


<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

#### A. TUJUAN

1. Mahasiswa mengenal struktur data link list
2. Mahasiswa mampu menerapkan operasi dasar dalam link list
3. Mahasiswa mampu menyelesaikan aplikasi link list, implementasi simpul/node.

#### B. HARDWARE & SOFTWARE

1. Personal Computer
2. Notepad++
3. DevC++ IDE

#### C. TEORI SINGKAT

##### 1. Simpul/Nodes

Sebuah simpul/node berisi alamat dan kumpulan data, dalam sebuah simpul/node keduanya dibungkus menjadi sebuah objek berupa struct seperti berikut :


```

1. struct node
2. {
3.     int data;
4.     struct node *next;
5. };

```

##### 2. Penggunaan Alokasi Memory

Ketika kita mempelajari tipe data array, nampak kelemahan tipe data ini adalah sifatnya yang statis. Artinya ketika kita mendeklarasikan sebuah variable dengan tipe data array maka data yang kita deklarasikan disimpan pada memori harus dalam kondisi teratur. Selain itu selama program berjalan ukuran dari array bersifat tetap atau kita tidak dapat merubahnya. Adakalanya dalam pemrograman ukuran sebuah obyek belum dapat kita tentukan sampai program kita jalankan. Alokasi memori menyediakan fasilitas untuk membuat ukuran buffer dan array secara dinamik. Dinamik artinya bahwa ruang dalam memori

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

akan dialokasikan ketika program dieksekusi (run time). Fasilitas ini memungkinkan user untuk membuat tipe data dan struktur dengan ukuran dan panjang berapapun yang disesuaikan dengan kebutuhan di dalam program.

#### a. Perintah sizeof()

Sebelum kita menggunakan alokasi memori, kita harus mengenal perintah sizeof. Perintah ini digunakan untuk


- Untuk mendapatkan ukuran dari berbagai tipe data, variabel ataupun struktur.
- Return value : ukuran dari obyek yang bersangkutan dalam byte.
- Parameter dari sizeof() : sebuah obyek atau sebuah tipe data

#### b. Perintah malloc()

Fungsi standar dalam C yang digunakan untuk mengalokasikan memori adalah malloc(). Prototype dari fungsi ini adalah sebagai berikut:

`void *malloc(int jml_byte)` Banyaknya byte yang akan dipesan dinyatakan sebagai parameter fungsi. Return value dari fungsi ini adalah sebuah pointer yang tak bertipe (pointer to void) yang menunjuk ke buffer yang dialokasikan. Pointer tersebut haruslah dikonversi kepada tipe yang sesuai (dengan menggunakan type cast) agar bisa mengakses data yang disimpan dalam buffer. Jika proses alokasi gagal dilakukan, fungsi ini akan memberikan return value berupa sebuah pointer NULL.

```
int *x;
x = (int *) malloc(3 * sizeof(int));
if (x== NULL) {
    printf("Error on malloc\n");
    exit(0);
} else {
    lakukan operasi memori dinamis...
}
```

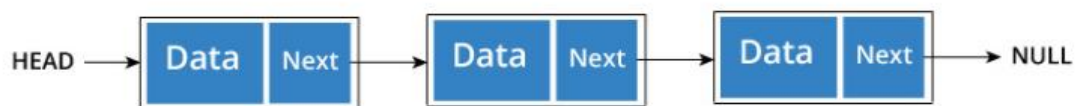
<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

```
int *x;
x = (int *) malloc(3 * sizeof(int));
```

### 3. Single Link List

Di dalam game Treasure Hunt, anda mulai menjelajahnya dengan mencari clue pertama. Di saat anda menemukannya, anda tidak menemukan harta karunnya tapi menemukan clue berikutnya dan begitu seterusnya sampai anda menemukan harta karun.

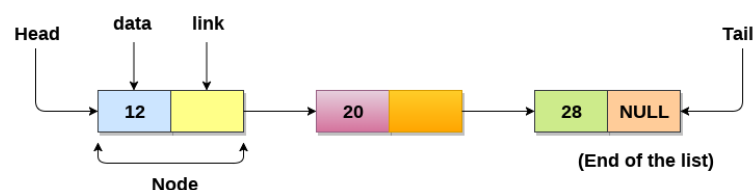
Sebuah linked list sama halnya dengan logika di atas, merupakan sebuah data yang berupa simpul atau node beralamat yang saling bertaut. Setiap simpul bisa menyimpan data yang mana isinya bisa char, int, string atau tipe data lainnya.




**Gambar 1. Representasi Linked List**

Gambar di atas menerangkan sebuah link list sederhana/ Single Linked List, untuk memulainya harus membuat sebuah simpul special yang hanya memiliki alamat/pointer simpul ini dinamakan HEAD.

Kemudia pada akhir simpul/node diberi alamat kosong yang tidak menuju kemanapun yang disebut simpul TAIL.



