

Nama : Nugraha Suryapratama

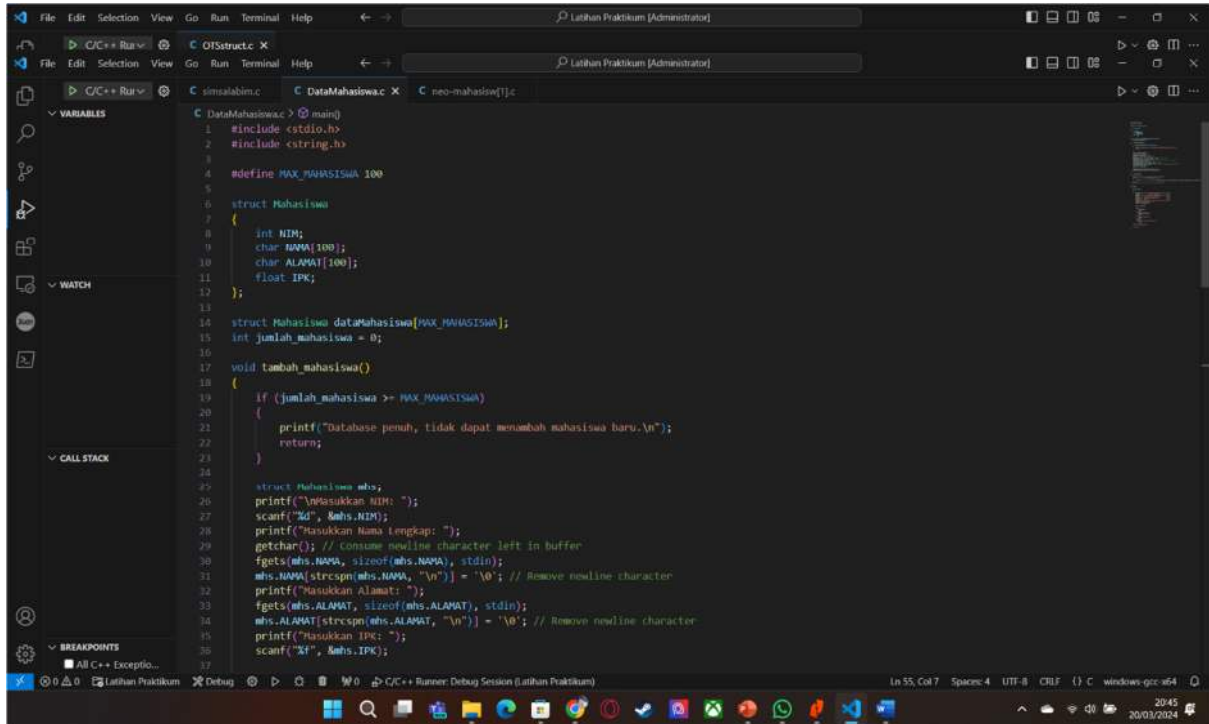
NIM : 1203230113

Kelas : IF 03-03

## Laporan Tugas OTH Week 6

### Soal 1

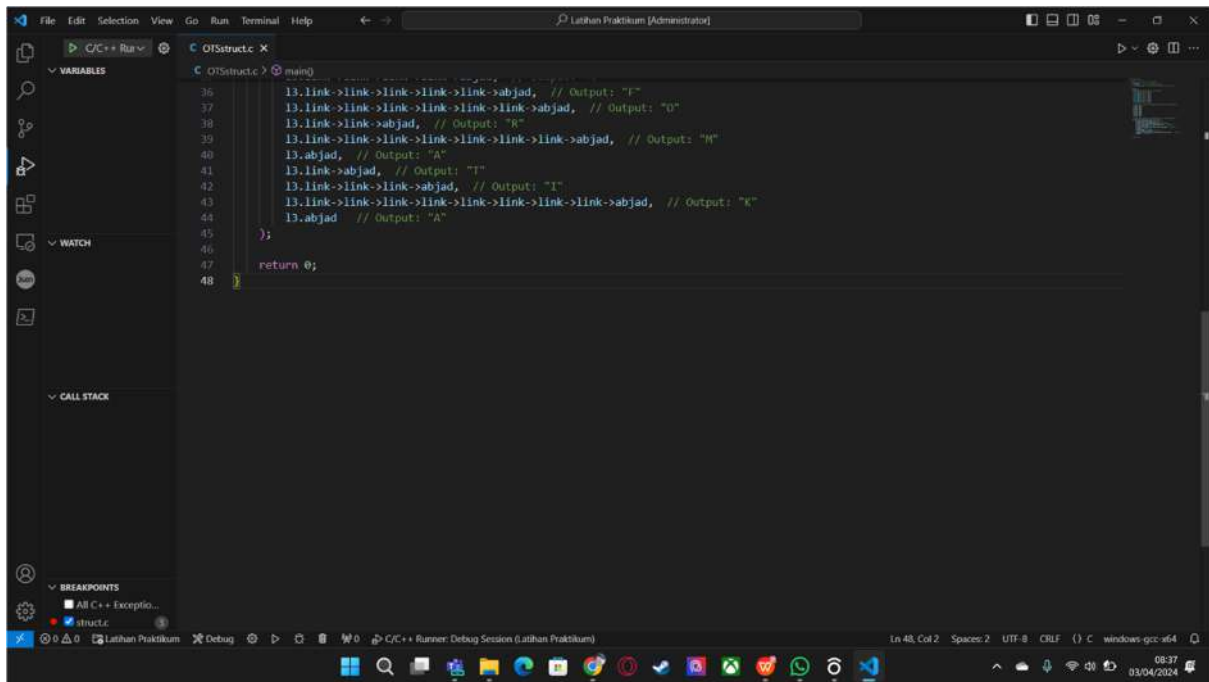
### Screenshot Program



The screenshot shows a C++ program in a debugger (Visual Studio) with the following code:

```
1 #include <stdio.h>
2 #include <string.h>
3
4 #define MAX_MAHASISWA 100
5
6 struct Mahasiswa
7 {
8     int NIM;
9     char NAMA[100];
10    char ALAMAT[100];
11    float IPK;
12 };
13
14 struct Mahasiswa dataMahasiswa[MAX_MAHASISWA];
15 int jumlah_mahasiswa = 0;
16
17 void tambah_mahasiswa()
18 {
19     if (jumlah_mahasiswa >= MAX_MAHASISWA)
20     {
21         printf("Database penuh, tidak dapat menambah mahasiswa baru.\n");
22         return;
23     }
24
25     struct Mahasiswa mhs;
26     printf("\nMasukkan NIM: ");
27     scanf("%d", &mhs.NIM);
28     printf("Masukkan Nama lengkap: ");
29     getchar(); // Consume newline character left in buffer
30     fgets(mhs.NAMA, sizeof(mhs.NAMA), stdin);
31     mhs.NAMA[strcspn(mhs.NAMA, "\n")] = '\0'; // Remove newline character
32     printf("Masukkan Alamat: ");
33     fgets(mhs.ALAMAT, sizeof(mhs.ALAMAT), stdin);
