

## Tugas Praktikum ASD - Struct dan Stack

Nama: Gibraltar Aldebran Abdallah Dj

Nim: 1203230033

Kelas: IF-03-02

### 1.Source Code

```
#include <stdio.h>

struct node
{
    struct node *link;
    char alphabet;
};

int main()
{
    // Node initialization
    struct node l1, l2, l3, l4, l5, l6, l7, l8, l9;

    l1.link = NULL;
    l1.alphabet = 'F';

    l2.link = NULL;
    l2.alphabet = 'M';

    l3.link = NULL;
    l3.alphabet = 'A';

    l4.link = NULL;
    l4.alphabet = 'I';

    l5.link = NULL;
    l5.alphabet = 'K';

    l6.link = NULL;
    l6.alphabet = 'T';

    l7.link = NULL;
    l7.alphabet = 'N';

    l8.link = NULL;
    l8.alphabet = 'O';
```

```

19.link = NULL;
19.alphabet = 'R';

// Linking nodes
14.link = &l7; // N
17.link = &l1; // F
11.link = &l8; // O
18.link = &l9; // R
19.link = &l2; // M
12.link = &l3; // A
13.link = &l6; // T
16.link = &l4; // I

// Print linked list
printf("%c",
14.alphabet); // I
printf("%c", 14.link-
>alphabet); // N
printf("%c", 14.link->link-
>alphabet); // F
printf("%c", 14.link->link->link-
>alphabet); // O
printf("%c", 14.link->link->link->link-
>alphabet); // R
printf("%c", 14.link->link->link->link->link->link-
>alphabet); // M
printf("%c", 14.link->link->link->link->link->link->link-
>alphabet); // A
printf("%c", 14.link->link->link->link->link->link->link->link-
>alphabet); // T
printf("%c", 14.link->link->link->link->link->link->link->link->link->alphabet);
// I

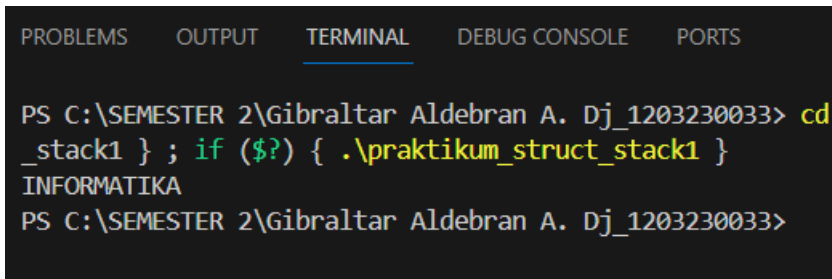
14.link = &l5;
15.link = &l3;

printf("%c", 14.link->alphabet); // K
printf("%c", 14.link->link->alphabet); // A

return 0;
}

```

## Output



```
PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS

PS C:\SEMESTER 2\Gibraltar Aldebran A. Dj_1203230033> cd
_stack1 } ; if ($?) { .\praktikum_struct_stack1 }
INFORMATIKA
PS C:\SEMESTER 2\Gibraltar Aldebran A. Dj_1203230033>
```

## Pejelasan

### Struktur Node

-Program mendefinisikan `struct node` yang memiliki dua field: pointer `link` ke node berikutnya dan `char alphabet` untuk menyimpan karakter.

### Inisialisasi Node

-Delapan node ('11' hingga '19') diinisialisasi dengan huruf tertentu ('F', 'M', 'A', 'T', 'K', 'T', 'N', 'O', 'R') dan link mereka diatur ke `NULL`, artinya awalnya tidak ada node yang terhubung satu sama lain.

### Pembentukan Linked List

-Node-node tersebut kemudian dihubungkan untuk membentuk kata "INFORMATI". Ini dilakukan dengan menetapkan pointer `link` dari setiap node ke node berikutnya dalam urutan kata tersebut.

-Misalnya, `l4.link = &l7;` menghubungkan node yang menyimpan 'T' ke node yang menyimpan 'N', dan seterusnya, sampai terbentuk siklus yang kembali ke node awal (node yang menyimpan 'I').

