Program Pembayaran Parkir

Oleh:
Aditya Fadhil Herindro - 51679
Steven Vincent Hendrawan - 43664
Farren Yazid Pasha - 44665
Irwin Riyadi - 53023

Tentang Aplikasi

Aplikasi Program Pembayaran Parkir ini merupakan aplikasi sederhana yang bisa membantu operator melayani pengguna lapangan parkir dengan beberapa fitur.

- Parkir Harian
- Membership System (Pendaftaran dan Penggunaan)
- History saving serta melihat kembali data tersimpan dalam dua opsi

Program ini dilengkapi dengan dua database yang dimana satu untuk menyimpan data member dan yang lain untuk data parkir. Penggunaan data member juga akan tercatat pada data parkir dengan perbedaan pada harga.

Data Structures

```c
// Stack
typedef struct Node{
    char jenis[30];
    char noKendaraan[10];
    int tahun, bulan, tanggal;
    float jamMasuk, jamKeluar;
    int harga;

    struct Node *next;
}Node;

// Linked List untuk hashTable daftar membership
typedef struct LL{
    char nama[50];
    int tahun, bulan, tanggal;

    struct LL *next;
}LL;

//Tree Tahun
typedef struct tree
{
    int bulan;
    int tanggal;
    int tahun;
    char jenis[30];
    char noKendaraan[10];
    float jamMasuk, jamKeluar;
    int harga;
    struct tree *next, *left, *right;

}Tree;
```

Functions

```
//Salin Data ke Node pada Tree
Tree *new(char type[], char number[], int date, int month, int year, float hourIn, float hourOut, int price)
{...

//Mencari Node pada Tree untuk diinput dari File atau data Baru pada Tree
Tree *pushToTree(Tree **node, char type[], char number[], int date, int month, int year, float hourIn, float hourOut, int price)

// Insert LL untuk Hash Table
void insert(LL **HEAD, char nama[], int tanggal, int bulan, int tahun){...

//Print Data from HashTable
LL *memberData(LL *HEAD){...

// HashTable Print
void printHash(LL *HEAD[26][26]){...

//Cek apakah Stack kosong
int isEmpty(Node *head){...

//Proses Opsi 1
//Masukkan Input ke Stack
void pushInput(Node **head, char type[], char number[], int year, int month, int date, float hourIn){...

//Memasukkan input data Baru
void scanInput(Node **head, char type[], char number[], int *year, int *month, int *date, float *hourIn, float *hourOut){...

//Proses Opsi 3
//print Daftar Parkir (keseluruhan) pada Stack
void printHistory(Node *head)...

//Print Tree
void printTree(Tree *cTree, int li)...
```

```
//Memilih Tree
void chooseTree(Tree *yTree[100])...

//Proses No. 2
//Mencari Node pada Tree untuk melengkapi data pada new Input
Tree *searchTree(Tree **node, char type[], char number[], int date, int month, int year, float hourIn, float hourOut, int price)...

//Masukkan pembayaran ke Stack
void pushOutput(Node **pay, float hourOut, int price)...

//Menentukan Harga
void checkPrice(Node **pay, float hourOut, int *price){...

//Mencari Data pada Stack
Node *findData(Node *head, char number[], int *year, int *month, int *date, float *hourIn)...

//Opsi No. 2
//Pengecekan membership
bool membershipPay(LL *hash[26][26], int *date, int *month, int *year) {...

//Masukkan data member ke database
void saveMemberData(char nama[], int tanggal, int bulan, int tahun)...

//Proses Pembayaran
void payment(LL *hash[26][26], Node **pay, Node *head, char number[], int year, int month, int date, float *hourIn, float *hourOut, int *price)

//Proses Penukaran Node
void swap(Node *a, Node *b)...

//Proses Bubble Sort
void bubbleSort(Node *head)...

//Proses Opsi 4
//Daftar sebagai member
void regMember(LL *HashT[26][26]){...
```

