PERTEMUAN 3
TIPE DATA

A. TUJUAN PEMBELAJARAN

Setelah mempelajari materi ini diharapkan mahasiswa Mahasiswa mengetahui tentang Tipe data, ADT dan implementasi ADT pada Python.

B. URAIAN MATERI

1. Pengertian Tipe Data

Item data direpresentasikan dalam komputer sebagai urutan digit biner. Ini urutan bisa tampak sangat mirip tetapi memiliki arti yang berbeda sejak komputer dapat menyimpan dan memanipulasi jenis data yang berbeda. Misalnya, biner sequence 01001100110010110101110011011100 bisa berupa string karakter, nilai yang lebih kuat, atau nilai nyata. Untuk membedakan antara berbagai jenis data, file tipe istilah sering digunakan untuk merujuk pada kumpulan nilai dan istilah tipe data untuk merujuk ke tipe yang diberikan bersama dengan kumpulan operasi untuk memanipulasi nilai dari tipe yang diberikan.

Bahasa pemrograman biasanya menyediakan tipe data sebagai bagian dari bahasa diri. Tipe data ini, yang dikenal sebagai primitif, hadir dalam dua kategori: sederhana dan kompleks. Tipe data sederhana terdiri dari nilai yang paling banyak bentuk dasar dan tidak dapat diuraikan menjadi bagian-bagian yang lebih kecil. Bilangan bulat dan tipe nyata, misalnya, terdiri dari nilai numerik tunggal. Tipe data kompleks, disisi lain, dibangun dari beberapa komponen yang terdiri dari tipe sederhana atau tipe kompleks lainnya. Di Python, objek, string, list, dan kamus, yang bisa mengandung banyak nilai, semuanya adalah contoh tipe kompleks. Tipe primitif yang disediakan oleh bahasa mungkin tidak cukup untuk memecahkan masalah kompleks yang besar.

Dengan demikian, sebagian besar bahasa memungkinkan konstruksi tipe data tambahan, yang dikenal sebagai tipe yang ditentukan pengguna karena mereka didefinisikan oleh pemrogram, bukan bahasa. Beberapa dari tipe data ini sendiri bisa sangat kompleks.

2. Tipe data abstrak (ADT)

Tipe data abstrak adalah kelas/ tipe untuk objek yang perilakunya ditentukan oleh satu set nilai dan satu set operasi. Definisi ADT hanya menyebutkan operasi apa yang akan dilakukan, tetapi tidak menyebutkan bagaimana operasi ini akan dilakukan. Itu tidak menentukan bagaimana mengatur data dalam memori dan algoritma apa yang akan digunakan untuk mengimplementasikan operasi. Ini disebut "abstraksi" karena memberikan tampilan yang tidak bergantung pada implementasi. Proses memberikan hanya informasi dasar dan menyembunyikan detailnya disebut abstraksi.

Tipe data abstrak (ADT) adalah model matematika dari objek data yang meningkatkan tipe data dengan mengaitkan tipe data dengan fungsi yang beroperasi pada data terkait. Selain itu, penting untuk disadari bahwa operasi pada data objek yang bersangkutan termasuk dalam spesifikasi ADT. Misalnya, himpunan ADT didefinisikan sebagai kumpulan data yang diakses oleh operasi himpunan (seperti gabungan, perpotongan, dan perbedaan himpunan).

Penggunaan ADT harus menyediakan beberapa cara untuk merepresentasikan elemen dari tipe data (seperti matriks) dan cara untuk mengimplementasikan operasi matriks. Secara umum, akan digunakan beberapa algoritma untuk menggambarkan operasi ADT. Algoritma biasanya merupakan serangkaian instruksi yang digunakan untuk menentukan dengan tepat bagaimana komputer akan melakukan operasi. ADT adalah tipe data spesifik yang didefinisikan oleh programmer untuk memudahkan pemrograman dan beradaptasi dengan tipe data yang tidak secara spesifik disesuaikan dengan bahasa pemrograman yang digunakan. Oleh karena itu, untuk menyatakan ADT secara informal:

a. Tipe data abstrak ADT pertama kali ditemukan oleh ilmuwan komputer dan digunakan untuk memisahkan struktur penyimpanan dari perilaku tipe data abstrak (seperti tumpukan dan antrian). Seperti yang kita harapkan, programmer tidak perlu tahu bagaimana mengubah stack ke implementasi ADTdan tidak mengubah program yang menggunakannya secara keseluruhan., dengan catatan bahwa interface ADT tersebut dengan ‘dunia luar’ tetap dipertahankan.

