PERTEMUAN 12
STACK (LANJUT)

A. TUJUAN PEMBELAJARAN

Mampu menjelaskan pengertian dan pembuatan stack,melakukan operasi penyisipan dan penghapusan elemen dalam stack, dapat mengimplementasikan stack pada *Linked List* dalam bahasa pemrograman C++. Dimodul ini anda harus mampu:

Ketepatan dan penguasaan dalam penggunaan konsep *Stack* pada *Linked list* dalam membangun aplikasi dengan bahasa pemrograman *C++,* Latihan Dan tugas.

B. URAIAN MATERI

1. STACK

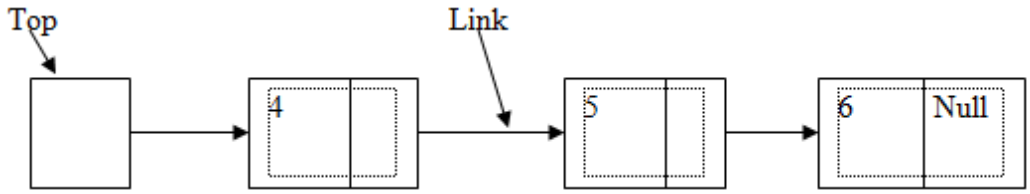
Setelah di sebelumnya diperkenalakan beberapa operasi *Stack* dengan fungsi-fungsi *stack*, Pertemuan ini akan melanjutkan pembahasan tentang *Stack* (Tumpukan) .

Setelah implementasi stack dengan array, stack juga diimplementasikan dengan single linked list. dibandingkan dengan array, Linked List penggunaan alokasi memori yang dinamis sehingga menghindari pemborosan.

Ilustrasi stack dengan array missal disediakan tempat berisi 100 elemen, ketika dipakai hanya diisi 50 elemen, maka pemborosan memori 50 elemen. Dengan linked list maka tempat yang disediakan akan sesuai banyaknya dengan elemen yang mengisi tumpukan. Tidak ada Full didalam tumpukan menggunakan linked list, karena program tidak menentukan jumlah elemen tumpukan kecuali dibatasi oleh memori yang tersedia.

implementasi Stack dengan Linked List

Contoh gambaran stack menggunakan Linked List.



Gambar 12.28 Stack menggunakan Linked List

Contoh implementasi struktur data stack dengan Linked List bahasa C.

```
#include<stdio.h>
#include<stdlibh
#define size 10

Struct stack
        {
                Int data;
                Struct stack*Next;
        };
        Typedef struct stack stk:
        d*tos=Null:
        Void push():
        Int pop();
        Void display();
        Void main();
        Int pilih=0;
        Int val;
```

```
        Do
        {
                Print("----menu----");
                Printf("1.Push data ke dalam tumpukan dan tampilkan");
                Printf(2.Pop data dari dalam tumpukan dan tampilkan");
                Printf("3.Tampilkan");
                Printf("4.Keluar");
                Printf("masukan pilihan);
                Scanf("%d",Pilihan);
                Switch(pilihan)
{
                Kasus 1:        push();
                                Display();
                                Break();
                Kasus 2:        val=Pop();
                                Printf("Pop up value=%d",val);
                                Printf("tumpukan setelah diambil");
                                Display();
                                Break;
                Kasus 3:        display();
                                 Break;
                Kasus 4:        printf("Keluar Dari Program");
                                Break;
        }while(pilihan!=4);
}
Void Push(){
                d*node;
```

```
        Node=(d*)malloc(sizeof(d));
        Printf("masukan data yang dimasukkan di dalam tumpukkan:");
        Scanf("%d",node->data);
        Node->nest=Tos;
        Tos=node;//Top of stack pindah ke node baru
        }
Int Pop(){
        Int val;
        D*temp;
        Temp=tos;// periksa tumpukan apabila kosong
        If(tos==NULL)
        Printf("tumpukkan penuh");
        Exit(0);{
                Else
                {
                        Val=tos->data;
                        Tos=tos->next;
                        Free(temp);}
                Return val:
        }
        Void display(){
                        D*temp;
                        Temp=tos;
                        Printf("tumpukan elemen..")
                        If(temp==NULL)
                        Printf("tumpukkan kosong");
        Else
```

```
{
        While(temp->next!=NULL){
        Printf("%d",temp->data);
}
```

2. Aplikasi Stack

Postfix atau notasi polish sebuah evaluasi expresi matematika diberi ekspresi *Aritmatika Postfix (**'.', ':', '+, '-", '^'**).* Dan juga ada yang memiliki prioritas (makin besar adalah makin tinggi) :

Kondisi Operator :

OPERATOR	ARTI	PRIORITAS
^	PANGKAT	3
***/**	KALI DAN BAGI	2
+ -	TAMBAH DAN KURANG	1

Biasanya kita akan mengkodekan ekspresi kita dalam program menggunakan notasi infix, tetapi compiler memahami dan hanya menggunakan notasi postfix dan oleh karena itu untuk mengubah notasi infix menjadi notasi post fix. Untuk kali ini struktur data stack digunakan dan akan mempertimbangkan mengikuti operator biner dan aturan prioritas tersebut.