File Processing

```c
//Masukkan data member ke database
void saveMemberData(char nama[], int tanggal, int bulan, int tahun)
{
    FILE *fim = fopen("dataMember.txt", "a+");
    fprintf(fim, "%s#%d/%d/%d\n", nama, tanggal, bulan, tahun);
    fclose(fim);
}
```

```c
FILE *fin = fopen("dataParkir.txt","r");
while(!feof(fin)){
    fscanf(fin,"%[^#]#%[^#]#%d/%d/%d#%f#%f#%d\n", type, number, &date, &month, &year, &hourIn, &hourOut, &price);
    pushInput(&head, type, number, year, month, date, hourIn);
    pushOutput(&head, hourOut, price);
    mod = year % 100;
    yTree[mod] = pushToTree(&yTree[mod], type, number, date, month, year, hourIn, hourOut, price);
}
fclose(fin);

//Proses Input dari File ke Hash Table
FILE *fim = fopen("dataMember.txt", "a+");
while(!feof(fim))
{
    fscanf(fim, "%[^#]#%d/%d/%d\n", namaMember, &tanggalMember, &bulanMember, &tahunMember);
    key1 = namaMember[0] % 65;
    key2 = namaMember[1] % 97;
    insert(&HashT[key1][key2], namaMember, tanggalMember, bulanMember, tahunMember);
}
fclose(fim);
```

```c
//Push ke Stack dan File
pushInput(head, type, number, *year, *month, *date, *hourIn);
FILE *fin = fopen("dataParkir.txt","a+");
fprintf(fin, "%s#%s#%d/%d/%d#%.2f#0.00#0\n", type, number, *date, *month, *year, *hourIn);
fclose(fin);
```

```c
FILE *fout = fopen("dataParkir.txt","w");
while(temp != NULL){
    fprintf(fout, "%s#%s#%d/%d/%d#%.2f#%.2f#%d\n", temp->jenis, temp->noKendaraan, temp->tanggal, temp->bulan, temp->tahun, temp-
    temp = temp->next;
}
fclose(fout);
```

Pointer Implementation

main.c

D: > ofukuroo > materi kuliah > AlgoStrukdat LEC > Tugas UAS > main.c > ...

```
328 Node *findData(Node *head, char number[], int *year, int *month, int *date, float *hourIn)···

357

358  //Pengecekan membership
359 bool membershipPay(LL *hash[26][26], int *date, int *month, int *year) {···

434

435  //Masukkan data member ke database
436 void saveMemberData(char nama[], int tanggal, int bulan, int tahun)···

442

443  //Daftar sebagai member
444 void regMember(LL *HashT[26][26]){···

503

504  //Proses Pembayaran
505 void payment(LL *hash[26][26], Node **pay, Node *head, char number[], int year, int month, int date, float *hourIn, f

555

556  //Proses Penukaran Node
557 void swap(Node *a, Node *b)···

587

588  //Proses Bubble Sort
589 void bubbleSort(Node *head)···

615

616  //Print Tree
617 void printTree(Tree *cTree, int li)···

632

633  //Memilih Tree
634 void chooseTree(Tree *yTree[100])···

656
```

```c
//Proses Input dari File ke Stack dan Tree
FILE *fin = fopen("dataParkir.txt","r");
while(!feof(fin)){
    fscanf(fin,"%[^#]#%[^#]#%d/%d/%d#%f#%f#%d\n", type, number, &date, &month, &year, &hourIn, &hourOut, &price);
    pushInput(&head, type, number, year, month, date, hourIn);
    pushOutput(&head, hourOut, price);
    mod = year % 100;
    yTree[mod] = pushToTree(&yTree[mod], type, number, date, month, year, hourIn, hourOut, price);
}
fclose(fin);

//Proses Input dari File ke Hash Table
FILE *fim = fopen("dataMember.txt", "a+");
while(!feof(fim))
{
    fscanf(fim, "%[^#]#%d/%d/%d\n", namaMember, &tanggalMember, &bulanMember, &tahunMember);
    key1 = namaMember[0] % 65;
    key2 = namaMember[1] % 97;
    insert(&HashT[key1][key2], namaMember, tanggalMember, bulanMember, tahunMember);
}
fclose(fim);
case 1 :
    //Input data baru
    scanInput(&head, type, number, &year, &month, &date, &hourIn, &hourOut);
    //Masukkan data baru ke Tree
    mod = year % 100;
    hourOut = 0;
    price = 0;
    yTree[mod] = pushToTree(&yTree[mod], type, number, date, month, year, hourIn, hourOut, price);
    printf("Press any key to continue\n");
    getch();
    break;

case 2 :
    //Melengkapi info pada Inout saat membayar
    payment(HashT, &pay, head, number, year, month, date, &hourIn, &hourOut, &price);
    //Input to Tree
    mod = year % 100;
    yTree[mod] = searchTree(&yTree[mod], type, number, date, month, year, hourIn, hourOut, price);
    //Rewrite ke File
    temp = head;
    FILE *fout = fopen("dataParkir.txt","w");
    while(temp != NULL){
        fprintf(fout, "%s#%s#%d/%d/%d#%.2f#%.2f#%d\n", temp->jenis, temp->noKendaraan, temp->tanggal, temp
        temp = temp->next;
    }
```