b. Pemakaian dan pembuatan ADT dapat dilakukan secara terpisah. yang perlu dibicarakan antara pembuat dan pengguna ADT adalah interface ADT yang bersangkutan.

c. ADT adalah suatu alat dalam pengembangkan sistem dimana memiliki sifat modular, yang dapat dimungkinkan suatu sistem dikembangkan beberapa orang anggota tim kerja dimana setiap anggota tim dapat melakukan bagian masing-masing secara tetap mempertahankan keterpaduan dengan anggota tim lainnya.

Dalam hal tersebut diperlukan untuk pembeda antara stuktur data dengan ADT. Struktur data hanya melihatkan bagaimana data di organisis, sedang pada ADT memiliki cakupan yang lebihluas, yakni dengan memuat struktur data yang tertentu sekalian dengan operasi yang dibuat pada struktur data tersebut. Sehingga dapat didefinisikan secara umumya ADT secara luas sebagai berikut:

3. Implementasi ADT

Bahasa pemrograman memiliki bentuk tipe data:

a. Built-in Type

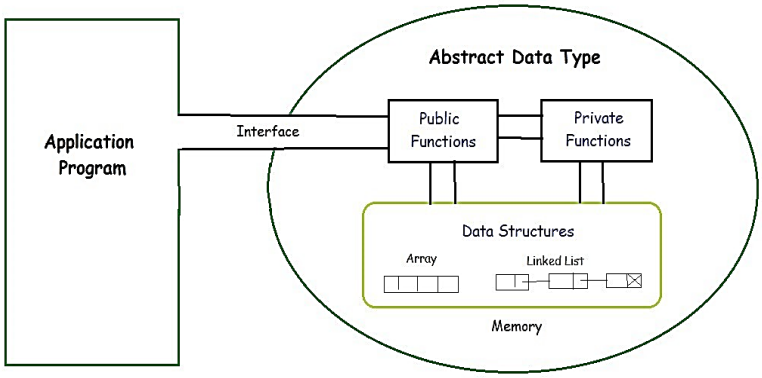
Built-in ini bermakna bahwa bahasa pemograman telah menyediakan tipe data dimana torientasi tidak pada permasalahan yang dihadapi.

b. User Defined Type(UDT)

Pada UDT pemrogram telah melakuakan pembuatan tipe data dimana memiliki kedekatan dalam menyelesaikan masalah yang dihadapi, Mysalnya pada record Pascal, struct di bahasa C, class di bahasa Java.

c. Abstract Data Type(ADT)

Konsep UDT dapat diperluas dengan menambah enkapsulasi diman memiliki isi sifat dan operasi yang dapat dibuat pada kelas tersebut

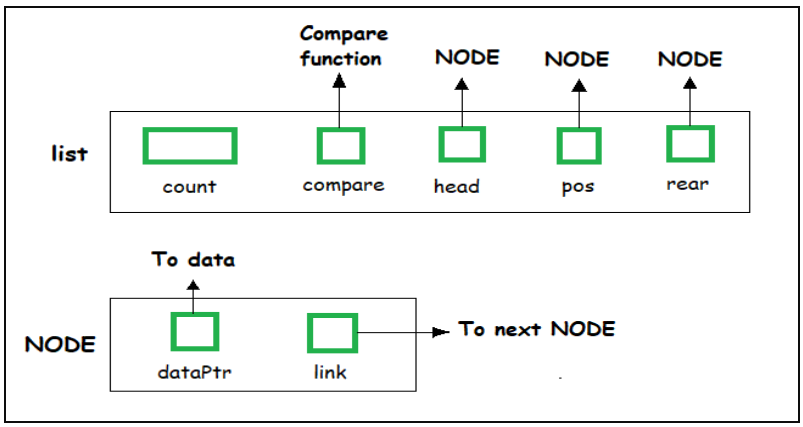


Gambar 3. 1 Implementasi Abstract Data Type

Pengguna tipe data tidak perlu mengetahui bagaimana tipe data tersebut diimplementasikan, misalnya kita telah menggunakan nilai-nilai Primitif seperti tipe data int, float, char hanya dengan pengetahuan bahwa tipe data ini dapat beroperasi dan dijalankan tanpa gagasan tentang bagaimana mereka diimplementasikan. Jadi pengguna hanya perlu tahu apa yang bisa dilakukan oleh tipe data, tetapi tidak bagaimana implementasinya. Pikirkan ADT sebagai kotak hitam yang menyembunyikan struktur dalam dan desain tipe data. Sekarang mengulang kembali pembahasan diatas mendefinisikan tiga ADT yaitu List ADT, Stack ADT, Queue ADT.

