

LAPORAN UJIAN TENGAH SEMESTER GENAP
ALGORITMA DAN DATA STRUKTUR

PIZZAHUT (FOOD & DRINKS ORDERING)



Kevin Mikael - 00000111440

Bagus Kuncoro A Y - 00000113628

Christian Surya T - 00000116930

Adam Rifqy Hajat - 000001133876

PROGRAM STUDI INFORMATIKA
FAKULTAS TEKNIK DAN INFORMATIKA
UNIVERSITAS MULTIMEDIA NUSANTARA

2025

DAFTAR ISI

DAFTAR ISI.....	I
1.1. Rumusan Masalah.....	1
1.2. Batasan Masalah.....	1
1.3. Tujuan dan Manfaat.....	1
BAB 2 PEMBAHASAN.....	2
2.1. Tampilkan Menu.....	2
2.2 Tambah Pesanan.....	4
2.3. Hapus Pesanan.....	7
2.4. Tampilkan Keranjang.....	8
2.5. Check Out.....	10
2.6. Cek Riwayat Pesanan.....	12
2.7. Hapus Riwayat Pesanan.....	13
2.8. Keluar.....	13
BAB 3 PEMBAGIAN TUGAS.....	14
3.1 Ketua Kelompok.....	14
3.2 Anggota Kelompok.....	14
BAB 4 EVALUASI.....	15
4.1 Ketua Kelompok.....	15
4.2 Anggota Kelompok.....	15
BAB 5 KESIMPULAN.....	16

BAB 1

PENDAHULUAN

1.1. Rumusan Masalah

- ❖ Bagaimana merancang aplikasi pemesanan makanan dan minuman berbasis teks yang dapat menampilkan menu makanan, minuman, dan topping dari file eksternal?
- ❖ Bagaimana menyusun sistem pemesanan yang dapat menambahkan, menghapus, dan menampilkan keranjang pesanan secara dinamis?
- ❖ Bagaimana memproses pembayaran dengan berbagai metode dan mencatat riwayat transaksi ke dalam file?
- ❖ Bagaimana menyimulasikan proses pembuatan pesanan dengan antrian dan status masak?

1.2. Batasan Masalah

- ❖ Aplikasi ditulis menggunakan bahasa C dengan sistem operasi Windows.
- ❖ Menu makanan, minuman, dan topping dibaca dari file teks eksternal (menuMakanan.txt, menuMinuman.txt, dan topping.txt).
- ❖ Tidak ada antarmuka grafis (GUI) maupun koneksi ke sistem online.
- ❖ Riwayat transaksi hanya disimpan dalam file history.txt, tanpa database.
- ❖ Antrian pesanan hanya simulasi menggunakan struktur data queue (tidak real-time atau multi-user).
- ❖ Pembayaran hanya berupa input pilihan tanpa integrasi dengan sistem pembayaran asli.
- ❖ Tidak ada validasi atau otentikasi pengguna.

1.3. Tujuan dan Manfaat

a) Tujuan

- ❖ Membangun aplikasi pemesanan makanan dan minuman berbasis teks yang interaktif dan efisien.
- ❖ Memudahkan pelanggan dalam memilih menu makanan, minuman, dan topping secara fleksibel.
- ❖ Menyediakan sistem keranjang belanja yang dapat diubah sebelum proses checkout.
- ❖ Mengimplementasikan fitur riwayat pemesanan untuk dokumentasi transaksi pengguna.
- ❖ Menyimulasikan proses antrian pesanan dengan pendekatan *queue* untuk mencerminkan status penyajian makanan.

b) Manfaat

- ❖ Pengguna dapat memesan makanan dengan cepat tanpa harus datang langsung ke kasir.
- ❖ Fitur visual (seperti bentuk pizza) dan sistem menu dinamis memberikan pengalaman yang menyenangkan meskipun berbasis teks.
- ❖ Sistem mencatat pesanan dan transaksi ke dalam file, memudahkan pelacakan dan evaluasi.
- ❖ Menggunakan queue untuk memproses pesanan memberi gambaran nyata tentang alur pemesanan di restoran.

BAB 2

PEMBAHASAN

```
===== Welcome To Pizza Hut (Food And Drink Ordering) =====
      /\
     /\
    /\
   /\
  /\
 /\
/\
(o)
(o)
(o)
(o)

      /\
     /\
    /\
   /\
  /\
 /\
/\
(o)
(o)
(o)
(o)

      /\
     /\
    /\
   /\
  /\
 /\
/\
(o)
(o)
(o)
(o)

===== Pizza Hut Main Menu =====
1. Tampilkan Menu
2. Tambah Pesanan
3. Hapus Pesanan
4. Tampilkan Keranjang
5. Check Out
6. Cek Riwayat Pesanan
7. Hapus Riwayat Pesanan
8. Keluar
=====
Masukkan pilihan anda : |
```