34     mhs.ALAMAT[strcspn(mhs.ALAMAT, "\n")] = '\0'; // Remove newline character
35     printf("Masukkan IPK: ");
36     scanf("%f", &mhs.IPK);
37 }
```

The program is a student database management system. It defines a constant `MAX_MAHASISWA` of 100, a `Mahasiswa` struct with fields `NIM`, `NAMA`, `ALAMAT`, and `IPK`, and an array `dataMahasiswa` of size `MAX_MAHASISWA`. The `tambah_mahasiswa` function checks if the database is full and, if not, prompts the user to enter student details (NIM, Nama, Alamat, and IPK) and adds them to the database.



```
36 13.link->link->link->link->link->abjad, // Output: "F"
37 13.link->link->link->link->link->abjad, // Output: "D"
38 13.link->link->abjad, // Output: "R"
39 13.link->link->link->link->link->link->abjad, // Output: "M"
40 13.abjad, // Output: "A"
41 13.link->abjad, // Output: "I"
42 13.link->link->link->abjad, // Output: "I"
43 13.link->link->link->link->link->link->link->abjad, // Output: "K"
44 13.abjad // Output: "A"
45
46
47
48 return 0;
```

## Penjelasan Source Code :

### 1. Pemasukan Header:

- `#include <stdio.h>`: Baris ini memasukan library standar input/output, menyediakan fungsi seperti `printf` untuk mencetak ke konsol.

