Pendahuluan Struktur Data

Nisa'ul Hafidhoh
nisa@dsn.dinus.ac.id
08156114760

Tujuan

- Mahasiswa dapat melakukan pemrograman dalam skala menengah dengan memanfaatkan struktur data internal yang kompleks dan mengimplementasikan dalam bahasa pemrograman yang di pilih.

RENCANA KEGIATAN PERKULIAHAN SEMESTER

W	Pokok Bahasan
1	ADT Stack
2	ADT Queue
3	List Linear
4	List Linear
5	List Linear
6	Representasi Fisik List Linear
7	Variasi List Linear
8	**Ujian Tengah Semester**

W	Pokok Bahasan
9	Variasi List Linear
10	Variasi List Linear
11	Stack dengan Representasi List
12	Queue dengan Representasi List
13	List Rekursif
14	Pohon dan Pohon Biner
15	Multi List
16	**Ujian Akhir Semester**

Deskripsi Tugas

- UTS 30%
- UAS 30%
- Tugas 40%

Outline

- Review tipe data, array, pointer
- Stack

Tipe Data

- Pola representasi suatu data dalam komputer -> menentukan secara internal data disimpan
- Jenis:
  - Tipe Data Dasar / Primitif

    Tipe data yang tersedia / didefinisikan dalam suatu bahasa
  - Tipe Data Bentukan / Komposisi

    Tipe data yang disusun dari berbagai tipe data dasar

Objek Data

- Kumpulan elemen yang mungkin untuk suatu tipe data tertentu
- Contoh:
  - Integer mengacu pada nilai -32768 s/d 32767
  - String adalah kumpulan karakter maks. 255 huruf

Struktur Data

- Cara penyimpanan dan pengorganisasian data pada memory komputer maupun file secara efektif sehingga dapat digunakan secara efisien termasuk operasi di dalamnya
- Di dalam struktur data terdapat 2 aktivitas :
  1. Mendeskripsikan kumpulan objek data yang sesuai dengan tipe data yang ada
  2. Menunjukkan mekanisme kerja operasi-operasinya

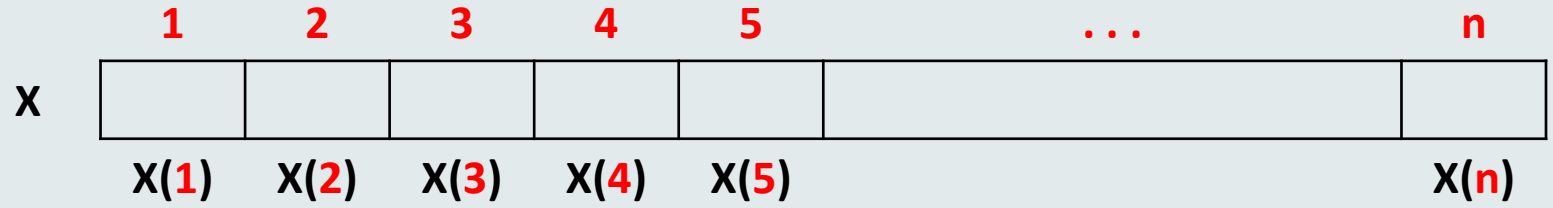
     contoh :

     integer (-32768 s/d 32767) jenis operasi yang diperbolehkan : +, - , *, / , mod,< , >, !=

Struktur Data = Objek Data + Operasi Manipulasi Data

Array

- Tipe yang mengacu kepada sekumpulan elemen yang banyak dan bertipe sama melalui indeks



- Deklarasi:

```
tipe_data nama_var_array [ukuran_indeks];
```

- Akses:

```
nama_var_array [indeks]
```

Contoh Array

- Deklarasi Arr untuk 10 elemen, input & output nilai sesuai data yang dimasukan

```
int main()
{
    int x[10],i,n;
      scanf("%d",&n);

     for(i=0;i<n;i++)
       scanf("%d",&x[i]);

     for(i=0;i<n;i++)
       printf("%d",x[i]);
     return 0;
}
```

