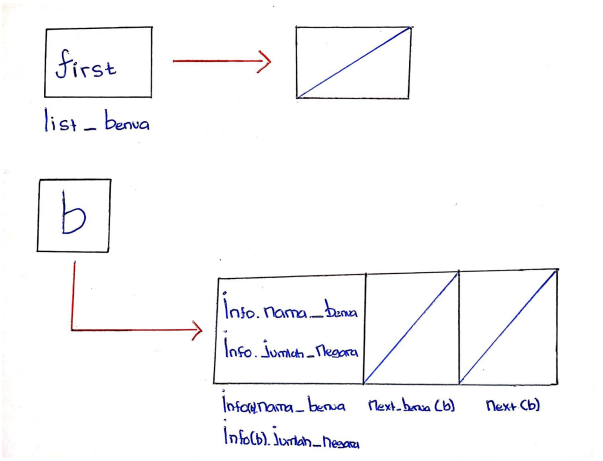
Deskripsi Algoritma:

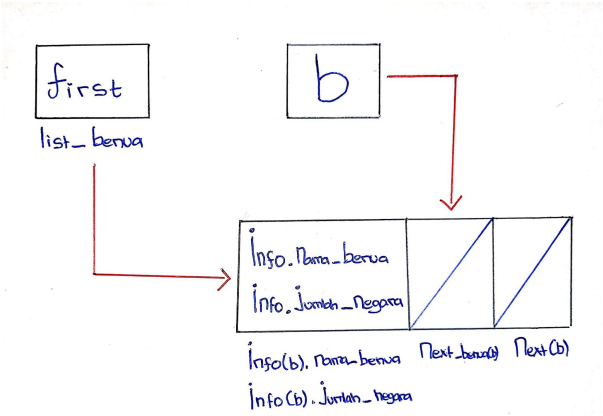
Procedure insert_after_benua berfungsi sama seperti insert_first_benua, yang membedakan adalah letak node yang akan diletakkan di list berada di antara node yang ditetapkan user, prosedur tersebut juga menggunakan parameter tambahan yaitu *"prec"* sebagai tempat tujuan node yang akan diletakkan selanjutnya.

```
Procedure insert_last_benua(input/output list_benua: Multi_Linked_List, input b:
adr_benua)
Kamus
        p: adr_benua
Algoritma
        If (first(list_benua) == NULL) then
                first(list_benua) ← b
        Else then
                p ← first(list_benua)
                While (next(p) <> NULL) do
                        p ← next(p)
                Endwhile
                next(p) ← b
        Endif
Endprocedure
```

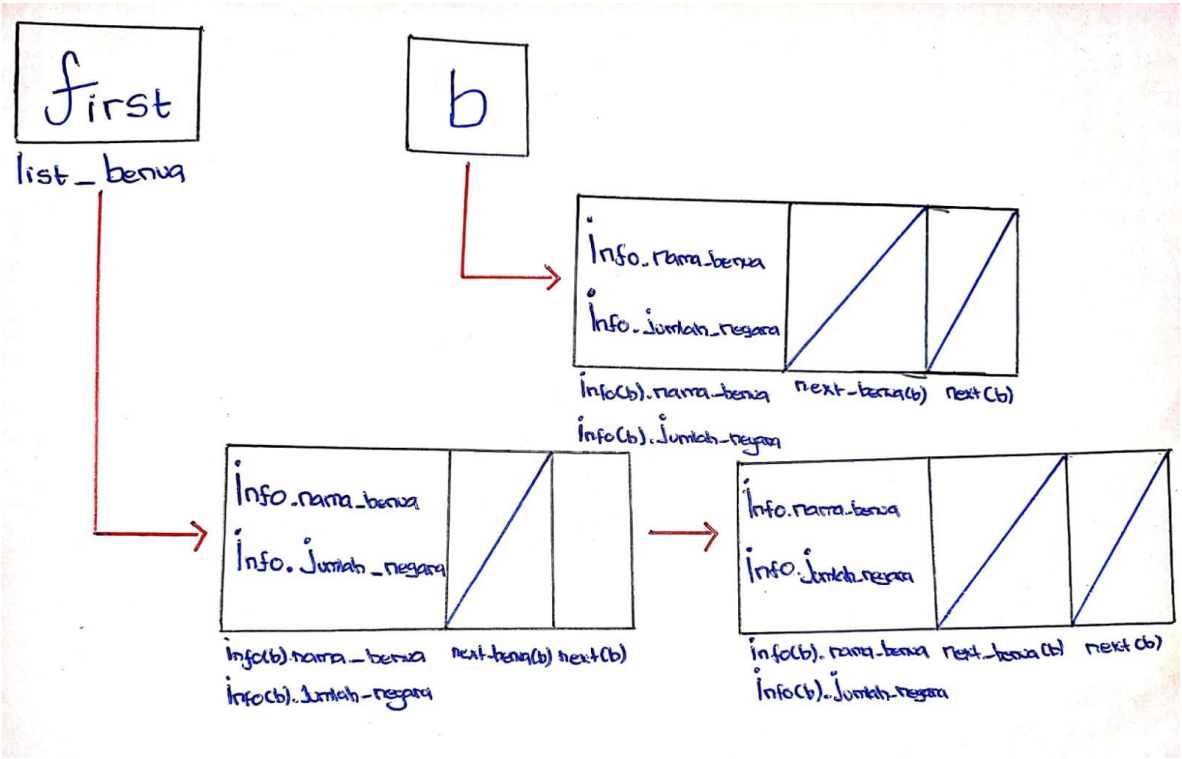
Visual Algoritma:

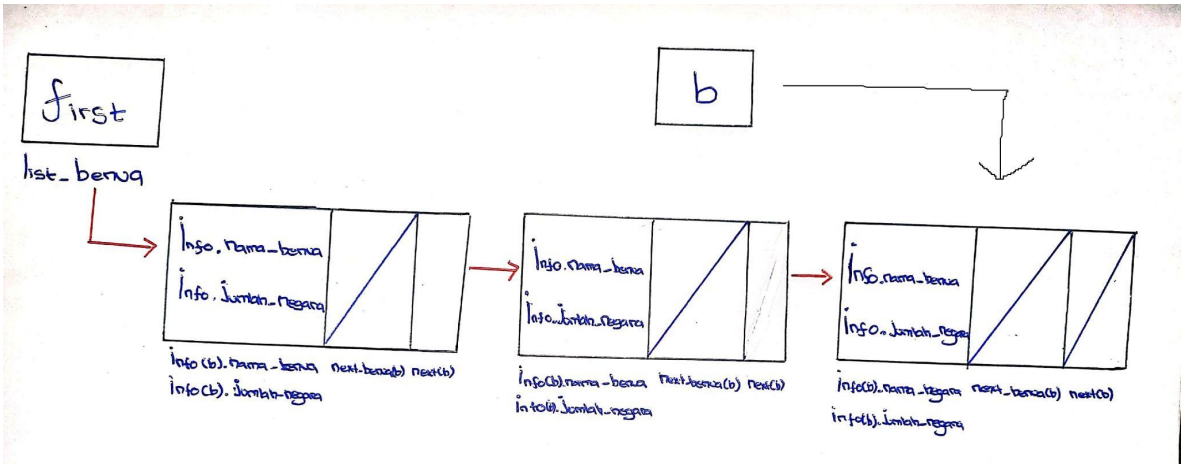
Kondisi If:





Kondisi Else:



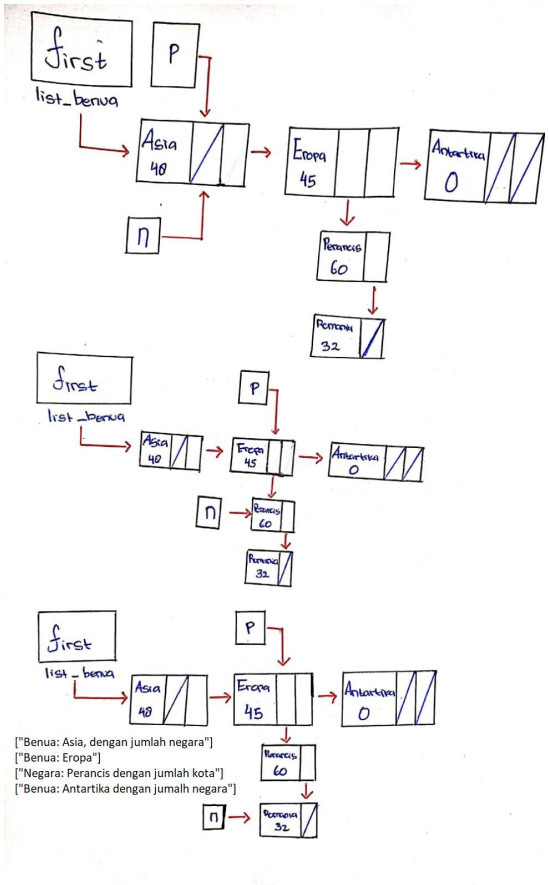


Deskripsi Algoritma:

Procedure insert_last_benua berfungsi untuk meletakkan node b ke Multi Linked List, mirip seperti procedure insert_first_benua dan insert_after_benua, tetapi bedanya adalah letak dari nodenya akan diletakkan di bagian paling akhir dari Multi Linked List. Kondisi yang digunakan ada 2 kondisi yaitu, jika list kosong, maka akan langsung diletakkan dan jika kondisi list tidak kosong, maka node b akan diletakkan di bagian paling akhir list, tidak di depan dan tidak juga di tengah 2 node.

```
Procedure show_data_benua(input list_benua: Multi_Linked_List)
Kamus
        n: adr_negara
        b: adr_benua
Algoritma
        b ← first(list_benua)
        While (b <> NULL) do
                output(“[Benua: “, info(b).nama_benua, “ dengan”, “ ”)
                output(“Jumlah negara: “, info(b).jumlah_negara, “]”)
                n ← next_negara(b)
                While (n <> NULL) do
                        output(“[Negara: “, info(n).nama_negara, “ dengan”, “ ”)
                        output(“Jumlah kota: “, info(n).jumlah_kota, “]”)
                        n ← next(n)
                Endwhile
                b ← next(b)
        Endwhile
Endprocedure
```

Visual Algoritma:



Deskripsi Algoritma:

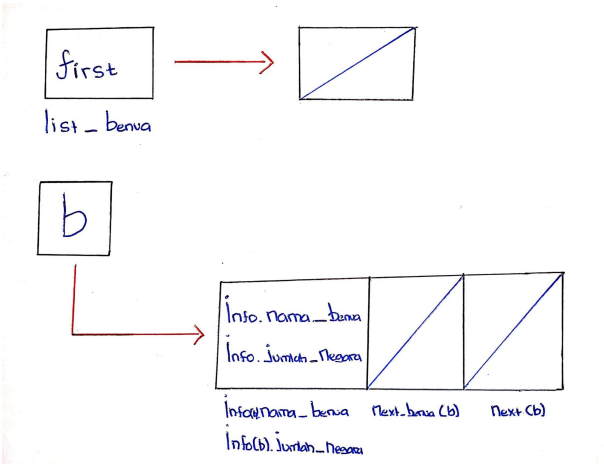
Procedure show_data_benua akan mengoutputkan semua info dari Multi Linked List dari info(b) sampai info(n).

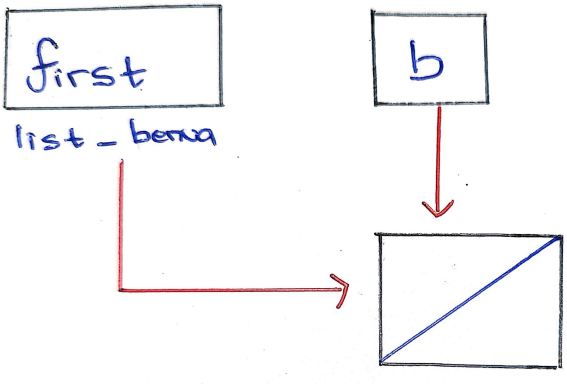
Procedure delete_last_benua(**input/output** list_benua: Multi_Linked_List, **output** b: adr_benua)
Kamus
    q: adr_benua
Algoritma
    If (first(list_benua) == NULL) then
        b ← NULL
    Else if (next(first(list_benua)) == NULL) then
        b ← first(list_benua)
        first(list_benua) ← NULL
    Else then
        q ← first(list_benua)
        b ← first(list_benua)
        While (next(b) <> NULL) do
            q ← b
            b ← next(b)

```
                Endwhile
                next(q) ← NULL
        Endif
Endprocedure
```

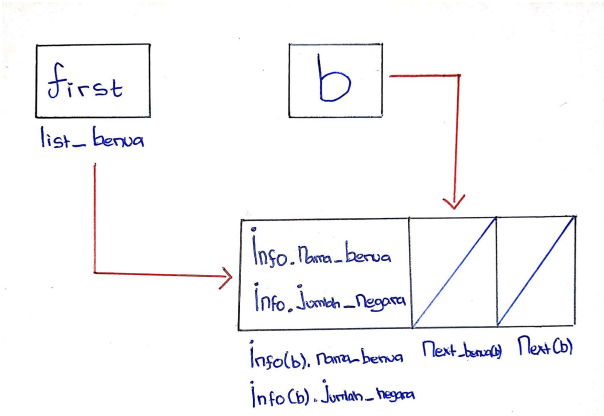
Visual Algoritma:

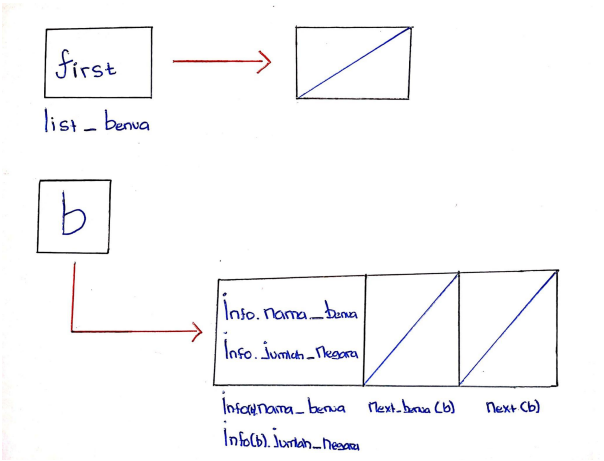
Kondisi If:



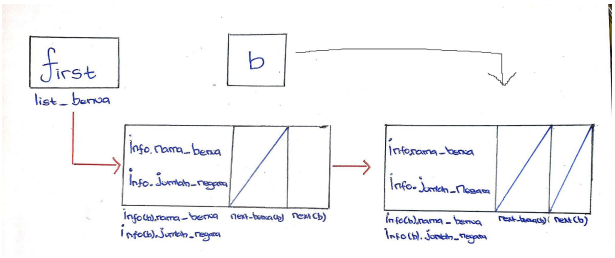


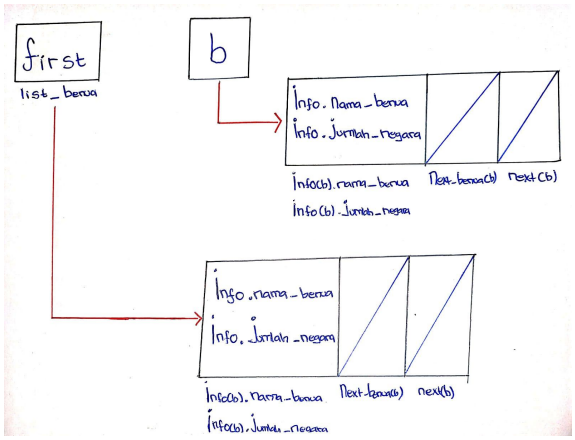
Kondisi Else If:





Kondisi Else:





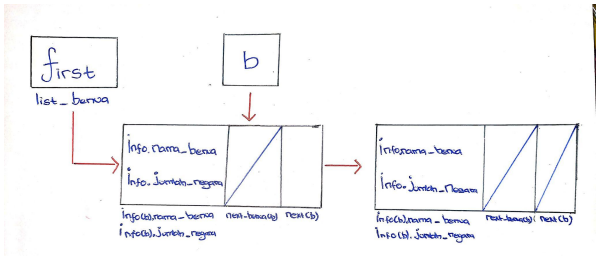
Deskripsi Algoritma:

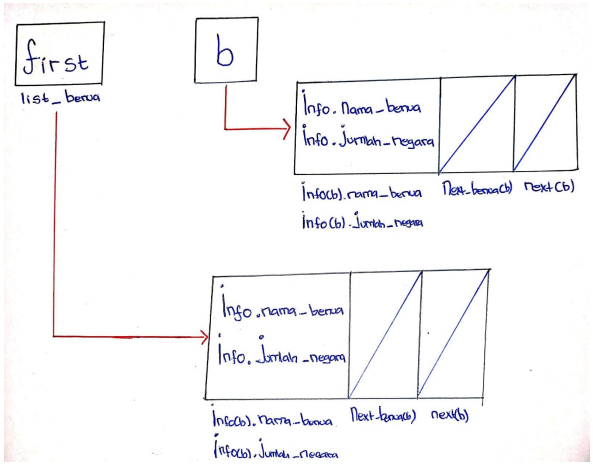
Procedure delete_last_benua akan menghapus node b yang sudah di letakkan di Multi Linked List. Penghapusan prosedur ini memiliki 3 kondisi, jika first(list_benua) itu NULL, maka node b akan di NULLkan juga. Untuk kondisi 2, jika Multi Linked List hanya memiliki satu node, maka node itu akan dihapus dari Multi Linked List dan node tersebut akan kembali ke node asalnya yaitu b. Dan kondisi terakhir jika Multi Linked List memiliki node lebih dari satu, maka node terakhir dari Multi Linked List akan dihapus.

Procedure delete_first_benua(**input/output** list_benua: Multi_Linked_List, **output** b: adr_benua)
Algoritma
    If (first(list_benua) == NULL) then
        b ← NULL
    Else if (next(first(list_benua)) == NULL) then
        b ← first(list_benua)
        first(list_benua) ← NULL
    Else then
        b ← first(list_benua)
        first(list_benua) ← next(b)
        next(b) ← NULL
    Endif
Endprocedure

Visual Algoritma:

Kondisi Else:



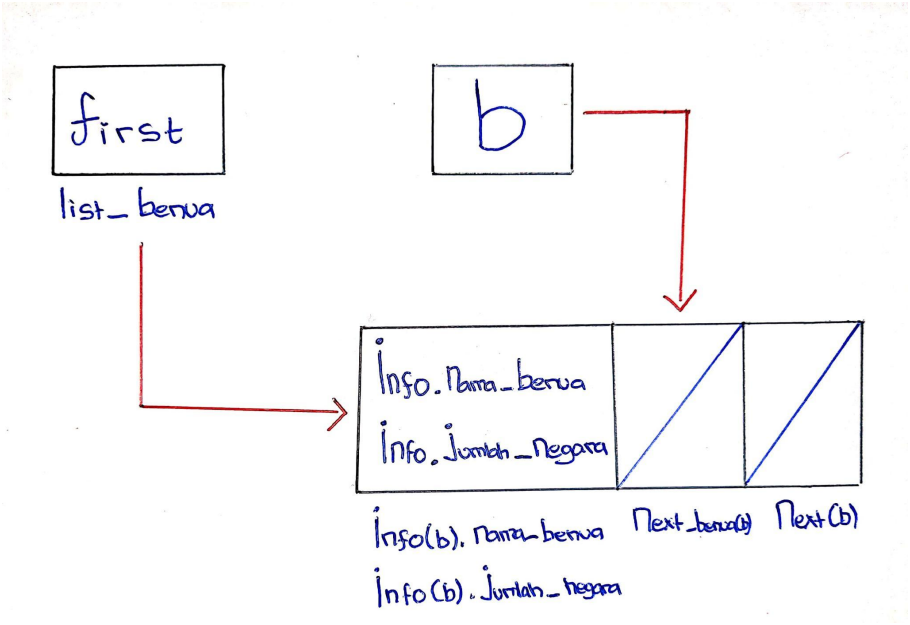


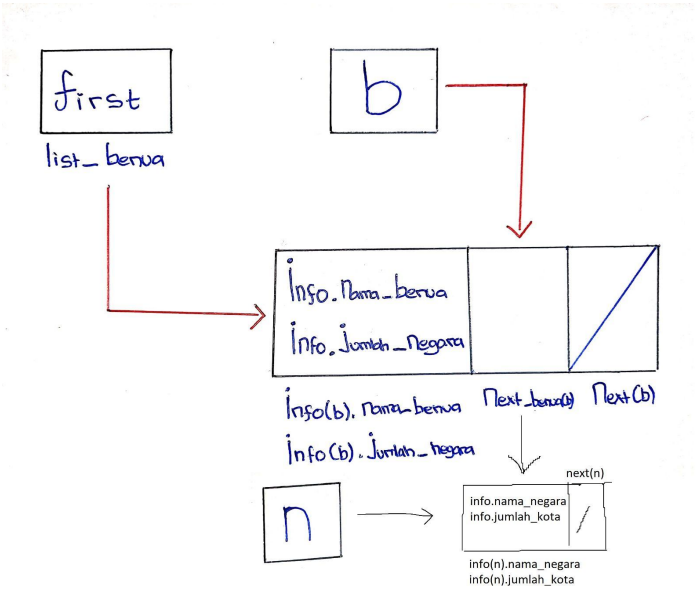
Deskripsi Algoritma:

Procedure delete_first_benua memiliki fungsi algoritma yang sama dengan prosedur delete_last_benua, perbedaannya hanyalah letak node yang dihapus adalah node yang berada di paling depan, kedua kondisi dari delete_first_benua tentu sama.
Procedure insert_new_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, n: adr_negara)
Kamus
    q: adr_negara
Algoritma
    If (next_negara(b) == NULL) then
        next_negara(b) ← n
    Else then
        q ← next_negara(b)
        While (next(q) <> NULL) do
            q ← next(q)
        Endwhile
        next(q) ← n
    Endif
Endprocedure