Notasi Infix ubah ke Postfix

Contoh notasi *Infix :* A+B*C-D

Berikut cara-cara yang dapat dilakukan*:*

a. Berdasarkan prioritas operator, beri tanda kurung pada expresi, prioritas expresi adalah (*, /) dan (+,-).

   ((A+B(B*C))-D)

b. Periksa tanda kurung apakah benar dan tanda kurung buka tutup cocok.

Tanda kurung buka=tanda kuruk tutup=3

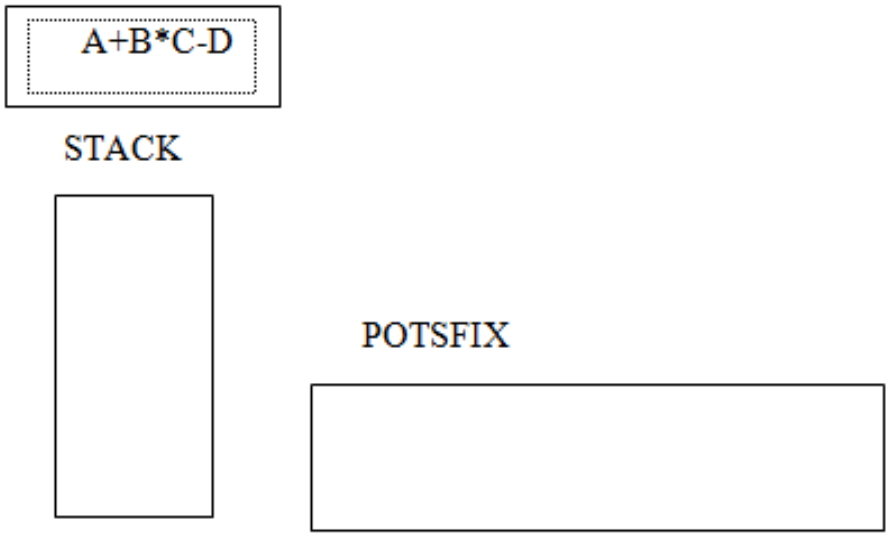
c. Beri tanda kurung di awali dari sisi kanan, dan berikan tanda kurung yang sama dari sisi kiri.

((A+B*C))-D)

12 3 32 1

d. Scan dari sisi kanan ketika memasang tanda kurung. Lanjutkan sampai tanda kurung selanjutnya. Lalu Push variable ke stack.
e. Utuk menutup dan membuka pasangan tanda kurung nomer 2, operator yang mengatur adalah ‘ + ’ lalu push ke dalam stack. Stack sekarang (+ D -) lalu tanda kurung nomer 3 operator yang mengatur adalah ‘ * ’. push ke stack. Stack sekarang (* + D -) lanjut scan akan menemukan variable C, B dan A dan push ke stack. Jadi stack akhir :