Tampilan awal program

2.1. Tampilkan Menu

Kami membuat fungsi menuMakanan untuk untuk menampilkan menu makanan/minuman dengan cara membaca data yang sudah disediakan.

```
30 // Penginisialisasi menu makanan dan minuman dengan file processing
31 void menuMakanan(menuStruct makanan[], menuStruct minuman[], menuStruct topping[])
32 {
33     // Inisialisasi menu makanan
34     int counter = 0;
35     FILE *fp = fopen("menuMakanan.txt", "r");
36     if (!fp)
37     {
38         printf("Error: Tidak bisa membuka file menuMakanan.txt\n");
39         return;
40     }
41     while (counter < 40 && (fscanf(fp, "%[^#]#%d\n", makanan[counter].name, &makanan[counter].harga)) == 2)
42     {
43         counter++;
44     }
45     fclose(fp);
46
47     // Inisialisasi menu minuman
48     counter = 0;
49     fp = fopen("menuMinuman.txt", "r");
50     if (!fp)
51     {
52         printf("Error: Tidak bisa membuka file menuMinuman.txt\n");
53         return;
54     }
55     while (counter < 15 && (fscanf(fp, "%[^#]#%d\n", minuman[counter].name, &minuman[counter].harga)) == 2)
56     {
57         counter++;
58     }
59     fclose(fp);
60
61     //inisialisasi menu untuk topping
62     counter = 0;
63     fp = fopen("topping.txt", "r");
64     if (!fp)
65     {
66         printf("Error: Tidak bisa membuka file topping.txt\n");
67         return;
68     }
69     while (counter < 20 && (fscanf(fp, "%[^#]#%d\n", topping[counter].name, &topping[counter].harga)) == 2)
70     {
71         counter++;
72     }
73     fclose(fp);
74 }
```

Fungsi menuMakan adalah untuk membuka dan membaca file makanan.txt, minuman.txt, dan topping.txt, kemudian di fscanf kedalam struct-struct makanan, minuman, dan topping.

Fungsi tampilkanMenu akan menampilkan setiap elemen item dari struct makanan dan minuman. Setiap item diurutkan berdasarkan ID.

Tampilan setelah memilih menu 1:

```

===== Pizza Hut Main Menu =====
1. Tampilkan Menu
2. Tambah Pesanan
3. Hapus Pesanan
4. Tampilkan Keranjang
5. Check Out
6. Cek Riwayat Pesanan
7. Hapus Riwayat Pesanan
8. Keluar
=====
Masukkan pilihan anda : 1

Menampilkan Menu Makanan
=====
|ID| Nama Makanan | Harga |
=====
1| Splitza Signature | Rp. 90000 |
2| Splitza Classic | Rp. 80000 |
3| Veggie Garden | Rp. 76000 |
4| Meat Lovers | Rp. 86000 |
5| Super Supreme | Rp. 86000 |
6| Tuna Melt | Rp. 76000 |
7| Cheesy Galore | Rp. 76000 |
8| Meat Lovers Cheesy Mayo | Rp. 86000 |
9| American Favourite | Rp. 86000 |
10| Super Supreme Chicken | Rp. 86000 |
11| Pepperoni | Rp. 86000 |
12| Meaty | Rp. 76000 |
13| Chicken Puff Pizza | Rp. 86000 |
14| Meat Lovers Chicken | Rp. 86000 |
15| Hawaiian Chicken | Rp. 76000 |
16| Beef Lasagna | Rp. 65000 |
17| Beef Spaghetti | Rp. 46000 |
18| Chicken Fettuccine alla Italia | Rp. 65000 |
19| Black Pepper Chicken/Beef Fettuccine | Rp. 54000 |
20| Tuna Aglio Olio Spaghetti | Rp. 54000 |
21| Oriental Chicken Spaghetti | Rp. 66000 |
22| Cumi Cabe Ijo | Rp. 73000 |
23| Meatballs Beef Mushroom Rice | Rp. 43000 |
24| Black Pepper Chicken Rice | Rp. 43000 |
25| Oriental Chicken Rice | Rp. 43000 |
26| Thai Chicken Rice | Rp. 43000 |
27| New Orleans Chicken Wings | Rp. 43000 |
28| Baked Chicken Chunks | Rp. 42000 |
29| Sausage Pastry Roll | Rp. 34000 |
30| Garlic Bread | Rp. 24000 |
31| Chicken Royale | Rp. 41000 |
32| Nacho Cheese | Rp. 34000 |
33| Deluxe Beef Bruschetta | Rp. 28000 |
34| Deluxe Chicken Bruschetta | Rp. 28000 |
35| Garlic Cheese Bread | Rp. 28000 |
36| Choco Puff | Rp. 34000 |
37| Cheese Rolls | Rp. 34000 |
38| Potato Wedges | Rp. 26000 |
39| Cheese Martabak Pizza | Rp. 37000 |
40| Fresh Salad | Rp. 42000 |

Menampilkan Menu Minuman
=====
|ID| Nama Minuman | Harga |
=====
1| Tropical Punch | Rp. 33000 |
2| Lychee Breeze | Rp. 33000 |
3| Orange Lychee Sparkle | Rp. 31000 |
4| Blue Ocean | Rp. 31000 |
5| Mixberry | Rp. 31000 |
6| Winter Punch | Rp. 33000 |
7| Green Tea Yakult | Rp. 29000 |
8| Honey Lime Tea | Rp. 25000 |
9| Lychee Spring | Rp. 29000 |
10| Watermelon Juice | Rp. 36000 |
11| Melon Juice | Rp. 36000 |
12| Strawberry Watermelon Juice | Rp. 36000 |
13| Strawberry Orange Juice | Rp. 36000 |
14| Melon Peach Juice | Rp. 36000 |
15| Avocado Juice | Rp. 36000 |
=====

