**TAS**



**NAMA:**

**ARBAI - 223220066**

**TEKNIK INFORMATIKA-S1**

**FAKULTAS TEKNIK DAN INFORMATIKA**

**UNIVERSITAS AKI**

**2023**

1. Array, record maupun pointer dapat digunakan untuk membuat struktur data.

Apa kelebihan dan kekurangan dari masing-masing struktur data tersebut ?Uraikan jawaban anda dengan ilustrasi data !

* Array:
* Kelebihan:

1. Akses elemen secara langsung dengan menggunakan indeks.
2. Penggunaan memori yang efisien karena elemen-elemen disimpan secara berurutan.
3. Pengoperasian dan pencarian data yang cepat jika indeks diketahui.
   * Kekurangan:
4. Ukuran array biasanya tetap dan sulit diubah setelah dideklarasikan.
5. Memori yang dialokasikan harus mencukupi untuk ukuran array yang dideklarasikan, terlepas dari jumlah elemen yang sebenarnya digunakan.
6. Penambahan atau penghapusan elemen memerlukan operasi yang kompleks dan membutuhkan alokasi memori ulang.

Contoh:

Misalkan kita memiliki array bilangan bulat [5, 8, 2, 1, 9]. Untuk mengakses elemen ke-3, kita dapat menggunakan indeks 2 dan menghasilkan nilai 2.

* Record:
  + Kelebihan:

1. Dapat menyimpan beberapa jenis data yang berbeda dalam satu struktur.
2. Memungkinkan pengorganisasian yang terstruktur dan representasi data yang lebih kompleks.
3. Memudahkan pengaksesan dan manipulasi data melalui nama bidang.

* Kekurangan:

1. Memerlukan ruang memori yang lebih besar untuk menyimpan informasi tambahan mengenai setiap bidang.
2. Operasi pencarian atau pembaruan pada bidang-bidang tertentu dalam record mungkin memerlukan waktu yang lebih lama.

Contoh:

Misalkan kita memiliki record "Mahasiswa" yang memiliki bidang "nama" (string) dan "usia" (integer). Dengan menggunakan record, kita dapat menyimpan informasi tentang seorang mahasiswa, seperti Mahasiswa1 = {nama: "John", usia: 20}. Kemudian, kita dapat mengakses atau memodifikasi bidang-bidang tersebut menggunakan nama bidangnya.

* Pointer:
  + Kelebihan:

1. Memungkinkan efisiensi penggunaan memori dengan menghindari penggandaan data yang sama.
2. Memungkinkan manipulasi dan pengubahan data secara efisien dengan mengacu pada alamat memori yang sama.
3. Memungkinkan representasi struktur data yang kompleks dengan referensi saling mengacu satu sama lain.
   * Kekurangan:
4. Dapat menyebabkan kesalahan program jika pointer digunakan secara tidak benar.
5. Memerlukan pengelolaan memori yang hati-hati, termasuk alokasi dan dealokasi yang tepat.

Contoh:

Misalkan kita memiliki variabel integer "a" dengan nilai 5. Kemudian, kita mendeklarasikan pointer "ptr" yang menunjuk ke alamat memori variabel "a". Dengan menggunakan pointer, kita dapat mengakses nilai variabel "a" melalui dereferensi pointer dengan menggunakan tanda asterisk (\*ptr), sehingga jika kita mencetak nilai \*ptr, kita akan mendapatkan nilai 5.

1. Ada berapa macam operasi dapat dilakukan pada File ( berkas ) ?Berikan contohnya masing-masing !
2. Pembukaan File (Opening a File):

Operasi ini digunakan untuk membuka file yang ada di sistem dan membuat koneksi antara program dengan file tersebut. Contoh penggunaannya adalah:

file = open("data.txt", "r

1. Penulisan ke File (Writing to a File):

Operasi ini digunakan untuk menulis data ke dalam file yang sudah dibuka. Contoh penggunaannya adalah:

file.write("Hello, world!")

1. Pembacaan dari File (Reading from a File):

Operasi ini digunakan untuk membaca data dari file yang sudah dibuka. Contoh penggunaannya adalah:

content = file.read()

1. Penutupan File (Closing a File):

Operasi ini digunakan untuk menutup file setelah selesai digunakan. Contoh penggunaannya adalah:

file.close()

1. Pemindahan Posisi dalam File (File Positioning):

Operasi ini digunakan untuk memindahkan posisi pembacaan atau penulisan dalam file. Contoh penggunaannya adalah:

file.seek(0)

1. Pemeriksaan Ketersediaan File (File Existence Checking):

Operasi ini digunakan untuk memeriksa apakah file ada atau tidak di dalam sistem. Contoh penggunaannya adalah:

import os

if os.path.exists("data.txt"):

print("File exists")

else:

print("File does not exist")

1. Penghapusan File (File Deletion):

Operasi ini digunakan untuk menghapus file dari sistem. Contoh penggunaannya adalah:

import os

os.remove("data.txt")

1. Jelaskan Konsep, Fungsi ( Kegunaan ), Deklarasi dan contoh Implementasi

dari Data Pointer dikaitkan implementasinya pada STACK dan QUEUE !

1. Konsep Data Pointer:

Pointer adalah sebuah variabel yang berisi alamat memori dari suatu objek atau variabel lainnya. Dalam konteks struktur data, pointer digunakan untuk mengaitkan dan menghubungkan data dengan cara menunjuk ke alamat memori tempat data tersebut disimpan. Dengan menggunakan pointer, kita dapat mengakses dan memanipulasi data secara efisien dengan mengacu pada alamat memori yang sama.