Notasi Postfix A B C * + D -



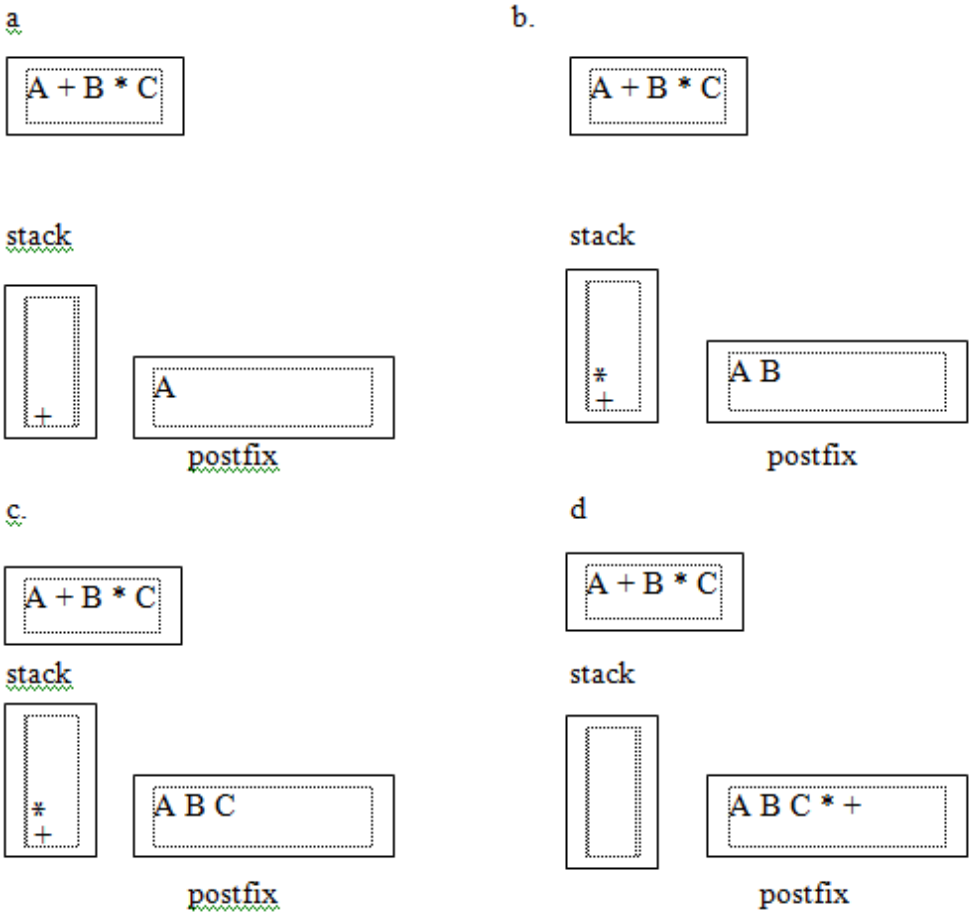
Gambar 12.29 Notasi Postfix

1) Baca pertanyaan dari kiri ke kanan atau dari depan ke belakang.
2) Masuk ke Postfix, apabila Operand.
3) Apabila operator :
   a) Push ke Tuumpukan(Stack), apabila kosong.
   b) Push operator soal ke Tumpukan(Stack) apabila derajat operator soal lebih tinggi derajat operator TOS (Top Of Stack).

c) Jika selama derajat operator soal lebih kecil sama dengan derajat TOS.
   (1) Pop Tos di masukan ke dalam Postfix,
   (2) Apabila sudah, Push operator soal ke Stack.

4) Apabila semua sudah, pop isi tumpukan(Stack) dan Push di masukkan ke Postfix secara berurtutan.

Ilustrasi infix to postfix



Gambar 12.30 Infix ke Postfix

Sekarang coba semua masalah dalam latihan sebelum melanjutkan menganalisa dalam bahasa c , algoritmanya sebagai berikut :

Cara 1 : lakukan untuk setiap karakter dalam string infix ulangi langkah 2 sampai 5.

Cara 2 : if (karakter = = operand)

Tambahkan ke string postfix.

Cara 3 : if (karakter= = “(“)

Push ke dalam stack.

Cara 4 : if (karakter = = operator)

{ while (prioritas dari operator top of stack>=prioritas dari operator)

{pop ()

Tambah ke string postfix

}

Push(operator)

}

Cara 5 : if(simbol == “)”)

{

While(pop()!=”(“)

{

Pop()

Tambah ke srign Postfix

}

//penurunan “(“tanda kurung

}

Cara 6 : apabila string input mengakhiri pop dan menambahkan konten ke string output.

3. Postfix Evaluator

struktur data stack dapat digunakan untuk mengevaluasi ekspresi notasi posfix. Disini akan memberikan contoh algoritma dan kode. menganalisis dan memahami algoritma dan kode. perhatikan bahwa algoritma ini digunakan oleh penyusun untuk mengevaluasi ekspresi yang ditulis dalam di kode Anda.

Contoh evaluator dari notasi expresi postfix.

Notasi infix : A + B * C –D

Notasi Postfix : A B C * + D –

Nilai A= 2, B= 3, C= 4, dan D= 5 ;

Ekspresi di atas akan menghasilkan 2 + * 4 – 5 = 9 mari lihat bagaimana stack menjalankan tugasnya…

Cara 1 : tiga symbol pertama adalah operand A B C i.e 2, 3, 4 ; karena itu push kedalah stack , tambahan stack adalah..

C B A i.e 4 3 2 ⟶

Cara 2 : lalu ada * karena itu pop dua operand paling banyak di stack dan melakukan operasi push hasilny akan ke stack.

Operand poped : C & B i.e 4 * 3 = 12

Push hasilnyake stack.

Penambahan ke dalam stack adalah..