a. List ADT

Secara umum data tersimpan pada urutan kunci pada lisy dimana mempunyai bentuk head terdiri berupa nilai, ppinter dan alamat fungsi di buatpembanding sebagai pembanding data pada list.

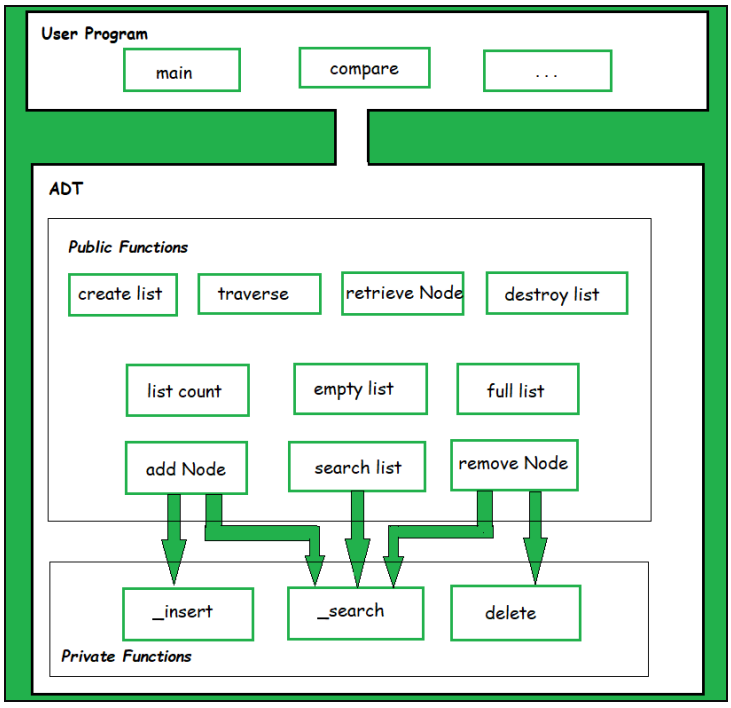


Gambar 3. 2 List ADT

Node data berisi pointer ke struktur data dan pointer self-referensial yang menunjuk ke node berikutnya dalam list .

```
//List  Definisi Tipe ADT
simpul struktur typedef
{
 batal *DataPtr;
 struct simpul *tautan;
} simpul;
struktur typedef
{
 jumlah int;
 simpul *pos;
 simpul * kepala;
 Simpul *belakang;
 int (*bandingkan) (batal *argumen1, batal *argumen2)
} LIST ;
```

List Fungsi ADT diberikan di bawah ini:



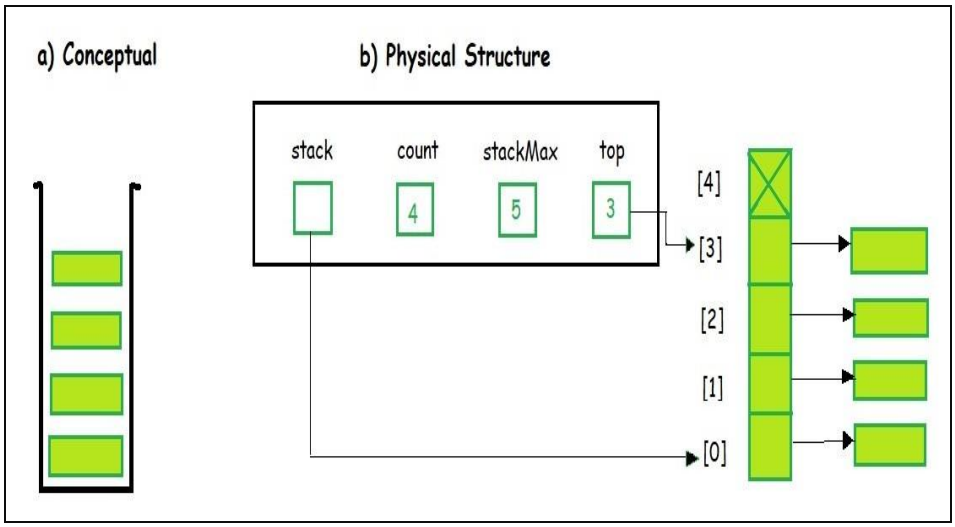
Gambar 3. 3 List Fungsi ADT

List berisi elemen dari jenis yang sama yang diatur dalam urutan berurutan dan operasi berikut dapat dilakukan pada list .

