MINGGU 4
Linked List

DESKRIPSI TEMA

Pada pertemuan minggu ini, mahasiswa akan belajar mengenai Dynamic Memory Allocation dan macam-macam Linked List. Tipe Linked List yang akan dipelajari adalah Single Linked List, Double Linked List, dan Circular Linked List.

CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

1. Mahasiswa mampu menerapkan Dynamic Memory Allocation menggunakan Bahasa C.
2. Mahasiswa mampu menerapkan Single Linked List menggunakan Bahasa C.
3. Mahasiswa mampu menerapkan Double Linked List menggunakan Bahasa C.
4. Mahasiswa mampu menerapkan Circular Linked List menggunakan Bahasa C.

PENUNJANG PRAKTIKUM

1. Aplikasi CodeBlock
2. Aplikasi Dev-C++ (alternatif)

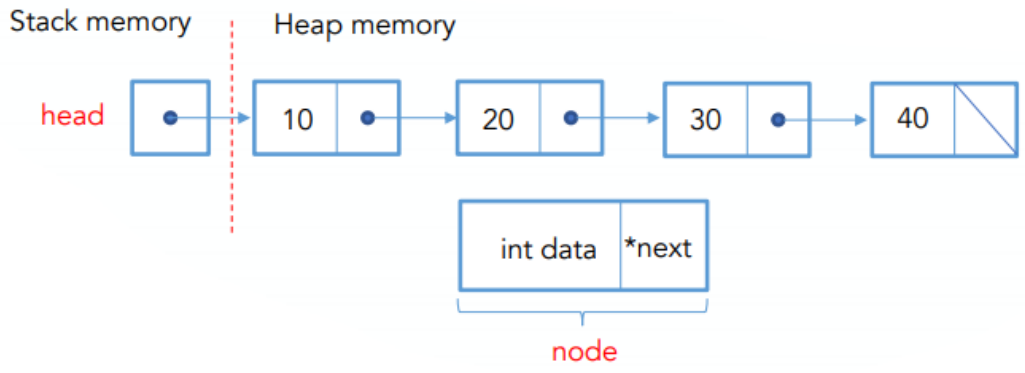
LANGKAH-LANGKAH PRAKTIKUM

A. Dynamic Memory Allocation

Sesuai dengan namanya, memory dapat dialokasikan saat aplikasi berjalan, bukan saat di-compile. Limitasi ukuran dari memory yang dapat dialokasikan terbatas pada jumlah RAM yang terdapat pada perangkat yang digunakan. Implementasi Dynamic Memory Allocation akan dilakukan bersamaan dengan Linked List.

B. Single Linked List

Konsep dasar Single Linked List



- Tutorial 1.1 – Introductory to Single Linked List
  a) Buat sebuah file dengan nama W04_SLL1.c
  b) Salinlah code berikut ini. Baca juga komentar yang terdapat pada code untuk membantu menjelaskan mengenai tahapan-tahapan yang dilakukan.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

// #1 Siapkan tipe data untuk Linked List
typedef struct Mahasiswa{
  char nama[100];
  char jurusan[30];
  int nim;
  // *next digunakan untuk menunjuk ke node selanjutnya
  struct Mahasiswa *next;
}Mahasiswa;

int main(){
  // #2 Siapkan head (kepala) dan sebuah node dari Linked List
  Mahasiswa *head, *node;
  // #3 Selalu buat head sebagai NULL;
  head = NULL;

  // #4 Tahap ini head sudah siap, hanya tinggal diisi dengan *node
  // Sebelum mengisinya dengan *node,
  // *node harus dialokasikan terlebih dahulu di dalam memory
  node = (Mahasiswa*) malloc(sizeof(Mahasiswa));

  // #5 Memasukkan data ke dalam node
  // Terdapat dua cara untuk memasukkan data ke dalam node
  // 1. Tampung terlebih dahulu dalam sebuah variabel
  //     baru setelahnya memasukkannya ke dalam node (line 33-34)
  // 2. Langsung memasukkan data ke dalam node tanpa
  //     perantara variabel lain. (line 35)

  // int nim = 14026;
  // node->nim = nim;
  node->nim = 14026;
  strcpy(node->nama, "James Christian Wira");
  strcpy(node->jurusan, "Informatika");