<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

### Gambar 2. Bentuk lain Linked List

Pada gambar di atas setiap simpul berisi

1. Data lebih dari satu tipe variable
2. Alamat dari simpul lain

Contoh program Linked list sederhana, setiap simpul struct memiliki item data dan sebuah pointer menuju simpul struct lain. Berikut linked list sederhana dengan 3 simpul :

```


6. struct node
7. {
8.     int data;
9.     struct node *next;
10. };

```

```

1. /* Initialize nodes */
2. struct node *head;
3. struct node *one = NULL;
4. struct node *two = NULL;
5. struct node *three = NULL;
6.
7. /* Allocate memory */
8. one = malloc(sizeof(struct node));
9. two = malloc(sizeof(struct node));
10. three = malloc(sizeof(struct node));
11.
12. /* Assign data values */
13. one->data = 1;
14. two->data = 2;
15. three->data=3;
16.
17. /* Connect nodes */
18. one->next = two;
19. two->next = three;
20. three->next = NULL;
21.
22. /* Save address of first node in head */
23. head = one;

```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	


Linked list ini dibuat demikian dengan tujuan mengurutkan data, linked list ini memiliki kemampuan untuk memutuskan urutan data kemudian menyambungkannya lagi, contoh jika ingin menyisipkan sebuah elemen 4 diantara 1 dan 2, langkah-langkahnya sebagai berikut :

1. Create a new struct node and allocate memory to it.
2. Add its data value as 4
3. Point its next pointer to the struct node containing 2 as data value
4. Change next pointer of "1" to the node we just created.
5. Doing something similar in an array would have required shifting the positions of all the subsequent elements.

#### D. PERCOBAAN

##### 1. Alokasi Memori

```
#include<stdio.h>
#include<stdlib.h>
typedef struct employee_st {
    char name[40];
    int id;
}
main()
{
    int myInt;
    Employee john;
    printf("Size of int is %d\n",sizeof(myInt));
    printf("Size of int is %d\n",sizeof(int));
    printf("Size of Employee is %d\n",sizeof(Employee));
    printf("Size of john is %d\n",sizeof(john));
    printf("Size of char is %d\n",sizeof(char));
    printf("Size of short is %d\n",sizeof(short));
    printf("Size of int is %d\n",sizeof(int));
    printf("Size of long is %d\n",sizeof(long));
    printf("Size of float is %d\n",sizeof(float));
    printf("Size of double is %d\n",sizeof(double));
    return 0;
}
```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

## 2. Single Link list without Typedef

```
#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
};

int main()
{
    struct node* head = NULL;
    struct node* second = NULL;
    struct node* third = NULL;

    head      = (struct node*)malloc(sizeof(struct node));
    second    = (struct node*)malloc(sizeof(struct node));
    third     = (struct node*)malloc(sizeof(struct node));

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = NULL;

    return 0;
}
```

## 3. Single Link List (Typedef)


```
#include <stdio.h>
#include <stdlib.h>

struct LinkedList {
    int data;
    struct LinkedList *next;
};

typedef struct LinkedList node;

node *head = NULL;
node *second = NULL;
node *third = NULL;
int main()
{

```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

```

head    = (node*)malloc(sizeof(node));
second  = (node*)malloc(sizeof(node));
third   = (node*)malloc(sizeof(node));

head->data = 1;
head->next = second;

second->data = 2;
second->next = third;

third->data = 3;
third->next = NULL;

return 0;
}

```

#### 4. Single Link List

```

#include <stdio.h>
#include <stdlib.h>

typedef struct LinkedList {
    int data;
    struct LinkedList *next;
} node;

node *head = NULL;
node *second = NULL;
node *third = NULL;

int main()
{
    head    = (node*)malloc(sizeof(node));
    second  = (node*)malloc(sizeof(node));
    third   = (node*)malloc(sizeof(node));


    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = NULL;

    return 0;
}

```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

## 5. Menampilkan Link List

```
#include <stdio.h>
#include <stdlib.h>

typedef struct LinkedList {
    int data;
    struct LinkedList* next;
}node;

int main()
{
    node* head = NULL;
    node* second = NULL;
    node* third = NULL;

    head    = (node*)malloc(sizeof(node));
    second  = (node*)malloc(sizeof(node));
    third   = (node*)malloc(sizeof(node));

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = NULL;

    printf("%d", head->data);

    return 0;
}
```


## 6. Menampilkan Link List

```
#include <stdio.h>
#include <stdlib.h>

struct Node {
    int data;
    struct Node* next;
};

void printList(struct Node* n)
{
    while (n != NULL) {
        printf(" %d ", n->data);
    }
}
```



<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

```

        n = n->next;
    }
}

int main()
{
    struct Node* head = NULL;
    struct Node* second = NULL;
    struct Node* third = NULL;

    head = (struct Node*)malloc(sizeof(struct Node));
    second = (struct Node*)malloc(sizeof(struct Node));
    third = (struct Node*)malloc(sizeof(struct Node));

    head->data = 1;
    head->next = second;

    second->data = 2;
    second->next = third;

    third->data = 3;
    third->next = NULL;

    printList(head);

    return 0;
}