Pointer

- Menunjuk kepada nama yang diacu sehingga informasi pada nama dapat diakses
- Memungkinkan alokasi dinamik → memori baru dialokasi berdasarkan kontrol pemrogram.

  jika sudah tidak dibutuhkan, dapat di dealokasi (harus hati - hati)
- Dalam bahasa C, nilai variabel bertype pointer dapat dimanipulasi sebagaimana halnya nilai numerik

Pointer

- Format deklarasi

  <type> * <nama>;

Contoh :

```
int *i;          /*pointer ke integer*/
float *f;        /*pointer ke real*/
char *cc;        /*pointer ke character*/
int *(T)[10]; /*pointer ke array dg 10 elemen integer*/
int *T[10];   /*pointer ke array dg 10 elemen bertype
                 pointer ke integer*/
```

Operator

- Operator address of '&' digunakan untuk mendapatkan alamat memori dari operandnya.

  &almt -> address

- Operator reference '*' digunakan untuk mendapatkan nilai dari operandnya

  *almt -> value

Contoh

```
int main()
{
    int x,y;
    int *ptr;
        ptr=&x;

    printf("%p\n",ptr);
    printf("%p\n",&x);
    printf("%d\n",x);

    y=*ptr;
    printf("%d\n",y);

    *ptr=120;
    printf("%d\n",x);

    ptr=&y;
    printf("%p\n",ptr);

    *ptr=50;
    printf("%d\n",y);

    return 0;
}
```

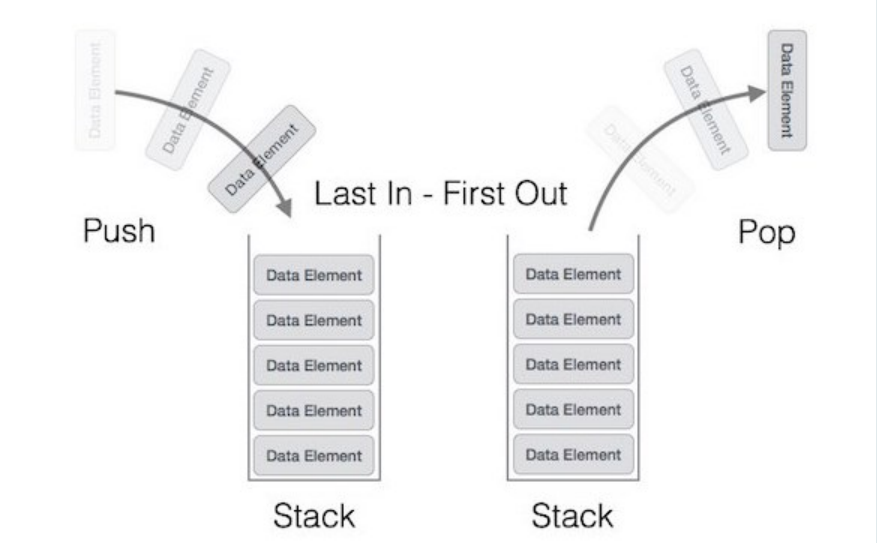
Output?

STACK

- Jenis data abstrak (ADT) yang biasa digunakan dalam bahasa pemrograman.
- Dinamai tumpukan karena berperilaku seperti tumpukan dunia nyata, misalnya - setumpuk kartu atau tumpukan piring dll.
- Pemanfaatan:
  - Dynamic Memory Management
  - Arithmetic Calculation
  - Redo-undo / Forward and backward feature
  - Backtracking Algorithm

