Baik, mari kita lihat penjelasan kode di atas secara detail, baris per baris.

### Header dan Deklarasi Struct

```c

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

```

- `#include <stdio.h>`: Mengimpor pustaka standar untuk fungsi input-output (`printf`, `scanf`, dll.).

- `#include <stdlib.h>`: Mengimpor pustaka standar untuk fungsi manajemen memori (`malloc`, `free`).

- `#include <string.h>`: Mengimpor pustaka string untuk fungsi manipulasi string (`strcpy`, `strcmp`, dll.).

```c

struct Buku {

char judul[50];

char pengarang[50];

struct Buku\* next;

};

```

- `struct Buku`: Mendefinisikan struktur `Buku` yang berfungsi sebagai node dari linked list.

- `char judul[50]`: Menyimpan judul buku sebagai string berukuran 50 karakter.

- `char pengarang[50]`: Menyimpan nama pengarang buku sebagai string berukuran 50 karakter.

- `struct Buku\* next`: Pointer yang menunjuk ke buku berikutnya dalam linked list.

### Fungsi `buatBukuBaru`

```c

struct Buku\* buatBukuBaru(const char\* judul, const char\* pengarang) {

struct Buku\* bukuBaru = (struct Buku\*)malloc(sizeof(struct Buku));

strcpy(bukuBaru->judul, judul);

strcpy(bukuBaru->pengarang, pengarang);

bukuBaru->next = NULL;

return bukuBaru;

}

```

- `struct Buku\* buatBukuBaru(const char\* judul, const char\* pengarang)`: Fungsi ini membuat node buku baru dan mengembalikannya.

- `struct Buku\* bukuBaru = (struct Buku\*)malloc(sizeof(struct Buku))`: Mengalokasikan memori untuk node buku baru dan mengembalikannya sebagai pointer `bukuBaru`.

- `strcpy(bukuBaru->judul, judul)`: Menyalin nilai `judul` yang diberikan ke anggota `judul` pada `bukuBaru`.

- `strcpy(bukuBaru->pengarang, pengarang)`: Menyalin nilai `pengarang` yang diberikan ke anggota `pengarang` pada `bukuBaru`.

- `bukuBaru->next = NULL`: Menetapkan pointer `next` pada `bukuBaru` ke `NULL`, karena buku ini akan menjadi node terakhir.

- `return bukuBaru`: Mengembalikan pointer `bukuBaru`.

### Fungsi `tambahBuku`

```c

void tambahBuku(struct Buku\*\* head, const char\* judul, const char\* pengarang) {

struct Buku\* bukuBaru = buatBukuBaru(judul, pengarang);

if (\*head == NULL) {

\*head = bukuBaru;

} else {

struct Buku\* temp = \*head;

while (temp->next != NULL) {

temp = temp->next;

}

temp->next = bukuBaru;

}

printf("Buku '%s' oleh %s berhasil ditambahkan.\n", judul, pengarang);

}

```

- `void tambahBuku(struct Buku\*\* head, const char\* judul, const char\* pengarang)`: Fungsi untuk menambahkan buku ke akhir linked list.

- `struct Buku\* bukuBaru = buatBukuBaru(judul, pengarang)`: Membuat node buku baru dengan judul dan pengarang yang diberikan.

- `if (\*head == NULL)`: Mengecek apakah linked list kosong.

- `\*head = bukuBaru`: Jika kosong, buku baru menjadi head dari linked list.

- `else`: Jika linked list tidak kosong, cari node terakhir.

- `struct Buku\* temp = \*head`: Menyimpan pointer awal linked list dalam variabel `temp`.

- `while (temp->next != NULL)`: Loop hingga `temp` menunjuk ke node terakhir (yang `next`-nya `NULL`).

- `temp->next = bukuBaru`: Menambahkan node baru di akhir linked list.

- `printf(...)`: Menginformasikan bahwa buku berhasil ditambahkan.

### Fungsi `hapusBuku`

```c

void hapusBuku(struct Buku\*\* head, const char\* judul) {

struct Buku\* temp = \*head;

struct Buku\* prev = NULL;

if (temp != NULL && strcmp(temp->judul, judul) == 0) {

\*head = temp->next;

free(temp);

printf("Buku '%s' berhasil dihapus.\n", judul);

return;

}

while (temp != NULL && strcmp(temp->judul, judul) != 0) {

prev = temp;

temp = temp->next;

}

if (temp == NULL) {

printf("Buku '%s' tidak ditemukan.\n", judul);

return;

}

prev->next = temp->next;

free(temp);

printf("Buku '%s' berhasil dihapus.\n", judul);

}

```

- `void hapusBuku(struct Buku\*\* head, const char\* judul)`: Fungsi ini menghapus buku berdasarkan judulnya.