```

2.2 Tambah Pesanan

Kami membuat fungsi menambah pesanan bertujuan agar pengguna dapat memilih makanan/minuman yang sebelumnya sudah ditampilkan untuk dimasukkan kedalam keranjang, dan khusus untuk makanan pizza, pengguna dapat menambahkan topping sesuai selera.

```
137 // Menambahkan pesanan ke keranjang
138 void tambahPesanan(menuStruct **head, menuStruct **tail, menuStruct makanan[], menuStruct minuman[], menuStruct topping[])
139 {
140     int choice, pilih;
141     menuStruct *node;
142     node = (menuStruct *)malloc(sizeof(menuStruct));
143     if (!node)
144     {
145         printf("Gagal mengalokasikan memori!\n");
146         return;
147     }
148     printf("\n===== Jenis Menu =====\n");
149     printf("[1] Makanan\n");
150     printf("[2] Minuman\n");
151     printf("===== \n");
152     printf("Masukkan pilihan: ");
153     scanf("%d", &choice);
154     getchar();
155     if (choice == 1)
156     {
157         printf("Masukkan ID makanan pilihan anda : ");
158         scanf("%d", &pilih);
159         getchar();
160         if (pilih > 40 || pilih < 1)
161         {
162             printf("Masukkan pilihan yang benar!");
163             return;
164         }
165         strcpy(node->name, makanan[pilih - 1].name);
166         node->harga = makanan[pilih - 1].harga;
167         if (pilih <= 15)
168             tambhahtopping(topping, node);
169     }
170 }
171 else if (choice == 2)
172 {
173     printf("Masukkan ID minuman pilihan anda : ");
174     scanf("%d", &pilih);
175     getchar();
176     if (pilih > 15 || pilih < 1)
177     {
178         printf("Masukkan pilihan yang benar!");
179         return;
180     }
181     strcpy(node->name, minuman[pilih - 1].name);
182     node->harga = minuman[pilih - 1].harga;
183 }
184 else
185 {
186     printf("Masukkan pilihan yang benar!");
187     return;
188 }
189 node->next = NULL;
190 if ((*head) == NULL)
191 {
192     (*head) = (*tail) = node;
193 }
194 else
195 {
196     (*tail)->next = node;
197     (*tail) = node;
198 }
199 }
```

```

101 void tambahtopping(menuStruct topping[], menuStruct *node){
102     int temp;
103     printf("Apakah anda ingin menambahkan topping?\n");
104     printf("[0] Tidak\n");
105     printf("[1] Ya\n");
106     printf("Masukkan pilihan: ");
107     scanf("%d",&temp);
108     if (temp == 0){
109         return;
110     }
111     else if (temp > 1 || temp < 0){
112         return;
113     }
114
115     printf("=====\n");
116     printf("|ID| %27s%12s | %7s | \n", "Nama Topping", "", "Harga");
117     printf("=====\n");
118     for (int i = 0; i < 20; i++) {
119         printf("|%-2d| %-40s | Rp. %-6d | \n", i + 1, topping[i].name, topping[i].harga);
120     }
121     printf("=====\n");
122     hm:
123     printf("Pilih topping no : ");
124     scanf("%d",&temp);
125     getchar();
126     if(temp < 1 || temp > 20){
127         printf("Input invalid please try again\n");
128         goto hm;
129     }
130     char top [40] = " + ";
131     strcat(top, topping[temp - 1].name);
132     strcat(node->name, top);
133     node->harga = node->harga + topping[temp-1].harga;
134     printf("Topping ditambahkan!\n");
135 }

```

Fungsi ini bertugas menambahkan pesanan baru ke dalam keranjang belanja. Saat fungsi ini dipanggil, pertama-tama dibuat sebuah node baru yang akan menyimpan data pesanan.

Pengguna diminta untuk memilih jenis menu, apakah ingin memesan makanan atau minuman. Jika memilih makanan, maka pengguna harus memasukkan ID makanan yang tersedia. Jika ID yang dimasukkan valid, maka nama dan harga makanan tersebut akan disalin ke dalam node pesanan.

Jika makanan yang dipilih termasuk dalam daftar makanan yang bisa ditambahkan topping (misalnya hanya makanan dengan ID 1 sampai 15), maka fungsi akan memanggil tambahTopping untuk menambahkan topping sesuai keinginan pengguna. Jika pengguna memilih minuman, prosesnya hampir sama. Sistem akan meminta ID minuman, lalu menyimpan nama dan harga minuman ke dalam node jika ID valid.

Setelah semua informasi dimasukkan, node baru ini akan ditambahkan ke akhir dari daftar keranjang belanja. Jika keranjang masih kosong, maka node ini akan menjadi pesanan pertama.

Fungsi ini digunakan untuk menambahkan topping ke makanan yang dipesan. Pertama-tama, pengguna ditanya apakah ingin menambahkan topping. Jika jawabannya tidak, maka fungsi akan langsung keluar.

Jika pengguna memilih untuk menambahkan topping, maka akan ditampilkan daftar topping yang tersedia lengkap dengan harga dan nomor ID-nya. Pengguna kemudian diminta untuk memilih topping berdasarkan nomor. Jika pilihan topping tidak valid, pengguna akan diminta untuk mengulang sampai memilih topping yang benar.