Array Implementation

```
682    //Variabel untuk daftar member
683    char namaMember[50], huruf[2];
684    int tanggalMember, bulanMember, tahunMember;
```

```
692    //Variable untuk data input
693    char type[30];
694    int year;
695    int month;
696    int date;
697    char number[10];
```

```
672    //Inisialisasi Tree
673    Tree *yTree[100];
```

```
663    //Inisialisasi Hash Table
664    int i, j, k, key1, key2;
665    LL *HashT[26][26];
666    for(i = 0; i < 26; i++){
667        for(j = 0; j < 26; j++){
668            HashT[i][j] = NULL;
669        }
670    }
```

Linked List Implementation

```c
// Stack
typedef struct Node{
    char jenis[30];
    char noKendaraan[10];
    int tahun, bulan, tanggal;
    float jamMasuk, jamKeluar;
    int harga;

    struct Node *next;
}Node;
```

```c
//Tree Tahun
typedef struct tree
{
    int bulan;
    int tanggal;
    int tahun;
    char jenis[30];
    char noKendaraan[10];
    float jamMasuk, jamKeluar;
    int harga;
    struct tree *next, *left, *right;

}Tree;
```

```c
// Linked List untuk daftar membership
typedef struct LL{
    char nama[50];
    int tahun, bulan, tanggal;

    struct LL *next;
}LL;
```

Stack Implementation

```
9   // Stack
10  typedef struct Node{
11      char jenis[30];
12      char noKendaraan[10];
13      int tahun, bulan, tanggal;
14      float jamMasuk, jamKeluar;
15      int harga;
16
17      struct Node *next;
18  }Node;
```

```
228         if(!isEmpty(*head))
229             node->next = *head;
230         *head = node;
```

```
346     //search menggunakan linear
347     for(curr = head; curr != NULL; curr = curr->next)
348     {
349         if(strcmp(curr->noKendaraan, number) == 0 &&
350             curr->tanggal == *date && curr->bulan == *month
351             && curr->tahun == *year && curr->jamMasuk == *hourIn){
352             return curr;
353         }
354     }
355     return NULL;
356 }
```