Karakteristik

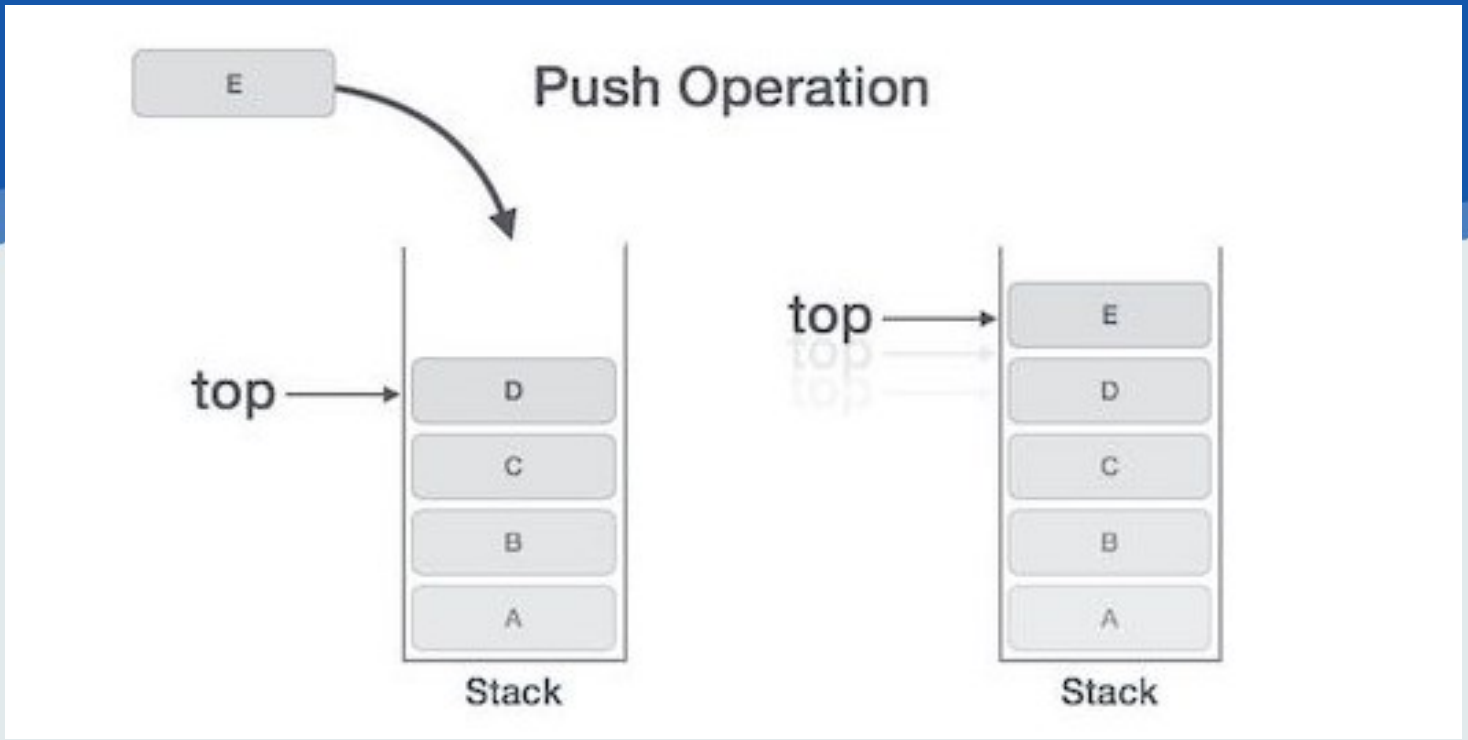
- LIFO (Last In First Out)
- Elemen Paling atas untuk operasi : TOP
- Stack memiliki dua perilaku utama:
  - PUSH: Tambahkan barang ke atas
  - POP: Hapus item di atas



Operasi PUSH

Proses menempatkan elemen data baru ke stack dikenal sebagai Operasi PUSH.

- Langkah 1 – Periksa apakah stack penuh.
- Langkah 2 - Jika tumpukan penuh, menghasilkan kesalahan dan keluar.
- Langkah 3 - Jika stack tidak penuh, kenaikan TOP untuk menunjukkan ruang kosong di sebelah.
- Langkah 4 - Tambahkan elemen data ke lokasi tumpukan, di mana TOP menunjuk.
- Langkah 5 - kembali sukses.