Setelah topping dipilih, nama topping akan ditambahkan ke nama makanan (misalnya jadi "Meat Lovers + Keju"), dan harga topping akan ditambahkan ke total harga makanan tersebut. Setelah itu, sistem memberitahu bahwa topping berhasil ditambahkan.

```
=====
Masukkan pilihan anda : 2
===== Jenis Menu =====
[1] Makanan
[2] Minuman
=====
Masukkan pilihan: 1
Masukkan ID makanan pilihan anda : 1
Apakah anda ingin menambahkan topping?
[0] Tidak
[1] Ya
Masukkan pilihan: 1
=====
|ID|                Nama Topping                |  Harga  |
=====
[1] | Pepperoni                | Rp. 15000 |
[2] | Italian Sausage          | Rp. 17000 |
[3] | Ham                      | Rp. 14000 |
[4] | Chicken                  | Rp. 16000 |
[5] | Meatballs                | Rp. 15000 |
[6] | Beef                    | Rp. 16000 |
[7] | Mozzarella               | Rp. 13000 |
[8] | Parmesan                 | Rp. 14000 |
[9] | Provolone                | Rp. 13000 |
[10] | Mushrooms                | Rp. 10000 |
[11] | Onions                   | Rp. 8000  |
[12] | Tomatoes                 | Rp. 9000  |
[13] | Jalapenos                | Rp. 10000 |
[14] | Banana Peppers           | Rp. 10000 |
[15] | Roasted Red Peppers      | Rp. 12000 |
[16] | Pineapple                | Rp. 11000 |
[17] | Green Peppers            | Rp. 9000  |
[18] | Anchovies                | Rp. 20000 |
[19] | Garlic                   | Rp. 7000  |
[20] | BBQ Sauce Drizzle        | Rp. 8000  |
=====
Pilih topping no : 2
Topping ditambahkan!
```


2.3. Hapus Pesanan

Pengguna dapat menghapus/membatalkan pesanan di keranjang yang telah ditambahkan sebelumnya.

```
// Menghapus pesanan berdasarkan list keranjang
void hapusPesanan(menuStruct **head, menuStruct **tail)
{
    if (*head == NULL)
    {
        printf("\nBelum ada pesanan yang terisi\n");
        return;
    }
    menuStruct *temp = *head;
    int i = 1, del;
    printf("\nMenampilkan keranjang\n");
    printf("=====\n");
    printf("|No| %27s%13s | %7s%4s|\n", "Nama", "", "Harga ", "");
    printf("=====\n");
    while (temp != NULL)
    {
        printf("|%-2d| %-40s | Rp. %-5d |\n", i, temp->name, temp->harga);
        temp = temp->next;
        i++;
    }
    printf("=====\n");
    printf("Hapus pesanan No: ");
    scanf("%d", &del);
    if (del < 1 || del >= i)
    {
        printf("Nomor pesanan tidak valid!\n");
        return;
    }
    temp = *head;
    if (del == 1)
    {
        *head = (*head)->next;
        free(temp);
        if (*head == NULL)
            *tail = NULL;
        return;
    }
    for (int j = 1; j < del - 1; j++)
    {
        temp = temp->next;
    }
    menuStruct *deleteNode = temp->next;
    temp->next = deleteNode->next;
    if (deleteNode == *tail)
    {
        *tail = temp;
    }
    free(deleteNode);
    printf("Pesanan nomor %d berhasil dihapus.\n", del);
}
```

Fungsi `hapusPesanan` bertujuan untuk menghapus salah satu pesanan dari daftar yang tersimpan dalam bentuk linked list. Pertama, fungsi ini memeriksa apakah daftar pesanan kosong. Jika kosong, maka akan ditampilkan pesan bahwa belum ada pesanan, dan fungsi langsung keluar. Jika tidak kosong, fungsi akan menampilkan seluruh isi keranjang beserta nomor urut, nama pesanan, dan harganya.

Setelah itu, pengguna diminta untuk memasukkan nomor pesanan yang ingin dihapus. Jika nomor yang dimasukkan tidak valid (kurang dari 1 atau melebihi jumlah pesanan), maka akan muncul pesan kesalahan dan proses dibatalkan.

Jika pesanan yang dihapus adalah yang pertama, maka pointer head akan dipindahkan ke node berikutnya dan node awal akan dihapus. Jika setelah penghapusan daftar menjadi kosong, maka tail juga akan diatur menjadi NULL. Jika pesanan yang ingin dihapus berada di tengah atau akhir daftar, maka fungsi akan menelusuri node hingga ke posisi sebelum node yang ingin dihapus, lalu mengatur pointer agar melewati node tersebut. Jika node yang dihapus adalah node terakhir, maka tail akan diperbarui. Setelah node berhasil dihapus, akan ditampilkan pesan bahwa penghapusan berhasil.

```

===== Pizza Hut Main Menu =====
1. Tampilkan Menu
2. Tambah Pesanan
3. Hapus Pesanan
4. Tampilkan Keranjang
5. Check Out
6. Cek Riwayat Pesanan
7. Hapus Riwayat Pesanan
8. Keluar
=====
Masukkan pilihan anda : 3

Menampilkan keranjang
=====
|No|                Nama                | Harga |
=====
|1 | Splitza Signature + Italian Sausage | Rp. 107000 |
|2 | Beef Spaghetti                     | Rp. 46000 |
=====
Hapus pesanan No: 1

