## Studi Kasus

Dengan adanya internet, di masa sekarang seseorang tidak perlu membutuhkan sebuah toko fisik untuk mulai berbisnis. Banyaknya media sosial dan platform penjualan online memungkinkan munculnya berbagai macam toko online. Di toko online, pembeli tidak perlu datang secara langsung ke toko, melainkan hanya perlu memilih produk yang diinginkan melalui gawai masing-masing, dan melakukan order atau pemesanan kepada penjual. Produk kemudian akan dikirimkan penjual ke alamat pembeli.

Dari sisi penjual juga terdapat banyak kemudahan. Penjual tidak perlu memiliki toko fisik dan memajang barang-barang, melainkan hanya perlu mengiklankan produk di toko online. Setelah itu, penjual hanya perlu menunggu pemesanan dari pelanggan dan nantinya mengirimkan pesanan tersebut. Namun, dengan tidak adanya toko fisik juga sangat dimungkinkan penyimpanan barang menjadi tidak terorganisir. Oleh karena itu, diperlukan sistem yang dapat mencatat semua barang-barang yang dijual oleh toko online. Selain itu, diperlukan juga sistem untuk menangani pemesanan dari pembeli, sebab tentu saja ada batasan mengenai seberapa banyak pesanan yang dapat ditangani oleh pembeli dalam satu hari.

## Rancangan Program

Dari studi kasus di atas, dibuat sebuah program untuk menangani penyimpanan barang yang dijual serta pemesanan barang. Sistem penyimpanan barang akan ditangani dengan menggunakan struktur data linked list. Sistem pemesanan barang akan ditangani dengan menggunakan struktur data queue.

Program akan terdiri dari dua sisi, yakni sisi penjual dan sisi pembeli. Sisi penjual akan memuat menu seperti melihat daftar produk, menambah produk ke daftar, menghapus produk dari daftar, melihat daftar antrian, serta memroses antrian terdepan. Sisi pembeli akan memuat menu melihat daftar produk dan memesan produk (menambahkan antrian).

Antrian pemesanan akan dibatasi dengan jumlah yang sudah ditentukan sebelumnya. Apabila panjang antrian sudah mencapai batas, maka otomatis pembeli tidak bisa menambah pemesanan baru.

Program akan memiliki tiga buah struct. Struktur data linked list akan menggunakan sebuah struct sebagai masing-masing node yang ada di dalamnya. Struktur data queue akan menggunakan dua buah struct. Sebuah struct sebagai masing-masing node yang terdapat dalam queue dan sebuah struct sebagai antrian itu sendiri.

## Algoritma

1. Fungsi main
2. Mulai
3. Deklarasi struct ListNode dengan member:

* nama dengan tipe data string
* stok dengan tipe data integer
* pointer next dengan tipe data ListNode

1. Deklarasi struct QueueNode dengan member:

* namaBarang dengan tipe data string
* namaPembeli dengan tipe data string
* pointer next dengan tipe data QueueNode

1. Deklarasi struct Queue dengan member:

* length dengan tipe data integer
* pointer front dengan tipe data QueueNode
* pointer rear dengan tipe data QueueNode

1. Deklarasi variabel pointer pHead dengan tipe data struct ListNode dan inisialisasi dengan NULL
2. Deklarasi variabel pQueue dengan tipe data struct Queue
3. Inisialisasi member length dengan 0
4. Inisialisasi member front dan rear dengan NULL
5. Looping: selama pilihan != ‘q’ lakukan:
6. Tampilkan menu penjual atau pembeli
7. Minta masukkan untuk variabel pilihan
8. Periksa:
   * Jika pilihan == ‘1’, looping: selama option != ‘q’ lakukan:
9. Tampilkan menu untuk penjual
10. Minta masukkan untuk variabel option
11. Panggil fungsi yang sesuai dengan option berdasarkan menu yang ada
    * Jika pilihan == ‘2’, looping: selama option != ‘q’ lakukan:
12. Tampilkan menu untuk pembeli
13. Minta masukkan untuk variabel option
14. Panggil fungsi yang sesuai dengan option berdasarkan menu yang ada
15. Selesai
16. Fungsi addBarang
17. Mulai, menerima parameter pointer to pointer pHead
18. Deklarasi variabel nama dan masukkanSetelah dengan tipe data string
19. Deklarasi variabel stok dengan tipe data integer
20. Deklarasi variabel pointer pNew dan pointer pCur dengan tipe data struct ListNode
21. Minta masukkan untuk variabel namaBarang
22. Minta masukkan untuk variabel stok
23. Alokasikan memori ke pNew
24. Jika pNew == NULL, tampilkan pesan error bahwa memori tidak cukup dan kembalikan fungsi
25. Simpan nilai dari variabel nama ke member nama dari pNew
26. Simpan nilai dari variabel stok ke member stok dari pNew
27. Inisialisasi member next dari pNew dengan NULL
28. Dereference pHead dan simpan nilainya ke pCur
29. Jika pCur == NULL:

* Simpan nilai dari pNew ke dereference dari pHead
* Kembalikan fungsi

1. Panggil fungsi traverseList dan masukkan dereference dari pHead sebagai parameter
2. Minta masukkan untuk variabel masukkanSetelah
3. Looping: selama pCur != NULL dan member nama dari pCur != masukkan setelah lakukan:

* Simpan nilai member next dari pCur ke variabel pCur

1. Jika pCur == NULL tampilkan pesan error bahwa barang tidak ditemukan dan kembalikan fungsi
2. Jika member next dari pCur == NULL, maka masukkan NULL ke member next dari pNew
3. Jika kondisi pada poin (18) salah, maka masukkan member next dari pCur ke member next dari pNew
4. Masukkan pNew ke member next dari pCur
5. Selesai
6. Fungsi hapusBarang
7. Mulai, menerima variabel pointer to pointer pHead dengan tipe data struct ListNode
8. Deklarasi variabel pointer pPrev dan variabel pointer pCur dengan tipe data struct ListNode
9. Deklarasi variabel namaBarang dengan tipe data string
10. Inisialisasi pCur dengan dereference dari pHead
11. Inisialisasi pPrev dengan NULL
12. Jika pCur == NULL tampilkan pesan error bahwa daftar produk kosong dan kembalikan fungsi
13. Panggil fungsi traverseList, masukkan dereference dari pHead sebagai parameter
14. Minta masukkan untuk variabel namaBarang
15. Looping: selama pCur != NULL dan member nama dari pCur != namaBarang lakukan:
16. Masukkan nilai dari pCur ke pPrev
17. Masukkan nilai member next dari pCur ke pCur
18. Jika pCur == NULL tampilkan pesan error bahwa nama barang tidak ditemukan dan kembalikan fungsi
19. Jika pCur == dereference dari pHead, maka:
20. Masukkan nilai member next dari pCur ke dereference pHead
21. Bebaskan memori pCur
22. Kembalikan fungsi
23. Jika member next dari pCur == NULL, maka:
24. Masukkan NULL ke member next dari pPrev
25. Bebaskan memori pCur
26. Kembalikan fungsi
27. Masukkan member next dari pCur ke member next dari pPrev
28. Bebaskan memori pCur
29. Kembalikan fungsi
30. Selesai
31. Fungsi enQueue
32. Mulai, menerima parameter variabel pointer pQueue dengan tipe data struct Queue dan variabel pointer pHead dengan tipe data struct ListNode
33. Deklarasi variabel MAX dengan tipe data integer dan inisialisasi dengan 5
34. Deklarasi variabel pointer pNew dengan tipe data struct QueueNode
35. Deklarasi variabel pointer pCur dengan tipe data struct ListNode
36. Deklarasi variabel namaBarang dan namaPembeli dengan tipe data string
37. Periksa, jika member length dari pQueue >= MAX, maka tampilkan pesan error bahwa antrian telah penuh dan kembalikan fungsi
38. Panggil fungsi traverseList, masukkan pHead sebagai parameter
39. Minta masukkan untuk variabel namaBarang
40. Minta masukkan untuk variabel namaPembeli
41. Inisialisasi pCur dengan nilai dari pHead
42. Looping, selama pCur != NULL dan member nama dari pCur != namaBarang lakukan:

* Masukkan nilai member next dari pCur ke pCur

1. Jika pCur == NULL, maka tampilkan pesan error bahwa produk tidak tersedia dan kembalikan fungsi
2. Alokasikan memori ke pNew
3. Jika pNew == NULL, tampilkan pesan error bahwa memori tidak cukup dan kembalikan fungsi
4. Masukkan nilai dari namaBarang ke member namaBarang dari pNew
5. Masukkan nilai dari namaPembeli ke member namaPembeli dari pNew
6. Jika member front dari pQueue == NULL, maka:
7. Masukkan pNew ke member front dari pQueue
8. Masukkan pNew ke member rear dari pQueue
9. Inisialisasi member next dari pNew dengan NULL
10. Tambah member length dari pQueue sebanyak 1
11. Kembalikan fungsi
12. Masukkan pNew ke member next dari member rear dari pQueue
13. Masukkan pNew ke member rear dari pQueue
14. Inisialisasi member next dari pNew dengan NULL
15. Tambah member length dari pQueue sebanyak 1
16. Selesai
17. Fungsi deQueue
18. Mulai, menerima parameter variabel pointer pQueue dengan tipe data struct Queue
19. Deklarasi variabel pointer pCur dengan tipe data struct QueueNode
20. Inisialisasi pCur dengan member front dari pQueue
21. Jika pCur == NULL tampilkan pesan error bahwa antrian kosong dan kembalikan fungsi
22. Jika member next dari pCur == NULL, lakukan:
23. Masukkan NULL ke member front dari pQueue
24. Masukkan NULL ke member rear dari pQueue
25. Jika poin (5) bernilai salah, maka masukkan nilai member next dari pCur ke member front dari pQueue
26. Bebaskan memori pada pCur
27. Kurangi member length dari pQueue sebanyak 1
28. Selesai
29. Fungsi traverseList
30. Mulai, menerima parameter pointer pHead dengan tipe data struct ListNode
31. Deklarasi variabel pointer pCur dengan tipe data struct ListNode dan inisialisasi dengan nilai pada pHead
32. Jika pCur != NULL, looping selama pCur != NULL, lakukan:
33. Tampilkan member nama dan stok dari pCur
34. Masukkan nilai member next dari pCur ke variabel pCur
35. Selesai
36. Fungsi traverseQueue
37. Mulai, menerima parameter variabel pQueue dengan tipe data struct Queue
38. Deklarasi variabel pointer pCur dengan tipe data struct QueueNode dan inisialisasi dengan nilai member front dari pQueue
39. Jika pCur != NULL:
    1. Looping selama pCur != NULL, lakukan:
40. Tampilkan member namaBarang dan namaPembeli dari pCur
41. Masukkan nilai dari member next dari pCur ke variabel pCur
    1. Tampilkan member namaBarang dan namaPembeli dari pCur
42. Tampilkan panjang antrian
43. Selesai

## Tautan GitHub

[mdikavw/fp-strudat (github.com)](https://github.com/mdikavw/fp-strudat)

## Kode Program

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

struct ListNode {

char nama[30];

int stok;

ListNode \* next;

};

typedef struct listNode listNode;

void addBarang(ListNode \*\*Head);

struct QueueNode {

char namaBarang[30];

char namaPembeli[30];

QueueNode \* next;

};

typedef struct queueNode queueNode;

struct Queue {

int length;

QueueNode \* front;

QueueNode \* rear;

};

typedef struct Queue Queue;

void traverseList(ListNode \* pHead){

ListNode \* pCur;

pCur = pHead;

if(pCur != NULL){

while(pCur != NULL){

printf("%s(%i) -> ", pCur->nama, pCur->stok);

pCur = pCur->next;

}

}

printf("NULL\n");

return;

}

void addBarang(ListNode \*\* pHead){

char nama[30], masukkanSetelah[30];

int stok;

ListNode \* pNew, \* pCur;

printf("Masukkan nama barang: ");

scanf(" %s", nama);

printf("Masukkan stok barang: ");

scanf("%i", &stok);

pNew = (ListNode \*)malloc(sizeof(ListNode));

if(pNew == NULL){

printf("Memori tidak cukup\n");

system("pause");

return;

}

strcpy(pNew->nama, nama);

pNew->stok = stok;

pNew->next = NULL;

pCur = \*pHead;

if(pCur == NULL){

\*pHead = pNew;

return;

}

traverseList(\*pHead);

printf("Masukkan barang setelah (ketik 'depan' jika ingin memasukkan barang di depan): ");

scanf(" %s", masukkanSetelah);

if(strcmp(masukkanSetelah, "depan") == 0){

pNew->next = \*pHead;

\*pHead = pNew;

return;

}

while(pCur != NULL && strcmp(pCur->nama, masukkanSetelah) != 0){

pCur = pCur->next;

}

if(pCur == NULL){

printf("Barang tidak ditemukan\n");

system("pause");

return;

}

if(pCur->next == NULL){

pNew->next = NULL;

} else {

pNew->next = pCur->next;

}

pCur->next = pNew;

}

void hapusBarang(ListNode \*\* pHead){

ListNode \* pPrev, \* pCur;

char namaBarang[30];

pCur = \*pHead;

pPrev = NULL;

if(pCur == NULL){

printf("Daftar produk kosong\n");

system("pause");

return;

}

traverseList(\*pHead);

printf("Masukkan nama barang yang akan dihapus: ");

scanf(" %s", namaBarang);

while(pCur != NULL && strcmp(pCur->nama, namaBarang) != 0){

pPrev = pCur;

pCur = pCur->next;