  // #6 INI BAGIAN PENTING
  // Set node->next sebagai NULL
  // Selalu lakukan ini setiap kali membuat node baru
  node->next = NULL;

  // #7 Setelah node pertama dibuat, node tersebut dapat disambungkan
  // ke head
  head = node;
  // Node pertama dalam Linked List sudah berhasil dibuat

  return 0;
}
```

- Tutorial 1.2 – Insertion at Single Linked List & Showing Data inside Linked List
  Terdapat 2 cara melakukan insertion data ke dalam Linked List, yaitu memasukan data baru ke DEPAN linked list dan memasukkan data baru ke BELAKANG linked list. Tutorial ini hanya mencontohkan pemasukan data dari BELAKANG.
  1. Buatlah file dengan nama W01_SLL2.c
  2. Salinlah code berikut ini (gunakan code dari Tutorial 1.1 sebagai base code). Baca juga komentar yang terdapat pada code untuk membantu menjelaskan mengenai tahapan-tahapan yang dilakukan.
     a. Salin sturct Mahasiswa dari tutorial sebelumnya dan salin function main berikut ini.

```
38  int main(){
39    Mahasiswa *head, *node;
40    head = NULL;
41    node = (Mahasiswa*) malloc(sizeof(Mahasiswa));
42
43    node->nim = 14026;
44    strcpy(node->nama, "James Christian Wira");
45    strcpy(node->jurusan, "Informatika");
46    head = node;
47    // Code di atas merukapakan code dari Tutorial 1.1
48    // yang dihilangkan komentarnya
49
50    // #1 Tambahkan *tail untuk menunjuk node terakhir
51    Mahasiswa *tail;
52    tail = node;
53
54    // #2 Buat node baru. Pembuatan node baru dapat menggunakan
55    // variabel node yang sama namun melakukan ulang malloc
56
57    //Mahasiswa *newNode;
58    // Jika tidak ingin menggunakan variabel node yang sudah ada
59    node = createNewNode(17492, "Matthew Evans", "Informatika");
60    // #3 Jika penambahan dilakukan di belakang
61    tail->next = node ; // #3.1 Set tail->next = node;
62    tail = node; // #3.1 Set tail = node; untuk mengembalikan
63    // posisi tail ke paling akhir
64
65    // #4 Jika penambahan node dilakukan di depan
66    // Penggunaan tail tidak dibutuhkan
67    node = createNewNode(13633, "Justin Susanto", "Informatika");
68    node->next = head; // #4.1 Set node->next = head;
69    head = node; // #4.2 Set head = node; untuk mengembalikan
70    // posisi head ke paling awal
71
72    printLinkedList(head);
73    return 0;
74  }
```

b. Salinlah function printLinkedList dan createNewNode berikut ini. Perhatikan peletakan code berdasarkan nomor baris yang tertera.

```
13  Mahasiswa *createNewNode(int nim, char *nama, char *jurusan){
14      Mahasiswa *newNode = (Mahasiswa*) malloc(sizeof(Mahasiswa));
15      newNode->nim = nim;
16      strcpy(newNode->nama, nama);
17      strcpy(newNode->jurusan, jurusan);
18      newNode->next = NULL;
19      return newNode;
20  }
21
22  void printLinkedList(Mahasiswa *head) {
23      Mahasiswa *temp;
24      // Menyiapkan variabel yang akan berjalan dalam Linked List
25      temp = head; // Set temp ke list paling awal
26      int i = 1;
27      while(temp != NULL){
28          // Lakukan iterasi hingga tidak menemukan node selanjutnya
29          printf("Data ke %d\n", i);
30          printf("NIM     : %d\n", temp->nim);
31          printf("Nama    : %s\n", temp->nama);
32          printf("Jurusan : %s\n\n", temp->jurusan);
33          temp = temp->next;
34          i++;
35      }
36  }
```

- Tutorial 1.3 – Deleting a Node

  Terdapat 3 posisi penghapusan node pada Linked List, yaitu posisi awal, tengah, dan akhir.

  1. Posisi awal (Menggunakan code Tutorial 1.2). Salin dan jalankanlah program