1) Get() – Mengembalikan elemen dari list pada posisi tertentu.
2) Insert() – Menyisipkan elemen pada posisi apapun dari list .
3) Remove() – Hapus kemunculan pertama elemen apa pun dari list yang tidak kosong.
4) Removeat() – Menghapus elemen di lokasi tertentu dari list yang tidak kosong.
5) Replace() – Mengganti elemen pada posisi apa pun dengan elemen lain.
6) Size() – Mengembalikan jumlah elemen dalam list .
7) Isempty() – Mengembalikan nilai true jika list kosong, jika tidak mengembalikan false.
8) Isfull() – Mengembalikan nilai true jika list penuh, jika tidak mengembalikan false.

b. Stack ADT

Dalam Implementasi Stack ADT alih-alih data disimpan di setiap node, penunjuk ke data disimpan. Program mengalokasikan memori untuk data dan alamat diteruskan ke stack ADT.



Gambar 3. 4 Stack ADT

Node kepala dan node data dienkapsulasi dalam ADT. Fungsi panggilan hanya dapat melihat penunjuk ke Stack an. Struktur kepala Stack an juga berisi penunjuk ke atas dan jumlah entri yang saat ini ada di Stack an.

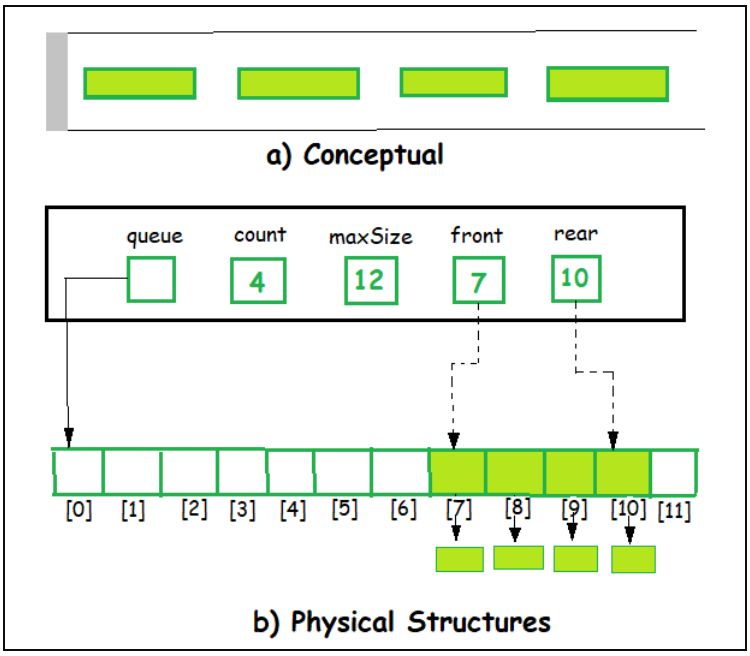
```
// Definisi Tipe ADT Stack
simpul struktur typedef
{
 batal *DataPtr;
 struct simpul *tautan;
} StackNode;
struktur typedef
{
 jumlah int;
 StackNode *atas;
} STACK AN;
```

Stack berisi elemen dengan tipe yang sama yang disusun secara berurutan. Semua operasi berlangsung di satu ujung yang berada di atas Stack an dan operasi berikut dapat dilakukan:

1) push() – Masukkan elemen di salah satu ujung Stack an yang disebut top.
2) pop() – Hapus dan kembalikan elemen di bagian atas Stack an, jika tidak kosong.
3) peek() – Mengembalikan elemen di bagian atas Stack an tanpa menghapusnya, jika Stack an tidak kosong.
4) size() – Mengembalikan jumlah elemen dalam Stack an.
5) isEmpty() – Mengembalikan nilai true jika Stack an kosong, jika tidak mengembalikan false.
6) isFull() – Mengembalikan nilai true jika Stack an penuh, jika tidak mengembalikan false.

c. Queue ADT

Tipe data abstrak Queue (ADT) mengikuti desain dasar tipe data abstrak Stack an.