### 2. Definisi Struktur:

- `struct Batu {`: Ini mendeklarasikan sebuah struktur bernama Batu.
  - `char abjad;`: Variabel member ini di dalam struktur menyimpan karakter alfabet (diperlakukan sebagai karakter tunggal).
  - `struct Batu *link;`: Ini adalah penunjuk ke struktur Batu lainnya, memungkinkan batu-batu untuk dihubungkan bersama dalam sebuah urutan.

### 3. Fungsi Utama:

- `int main() {`: Ini adalah titik masuk program.

### 4. Inisialisasi Batu:

- Serangkaian variabel struct Batu dideklarasikan dan diberi nilai:
  - `l1, l2, ..., l19`: Variabel-variabel ini merepresentasikan batu-batu individual, masing-masing menyimpan karakter alfabet tunggal ('F', 'M', dll.) dan awalnya memiliki penunjuk link yang diatur ke NULL.

### 5. Menghubungkan Batu-Batu:

- Penunjuk link dari setiap batu diatur untuk menunjuk ke batu berikutnya dalam urutan lingkaran yang diinginkan:

- `l7.link = &l1;`; Batu l7 sekarang menunjuk ke l1.
- Penugasan serupa dilakukan untuk batu-batu yang tersisa, menciptakan loop berbentuk lingkaran dimana batu terakhir (l4) menunjuk kembali ke l7.

## 6. Mengakses Alfabet (Mencetak):

- `printf("%c %c %c %c ...\\n", ...);`: Baris ini menggunakan printf untuk mencetak urutan karakter.
- Di dalam format string printf:
  - Ekspresi seperti `l3.link->link->link->abjad` mengakses karakter dalam linked list dengan mengikuti penunjuk link dari l3 untuk sejumlah langkah tertentu.

## Screenshot Output

```
PS C:\Users\Wugraha\Suryapratama\Documents\Latihan Praktikum> cd "C:\Users\Wugraha\Suryapratama\Documents\Latihan Praktikum\" ; if ($?) { gcc OTSstruct.c -o OTSstruct } ; if ($?) { .\OTSstruct }
INFORMATIKA
```

## Nomor 2

## Screenshot Program

```
File Edit Selection View Go Run Terminal Help
Latihan Praktikum [Administrator]

EXPLORER
  LATIHAN PRAKTIKUM
    tasks.json
    1203230113_NUGRA...
    1203230113_NUGRA...
    1203230113_NUGRA...
    1203230113_NUGRA...
    1203230113_NUGRA...
    1203230113_NUGRA...
    catur.c
    catur.exe
    DataMahasiswa.c
    DataMahasiswa.exe
    kasino.c
    OTSstruct1.c
    OTSstruct2.c
    OTSstruct2.exe
    OTSstruct.exe
    praktikumstack.c
    praktikumstack.exe
    simulabim.c
    simulabim.exe
    stack.c
    struct.exe
    tempCodeRunnerFile.c
  OUTLINE
    read_integers(int)
    twoStacks(int, int, L...)
    main()
  TIMELINE

C:\OTSstruct2.c
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4
5  int* read_integers(int size) {
6      int* arr = malloc(size * sizeof(int));
7      for (int i = 0; i < size; i++) {
8          scanf("%d", &arr[i]);
9      }
10     return arr;
11 }
12
13 int twoStacks(int maxSum, int a_count, int* a, int b_count, int* b) {
14     int i = 0, j = 0, count = 0, sum = 0;
15
16     // Simulate removing elements from stack a
17     while (i < a_count && sum + a[i] <= maxSum) {
18         sum += a[i];
19         i++;
20         count++;
21     }
22
23     // Simulate removing elements from stack b, while adjusting sum
24     while (j < b_count && i >= 0) {
25         sum += b[j];
26         j++;
27
28         // If sum exceeds maxSum, remove elements from stack a until it's within limit
29         while (sum > maxSum && i > 0) {
30             i--;
31             sum -= a[i];
32         }
33
34         // Update count if the current sum is within limit and greater than previous count
35         if (sum <= maxSum && i + j > count) {
36             count = i + j;
37         }
38     }
39 }
```

```
36
37
38
39
40
41
42
43
44 int main() {
45     int numTestCases;
46     scanf("%d", &numTestCases);
47
48     for (int i = 0; i < numTestCases; i++) {
49         int n, m, maxSum;
50         scanf("%d %d %d", &n, &m, &maxSum);
51
52         int* stackA = read_integers(n);
53         int* stackB = read_integers(m);
54
55         int result = twoStacks(maxSum, n, stackA, m, stackB);
56         printf("%d\n", result);
57
58         free(stackA);
59         free(stackB);
60     }
61
62     return 0;
}
```