}

if(pCur == NULL){

printf("Nama barang tidak ditemukan\n");

system("pause");

return;

}

if(pCur == \*pHead){

\*pHead = pCur->next;

free(pCur);

return;

}

if(pCur->next == NULL){

pPrev->next = NULL;

free(pCur);

return;

}

pPrev->next = pCur->next;

free(pCur);

return;

}

void enQueue(Queue \* pQueue, ListNode \* pHead){

int MAX = 5;

QueueNode \* pNew;

ListNode \* pCur;

char namaBarang[30], namaPembeli[30];

if(pQueue->length >= MAX){

printf("Antrian telah melebihi batas\n");

system("pause");

return;

}

printf("Masukkan nama barang: ");

scanf(" %s", namaBarang);

printf("Masukkan nama pembeli: ");

scanf(" %s", namaPembeli);

pCur = pHead;

while(pCur != NULL && strcmp(strlwr(pCur->nama), strlwr(namaBarang)) != 0){

pCur = pCur->next;

}

if(pCur == NULL){

printf("Produk tidak tersedia\n");

system("pause");

return;

}

pNew = (QueueNode \*)malloc(sizeof(QueueNode));

if(pNew == NULL){

printf("Memori tidak cukup\n");

system("pause");

return;

}

strcpy(pNew->namaBarang, namaBarang);

strcpy(pNew->namaPembeli, namaPembeli);

if(pQueue->front == NULL){

pQueue->front = pNew;

pQueue->rear = pNew;

pNew->next = NULL;

pQueue->length++;

system("pause");

return;

}

pQueue->rear->next = pNew;

pQueue->rear = pNew;

pNew->next = NULL;

pQueue->length++;

}

void deQueue(Queue \* pQueue){

QueueNode \* pCur;

pCur = pQueue->front;

if(pCur == NULL){

printf("Antrian kosong\n");

system("pause");

return;

}

if(pCur->next == NULL){

pQueue->front = NULL;

pQueue->rear = NULL;

system("pause");

} else {

pQueue->front = pCur->next;

}

free(pCur);

pQueue->length--;

system("pause");

return;

}

void traverseQueue(Queue pQueue){

QueueNode \* pCur;

pCur = pQueue.front;

if(pCur != NULL){

while(pCur != pQueue.rear){

printf("%s (%s) -> ", pCur->namaPembeli, pCur->namaBarang);

pCur = pCur->next;

}

printf("%s (%s) -> ", pCur->namaPembeli, pCur->namaBarang);

}

printf("NULL\n");

printf("Panjang antrian: %i\n", pQueue.length);

}

int main(){

char pilihan, option;

ListNode \* pHead;

pHead = NULL;

Queue pQueue;

pQueue.length = 0;

pQueue.front = NULL;

pQueue.rear = NULL;

printf("Program Pemesanan dan Penjualan Online Shop\n");

do{

printf("Masuk sebagai: \n");

printf("1. Penjual\n");

printf("2. Pembeli\n");

printf("q. Keluar\n");

printf("Masukkan pilihan: ");

scanf("%c", &pilihan);

system("cls");

if(pilihan == '1'){

do{

printf("Pilihan Menu: \n");

printf("1. Lihat daftar produk\n");

printf("2. Tambah produk baru\n");

printf("3. Hapus produk\n\n");

printf("4. Lihat antrian\n");

printf("5. Proses antrian terdepan\n");

printf("q. Keluar\n");

printf("Masukkan pilihan: ");

scanf(" %c", &option);

if(option == '1'){

traverseList(pHead);

system("pause");

} else if(option == '2'){

addBarang(&pHead);

} else if(option == '3'){

hapusBarang(&pHead);

} else if(option == '4'){

traverseQueue(pQueue);

system("pause");

} else if(option == '5'){

deQueue(&pQueue);

}

system("cls");

} while (option != 'q');

} else if(pilihan == '2'){

do{

printf("Pilihan Menu: \n");

printf("1. Lihat daftar produk\n");

printf("2. Pesan produk\n");

printf("q. Keluar\n");

printf("Masukkan pilihan: ");

scanf(" %c", &option);

if(option == '1'){

traverseList(pHead);

} else if(option == '2'){

enQueue(&pQueue, pHead);

}

system("cls");

} while (option != 'q');

}

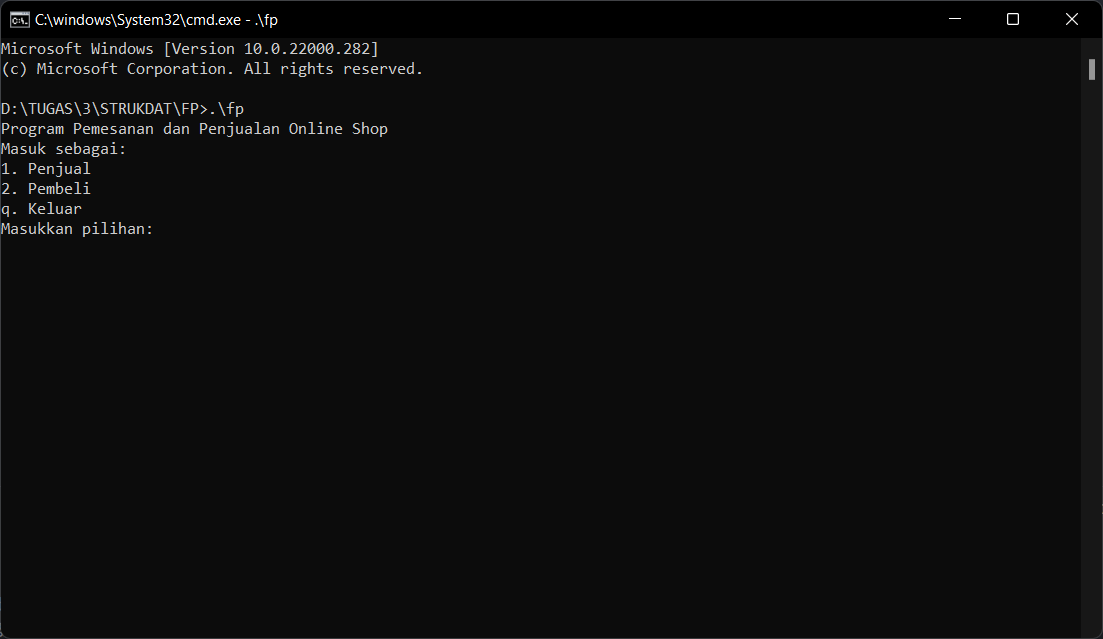
} while (pilihan != 'q');

return 0;