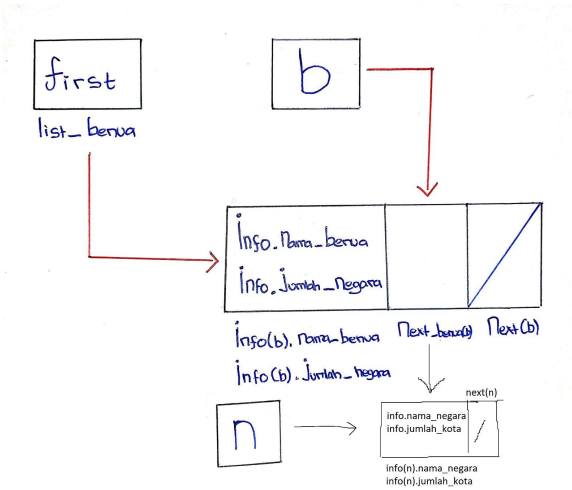
Visual Algoritma:

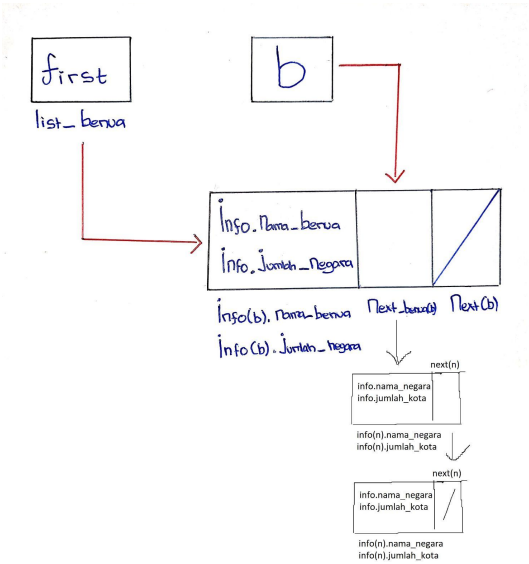
Kondisi If:





Kondisi Else:





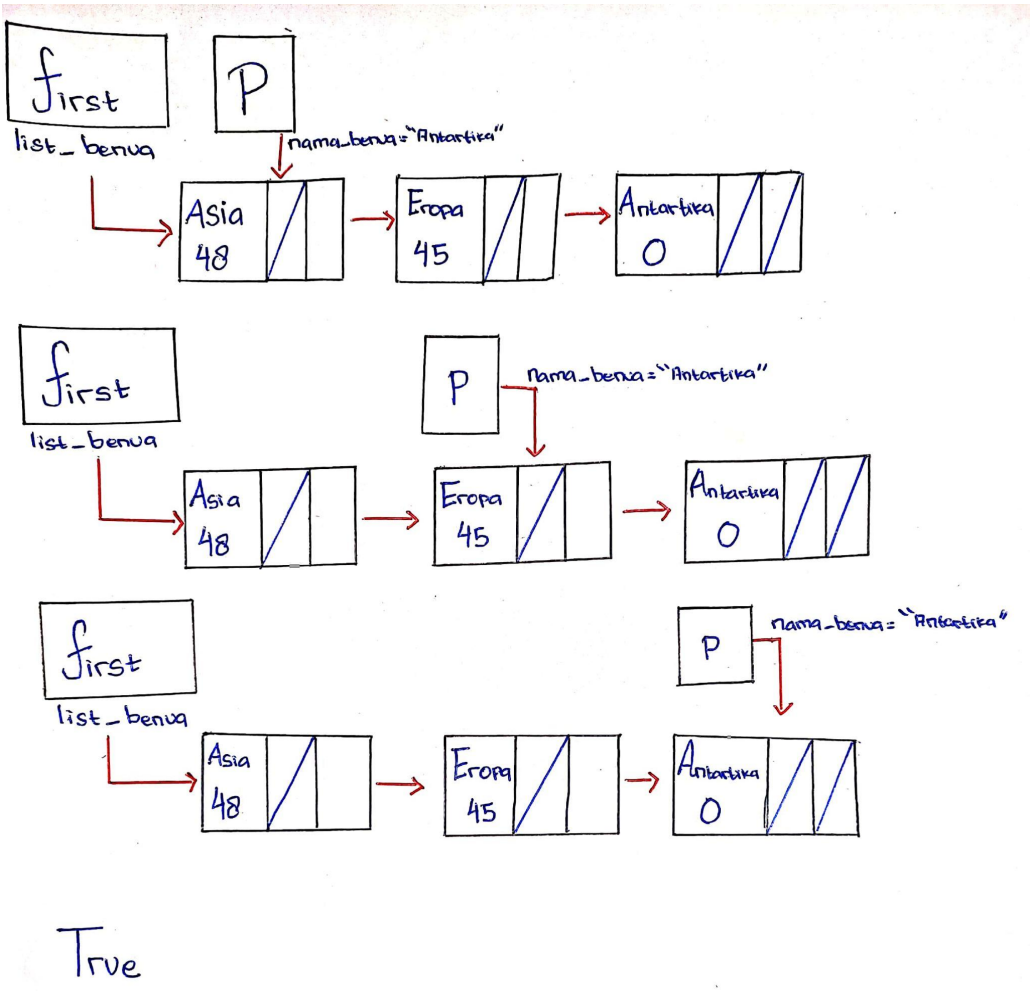
Deskripsi Algoritma:

Procedure insert_new_negara adalah suatu prosedur yang menambahkan node child yaitu node negara n ke salah satu node parent benua yang berada di Multi Linked List, prosedur ini memiliki 2 kondisi, yaitu jika tidak ada node child di suatu node parent maka child akan langsung ditambah di node parent yang dituju. Kondisi 2 adalah jika node parent sudah memiliki node child, maka node child akan ditambahkan menjadi node ke 2 child dari node parent tersebut.

Function search_benua(list_benua: Multi_Linked_List, **input** nama_benua: string) → boolean

```
Kamus
        p: adr_benua
Algoritma
        p ← first(list_benua)
        While (p <> NULL) do
                If (info(p).nama_benua == nama_benua) then
                        → true
                Endif
                p ← next(p)
        Endwhile
        → false
Endfunction
```

Visual Algoritma:



Deskripsi Algoritma:

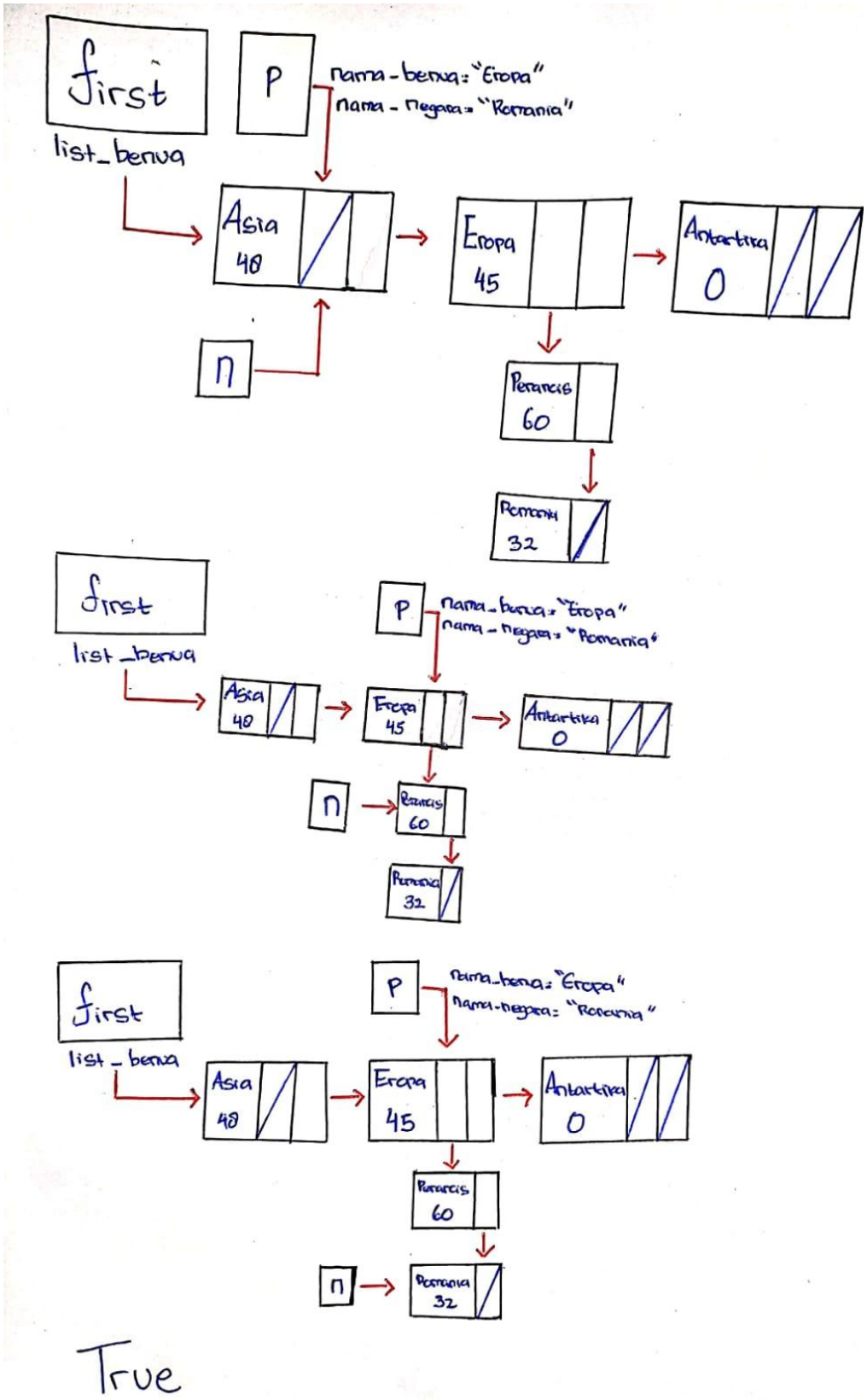
Function search_benua adalah fungsi yang mencari nama_benua apakah nama_benua tersebut ada di Multi Linked List atau tidak, jika tidak ada maka akan mengoutputkan false. Jika ada tetapi berada di node yang bukan pertama maka pointer penanda p akan bergerak menuju ke node berikutnya sampai ketemu lalu akan mengoutputkan true. Jika node pertama sudah ketemu nama_benua tersebut sama dengan info(p).nama_benua, maka akan langsung mengoutputkan true tanpa harus bergerak pointer tersebut.

```
Function search_negara_of_x(list_benua: Multi_Linked_List, input nama_negara: string, nama_benua: string) → boolean
Kamus
        p: adr_benua
        n: adr_negara
Algoritma
        p ← first(list_benua)
        While (p <> NULL) do
                If (info(p).nama_benua == nama_benua) then
                        n ← next_negara(p)
                        While (n <> NULL) do
                                If (info(n).nama_negara == nama_negara) then
                                        → true
                                Endif
                                n ← next(n)
                        Endwhile
                Endif
                p ← next(p)
        Endwhile
        → false
Endfunction
```

Deskripsi Algoritma:

Function search_negara_of_x merupakan fungsi yang memiliki kinerja hampir sama dengan function search_benua, bedanya adalah fungsi search_negara_of_x mencari apakah nama_negara sama dengan info(n).nama_negara di Multi Linked List. Fungsi ini membutuhkan dua pointer penanda untuk benua dan negara, jika negara tidak ketemu di pointer benua maka pointer benua akan bergerak. Jika pointer benuanya sudah tepat, tetapi pointer negara belum tepat, maka pointer negara akan bergerak sampai dapat lalu mengoutputkan hasil bahwa nama_negara sama dengan info(n).nama_negara, true.

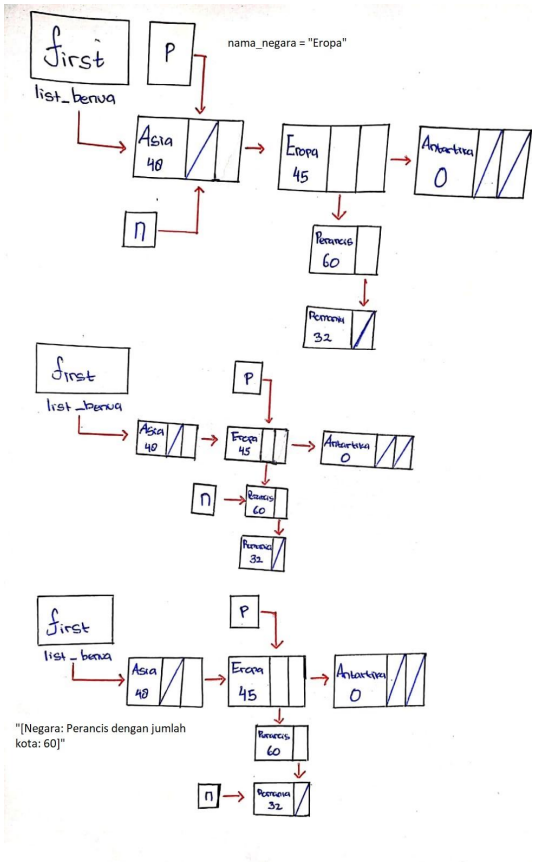
Visual Algoritma:



Procedure show_data_negara_of_x(**input** list_benua: Multi_Linked_List, nama_benua: string)

```
Kamus
        n: adr_negara
        b: adr_benua
Algoritma
        b ← first(list_benua)
        While (b <> NULL) do
                If (info(b).nama_benua == nama_benua) then
                        n ← next_negara(b)
                        While (n <> NULL) do
                                output("[Negara: ", info(n).nama_negara, " dengan", " ")
                                output("Jumlah kota: ", info(n).jumlah_kota, "]")
                                n ← next(n)
                        Endwhile
                Endif
                b ← next(b)
        Endwhile
Endprocedure
```

Visual Algoritma:

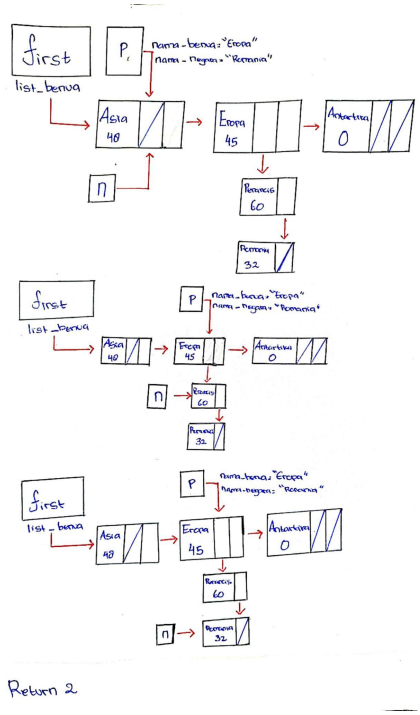


Deskripsi Algoritma:

Procedure show_data_negara_of_x akan mengoutputkan semua info(n).nama_negara beserta info(n).jumlah_kota dengan ketentuan hanya mengoutputkan daftar negara di satu benua saja. Perbandingan yang dilakukan adalah jika nama_benua sama dengan info(b).nama_benua maka akan langsung masuk ke node child dan mengoutput infonya.

```
Function count_jumlah_negara(list_benua: Multi_Linked_List,  b: adr_benua, n:
adr_negara) → integer
Kamus
        i: integer
Algoritma
        b ← first(list_benua)
        While (b <> NULL) do
                n ← next_negara(b)
                While (n <> NULL) do
                        i ← i + 1
                        n ← next(n)
                Endwhile
                b ← next(b)
        Endwhile
        → i
Endfunction
```

Visual Algoritma:

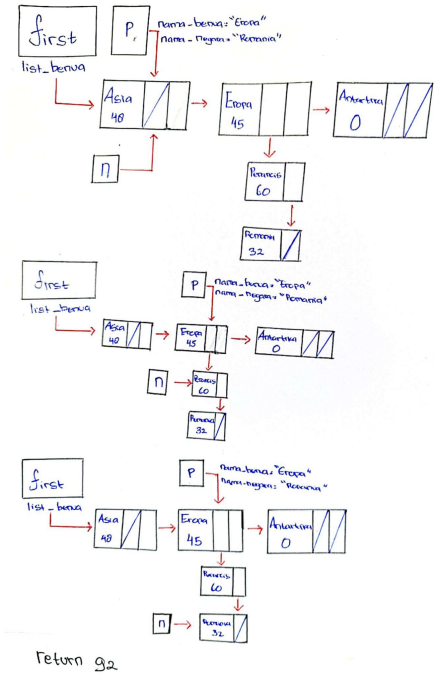


Deskripsi Algoritma:

Function count_jumlah_negara adalah suatu fungsi untuk menghitung jumlah negara yang ada di Multi Linked List, dengan menggunakan 2 pointer child negara n dan parent benua b. Kedua pointer tersebut akan bergerak sampai node Multi Linked List habis, lalu akan mengoutputkan hasil jumlah negara yang ada di multi linked list.

```
Function sum_jumlah_kota(list_benua: Multi_Linked_List,  b: adr_benua, n: adr_negara)
→ integer
Kamus
        sum: integer
Algoritma
        b ← first(list_benua)
        While (b <> NULL) do
                n ← next_negara(b)
                While (n <> NULL) do
                        sum ← sum + info(n).jumlah_kota
                        n ← next(n)
                Endwhile
                b ← next(b)
        Endwhile
        → sum
Endfunction
```

Visual Algoritma:

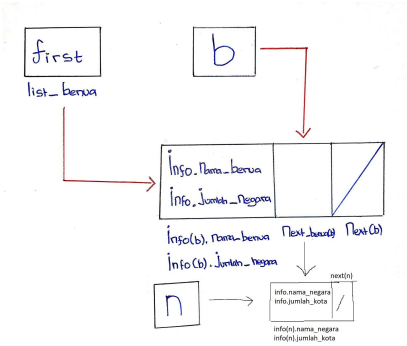


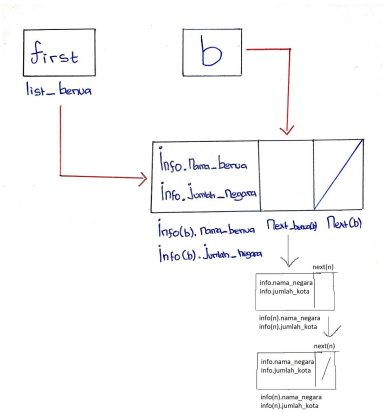
Deskripsi Algoritma:

Function sum_jumlah_kota akan menjumlah berapa jumlah kota yang dimasukkan di dalam Multi Linked List, jumlah kota tersebut didapatkan dari pointer child dan parent yang bergerak sama seperti di function count_jumlah_negara.

Procedure insert_relation(**input/output** list_benua: Multi_Linked_List, b: adr_benua, n: adr_negara)
Algoritma
    While (b <> NULL) do
        insert_new_negara(list_benua, b, n)
        n ← next_negara(b)
        While (n <> NULL)
            n ← next(n)
        Endwhile
        b ← next(b)
    Endwhile
Endprocedure

Visual Algoritma:



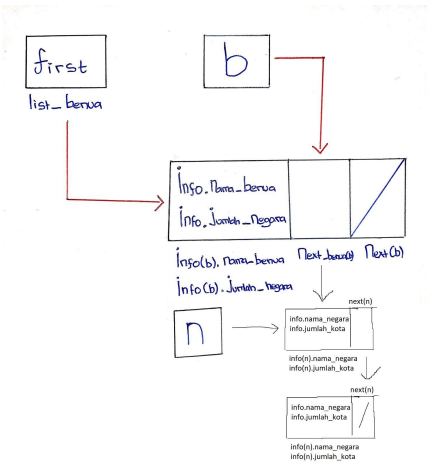


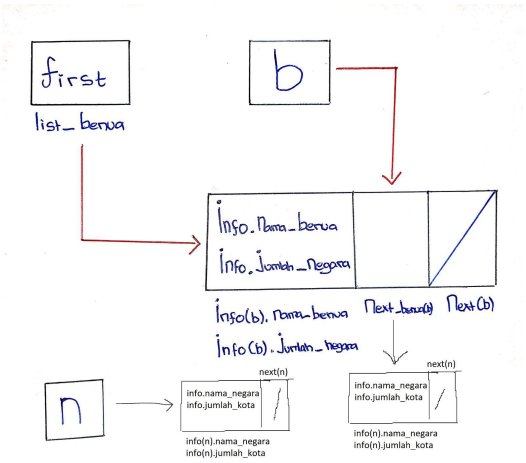
Deskripsi Algoritma:

Procedure insert_relation merupakan prosedur yang menambahkan node child ke node parent. Node dari parent tidak boleh kosong dan node child yang ditambahkan akan langsung dimasukkan ke Multi Linked List tanpa dilihat letaknya. Prosedur tersebut menggunakan procedure insert_new_negara.

Procedure delete_first_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, n: adr_negara)
Algoritma
    n ← next_negara(b)
    next_negara(b) ← next(next_negara(b))
    next(n) ← NULL
Endprocedure

Visual Algoritma:



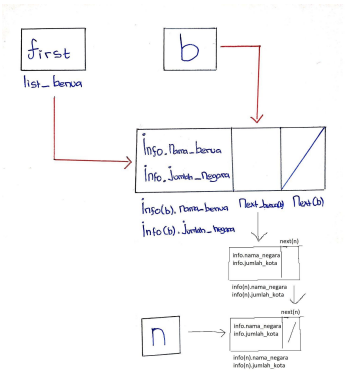


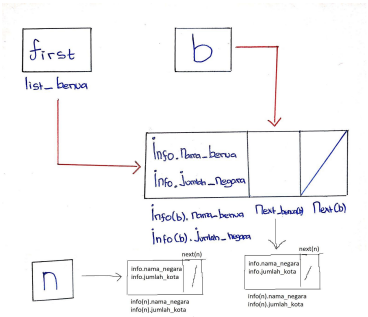
Deskripsi Algoritma:

Procedure delete_first_negara melakukan penghapusan node child negara yang terletak di bagian paling depan dari node child negara suatu node parent benua.