### Pencetakan Linked List

-Program mencetak karakter-karakter yang terhubung dalam linked list, dimulai dari `l4` ('I') dan mengikuti pointer `link` untuk menampilkan "INFORMATI". Hal ini dilakukan dengan mengakses field `alphabet` dari masing-masing node yang ditunjuk.

### Modifikasi Linked List

-Setelah pencetakan pertama, struktur linked list diubah dengan menetapkan `l4.link = &l5;` dan `l5.link = &l3;`, yang mengubah urutan linked list sehingga kini 'I' diikuti oleh 'K' dan 'A'.

-Modifikasi ini memutus siklus awal dan membentuk sebuah sub-sequence baru dari linked list.

### Pencetakan Setelah Modifikasi

-Program kemudian mencetak karakter-karakter yang terhubung dalam urutan baru, yaitu "KA", dengan mengikuti pointer `link` yang baru diatur.

## 2.Source Code

```
#include <assert.h>
#include <ctype.h>
#include <limits.h>
#include <math.h>
#include <stdbool.h>
#include <stddef.h>
#include <stdint.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

char *readline();
char *ltrim(char *);
char *rtrim(char *);
char **split_string(char *);

int parse_int(char *);

/*
 * Complete the 'twoStacks' function below.
 *
 * The function is expected to return an INTEGER.
 * The function accepts following parameters:
 * 1. INTEGER maxSum
 * 2. INTEGER_ARRAY a
 * 3. INTEGER_ARRAY b
 */

int twoStacks(int maxSum, int a_count, int *a, int b_count, int *b)
{
    int i = 0, j = 0, sum = 0, count = 0;
    while (i < a_count && sum + a[i] <= maxSum)
    {
        sum += a[i];
        i++;
    }
    count = i;
    while (j < b_count && i >= 0)
    {
        sum += b[j];
        j++;
        while (sum > maxSum && i > 0)
        {
            i--;
            sum -= a[i];
        }
        if (sum <= maxSum && i + j > count)
        {
            count = i + j;
        }
    }
    return count;
}
```

```

        {
            count = i + j;
        }
    }
    return count;
}

int main()
{
    FILE *fptr = fopen(getenv("OUTPUT_PATH"), "w");

    int g = parse_int(ltrim(rtrim(readline())));

    for (int g_itr = 0; g_itr < g; g_itr++)
    {
        char **first_multiple_input = split_string(rtrim(readline()));

        int n = parse_int(*(first_multiple_input + 0));

        int m = parse_int(*(first_multiple_input + 1));

        int maxSum = parse_int(*(first_multiple_input + 2));

        char **a_temp = split_string(rtrim(readline()));

        int *a = malloc(n * sizeof(int));

        for (int i = 0; i < n; i++)
        {
            int a_item = parse_int(*(a_temp + i));

            *(a + i) = a_item;
        }

        char **b_temp = split_string(rtrim(readline()));

        int *b = malloc(m * sizeof(int));

        for (int i = 0; i < m; i++)
        {
            int b_item = parse_int(*(b_temp + i));

            *(b + i) = b_item;
        }

        int result = twoStacks(maxSum, n, a, m, b);

        fprintf(fptr, "%d\n", result);
    }
}

```

```

    }

    fclose(fptr);

    return 0;
}

char *readline()
{
    size_t alloc_length = 1024;
    size_t data_length = 0;

    char *data = malloc(alloc_length);

    while (true)
    {
        char *cursor = data + data_length;
        char *line = fgets(cursor, alloc_length - data_length, stdin);

        if (!line)
        {
            break;
        }

        data_length += strlen(cursor);

        if (data_length < alloc_length - 1 || data[data_length - 1] == '\n')
        {
            break;
        }

        alloc_length <<= 1;

        data = realloc(data, alloc_length);

        if (!data)
        {
            data = '\0';

            break;
        }
    }

    if (data[data_length - 1] == '\n')
    {
        data[data_length - 1] = '\0';

        data = realloc(data, data_length);
    }
}

```

```

        if (!data)
        {
            data = '\0';
        }
    }
    else
    {
        data = realloc(data, data_length + 1);

        if (!data)
        {
            data = '\0';
        }
        else
        {
            data[data_length] = '\0';
        }
    }

    return data;
}

```

```

char *ltrim(char *str)
{
    if (!str)
    {
        return '\0';
    }

    if (!*str)
    {
        return str;
    }

    while (*str != '\0' && isspace(*str))
    {
        str++;
    }

    return str;
}