```

2.4. Tampilkan Keranjang

Pengguna dapat menampilkan pesanan yang ada di dalam keranjang

```

void tampilkanKeranjang(menuStruct **head){
    if(*head==NULL){...
    }
    menuStruct *temp = *head;
    int i = 1, totalHarga = 0;

    printf("\nMenampilkan keranjang\n");
    printf("=====\n");
    printf("|No| %19s%19s | %2s%2s |\n", "", "Nama", "", "Harga", "");
    printf("=====\n");
    while (temp != NULL)
    {
        printf("|%-2d| %-42s | Rp.%-6d |\n", i, temp->name, temp->harga);
        totalHarga += temp->harga;
        temp = temp->next;
        i++;
    }
    printf("=====\n");
    printf("|%16s%19s | Rp.%-6d |\n", "", "Total Harga", "", totalHarga);
    printf("=====\n");
    printf("\n");
}

```

```

===== Pizza Hut Main Menu =====
1. Tampilkan Menu
2. Tambah Pesanan
3. Hapus Pesanan
4. Tampilkan Keranjang
5. Check Out
6. Cek Riwayat Pesanan
7. Hapus Riwayat Pesanan
8. Keluar
=====
Masukkan pilihan anda : 4

Menampilkan keranjang
=====
|No|                Nama                | Harga |
=====
|1 | Lychee Breeze                     | Rp.33000 |
|2 | Lychee Breeze                     | Rp.33000 |
|3 | Lychee Breeze                     | Rp.33000 |
|4 | Lychee Breeze                     | Rp.33000 |
|5 | Lychee Breeze                     | Rp.33000 |
=====
|                Total Harga                | Rp.165000 |
=====

Total harga : Rp. 165000

```

```

===== Pizza Hut Main Menu =====
1. Tampilkan Menu
2. Tambah Pesanan
3. Hapus Pesanan
4. Tampilkan Keranjang
5. Check Out
6. Cek Riwayat Pesanan
7. Hapus Riwayat Pesanan
8. Keluar
=====
Masukkan pilihan anda : 4

Menampilkan keranjang
=====
|No|                Nama                |  Harga  |
=====
|1 | Meatballs Beef Mushroom Rice    | Rp.43000 |
|2 | Strawberry Orange Juice           | Rp.36000 |
=====
|                Total Harga          | Rp.79000 |
=====

Total harga : Rp. 79000

```

Fungsi ini digunakan untuk menampilkan isi keranjang belanja ke layar. Pertama-tama, program memeriksa apakah keranjang kosong dengan melihat apakah `*head == NULL`. Jika iya (tidak ada pesanan), maka isi keranjang tidak ditampilkan dan fungsi langsung berhenti.

Jika ada isinya, fungsi mulai dengan menginisialisasi variabel `temp` yang menunjuk ke awal keranjang, `i` sebagai penomoran pesanan, dan `totalHarga` untuk menyimpan total biaya semua pesanan. Kemudian ditampilkan header tabel seperti "No", "Nama", dan "Harga", lengkap dengan garis pemisah agar tampilannya rapi. Setelah itu, program masuk ke perulangan `while (temp != NULL)`, yang akan berjalan selama masih ada node dalam linked list (keranjang). Di dalam perulangan:

- Program menampilkan nomor pesanan (`i`), nama menu, dan harga menu dari node saat ini.
- Harga dari menu saat ini ditambahkan ke `totalHarga`.
- Pointer `temp` digeser ke node berikutnya.
- Nomor pesanan (`i`) juga ditambah.

Setelah semua isi keranjang ditampilkan, program mencetak garis pemisah dan kemudian mencetak total harga semua pesanan. Akhirnya, fungsi selesai dan kembali ke program utama.