Gambar 3. 5 Queue ADT

Setiap node berisi pointer kosong ke data dan pointer link ke elemen berikutnya dalam Queue. Tanggung jawab program adalah mengalokasikan memori untuk menyimpan data.

```
//Definisi Jenis ADT Queue
simpul struktur typedef
{
 batal *DataPtr;
 struct simpul *berikutnya;
} Node Queue;
struktur typedef
{
 QueueNode *depan;
 QueueNode *belakang;
 jumlah int;
} ANTRE;
```

Queue berisi elemen dari jenis yang sama diatur dalam urutan berurutan. Operasi berlangsung di kedua ujungnya, penyisipan dilakukan di ujung dan penghapusan dilakukan di depan. Operasi berikut dapat dilakukan:

1) Enqueue() – Menyisipkan elemen di akhir Queue.

2) Dequeue() – Hapus dan kembalikan elemen pertama dari Queue, jika Queue tidak kosong.

3) Peek() – Mengembalikan elemen Queue tanpa menghapusnya, jika Queue tidak kosong.

4) Size() – Mengembalikan jumlah elemen dalam Queue.

5) Isempty() – Mengembalikan nilai true jika Queue kosong

4. Jenis Struktur Data

Struktur data merupakan bagian dari bentuk suatu sistem program contohnya pada struktur maupun array dimana penerapan menginterprestasikan nilai data, hal tersebut dapat menbuat operasi yang ada menjadi spesifik dapat dilakukan terhadap data pada program.secara global jenis pada struktur data terbagai menjadi dua yakni Jenis data premitif diantaranya

a. Interger
b. Real
c. Boolean
d. Karakter
e. String dimana tipe struktur data ini memiliki jenis data campuran

Penjelasan jens data datas dibawah ini.

a. Bilangan Bulat/Integer

(....., -(n+1), -n, ..., -2, -1, 0, 1, 2, ..., n, n+1, ...)

Operasi mendasar terdapat pada interger diantaranya yaitu:

1) Jumlah

2) Kurang

3) Kali

4) Bagi

5) Pangkat

Data numerik selain Interger yang di kelompokkan dalam jenis data rea dimana ditulis dengan titik atau koma pada desimal. Bilangan reak tergolong pada memory komputer dimana menggunakan sistem floating point dimana versi itu sebagai Scientific Natation. Penyajian disini terdapat 2 yakni pecahan dan eksponen. Contoh pada interter dibawah ini.

Dalam sistem desimal, 456000 =0.456& 10^6

Dalam contoh tersebut 0.456 ialah pecahan sedang angka enam merupan eksponen, pada umumnya suatu bilangan real X dapat ditulis dengan M*RE sebagai(M=Mentisa/Pecahan, RE=real)

b. Boolean

Nama lain dari Boolean disebut dengan Logical. Unsur yang terdapat pada boolean memiliki satu dari True aytau False. Operator yang dengan Boolean pada jenis data terdiri dari:

1) Operator OR sebagai penghasil TRUE, apabila satu aatu keduannya memiliki nilai TRUE
2) Operator And sebagai penghasil TRUE, apabila AND dan OR memiliki FALSE atau kebalikannya
3) Operator NOT adalah Precedence yang berasal dari AND dan OR

Pada Suatu ekspresi dimana tidak menggunakan tanda kurung, operator NOT harus dievaluasi sebelum operator AND dan OR. Pada operator Relasional, yaitu : >, <, >=, <=, <> dan =.

c. Karakter dan String

Jenis data karakter merupakan elemen dari suatu himpunan yang terdiri atas bilangan, abjad dan simbol-simbol khusus. Sedangkan jenis data string merupakan jenis data campuran, karena elemen-elemennya dibentuk dari karakter-karakter di atas. Karakter yang digunakan untuk membentuk suatu string disebut sebagai alphabet. Dalam penulisannya, suatu string berada dalam tanda "aphosthrope".

Contoh :

Terdapat suatu himpunan alphabet A = { A, B, 2 }.