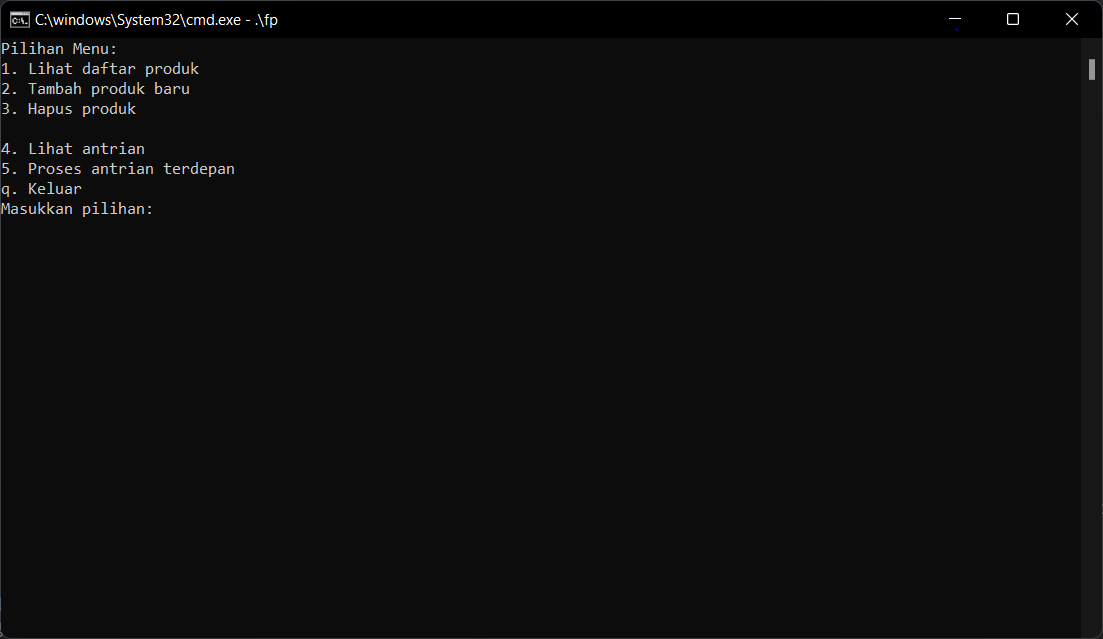
}

## Screenshot Program

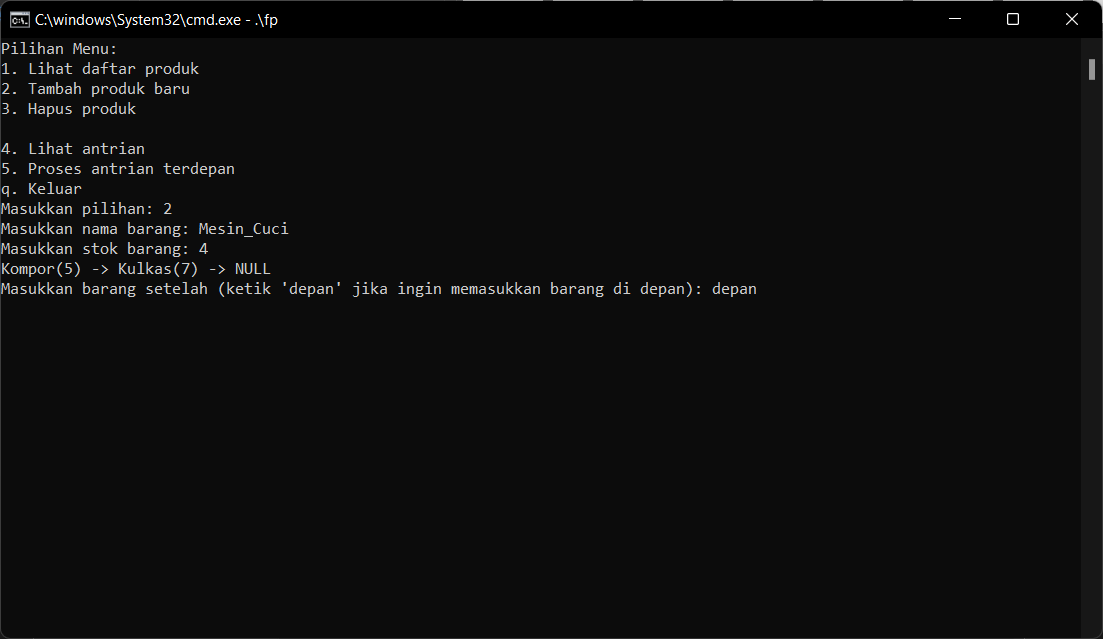
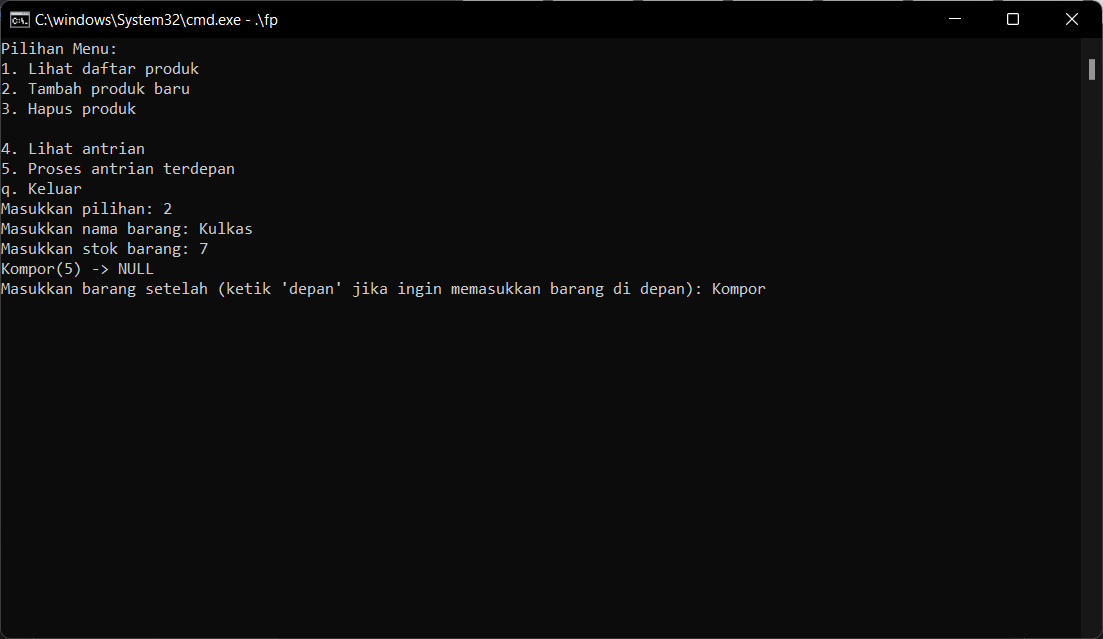
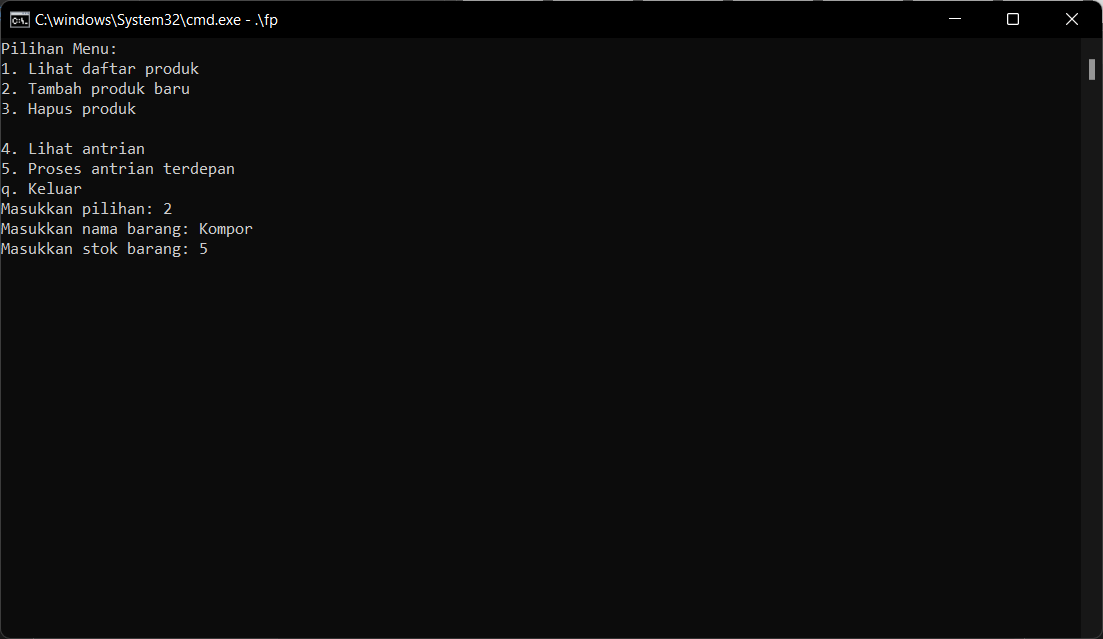
1. Tampilan awal