Procedure delete_last_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, **output** n: adr_negara)
Kamus
	q: adr_negara
Algoritma
	If (first(list_benua) <> NULL) then
		q ← next_negara(b)
		While (next(next(q)) <> NULL) do
			q ← next(q)
		Endwhile
		n ← q
		next(n) ← NULL
	Endif
Endprocedure

Visual Algoritma:



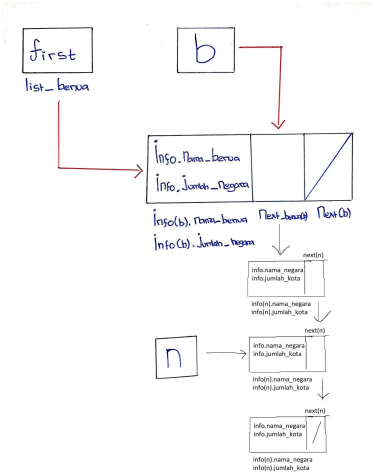


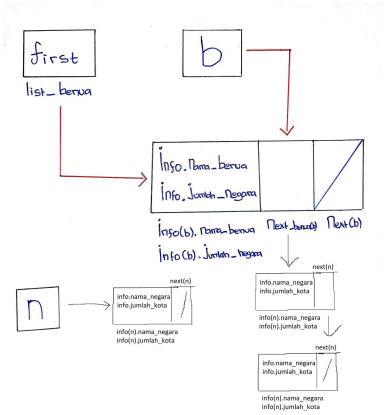
Deskripsi Algoritma:

Procedure delete_last_negara melakukan penghapusan node child negara yang terletak di bagian paling akhir dari node child negara yang berada di suatu node parent benua.

Procedure delete_after_negara(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, prec: adr_negara, **output** n: adr_negara)
Algoritma
	If (first(list_benua) <> NULL) then
		n ← next(prec)
		next(prec) ← next(n)
		next(n) ← NULL
	Endif
Endprocedure

Visual Algoritma:



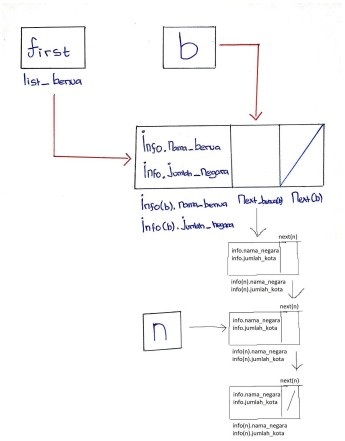


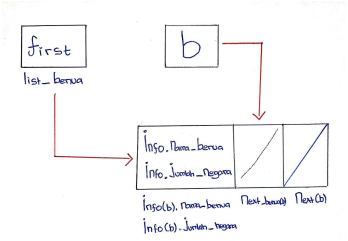
Deskripsi Algoritma:

Procedure delete_after_negara adalah prosedur yang menghapus node child negara yang berada setelah node child negara yang di pilih oleh user. Penanda pointer tersebut biasa dinamakan sebagai *“prec”*.

Procedure delete_relation(**input/output** list_benua: Multi_Linked_List, **input** b: adr_benua, n: adr_negara)
Algoritma
    b ← first(list_benua)
    While (b <> NULL) do
        If (next_negara(b) == n) then
            next_negara(b) ← NULL
        Endif
        b ← next(b)
    Endwhile
Endprocedure

Visual Algoritma:





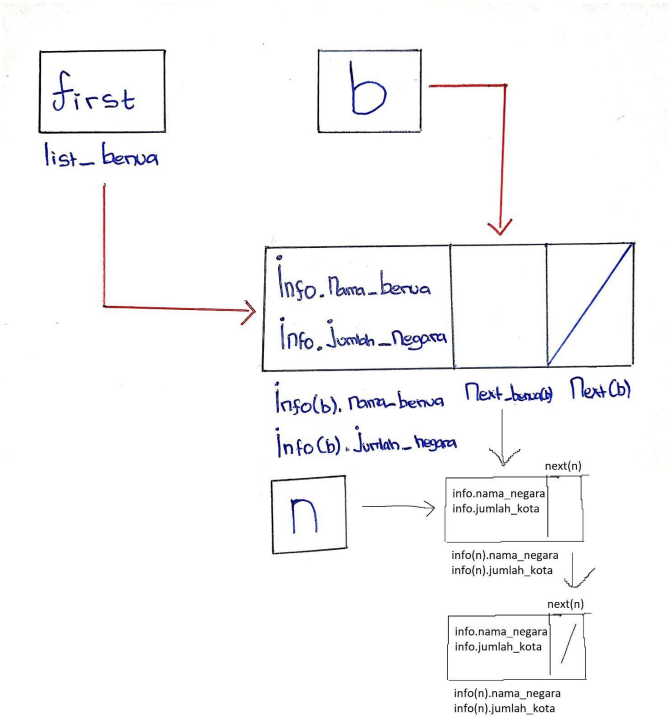
Deskripsi Algoritma:

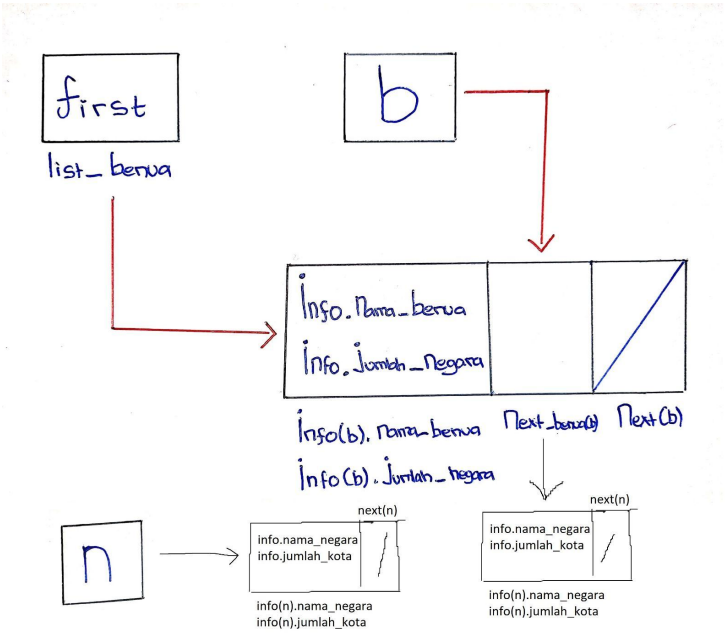
Procedure delete_relation melakukan penghapusan seluruh node child negara yang berada di semua node parent benua yang terdapat di Multi Linked List.

```
Procedure delete_negara_of_x(input/output list_benua: Multi_Linked_List, b:
adr_benua, n: adr_negara, input nama_benua: string, nama_negara: string)
Algoritma
        b ← first(list_benua)
        While (b <> NULL) do
                If (info(b).nama_benua == nama_benua) then
                        n ← next_negara(b)
                        While (n <> NULL) do
                                If (info(n).nama_negara == nama_negara) then
                                        If (n == next_negara(b)) then
                                                delete_first_negara(list_benua, b, n)
                                                n ← next_negara(b)
                                        Else if (next(n) == NULL) then
                                                delete_last_negara(list_benua, b, n)
                                                n ← next_negara(b)
                                        Else then
                                                delete_after_negara(list_benua, b, n, n)
                                                n ← next_negara(b)
                                        Endif
                                Endif
                                n ← next(n)
                        Endwhile
                Endif
                b ← next(b)
        Endwhile
Endprocedure
```