```

```

char *rtrim(char *str)
{
    if (!str)
    {
        return '\0';
    }
}

```

```

    }

    if (!*str)
    {
        return str;
    }

    char *end = str + strlen(str) - 1;

    while (end >= str && isspace(*end))
    {
        end--;
    }

    *(end + 1) = '\\0';

    return str;
}

char **split_string(char *str)
{
    char **splits = NULL;
    char *token = strtok(str, " ");

    int spaces = 0;

    while (token)
    {
        splits = realloc(splits, sizeof(char *) * ++spaces);

        if (!splits)
        {
            return splits;
        }

        splits[spaces - 1] = token;

        token = strtok(NULL, " ");
    }

    return splits;
}

int parse_int(char *str)
{
    char *endptr;
    int value = strtol(str, &endptr, 10);

```



```

    if (endptr == str || *endptr != '\0')
    {
        exit(EXIT_FAILURE);
    }

    return value;
}

```

## Input dan Output

```

PROBLEMS  OUTPUT  TERMINAL  DEBUG CONSOLE  PORTS

PS C:\SEMESTER 2\Gibraltar Aldebran A. Dj_1203230033> cd
_stack2 } ; if ($?) { .\praktikum_struct_stack2 }
1
5 4 11
4 5 2 1 1
3 1 1 2
PS C:\SEMESTER 2\Gibraltar Aldebran A. Dj_1203230033>

```

## Penjelasan

### Fungsi Utama `twoStacks`

Parameter:

- `maxSum`: Batas maksimum jumlah nilai yang bisa diambil dari kedua stack.
- `a\_count`: Jumlah elemen di stack pertama.
- `a`: Array yang merepresentasikan elemen-elemen di stack pertama.
- `b\_count`: Jumlah elemen di stack kedua.
- `b`: Array yang merepresentasikan elemen-elemen di stack kedua.

Kembali (Return): Jumlah maksimum elemen yang bisa diambil dari kedua stack tanpa melebihi `maxSum`.

### Logika `twoStacks`

1. Dimulai dengan mengambil elemen dari stack pertama (`a`) selama jumlah totalnya tidak melebihi `maxSum`.
2. Setiap kali elemen ditambahkan ke total, pointer `i` (untuk stack `a`) ditingkatkan.
3. Setelah tidak bisa mengambil lagi dari `a` tanpa melebihi `maxSum`, program mulai mengambil elemen dari stack kedua (`b`) menggunakan pointer `j`.
4. Jika penambahan elemen dari `b` membuat total melebihi `maxSum`, program kemudian mengurangi elemen yang sebelumnya diambil dari `a` (dengan mengurangi nilai `i` dan

mengurangi jumlah tersebut dari total) untuk mencoba memasukkan lebih banyak elemen dari `b`.

5. Proses ini berlanjut sampai tidak bisa lagi mengambil elemen dari `b` tanpa melebihi `maxSum` atau semua elemen dari `a` telah dikembalikan.

### **Pencapaian**

Fungsi `twoStacks` secara efisien mencari kombinasi pengambilan elemen dari kedua stack untuk memaksimalkan jumlah elemen yang diambil tanpa melebihi batas `maxSum`.

### **Fungsi Pendukung**

Program ini juga termasuk beberapa fungsi pendukung untuk memudahkan input/output dan manipulasi string:

- `getline`, `ltrim`, dan `rtrim` digunakan untuk membaca dan memformat input dari pengguna atau file.
- `split\_string` memecah string input berdasarkan spasi untuk mendapatkan elemen-elemen terpisah.
- `parse\_int` mengubah string menjadi integer.

### **Alur Eksekusi `main`**

1. Membaca jumlah kasus uji (`g`) dari input.
2. Untuk setiap kasus uji, program membaca nilai `n` (jumlah elemen di stack pertama), `m` (jumlah elemen di stack kedua), dan `maxSum`.
3. Stack `a` dan `b` diisi dengan nilai-nilai yang dibaca dari input.
4. Fungsi `twoStacks` dipanggil dengan stack dan batas yang diberikan, dan hasilnya dicetak ke file output yang ditentukan oleh `OUTPUT\_PATH`.