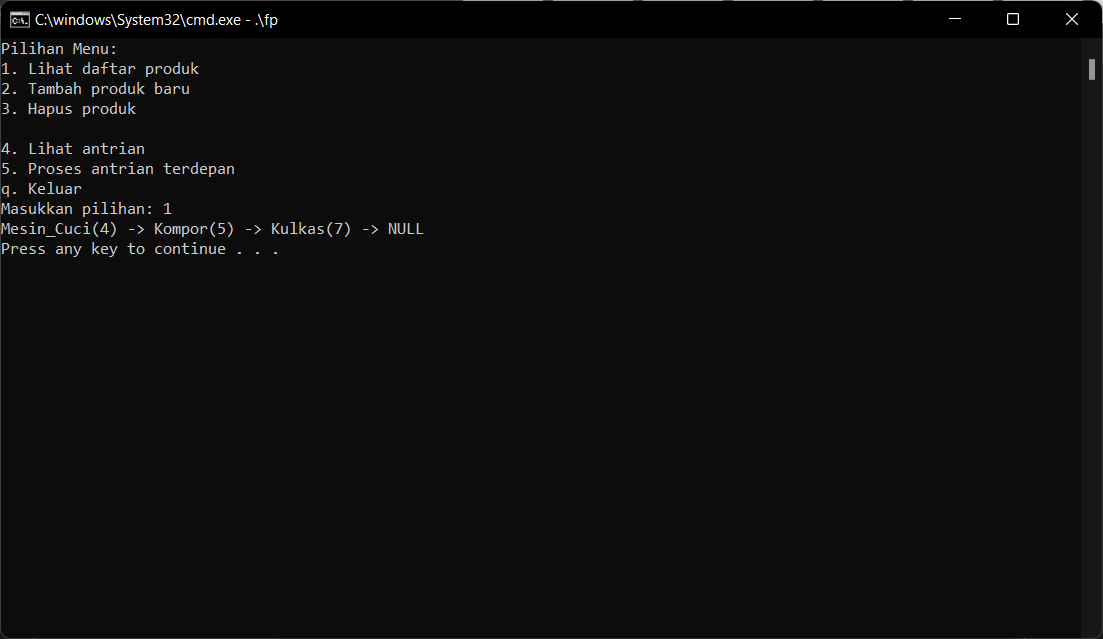
1. Menu penjual



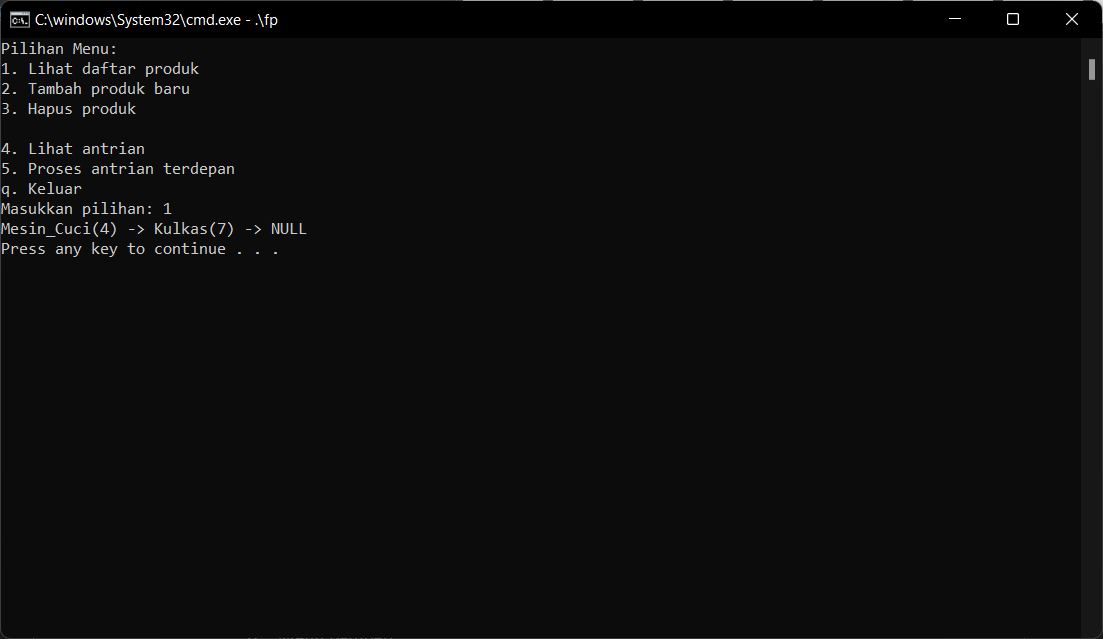
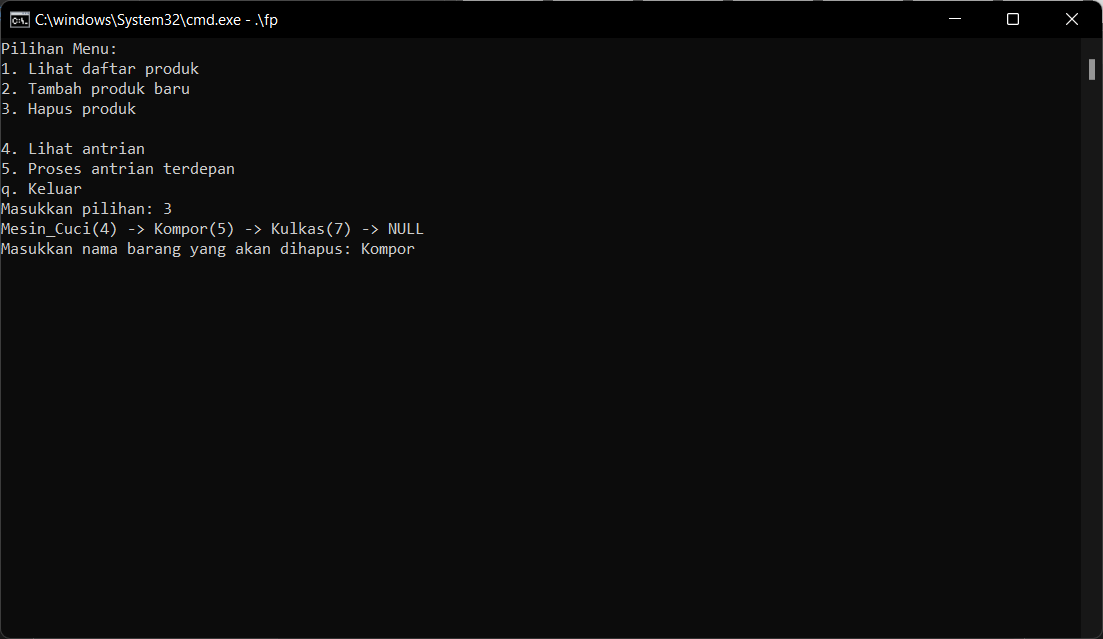
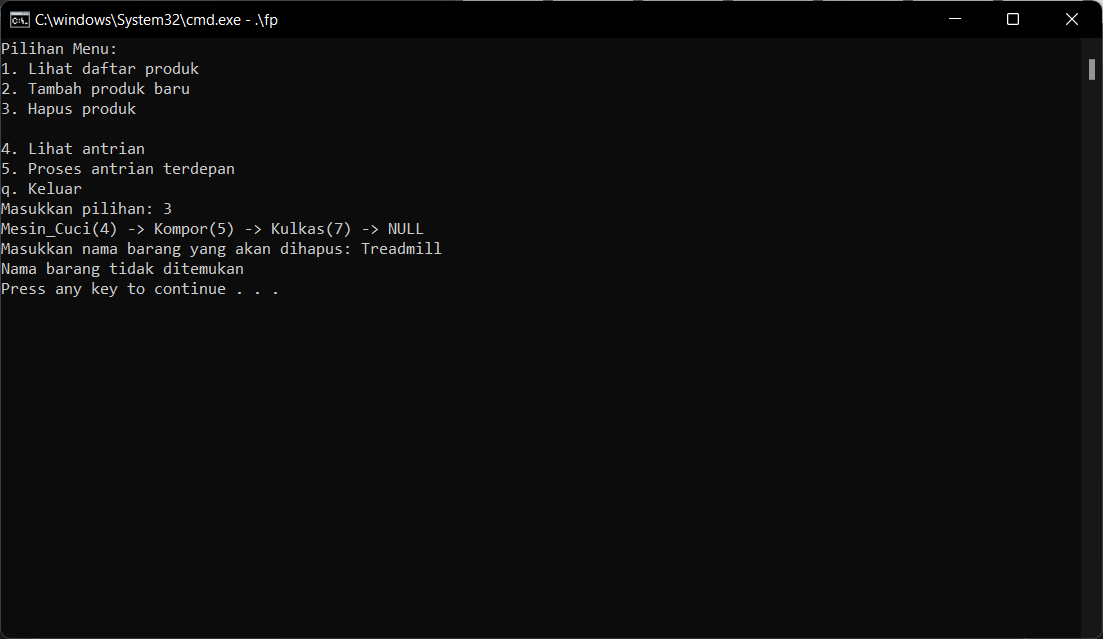
1. Menambahkan produk