## Penjelasan Kode :

### Fungsi-fungsi yang terlibat:

1. **read\_integers:** Fungsi ini membaca nilai integer dari masukan pengguna dan mengembalikan array integer yang berisi nilai-nilai tersebut.
2. **twoStacks:** Fungsi ini melakukan simulasi penghapusan elemen dari dua tumpukan untuk mencari jumlah maksimum elemen yang dapat dihapus tanpa melanggar batasan jumlah total.
3. **main:** Fungsi ini berfungsi sebagai titik masuk program, membaca kasus uji dari masukan pengguna, dan memanggil fungsi **twoStacks** untuk setiap kasus uji untuk menghitung hasilnya.

### Penjelasan detail fungsi twoStacks:

1. **Simulasi Penghapusan dari tumpukan a:**
  - o Fungsi ini melakukan iterasi melalui tumpukan a dan menambahkan elemen ke variabel sum selama jumlah tidak melebihi maxSum.
  - o Setiap elemen dihitung dalam variabel count untuk melacak jumlah elemen yang dihapus.
2. **Simulasi Penghapusan dari tumpukan b:**
  - o Fungsi ini melakukan iterasi melalui tumpukan b sambil juga menyesuaikan nilai sum.
  - o Jika sum melebihi maxSum setelah menambahkan elemen dari tumpukan b, fungsi ini menghapus elemen dari tumpukan a hingga sum kembali berada di bawah batas.

- Fungsi ini memperbarui count jika jumlah elemen yang dihapus saat ini lebih besar dari nilai count sebelumnya.

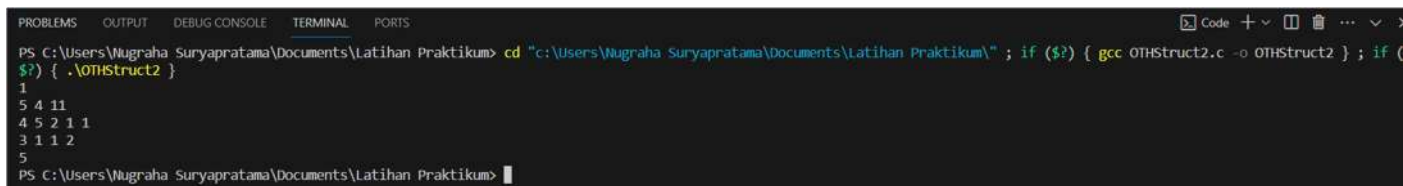
#### Fungsi main:

1. Membaca jumlah kasus uji dari masukan pengguna.
2. Untuk setiap kasus uji:
  - Membaca ukuran tumpukan a dan b, serta nilai maxSum.
  - Memanggil read\_integers untuk membaca elemen tumpukan a dan b.
  - Memanggil twoStacks untuk menghitung jumlah maksimum elemen yang dapat dihapus.
  - Mencetak hasil.
  - Membebaskan memori yang dialokasikan untuk array.

#### Inti dari kode:

Kode ini bertujuan untuk menemukan cara optimal untuk menghapus elemen dari dua tumpukan sekaligus, dengan tetap mematuhi batasan jumlah total yang dihapus. Hal ini dilakukan dengan simulasi penghapusan elemen dari tumpukan secara berurutan, menjaga nilai total agar tidak melebihi batas yang ditentukan.

#### Screenshot Output :



```
PS C:\Users\Nugraha Suryapratama\Documents\Latihan Praktikum> cd "c:\Users\Nugraha Suryapratama\Documents\Latihan Praktikum\" ; if ($?) { gcc 0THStruct2.c -o 0THStruct2 } ; if ($?) { .\0THStruct2 }
1
5 4 11
4 5 2 1 1
3 1 1 2
5
PS C:\Users\Nugraha Suryapratama\Documents\Latihan Praktikum>
```