```

## 7. Insertion at Beginning

```


#include <stdio.h>
#include <stdlib.h>

struct Node{
    int data;
    struct Node *next;
};

void push(struct Node** head_ref, int new_data){
    struct Node* new_node = (struct Node*) malloc(sizeof(struct
Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

void printList(struct Node *node)
{
    while (node != NULL)

```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

```

        {
            printf(" %d ", node->data);
            node = node->next;
        }
    }

int main()
{
    struct Node* head = NULL;
    push(&head, 7);
    push(&head, 1);
    printf("\n Created Linked list is: ");
    printList(head);

    return 0;
}

```

## 8. Insertion at Beginning

```

#include <stdio.h>
#include <stdlib.h>

struct node {
    int data;
    struct node *next;
};


void print(struct node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->data);
        node = node->next;
    }
}

int main()
{
    struct node *head = NULL;
    struct node *second = NULL;
    struct node *third = NULL;
    struct node *new_node = NULL;

    head      = (struct node*)malloc(sizeof(struct node));
    second    = (struct node*)malloc(sizeof(struct node));
    third     = (struct node*)malloc(sizeof(struct node));
    new_node  = (struct node*)malloc(sizeof(struct node));

    head->data = 1;

```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

```

        head->next = second;

        second->data = 2;
        second->next = third;

        third->data = 3;
        third->next = NULL;

        new_node->data = 4;
        new_node->next = head;
        head = new_node;

        print(head);
        return 0;
}

```

## 9. LinkedList

```

#include <stdio.h>
#include <stdlib.h>


struct Node
{
    int data;
    struct Node *next;
};

void push(struct Node** head_ref, int new_data)
{
    struct Node* new_node = (struct Node*) malloc(sizeof(struct
Node));
    new_node->data = new_data;
    new_node->next = (*head_ref);
    (*head_ref) = new_node;
}

void insertAfter(struct Node* prev_node, int new_data)
{
    if (prev_node == NULL)
    {
        printf("the given previous node cannot be NULL");
        return;
    }

    struct Node* new_node = (struct Node*) malloc(sizeof(struct
Node));
    new_node->data = new_data;
    new_node->next = prev_node->next;
}

```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

```

        prev_node->next = new_node;
    }

void append(struct Node** head_ref, int new_data)
{
    struct Node* new_node = (struct Node*) malloc(sizeof(struct
Node));
    struct Node *last = *head_ref;

    new_node->data = new_data;
    new_node->next = NULL;

    if (*head_ref == NULL)
    {
        *head_ref = new_node;
        return;
    }

    while (last->next != NULL)
        last = last->next;

    last->next = new_node;
    return;
}


void printList(struct Node *node)
{
    while (node != NULL)
    {
        printf(" %d ", node->data);
        node = node->next;
    }
}

int main()
{
    struct Node* head = NULL;
    append(&head, 6);
    push(&head, 7);
    push(&head, 1);
    append(&head, 4);
    insertAfter(head->next, 8);

    printf("\n Created Linked list is: ");
    printList(head);

    return 0;
}

```

<b>JOB SHEET 03</b>	
<b>Mata Kuliah :</b> Praktikum Struktur Data	<b>Kode :</b> INF1.62.2014
<b>Program Studi :</b> Teknik Informatika	<b>Waktu :</b> 2 x 50 Menit
<b>Jurusan :</b> Teknik Elektronika	<b>Fakultas :</b> Teknik
<b>Topik : Single Linked List</b>	

}

## E. TUGAS

1. Tampilkan bilangan Fibonacci pertama sampai ke-n menggunakan Pointer dengan Malloc, dimana n dimasukkan oleh user. Gunakan pengecekan pengalokasian dan fungsi free di akhir program.
2. Tampilkan bilangan Prima pertama sampai ke-n menggunakan Pointer dengan Malloc, dimana n dimasukkan oleh user. Gunakan fungsi realloc() untuk mengalokasikan ukuran memori yang baru (m), dimana m dimasukkan oleh user.

## F. DAFTAR PUSTAKA

1. Cipta Ramadhani. 2015. Dasar Algoritma & Struktur Data dengan Bahasa Java. Yogyakarta: ANDI.
2. Wu, C. Thomas. 2010. *An Introduction to Object-Oriented Programming with Java 5<sup>th</sup> Edition*. C. USA: McGraw – Hill Education.
3. Nemeyer, Patrick and Luck, Daniel. 2013. *Learning Java 4<sup>th</sup> Edition*. O'Reilly
4. Sharan, Kishori. 2014. *Beginning Java 8 Fundamentals*. Apress.
5. Schildt, Herbert. 2014. *Java: The Complete Reference 9<sup>th</sup> Edition*. McGraw – Hill Education.