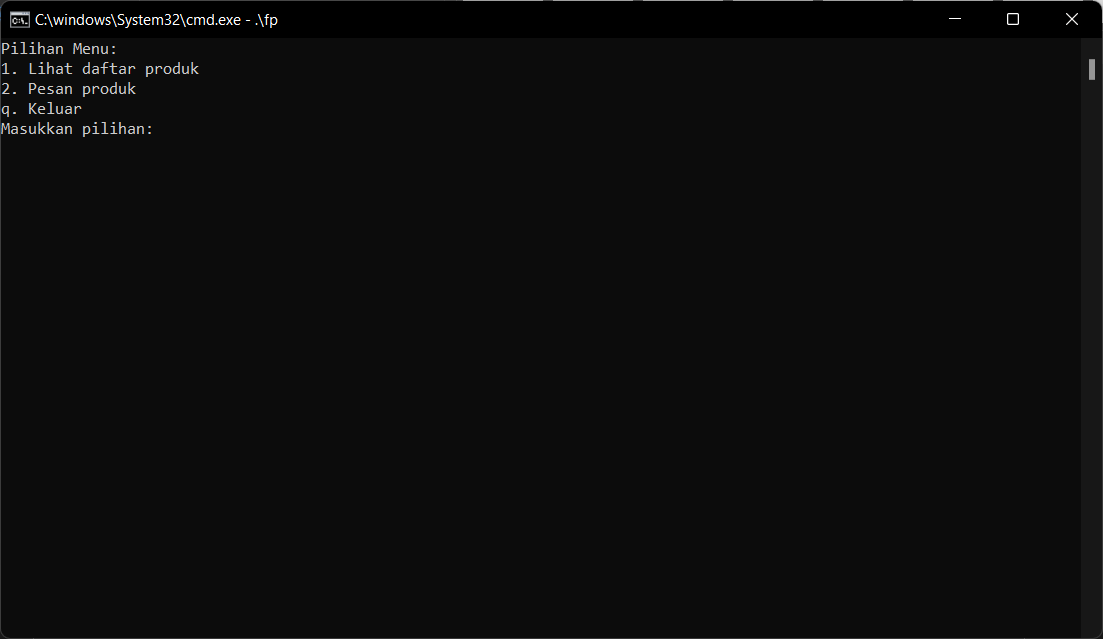
1. Melihat daftar produk



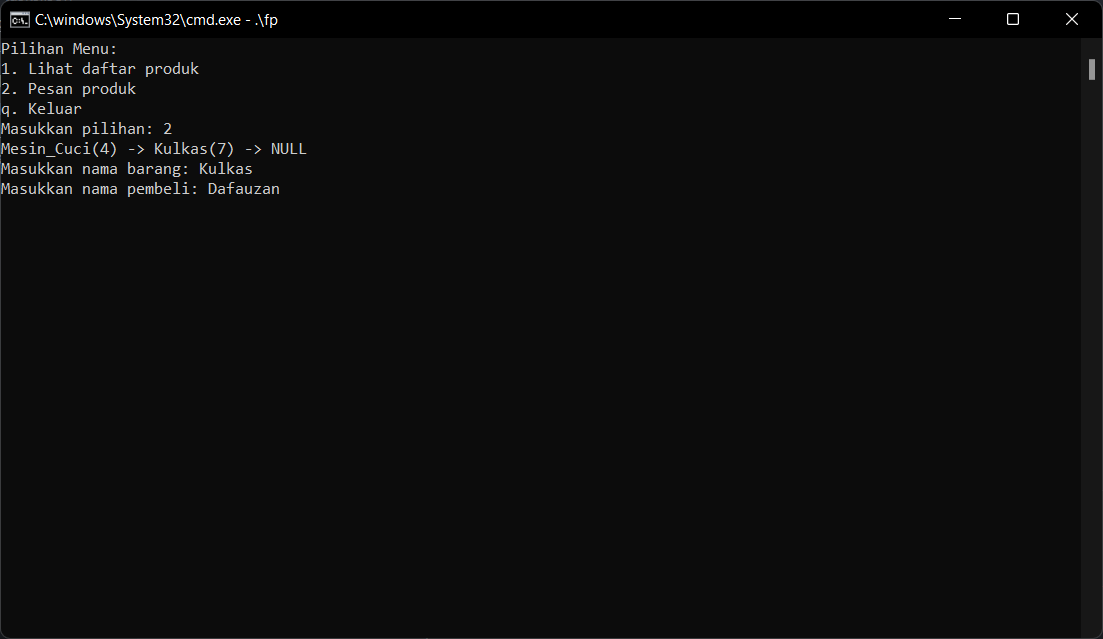
1. Menghapus produk

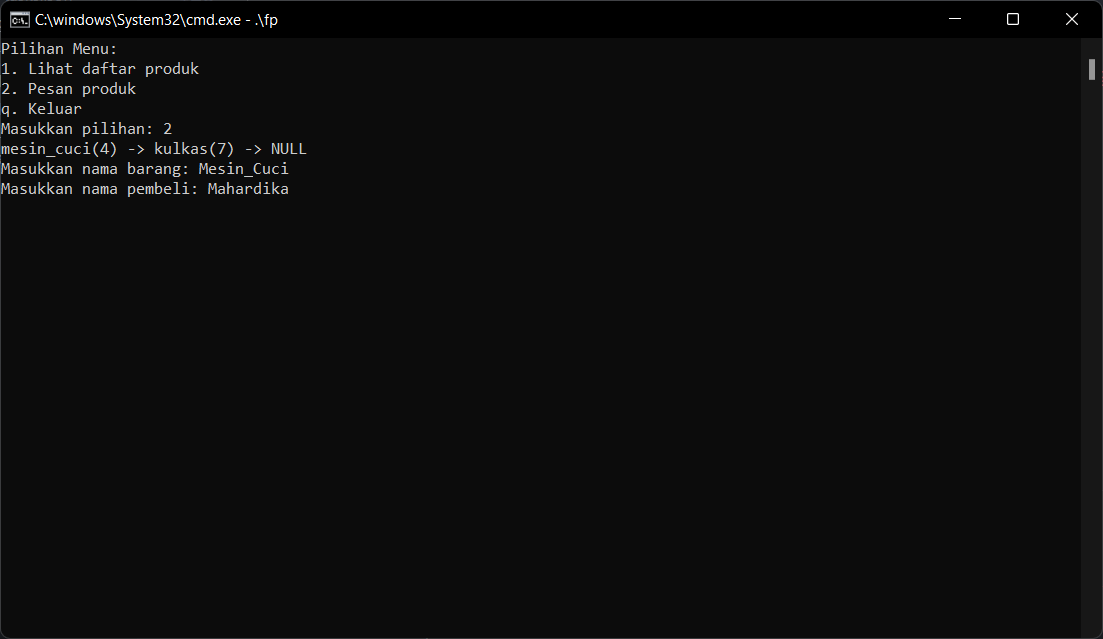