String yang dapat dibentuk dari alphabet tersebut yaitu 'AB2', 'ABB', 'BBA', 'ABA2', dan lainnya , termasuk "null string" atau "empty string". Himpunan yang memiliki anggota dari keseluruhan string dimana berasal dari himpunan alphabet yaang dikenaldengan istilah “ Vocabulary”. Suatu Vocabulari V dimana menghasilkan himpunan alphabet A dinotasi dengan VA atau A*, apabi;a string juga terbentul dari alphabet tersebut.

5. Memilih Struktur Data

Implementasi tipe data abstrak yang kompleks biasanya membutuhkan penggunaan struktur data untuk mengatur dan mengelola pengumpulan item data. Ada banyak struktur yang berbeda untuk dipilih. Jadi bagaimana kita tahu harus ke mana menggunakan? Kita harus mengevaluasi kesesuaian struktur data untuk mengimplementasikan diberi tipe data abstrak, yang kami berdasarkan pada kriteria berikut:

a. Apakah struktur data menyediakan persyaratan penyimpanan seperti yang ditentukan oleh domain dari ADT(Abstract Data Type)? Tipe data abstrak didefinisikan untuk bekerja dengan sebuah domain spesifik dari nilai data. Struktur data yang kita pilih harus mampu menyimpan semua kemungkinan nilai dalam domain itu, dengan mempertimbangkan semua batasan atau batasan yang ditempatkan pada masing-masing item.

b. Apakah struktur data menyediakan akses dan manipulasi data yang diperlukan fungsionalitas untuk sepenuhnya menerapkan ADT(Abstract Data Type)? Fungsionalitas abstrak tipe data disediakan melalui rangkaian operasi yang ditentukan. Struktur datanya harus memungkinkan implementasi ADT(Abstract Data Type) secara penuh dan benar tanpa harus untuk melanggar prinsip abstraksi dengan mengekspos detail implementasi ke pengguna.

c. Apakah struktur data cocok untuk implementasi operasi yang efisien asi? Tujuan penting dalam implementasi tipe data abstrak adalah untuk memberikan solusi yang efisien. Beberapa struktur data memungkinkan lebih banyak implementasi yang efisien daripada yang lain, tetapi tidak setiap struktur data cocok untuk menerapkan setiap ADT. Pertimbangan efisiensi dapat membantu memilih yang terbaik struktur dari beberapa kandidat.

Mungkin ada beberapa struktur data yang cocok untuk mengimplementasikan tipe abstrak, tetapi kami berusaha untuk memilih yang terbaik berdasarkan konteksnya di mana ADT akan digunakan. Untuk mengakomodasi konteks yang berbeda, bahasa perpustakaan biasanya akan menyediakan beberapa implementasi dari beberapa ADT, memungkinkan programmer untuk memilih yang paling tepat. Mengikuti pendekatan ini, kami memperkenalkan sejumlah tipe data abstrak di seluruh teks dan menyajikan beberapa implementasi sebagai struktur data baru diperkenalkan.

Efisiensi implementasi didasarkan pada analisis kompleksitas. Jadi, kami menunda pertimbangan efisiensi implementasi dalam pemilihan struktur data hingga saat itu. Sementara itu, kami hanya mempertimbangkan kesesuaian struktur data berdasarkan penyimpanan dan persyaratan fungsional dari tipe data abstrak. Kami sekarang mengalihkan perhatian kami untuk memilih struktur data untuk menerapkan file bag ADT. Kandidat yang mungkin pada saat ini termasuk list dan kamus struktur. List tersebut dapat menyimpan semua jenis objek yang sebanding, termasuk duplikat. Setiap item disimpan satu per satu, termasuk duplikatnya, yang artinya referensi untuk setiap objek individu disimpan dan kemudian dapat diakses bila diperlukan. Ini persyaratan penyimpanan ADT Bag, membuat list struktur kandidat untuk implementasinya.

Kamus menyimpan pasangan kunci atau nilai di mana komponen kunci harus berada sebanding dan unik. Untuk menggunakan kamus dalam mengimplementasikan ADT Bag, kami harus memiliki cara untuk menyimpan barang duplikat seperti yang dipersyaratkan oleh definisi tipe data abstract. Untuk mencapai ini, setiap item unik dapat disimpan di kunci bagian dari pasangan kunci / nilai dan penghitung dapat disimpan di bagian nilai. Itu counter akan digunakan untuk menunjukkan jumlah kemunculan yang sesuai barang di dalam bag. Ketika item duplikat ditambahkan, penghitung bertambah; kapan duplikat dihapus, penghitung dikurangi.