2.5. Check Out

Menyelesaikan pesanan sekaligus dengan memilih opsi pembayaran yang ingin dilakukan.

```
void checkout(menuStruct **head, menuStruct **tail)
{
    FILE *fp = fopen("history.txt", "a");

    if (!fp)
    {
        printf("\nGagal membuka history.txt untuk checkout!\n");
        return;
    }
    if (*head == NULL)
    {
        printf("\nKeranjang kosong! Silahkan pesan makanan atau minuman terlebih dahulu.\n");
        fclose(fp);
        return;
    }

    int metode, totalHarga = 0;
    char metodePembayaran[50];

    printf("\n=== Pilih Metode Pembayaran ===\n");
    printf("1. Tunai\n");
    printf("2. Kartu Kredit/Debit\n");
    printf("3. E-Wallet\n");
    printf("Pilihan: ");
    scanf("%d", &metode);
    getchar();

    switch (metode)
    {
        case 1:
            strcpy(metodePembayaran, "Tunai");
            break;
        case 2:
            strcpy(metodePembayaran, "Kartu Kredit/Debit");
            break;
        case 3:
            strcpy(metodePembayaran, "E-Wallet");
            break;
        default:
            printf("Metode tidak valid!\n");
            fclose(fp);
            return;
    }

    fprintf(fp, "\n==== History ==== \n");
    fprintf(fp, "Metode Pembayaran: %s\n", metodePembayaran);
    fprintf(fp, "Daftar Pesanan:\n");

    menuStruct *current = *head;
    while (current)
    {
        fprintf(fp, "- %s \t(Rp. %d)\n", current->name, current->harga);
        totalHarga += current->harga;
        current = current->next;
    }

    fprintf(fp, "\nTotal Harga: Rp. %d\n", totalHarga);
    fclose(fp);

    printf("\nCheckout berhasil!\n");
    printf("Total: Rp. %d\n", totalHarga);
    printf("Metode: %s\n", metodePembayaran);
    printf("apakah ingin melihat status pesanan anda?\n");
    int choice;
    printf("Yes[1]\n");
    printf("No [0]\n");
    scanf("%d", &choice);
    if (choice == 1)
    {
        cekStatusPesanan(head, tail);
    }

    while (*head)
    {
        menuStruct *temp = *head;
        *head = (*head)->next;
        free(temp);
    }
    *tail = NULL;
}

void dequeueAll(Node **front, Node **rear) {
    printf("\n=== Status Pesanan Sedang Diproses ===\n");
    while (*front != NULL) {
        Node* temp = *front;
        printf("sedang proses pesanan: %s\n", temp->namaPesanan);
        *front = (*front)->next;
        free(temp);
        Sleep(5000);
    }
    *rear = NULL;
    printf("Semua pesanan telah selesai dimasak.\n");
}

void enqueue(Node **front, Node **rear, char *namaPesanan) {
    Node* baru = (Node*)malloc(sizeof(Node));
    strcpy(baru->namaPesanan, namaPesanan);
    baru->next = NULL;

    if (*rear == NULL) {
        *front = *rear = baru;
    } else {
        (*rear)->next = baru;
        *rear = baru;
    }
}

void cekStatusPesanan(menuStruct **head, menuStruct **tail) {
    Node* front = NULL;
    Node* rear = NULL;
    menuStruct *temp = *head;
    while (temp != NULL) {
        enqueue(&front, &rear, temp->name);
        temp = temp->next;
    }
    dequeueAll(&front, &rear);
}

===== Pizza Hut Main Menu =====
1. Tampilkan Menu
2. Tambah Pesanan
3. Hapus Pesanan
4. Tampilkan Keranjang
5. Check Out
6. Cek Riwayat Pesanan
7. Hapus Riwayat Pesanan
8. Keluar
=====
Masukkan pilihan anda : 5

=== Pilih Metode Pembayaran ===
1. Tunai
2. Kartu Kredit/Debit
3. E-Wallet
Pilihan: 2

Checkout berhasil!
Total: Rp. 79000
Metode: Kartu Kredit/Debit
apakah ingin melihat status pesanan anda?
Yes[1]
No [0]
1

=== Status Pesanan Sedang Diproses ===
sedang proses pesanan: Meatballs Beef Mushroom Rice
sedang proses pesanan: Strawberry Orange Juice
Semua pesanan telah selesai dimasak.
```

Fungsi checkout digunakan untuk menyimpan daftar pesanan yang telah dipesan oleh pengguna ke dalam file bernama history.txt sekaligus menampilkan informasi total harga serta metode pembayaran yang dipilih. Fungsi ini pertama-tama memeriksa apakah file

history.txt bisa dibuka, lalu memeriksa apakah ada isi keranjang. Jika keranjang kosong, maka proses checkout dibatalkan.

Setelah itu, pengguna diminta untuk memilih metode pembayaran (tunai, kartu, atau e-wallet), yang kemudian disimpan dalam variabel metodePembayaran. Semua pesanan yang ada di keranjang dituliskan ke file history.txt satu per satu, bersamaan dengan nama menu dan harga masing-masing, lalu dihitung total harganya.

Setelah proses penulisan ke file selesai, pengguna diberi pilihan untuk melihat status pesanan mereka. Jika pengguna memilih “Ya”, maka fungsi cekStatusPesanan akan dijalankan, yang akan menampilkan daftar proses pesanan satu per satu dengan simulasi waktu.

Fungsi cekStatusPesanan digunakan untuk mensimulasikan proses pemrosesan pesanan setelah pengguna melakukan checkout. Proses ini menggunakan struktur data queue berbasis linked list, yang bekerja dengan prinsip First In, First Out (FIFO), artinya pesanan yang pertama kali masuk akan menjadi pesanan yang pertama kali diproses.

Pada awalnya, dua buah pointer yaitu front dan rear diinisialisasi dengan nilai NULL. Pointer front berfungsi untuk menunjukkan elemen pertama dalam queue (pesanan yang sedang atau akan diproses), sedangkan pointer rear menunjukkan elemen terakhir dalam queue (pesanan yang baru masuk).

Kemudian, seluruh isi keranjang pesanan yang disimpan dalam linked list *head akan dibaca satu per satu menggunakan pointer sementara temp. Nama dari setiap item pesanan akan dimasukkan ke dalam queue menggunakan fungsi enqueue.