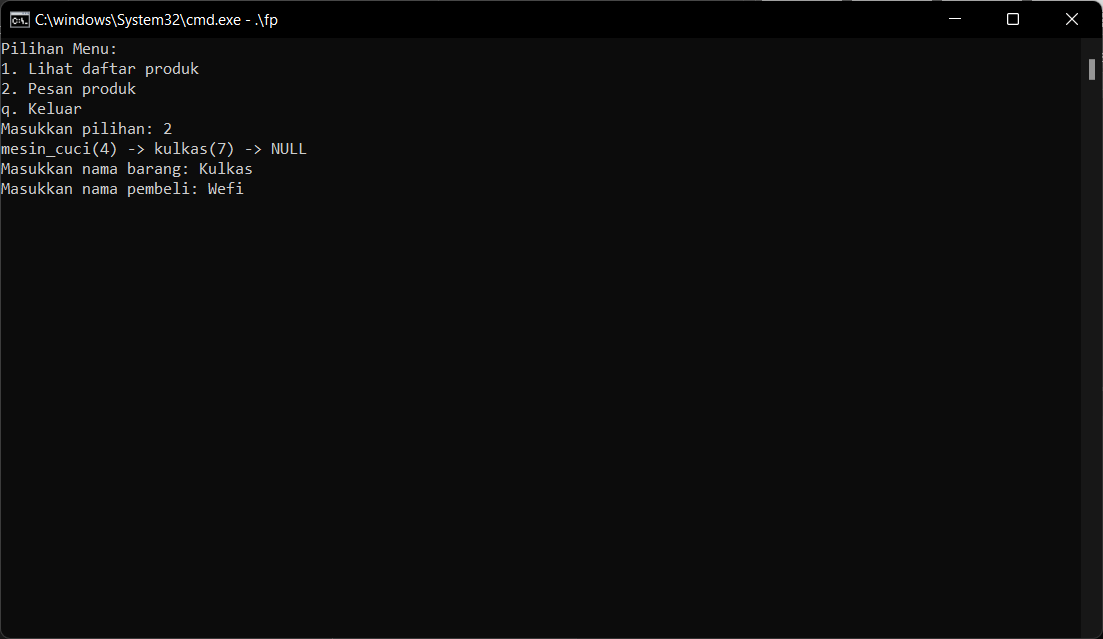
1. Menu pembeli

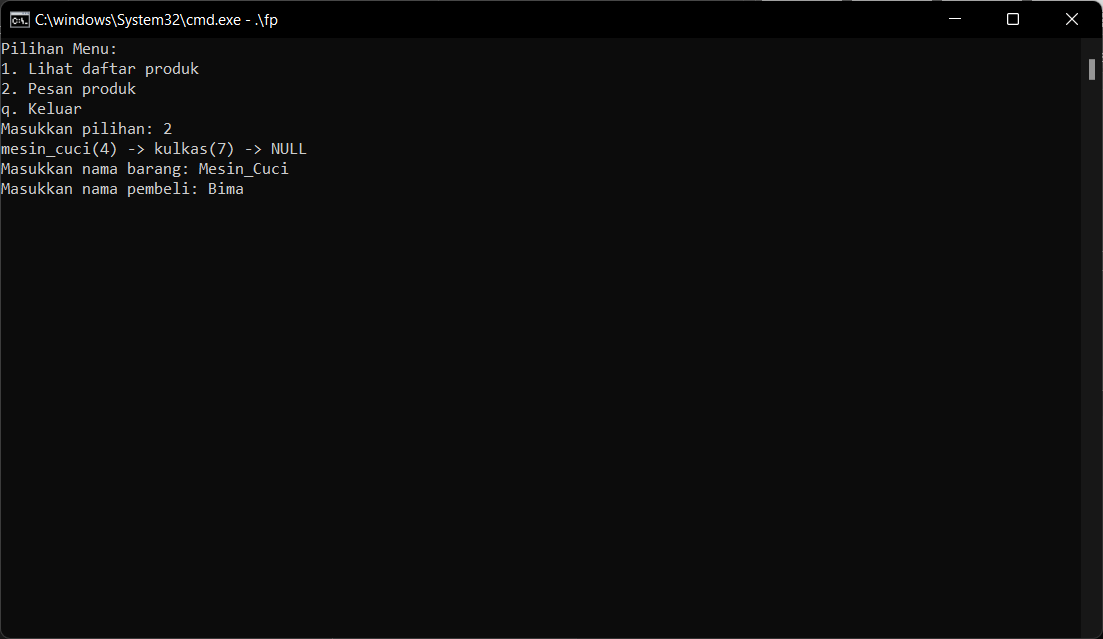


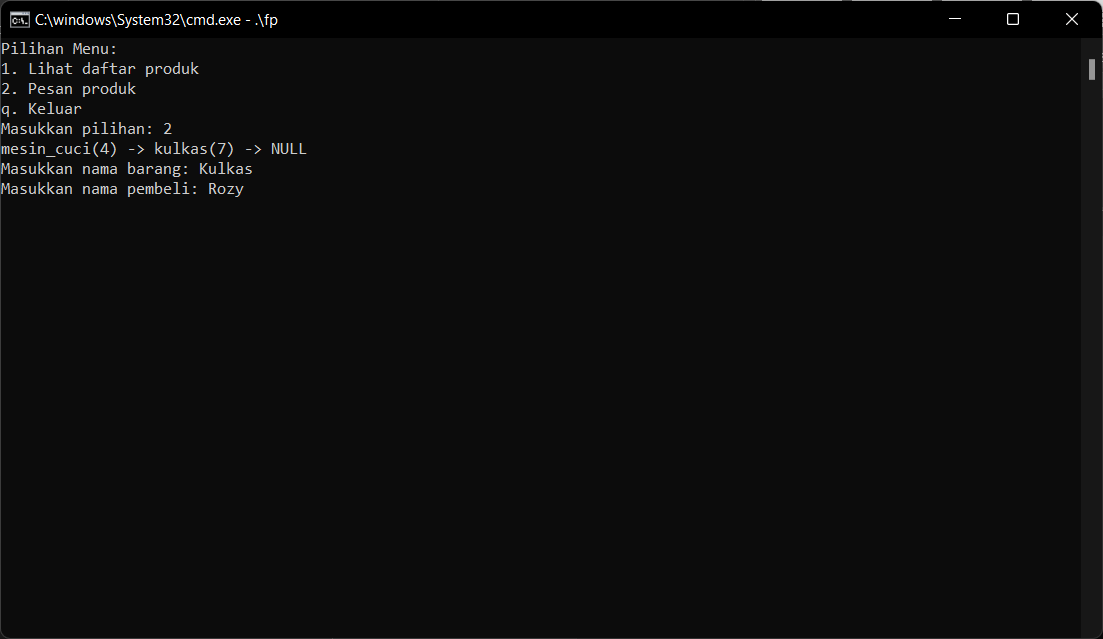