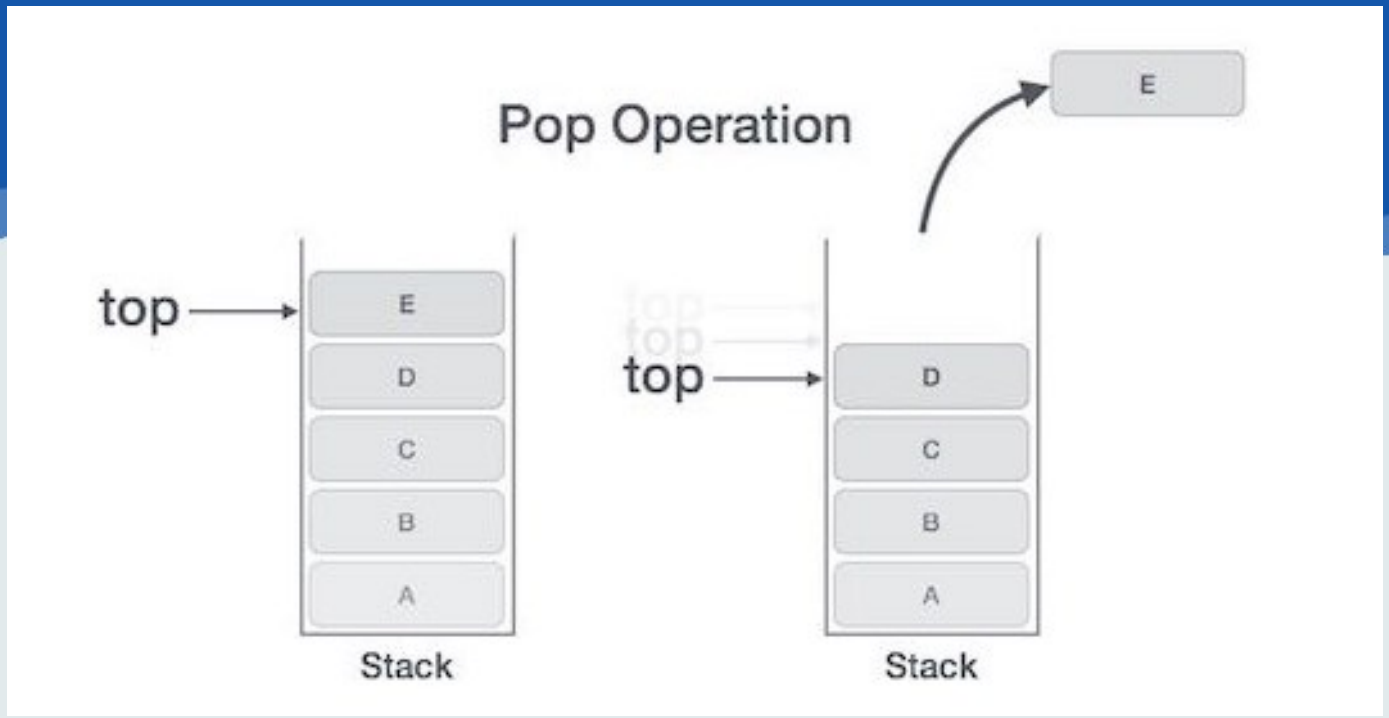
Algoritma untuk operasi PUSH

```
begin procedure push: stack, data

    if stack is full
       return null
    endif

    top ← top + 1

    stack[top] ← data

end procedure
```

Operasi POP

Mengakses konten sementara mengeluarkannya dari tumpukan, yang dikenal sebagai operasi pop.

- Langkah 1 – Periksa apakah stack kosong.
- Langkah 2 - Jika stack kosong, menghasilkan kesalahan dan keluar.
- Langkah 3 - Jika stack tidak kosong, mengakses elemen data di mana TOP menunjuk.
- Langkah 4 - Penurunan nilai TOP dengan 1.
- Langkah 5 - kembali sukses.



Algoritma untuk operasi POP

```
begin procedure pop: stack

   if stack is empty
      return null
   endif

   data ← stack[top]

   top ← top - 1

   return data

end procedure
```

Referensi

- Inggriani Liem, IF-ITB, Diktat Struktur Data (2007)

Terimakasih