```
48      node = createNewNode(13633, "Justin Susanto", "Informatika");
49      node->next = head;
50      head = node;
51
52      printf("Data SEBELUM di hapus\n");
53      printLinkedList(head);
54
55      Mahasiswa *trash; // Menyiapkan variabel untuk menghapus memory
56      trash = head; // menerima data yang mau dihapus
57      head = head->next; // memindahkan head ke data selanjutnya
58      free(trash); // menghapus data dari head (yang sekarang ada di trash)
59
60      printf("Data SETELAH di hapus\n");
61      printLinkedList(head);
62
63      return 0;
64  }
```

2. Posisi akhir (Menggunakan code Tutorial 1.2). Salin dan jalankanlah program

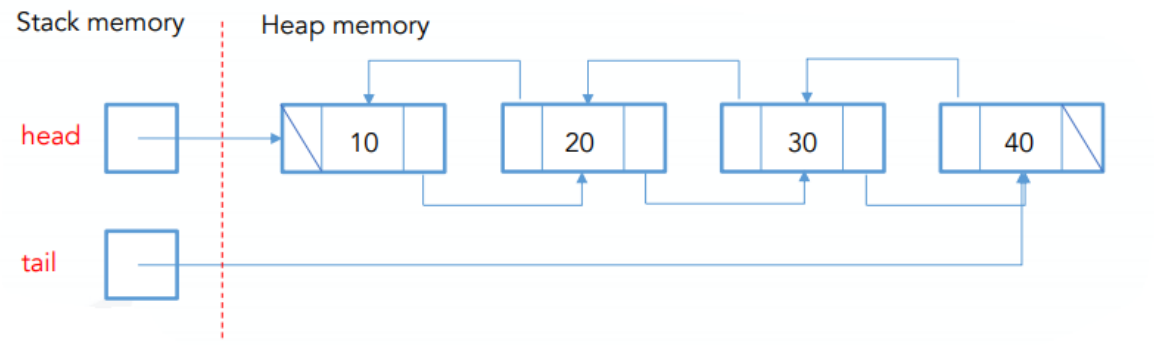
```
55      Mahasiswa *trash; // Menyiapkan variabel untuk menghapus memory
56      trash = head;
57      while(trash->next != tail){
58        // Iterasi sampai node sebelum tail
59        // Akan berhenti jika sudah 1 node sebelum tail
60        trash = trash->next;
61      }
62      tail = trash; // Pindahkan tail ke 1 node sebelumnya
63      trash = tail->next; // Pindahkan trash ke posisi node terakhir
64      tail->next = NULL; // Hapus koneksi tail->next agar tail menjadi node terakhir
65      free(trash); // Hapus trash
66
67      printf("Data SETELAH di hapus\n");
68      printLinkedList(head);
```

3. Posisi tengah (Menggunakan code Tutorial 1.2). Salin dan jalankanlah program

```
55      // Tujuannya adalah ingin menghapus node mahasiswa James Christian Wira
56      Mahasiswa *trash, *temp;
57      // trash: variabel untuk menghapus memory
58      // temp: variabel untuk menyambungkan LinkedList
59      trash = head;
60      while(strcmp(trash->next->nama, "James Christian Wira") == 1){
61        // Iterasi sampai node sebelum tail
62        // Akan berhenti jika sudah 1 node sebelum tail
63        printf("%s\n", trash->nama);
64        trash = trash->next;
65      }
66      if (trash->nim == head->nim) printf("true\n");
67      else printf("false\n");
68      temp = trash; // Simpan posisi sekarang
69      trash = trash->next; // Pindahakan trash ke node yang mau dihapus
70      temp->next = trash->next; // Sambungkan posisi temp ke node setelah trash
71      trash->next = NULL; // Hapus koneksi trash->next agar tidak terhubung ke program
72      free(trash); // Hapus trash
73
74      printf("Data SETELAH di hapus\n");
75      printLinkedList(head);
76
77      return 0;
78  }
```

C. Double Linked List

Konsep Double Linked List sama dengan Single Linked List. Pembeda keduanya adalah, node pada Double Linked List dapat menunjuk ke node sebelumnya, sedangkan node pada Single Linked List tidak dapat melakukannya.



Agar dapat diimplementasikan, di dalam struct harus terdapat sebuah variabel baru untuk melakukan penunjukan ke node sebelumnya.