Baik struktur list maupun kamus digunakan untuk mengimplementasikan Bag ADT. Untuk versi bag yang sederhana, bagaimanapun, list adalah pilihan yang lebih baik sejak saat itu kamus akan membutuhkan ruang dua kali lebih banyak untuk menyimpan isi bag dalam kasus di mana sebagian besar itemnya unik. Kamus itu luar biasa pilihan untuk implementasi variasi bag hitung dari ADT.

Setelah memilih list, kita harus memastikan list tersebut menyediakan sarana untuk mengimplementasikan set lengkap operasi tas. Saat mengimplementasikan ADT, kita harus menggunakan fungsionalitas yang disediakan oleh struktur data yang mendasarinya. Terkadang, opsi ADT identik dengan yang telah disediakan oleh struktur data. Dalam hal ini, file implementasi bisa sangat sederhana dan dapat terdiri dari satu panggilan ke operasi sponding struktur, sementara dalam kasus lain, penggunakan banyak operasi yang disediakan oleh struktur. Untuk membantu memverifikasi

implementasi yang benar dari Bag ADT menggunakan list, dapat menjelaskan bagaimana setiap pengoperasian bag nantinya dilaksanakan:

Bag kosong dapat diwakili oleh list kosong.

a. Ukuran bag bisa ditentukan oleh ukuran list.
b. Menentukan apakah bag berisi barang tertentu dapat dilakukan dengan ekuivalen operasi list. ˆ Ketika item baru ditambahkan ke bag, itu dapat ditambahkan ke akhir aftar karena tidak ada pemesanan khusus barang di dalam bag.
c. Menghapus item dari bag juga dapat ditangani dengan list yang setara operasi.
d. Item dalam list dapat dilintasi menggunakan for loop dan Python menyediakannya iterator yang ditentukan pengguna yang digunakan dengan bag.

Dari list yang diperinci ini, terlihat bahwa setiap operasi bag ADT dapat diimplementasikan disebutkan menggunakan fungsionalitas list yang tersedia. Jadi, list cocok untuk menerapkan bag.

C. LATIHAN SOAL

1. Jelaskan menurut anda tentang tipe data dan manfaatnya!
2. Jelaskandan sebutkan jenis mengenai ADT yang anda ketahui!
3. Bagaimana memilih tipe data yang sesuai dalam stuktur data?
4. Jelaskan proses operasi yang ada dalam ADT stack!
5. Jelaskan dan sebutkan proses operasi yang ada pada ADT queue!

D. REFERENSI

Emi Sita Eriana, A. Z. (2021). *Praktikum Algoritma Dan Pemrograman.* Tangerang Selatan: Unpam Press.

Eriana, E. S. (2020). Pemilihan Ketua Himtif Universitas Pamulang Dengan Metode Simple Additive Weighting (Saw). Jik(Jurnal Ilmu Komputer).

Jodi, U. R. (2020). Algoritma Dan Struktur Data.

Mohamad Aslam Katahman, M. F. (2021). Pembangunan Aplikasi Realiti Terimbuh Untuk Pengenalan Struktur Data. Information Technology And Computer Science.

Nasrullah, A. H. (2021). Implementasi Algoritma Decision Tree Untuk Klasifikasi Produk Laris. Jurnal Ilmiah Ilmu Komputer I.

Peng Qi, Y. Z. (2020). Stanza: A Python Natural Language Processing Toolkit For Many Human Languages.

Pradana Setialana, T. B. (2017). Pencarian Hubungan Kekerabatan Pada Struktur Data Genealogy Dalam Graph Databas.

Ranny Meilisa, D. P. (2020). Model Pembelajaran Flipped Classroom Pada Mata Kuliah Algoritma Dan Struktur Data. Jurnal Ilmiah Pendidikan Dan Pembelajaran (Jipp).

Revanza, M. G. (2020). Struktur Data Dan Bahasa Pemrograman.

Sianipar, R. H. (2013). Pemrograman & Struktur Data C: Belajar Dari Contoh Untuk Programmer Pemula Maupun Berpengalaman. Penerbit Informatika, 2013.

Thanaki, J. (2017). Python Natural Language Processing. Mambai.

Zein, A. (2018). Pendeteksian Kantuk Secara Real Time Menggunakan Pustaka Opencv Dan Dlib Python. Sainstech : Jurnal Penelitian Dan Pengkajian Sains Dan Teknologi.