Fungsi enqueue akan membuat node baru untuk setiap pesanan menggunakan malloc, kemudian menyalin nama pesanan ke dalam node tersebut. Jika queue masih kosong (pointer rear bernilai NULL), maka node baru tersebut menjadi elemen pertama dan terakhir dalam queue (baik front maupun rear menunjuk ke node tersebut). Jika queue sudah berisi elemen, maka node baru akan ditambahkan di belakang queue dan pointer rear diperbarui agar menunjuk ke node baru tersebut.

Setelah seluruh pesanan masuk ke dalam queue, fungsi dequeueAll akan dipanggil untuk memproses semua pesanan tersebut secara berurutan. Dalam fungsi ini, selama queue masih memiliki elemen, pesanan akan ditampilkan satu per satu dengan menampilkan pesan seperti "Sedang proses pesanan: [nama_pesanan]". Setiap pesanan akan diproses

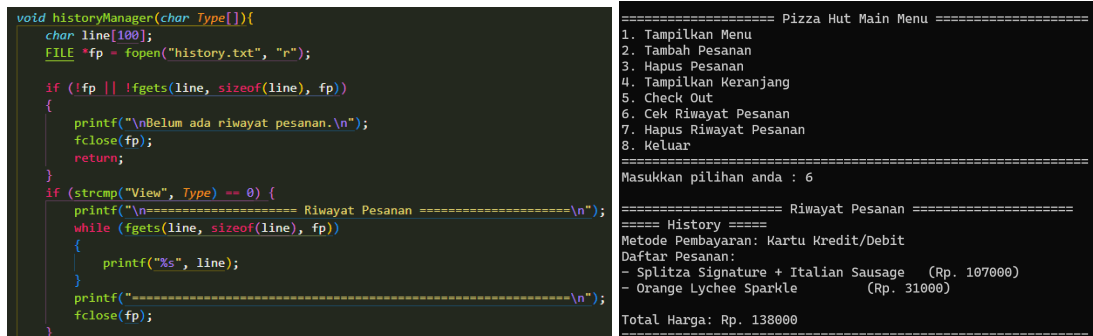
selama 5 detik menggunakan `Sleep(5000)` untuk mensimulasikan waktu memasak. Setelah itu, node tersebut dihapus dari memori menggunakan `free`, dan pointer `front` digeser ke node berikutnya. Proses ini diulangi hingga seluruh elemen dalam queue habis.

Jika pengguna memesan, misalnya, Meat Lovers dan lychee breeze, maka antrian queue akan berisi urutan pesanan tersebut. Fungsi `dequeueAll` kemudian akan memprosesnya satu per satu sesuai urutan, yaitu Meat Lovers terlebih dahulu, dilanjutkan dengan Lychee Breeze. Setelah seluruh pesanan selesai diproses, sistem akan menampilkan pesan bahwa semua pesanan telah selesai dimasak.

Dengan demikian, fungsi ini memberikan pengalaman realistis bagi pengguna mengenai bagaimana pesanan mereka diproses secara berurutan seperti di dapur restoran yang sebenarnya.

2.6. Cek Riwayat Pesanan

Dengan menu ini, program akan menampilkan riwayat pesanan yang telah dilakukan, yaitu setelah sudah melakukan checkout.



```
void historyManager(char Type){
    char line[100];
    FILE *fp = fopen("history.txt", "r");

    if (!fp || !fgets(line, sizeof(line), fp))
    {
        printf("\nBelum ada riwayat pesanan.\n");
        fclose(fp);
        return;
    }

    if (strcmp("View", Type) == 0) {
        printf("\n===== Riwayat Pesanan =====\n");
        while (fgets(line, sizeof(line), fp))
        {
            printf("%s", line);
        }
        printf("===== \n");
        fclose(fp);
    }
}
```

```
===== Pizza Hut Main Menu =====
1. Tampilkan Menu
2. Tambah Pesanan
3. Hapus Pesanan
4. Tampilkan Keranjang
5. Check Out
6. Cek Riwayat Pesanan
7. Hapus Riwayat Pesanan
8. Keluar
=====
Masukkan pilihan anda : 6
===== Riwayat Pesanan =====
===== History =====
Metode Pembayaran: Kartu Kredit/Debit
Daftar Pesanan:
- Splitza Signature + Italian Sausage (Rp. 107000)
- Orange Lychee Sparkle (Rp. 31000)
Total Harga: Rp. 138000
=====
```

Jika parameter `Type` bernilai "View", maka program akan menampilkan riwayat pesanan yang tersimpan dalam file `history.txt`. File akan dibuka dalam mode baca ("`r`") dan setiap baris dari file tersebut akan dibaca menggunakan fungsi `fgets`. Baris-baris tersebut kemudian akan ditampilkan ke layar menggunakan `printf`. Jika file tidak dapat dibuka atau tidak memiliki isi, maka akan ditampilkan pesan bahwa belum ada riwayat pesanan. Setelah proses pembacaan selesai, file akan ditutup menggunakan `fclose`.