```
6  typedef struct Mahasiswa{
7      char nama[100];
8      char jurusan[30];
9      int nim;
10     struct Mahasiswa *next, *prev;
11 }Mahasiswa;
```

Variabel *prev merupakan variabel yang akan dipakai untuk menunjuk node sebelumnya.

- Tutorial 2.1 – Creation, Insertion, and Node Deletion in Double Linked List
  1. Buatlah sebuah file dengan nama W04_DLL.c
  2. Pembuatan, penambahan data, dan penghapusan data pada Double Linked List secara serentak dijelaskan pada tutorial ini untuk menghemat waktu.
  3. Code dasar (template) yang digunakan menggunakan code dari Tutorial 1.2
  4. Salin dan jalankan code berikut ini. Perhatikan nomor baris saat menyalin code. Baca juga komentar yang terdapat pada code untuk membantu menjelaskan mengenai tahapan-tahapan yang dilakukan.
     1) Deklarasi include dan struct

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

typedef struct Mahasiswa{
  char nama[100];
  char jurusan[30];
  int nim;
  struct Mahasiswa *next, *prev; // prev untuk menunjuk node sebelumnya
}Mahasiswa;

```

     2) Function int main(). Pada function ini terdapat pembuatan node pertama secara manual.

```c
int main(){
  Mahasiswa *head, *node, *tail;
  head = tail = NULL;

  node = (Mahasiswa*) malloc(sizeof(Mahasiswa));
  node->nim = 14026;
  strcpy(node->nama, "James Christian Wira");
  strcpy(node->jurusan, "Informatika");
  // Untuk pembuatan node pertama. Selalu begini
  // Atau cek apakah head == null? (ada Linked List atau ga)
  node->next = node->prev = NULL;
  head = node;
  tail = node;

  node = (Mahasiswa*) malloc(sizeof(Mahasiswa));
  node->nim = 17492;
  strcpy(node->nama, "Matthew Evans");
  strcpy(node->jurusan, "Informatika");
  node->next = NULL;
  // Penambahan yang dilakukan pada tutorial ini adalah penambahan ke belakang
  node->prev = tail; // Menunjuk ke node 'terakhir'
  tail->next = node; // 'tail'->next menunjuk ke node baru
  tail = node; // Memindahkan tail ke node baru. node baru menjadi tail
  // Line 63-65 harus dilakuin terus setiap kali nambah node
  // Supaya ga diketik berkali-kali,
  // proses ini dipindahin aja ke function createNewNode
  // Yang perlu ditambahin hanya kirim tail ke function tersebut

```

3) Lanjutan function int main(). Gambar lanjutan ini mencontohkan pembuatan Linked List dengan menggunakan function. Function yang dimaksud adalah createNewNode() yang ada di step selanjutnya.

```
101
102     createNewNode(13633, "Justin Susanto", "Informatika", &tail);
103     createNewNode(12335, "Vionie Laorensa", "Informatika", &tail);
104     createNewNode(14045, "Mario Alexander", "Informatika", &tail);
105     createNewNode(11440, "Leonardo Pratama", "Informatika", &tail);
106     createNewNode(15840, "Handriki Kasa", "Informatika", &tail);
107     createNewNode(15773, "Cindy Michelle", "Informatika", &tail);
108     createNewNode(19587, "Aldric Leonardo", "Informatika", &tail);
109
110     printf("All data inside linked list\n");
111     printLinkedList(head);
112
113     printf("All data after 2x head deletion\n");
114     headDeletion(&head);
115     headDeletion(&head);
116     printLinkedList(head);
117
118     printf("All data after 1x tail deletion\n");
119     tailDeletion(&tail);
120     printLinkedList(head);
121
122     printf("All data after middle deletion (Mario Alexander - 14045)\n");
123     middleDeletion(&head, 14045);
124     printLinkedList(head);
125
126     return 0;
127 }
```

4) Function createNewNode(). Function ini digunakan untuk membuat node baru. Penjelasan baris 20-22 dapat dilihat pada baris 93-100

```
13  void *createNewNode(int nim, char *nama, char *jurusan, Mahasiswa **tail){
14      Mahasiswa *newNode = (Mahasiswa*) malloc(sizeof(Mahasiswa));
15      newNode->nim = nim;
16      strcpy(newNode->nama, nama);
17      strcpy(newNode->jurusan, jurusan);
18      newNode->next = NULL;
19      // Tambahin 3 baris ini
20      newNode->prev = *tail;
21      (*tail)->next = newNode;
22      *tail = newNode;
23  }
```