1 2 2 ⟶

Cara 3 : lalu ada +,

POP 1 2 & 2 hasilnya = 12 + 2

Push ke dalam stack . stack bertambah sekarang akan menjadi : 14

Cara 4 : selanjutnya D . push ke stack. Penambahan stack menjadi :

5 14 ⟶

Cara 5 : lalu ada - ,

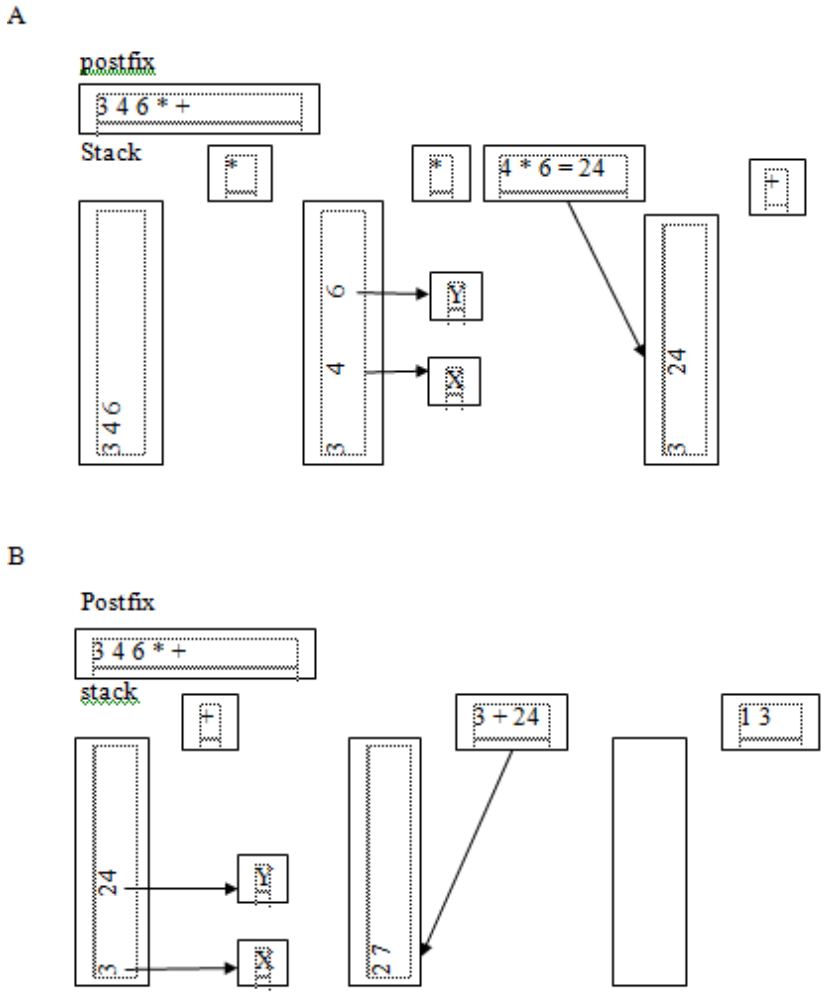
Karena ada penambahan 2 Pop dari stack i.e 5 & 14, lakukan operasi

14 – 5 = 9

Push hasilnya ke dalam stack :

9 ⟶

Ilustrasi postfix evaluator.



Gambar 12.4 Postfix evaluator

a. Scan Postfix dari kiri ke kanan.
b. Ciptakan stack kosong.
c. Apabila soal sebuah operand, tambahkan ke stack. Apabila operator , minimal dua operand dalam stack.

   Pop dua kali stack , pop pertama disimpan dalam Y, dan Pop ke dua dismpan dalam X , selanjutnya dievaluasi X < operator > Y, lalu save hasil selanjutnya push ke dalam stack.

d. Ulangi sampai semua soal discan.
e. Apabila selesai semua, elemen akhir di dalam stack adalah hasil.
f. Apabila lebih dari satu elemen berarti Eror.

Contoh algoritma evaluator Postfix :

1. scan symbol input dari string.

   If (symbol = = operand)

   Push ke dalam stack.

2. else

   {

   If (symbol = = operand)

   Gunakan operasi dua operand

   Didapat dengan dua operasi stack pop berturut-turut

   Push hasilnya kedalam stack.

   }

3. lanjutkan langkah satu dan dua,hingga akhir string input.

C. SOAL LATIHAN/TUGAS

Latihan	Petunjuk Pengerjaan Tugas
Latihan 12	1. Bagaimana prinsip kerja stack a)FIFO b) LIFO? 2. Apakah stack termasuk a)Linear List atau b) non linear? 3. sebutkan operasi didalam stack? 4. .jelaskan perbedaan stack dengan array dengan stack dengan linked list? 5. ubah notasi infix A + B * - D ke notasi postfix?

D. REFERENSI

C And Data Structure by practice by Ramesh Vasappanvara.

Implementasi Stack di C++ | Rahmat Subekti.

Struktur Data Array Stack Dan Queue | Aries Indra Gunawan