- `struct Buku\* temp = \*head`: Memulai pencarian dari `head`.

- `struct Buku\* prev = NULL`: Variabel `prev` untuk menyimpan node sebelumnya.

- `if (temp != NULL && strcmp(temp->judul, judul) == 0)`: Jika buku pertama adalah buku yang akan dihapus:

- `\*head = temp->next`: Memindahkan head ke node berikutnya.

- `free(temp)`: Menghapus node dari memori.

- `printf(...)` dan `return`: Menampilkan pesan dan mengakhiri fungsi.

- `while (temp != NULL && strcmp(temp->judul, judul) != 0)`: Loop untuk mencari buku sesuai judulnya.

- `prev = temp`: Menyimpan node sebelumnya.

- `temp = temp->next`: Melanjutkan ke node berikutnya.

- `if (temp == NULL)`: Jika tidak ditemukan, tampilkan pesan dan keluar.

- `prev->next = temp->next`: Menghubungkan node sebelumnya dengan node berikutnya, melewati node yang akan dihapus.

- `free(temp)`: Menghapus node yang sesuai dari memori.

- `printf(...)`: Menampilkan pesan sukses.

### Fungsi `tampilkanBuku`

```c

void tampilkanBuku(struct Buku\* head) {

struct Buku\* temp = head;

if (temp == NULL) {

printf("Daftar buku kosong.\n");

return;

}

printf("Daftar Buku:\n");

while (temp != NULL) {

printf("Judul: %s, Pengarang: %s\n", temp->judul, temp->pengarang);

temp = temp->next;

}

}

```

- `void tampilkanBuku(struct Buku\* head)`: Fungsi ini menampilkan seluruh buku dalam linked list.

- `struct Buku\* temp = head`: Memulai dari node pertama (head).

- `if (temp == NULL)`: Jika list kosong, tampilkan pesan.

- `printf("Daftar Buku:\n")`: Mencetak header daftar buku.

- `while (temp != NULL)`: Loop selama `temp` tidak `NULL`.

- `printf(...)`: Menampilkan judul dan pengarang buku.

- `temp = temp->next`: Beralih ke node berikutnya.

### Fungsi `main`

```c

int main() {

struct Buku\* head = NULL;

int pilihan;

char judul[50], pengarang[50];

do {

printf("\nMenu:\n");

printf("1. Tambah Buku\n");

printf("2. Hapus Buku\n");

printf("3. Tampilkan Daftar Buku\n");

printf("4. Keluar\n");

printf("Pilih opsi: ");

scanf("%d", &pilihan);

getchar();

switch (pilihan) {

case 1:

printf("Masukkan Judul Buku: ");

fgets(judul, sizeof(judul), stdin);

judul[strcspn(judul, "\n")] = '\0';

printf("Masukkan Nama Pengarang: ");

fgets(pengarang, sizeof(pengarang), stdin);

pengarang[strcspn(pengarang, "\n")] = '\0';

tambahBuku(&head, judul, pengarang);

break;

case 2:

printf("Masukkan Judul Buku yang akan dihapus: ");

fgets(judul, sizeof(judul), stdin);

judul[strcspn(judul, "\n")] = '\0';

hapusBuku(&head, judul);

break;

case 3:

tampilkanBuku(head);

break;

case 4:

printf("Keluar dari program.\n");

break;

default:

printf("Pilihan tidak valid.\n");

}

} while (pilihan != 4);

struct Buku\* temp;

while (head != NULL) {

temp = head;

head = head->next;

free(temp);

}

return 0;

}

```

- `struct Buku\* head = NULL`: Pointer untuk menyimpan head dari linked list.

- `int pilihan`: Variabel untuk menyimpan pilihan pengguna.

- `do ... while`: Menampilkan menu hingga pengguna memilih opsi "Keluar".

- `switch (pilihan)`: Mengeksekusi pilihan:

- `1`: Meminta input untuk judul dan pengarang, lalu menambahkannya.

- `2`: Meminta judul buku yang akan dihapus.

- `3`: Menampilkan daftar buku.

- `4`: Keluar program dan membersihkan memori.