1. Fungsi (Kegunaan) Data Pointer:
2. Alokasi Memori Dinamis: Pointer memungkinkan alokasi memori dinamis, di mana ruang memori dapat dialokasikan secara dinamis pada saat runtime. Hal ini berguna saat kita perlu mengalokasikan ruang memori sesuai dengan kebutuhan program.
3. Penanganan Struktur Data yang Kompleks: Pointer memungkinkan representasi dan penanganan struktur data yang kompleks, seperti linked list, tree, atau graph. Dengan menggunakan pointer, kita dapat membuat hubungan antara elemen-elemen dalam struktur data tersebut.
4. Penghematan Memori: Dengan menggunakan pointer, kita dapat menghindari penggandaan data yang sama dan menghemat penggunaan memori dengan berbagi referensi ke objek yang sama.
5. Deklarasi Pointer:

Dalam banyak bahasa pemrograman, deklarasi pointer biasanya menggunakan tanda asterisk (\*) sebelum nama variabel. Contoh deklarasi pointer dalam bahasa C adalah sebagai berikut:

int \*ptr;

1. Implementasi Data Pointer pada Stack:

Pada struktur data stack, pointer sering digunakan untuk mengimplementasikan tumpukan (stack) secara efisien. Biasanya, stack direpresentasikan menggunakan array dinamis atau linked list. Dalam implementasi menggunakan pointer, setiap elemen stack memiliki pointer yang menunjuk ke elemen sebelumnya dalam stack.

Contoh implementasi stack menggunakan pointer dalam bahasa C:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node \*top = NULL;

void push(int value) {

struct Node \*newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->next = top;

top = newNode;

}

void pop() {

if (top == NULL) {

printf("Stack is empty.\n");

return;

}

struct Node \*temp = top;

top = top->next;

free(temp);

}

void display() {

struct Node \*temp = top;

printf("Stack: ");

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

int main() {

push(1);

push(2);

push(3);

display(); // Output: Stack: 3 2 1

pop();

display(); // Output: Stack: 2 1

return 0;

}

1. Implementasi Data Pointer pada Queue:

Pada struktur data queue, pointer sering digunakan untuk mengimplementasikan antrian (queue) dengan memanfaatkan linked list. Dalam implementasi menggunakan pointer, setiap elemen queue memiliki pointer yang menunjuk ke elemen berikutnya dalam antrian.

Contoh implementasi queue menggunakan pointer dalam bahasa C:

#include <stdio.h>

#include <stdlib.h>

struct Node {

int data;

struct Node \*next;

};

struct Node \*front = NULL;

struct Node \*rear = NULL;

void enqueue(int value) {

struct Node \*newNode = (struct Node\*)malloc(sizeof(struct Node));

newNode->data = value;

newNode->next = NULL;

if (front == NULL && rear == NULL) {

front = rear = newNode;

return;

}

rear->next = newNode;

rear = newNode;

}

void dequeue() {

if (front == NULL) {

printf("Queue is empty.\n");

return;

}

struct Node \*temp = front;

front = front->next;

if (front == NULL)

rear = NULL;

free(temp);

}

void display() {

if (front == NULL) {

printf("Queue is empty.\n");

return;

}

struct Node \*temp = front;

printf("Queue: ");

while (temp != NULL) {

printf("%d ", temp->data);

temp = temp->next;

}

printf("\n");

}

int main() {

enqueue(1);

enqueue(2);

enqueue(3);

display(); // Output: Queue: 1 2 3

dequeue();

display(); // Output: Queue: 2 3

return 0;

}

1. Bagaimana kita mengatasi masalah penghapusan record pada file (arsip) beruntun ? Berikan penjelasan dan contoh algoritmanya !
2. Marking Deleted Records (Penandaan Record yang Dihapus):

Pada pendekatan ini, record yang ingin dihapus tidak dihapus secara fisik dari file. Sebagai gantinya, kita menandai record tersebut sebagai "dihapus" dengan menggunakan penanda khusus. Dengan demikian, record tersebut tetap ada di file, tetapi dianggap tidak aktif atau tidak digunakan.

1. Baca record yang ingin dihapus.

2. Tandai record sebagai "dihapus" dengan menggunakan penanda khusus (misalnya, bit penanda).

3. Simpan kembali record yang sudah ditandai ke dalam file.

1. Compact File (Kompaksi File):

Pada pendekatan ini, semua record yang tidak dihapus fisik akan digabungkan secara beruntun (berdekatan) untuk mengisi ruang kosong yang dihasilkan oleh record yang dihapus. Dengan demikian, file akan tetap berurutan dan tidak ada record yang dihapus secara fisik.

1. Baca record yang ingin dihapus.

2. Geser record-record yang ada setelah record yang dihapus ke posisi sebelumnya.

3. Simpan kembali record-record yang sudah digeser ke dalam file.

4. Perbarui metadata yang mengindikasikan jumlah record yang aktif.

Pendekatan yang digunakan tergantung pada kebutuhan aplikasi dan kompleksitas file. Pendekatan penandaan record yang dihapus cocok jika operasi penghapusan lebih sering terjadi daripada operasi pencarian atau pengurutan. Sedangkan pendekatan kompaksi file lebih cocok jika operasi pencarian atau pengurutan menjadi prioritas dan penghapusan jarang dilakukan.

Perlu dicatat bahwa kedua pendekatan tersebut tidak menghapus record secara fisik dari file, tetapi mengatur data dalam file agar record yang dihapus tidak lagi dianggap aktif atau digunakan.