1. Memesan produk (enqueue)

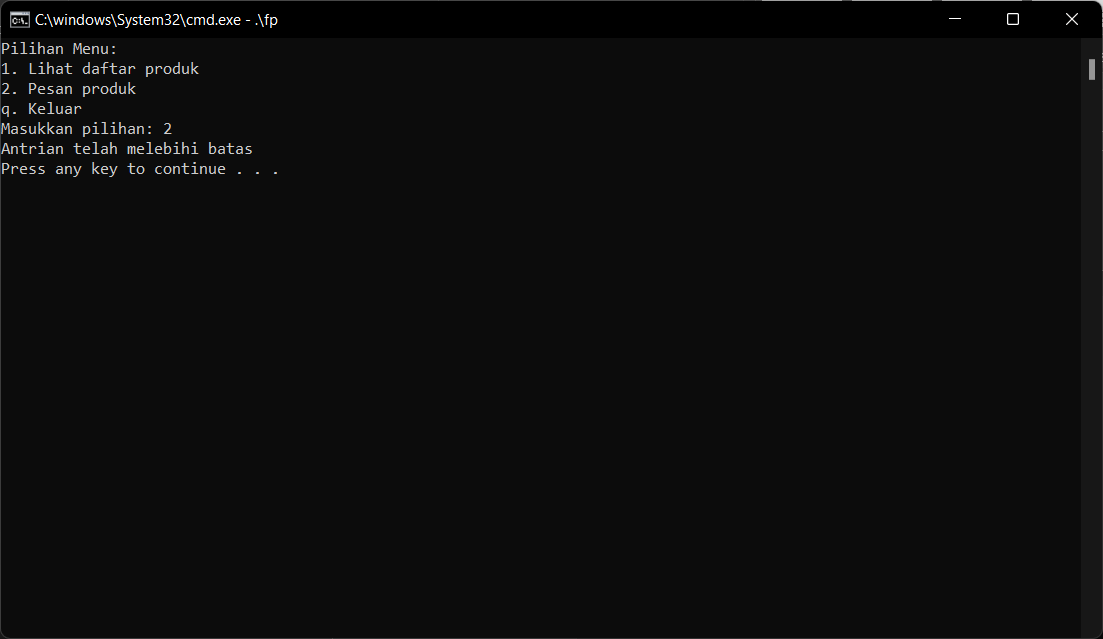












1. Melihat antrian



1. Memroses antrian terdepan (dequeue)