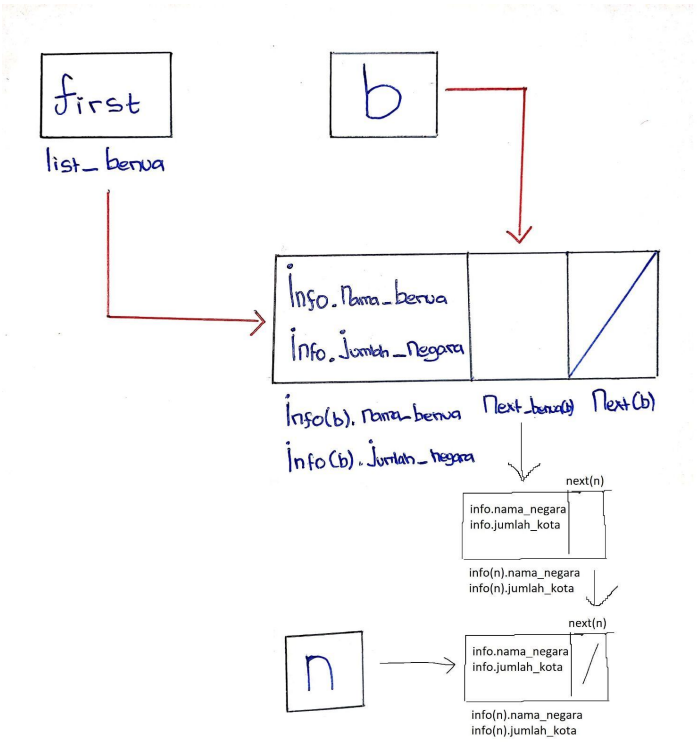
Visual Algoritma:

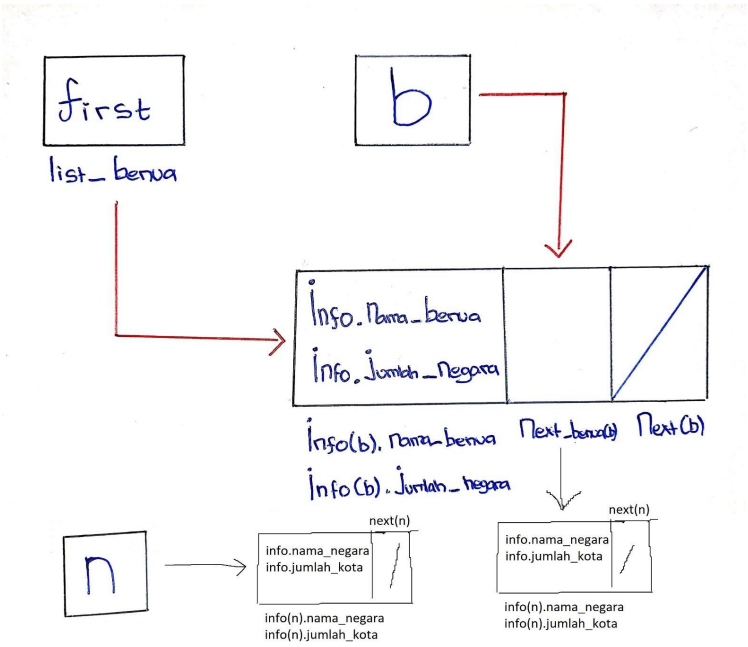
Kondisi If:



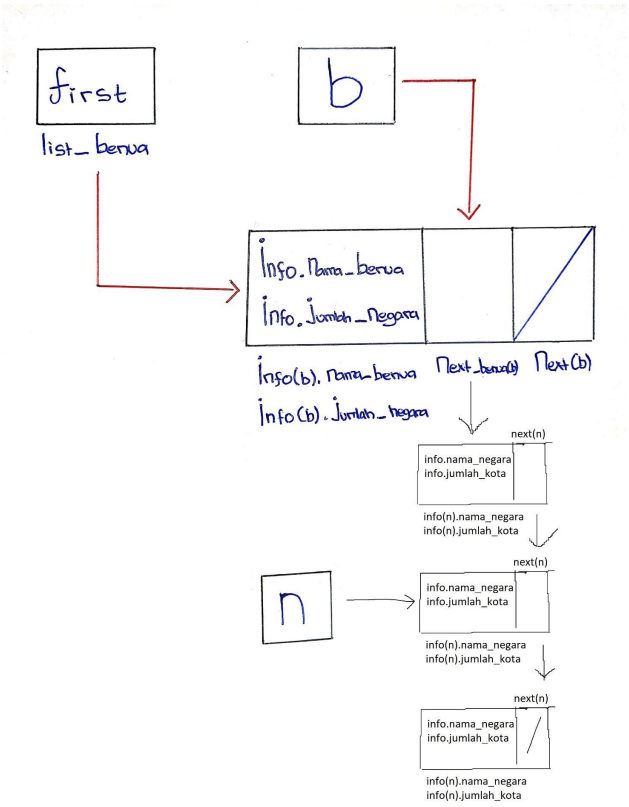


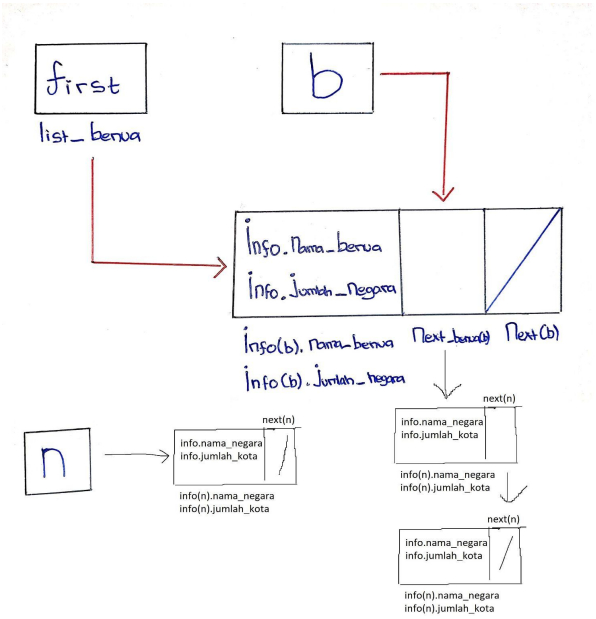
Kondisi Else If:





Kondisi Else:





Deskripsi Algoritma:

Procedure delete_negara_of_x merupakan prosedur yang menghapus node child negara yang memiliki info(n).nama_negara sama dengan nama_negara inputan user. Prosedur tersebut memiliki 3 kondisi, jika letak negara berada di paling depan, maka akan menghapus menggunakan prosedur delete_first_negara. Kondisi 2 adalah jika letak negara berada di paling belakang, maka akan menghapus menggunakan prosedur delete_last_negara, dan kondisi terakhir adalah jika letak negara bukan di paling depan dan bukan di paling belakang, maka akan menghapus menggunakan prosedur delete_after_negara.