Tree Implementation

```c
//Tree Tahun
typedef struct tree
{
    int bulan;
    int tanggal;
    int tahun;
    char jenis[30];
    char noKendaraan[10];
    float jamMasuk, jamKeluar;
    int harga;
    struct tree *next, *left, *right;

}Tree;
```

```c
//Mencari Node pada Tree untuk diinput dari File atau data Baru pada Tree
Tree *pushToTree(Tree **node, char type[], char number[], int date, int month, int year, float hourIn, float hourOut, int price)
{
    if(*node == NULL){
        //Masukkan data ke Node Kosong
        return new(type, number, date, month, year, hourIn, hourOut, price);
    }
    else
    {
        if((*node)->bulan == month && (*node)->tanggal == date)
        {
            //Geser ke Node bawah
            (*node)->next = pushToTree(&(*node)->next, type, number, date, month, year, hourIn, hourOut, price);
        }
        else if((*node)->bulan > month && (*node)->tanggal > date)
        {
            //Geser ke Node Kiri
            (*node)->left = pushToTree(&(*node)->left, type, number, date, month, year, hourIn, hourOut, price);
        }
        else if((*node)->bulan < month && (*node)->tanggal < date)
        {
            //Geser ke Node Kanan
            (*node)->right = pushToTree(&(*node)->right, type, number, date, month, year, hourIn, hourOut, price);
        }
    }
    return *node;
}
```

```c
//Proses Input dari File ke Stack dan Tree
FILE *fin = fopen("dataParkir.txt","r");
while(!feof(fin)){
    fscanf(fin,"%[^#]#%[^#]#%d/%d/%d#%f#%f#%d\n", type, number, &date, &month, &year, &hourIn, &hourOut, &price);
    pushInput(&head, type, number, year, month, date, hourIn);
    pushOutput(&head, hourOut, price);
    mod = year % 100;
    yTree[mod] = pushToTree(&yTree[mod], type, number, date, month, year, hourIn, hourOut, price);
}
fclose(fin);
```

Searching Feature

```
for(curr = head; curr != NULL; curr = curr->next)
{
    if(strcmp(curr->noKendaraan, number) == 0 &&
       curr->tanggal == *date && curr->bulan == *month
       && curr->tahun == *year && curr->jamMasuk == *hourIn){
        return curr;
    }
}
```

```
while(ptr != NULL){
    system("cls");
    printf("-------------------------------------\n"
           "Member Details\n");
    printf("Nama : %s\n",ptr->nama);
    printf("Batas aktif membership : %02d/%02d/%04d\n",ptr->tanggal, ptr->bulan, ptr->tahun);
    printf(".....\n\n");
    printf("Is this correct? (y/n): "); scanf(" %c", &valid);
    fflush(stdin);
    if(valid == 'y')
    {
        return ptr;
    }
    else
    {
        ptr = ptr->next;
    }
}
```

Sorting Feature

```c
//Proses Penukaran Node
void swap(Node *a, Node *b)
{
    Node temp[1];
    strcpy(temp->jenis,a->jenis);
    strcpy(temp->noKendaraan,a->noKendaraan);
    temp->tanggal = a->tanggal;
    temp->bulan = a->bulan;
    temp->tahun = a->tahun;
    temp->jamMasuk = a->jamMasuk;
    temp->jamKeluar = a->jamKeluar;
    temp->harga = a->harga;

    strcpy(a->jenis,b->jenis);
    strcpy(a->noKendaraan,b->noKendaraan);
    a->tanggal = b->tanggal;
    a->bulan = b->bulan;
    a->tahun = b->tahun;
    a->jamMasuk = b->jamMasuk;
    a->jamKeluar = b->jamKeluar;
    a->harga = b->harga;

    strcpy(b->jenis,temp->jenis);
    strcpy(b->noKendaraan,temp->noKendaraan);
    b->tanggal= temp->tanggal;
    b->bulan = temp->bulan;
    b->tahun = temp->tahun;
    b->jamMasuk= temp->jamMasuk;
    b->jamKeluar = temp->jamKeluar;
    b->harga = temp->harga;
}
```

```c
void bubbleSort(Node *head)
{
    int swapped;
    Node *ptr;
    Node *ptrEnd = NULL;

    if (head == NULL)
        return;

    do
    {
        swapped = 0;
        ptr = head;

        while (ptr->next != ptrEnd)
        {
            if (ptr->harga > ptr->next->harga)
            {
                swap(ptr, ptr->next);
                swapped = 1;
            }
            ptr = ptr->next;
        }
        ptrEnd = ptr;
    }while (swapped);
}
```

Application Demo