5) Pembuatan function headDeletion() yang akan digunakan untuk menghapus dan memindahkan head ke node selanjutnya.

```
39  void headDeletion(Mahasiswa **head){
40      // Menyiapkan variabel untuk dihapus
41      Mahasiswa *trash = *head;
42      *head = trash->next; // Memindahkan head ke new head
43      (*head)->prev = NULL; // Menghapus koneksi head ke node sebelumnya
44      trash->next = NULL; // Menghapus koneksi node sebelumnya ke new head
45      free(trash); // Menghapus trash
46  }
```

6) Pembuatan function tailDeletion() yang akan digunakan untuk menghapus dan memindahkan tail ke node sebelumnya.

```
48  void tailDeletion(Mahasiswa **tail){
49      // Menyiapkan variabel untuk dihapus
50      Mahasiswa *trash = *tail;
51      *tail = trash->prev; // Memindahkan head ke new head
52      (*tail)->next = NULL; // Menghapus koneksi head ke node sebelumnya
53      trash->prev = NULL; // Menghapus koneksi node sebelumnya ke new head
54      free(trash); // Menghapus trash
55  }
```

7) Pembuatan function middleDeletion() yang akan digunakan untuk menghapus dan menyambungkan kedua node yang terpisah.

```
57  void middleDeletion(Mahasiswa **head, int target){
58      // Menyiapkan variabel untuk dihapus
59      Mahasiswa *trash = *head, *tempBefore, *tempAfter;
60      while(trash->nim != target){ // Cari sampe ketemu yang mau dihapus
61          trash = trash->next;
62      }
63      // Sampai tahap ini, trash adalah node yang akan dihapus
64      // Untuk membuktikannya, silahkan melakukan printf nim atau namanya
65      tempBefore = trash->prev; // tempBefore = 1 node sebelum yang akan dihapus
66      tempAfter = trash->next; // tempAfter = 1 node setelah trash
67      // Menyambungkan 2 buah node yang terpisah oleh trash
68      tempBefore->next = tempAfter;
69      tempAfter->prev = tempBefore;
70      // Menghapus koneksi trash ke Linked List
71      trash->prev = NULL;
72      trash->next = NULL;
73      free(trash); // Menghapus trash
74  }
```

8) Terakhir adalah function printLinkedList() yang digunakan untuk menampilkan isi dari Linked List. Pembuatan function ini dapat didahulukan setelah tahap 4 agar kalian dapat melihat isi Linked List setelah menambahkan data baru ke dalam Linked List

```
25  void printLinkedList(Mahasiswa *head) {
26    Mahasiswa *temp;
27    temp = head;
28    int i = 1;
29    while(temp != NULL){
30      printf("Data ke %d\n", i);
31      printf("NIM     : %d\n", temp->nim);
32      printf("Nama    : %s\n", temp->nama);
33      printf("Jurusan : %s\n\n", temp->jurusan);
34      temp = temp->next;
35      i++;
36    }
37  }
```

D. Tugas

1. Konversikan Tutorial 1.3 menjadi sebuah Circular Linked List. Simpan code dengan nama W04_NIM_T01.c
2. Ketentuan tugas kedua adalah sebagai berikut:
   a) Mahasiswa dapat menggunakan (pilih salah satu)
      (1) Tugas yang sudah dikumpulkan pada Pertemuan/Week 3
      (2) Menggunakan file pendukung (jawaban tugas Pertemuan 3)
      (3) Membuat sendiri dari awal
      *note: melakukan konversi code jika memilih (1) dan (2).
   b) Ubahlah jumlah mahasiswa yang bisa diterima dari 100 menjadi tak terbatas / menggunakan memory allocation, dan penyimpanan data diganti dari array ke dalam bentuk Double Linked List.
   c) Aplikasi yang terdapat pada nomor 2 mempunyai fitur tambahan, yaitu menghapus data mahasiswa dan data nilai mahasiswa berdasarkan NIM yang di-input.
   d) Simpanlah aplikasi dengan nama W04_NIM_T02_XX.c
   e) XX pada nama file mengacu kepada base code yang digunakan sesuai yang tertera pada poin a), yaitu:
      XX = 01 jika memilih (1)
      XX = 02 jika memilih (2)
      XX = 03 jika memilih (3)

REFERENSI