2.7. Hapus Riwayat Pesanan

Program ini dapat membuat pengguna untuk menghapus Riwayat Pesanan sebelumnya, merupakan gabungan dari fungsi historyManager sebagai kondisi lainnya.

```
else if (strcmp("Delete", Type) == 0) {  
    fclose(fp);  
    FILE *fp = fopen("history.txt", "w");  
    printf("\nHistory telah dihapus\n");  
    fclose(fp);  
}
```

```
===== Pizza Hut Main Menu =====  
1. Tampilkan Menu  
2. Tambah Pesanan  
3. Hapus Pesanan  
4. Tampilkan Keranjang  
5. Check Out  
6. Cek Riwayat Pesanan  
7. Hapus Riwayat Pesanan  
8. Keluar  
=====  
Masukkan pilihan anda : 7  
History telah dihapus
```

Jika parameter Type bernilai "Delete", maka program akan menghapus seluruh isi file history.txt. Hal ini dilakukan dengan membuka file dalam mode tulis ("w"), yang secara otomatis akan mengosongkan file tersebut. Setelah file dibuka dan dikosongkan, program akan menampilkan pesan bahwa riwayat telah dihapus. File kemudian ditutup kembali menggunakan fclose.

2.8. Keluar

Fungsi ini akan menyelesaikan program.

```
case 8:  
{  
    printf("Terima kasih telah menggunakan aplikasi kami!!!");  
    return 0;  
}
```

```
===== Pizza Hut Main Menu =====  
1. Tampilkan Menu  
2. Tambah Pesanan  
3. Hapus Pesanan  
4. Tampilkan Keranjang  
5. Check Out  
6. Cek Riwayat Pesanan  
7. Hapus Riwayat Pesanan  
8. Keluar  
=====  
Masukkan pilihan anda : 8  
Terima kasih telah menggunakan aplikasi kami!!!  
Process returned 0 (0x0)   execution time : 2.961 s  
Press any key to continue.
```

BAB 3

PEMBAGIAN

TUGAS

3.1 Ketua Kelompok

Kevin Mikael

- Fungsi menuMakan
- Fungsi Tampilkan Menu
- Fungsi Tambah Pesanan

3.2 Anggota Kelompok

Adam Rifqy Hajat

- Fungsi Hapus Riwayat Pesanan
- Fungsi Cek Status Pesanan
- Fungsi Keluar

Bagus Kuncoro

- Fungsi CheckOut
- Fungsi Riwayat Pesanan

Christian Surya T

- Fungsi Hapus Pesanan
- Fungsi Tampilkan Keranjang
- Penampilan hasil program

BAB 4

EVALUASI

4.1 Ketua Kelompok

Kevin Mikael

- Dari pengalaman saya sendiri selama mengerjakan proyek besar pada semester ini, saya mengalami banyak hal positif dan menurut saya ini adalah kelompok terbaik yang pernah saya miliki.

4.2 Anggota Kelompok

Adam Rifqy Hajat

- Tugas ini telah memberikan saya perspektif lebih terhadap pengerjaan kode berkelompok. Walaupun di semester sebelumnya sudah pernah, skala kode di sini lebih signifikan daripada yang lalu. Saya belajar tentang aplikasi-aplikasi yang membantu kolaborasi dan komunikasi dalam

Bagus Kuncoro

- Setelah kurang lebih 2 minggu mengerjakan tugas ini, saya mempelajari cukup banyak hal, salah satunya adalah cara memakai Github, banyak hal yang sudah terjadi, salah satunya adalah missscom, namun saya rasa kelompok ini tetap dapat bekerjasama dengan baik.

Christian Surya T

- Berdasarkan pengalaman saya selama mengerjakan proyek ini, semuanya berjalan dengan lancar dengan kekurangan yang tidak banyak. Untuk pembagian tugas, menurut saya sudah aman atau adil karena kami disini saling melengkapi jadi jika ada yang di luar tugas menurut kami kurang tepat atau sesuai, maka kami akan langsung diskusikan bersama.

BAB 5

KESIMPULAN

Aplikasi *Pizza Hut Food and Drink Ordering* yang kami kembangkan menggunakan bahasa pemrograman C merupakan implementasi sistem pemesanan makanan dan minuman yang sederhana namun efektif. Aplikasi ini dirancang untuk memberikan pengalaman pemesanan yang menyerupai proses di restoran cepat saji, dengan mengedepankan kemudahan penggunaan, kecepatan interaksi, serta pengelolaan data yang efisien.

Melalui pemanfaatan struktur data linked list untuk manajemen keranjang pesanan dan queue untuk simulasi proses pemasakan, aplikasi ini mampu menangani berbagai kebutuhan fungsional seperti menampilkan menu dari file eksternal, menambah dan menghapus pesanan, menambahkan topping secara fleksibel, menghitung total harga, memilih metode pembayaran, serta menyimpan dan menampilkan riwayat pesanan ke dalam file *history.txt*.

Dari sisi teknis, aplikasi ini berhasil memadukan berbagai konsep penting dalam pemrograman, seperti file handling, dynamic memory allocation, modular programming, dan user input validation. Hal ini menjadikan aplikasi ini tidak hanya bermanfaat sebagai simulasi sistem pemesanan restoran, tetapi juga sebagai sarana edukatif yang efektif dalam memahami konsep-konsep dasar dan menengah pemrograman berorientasi prosedural.

Dengan demikian, aplikasi ini telah memenuhi tujuan utama pengembangan, yaitu menciptakan sistem pemesanan yang sederhana, interaktif, dan edukatif, yang dapat digunakan sebagai referensi dalam pengembangan aplikasi pemesanan makanan digital berbasis teks.