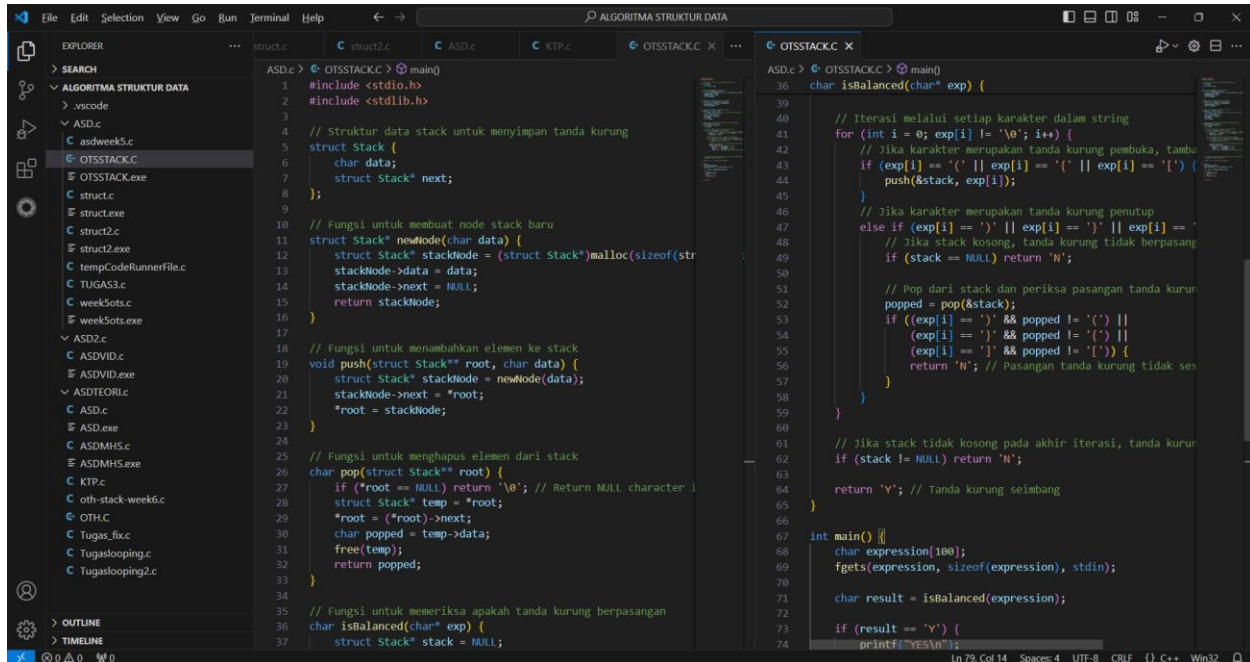


TUGAS PRAKTIKUM OTH

NAMA:KEVIN BETESDA KORNELIUS BANGUN

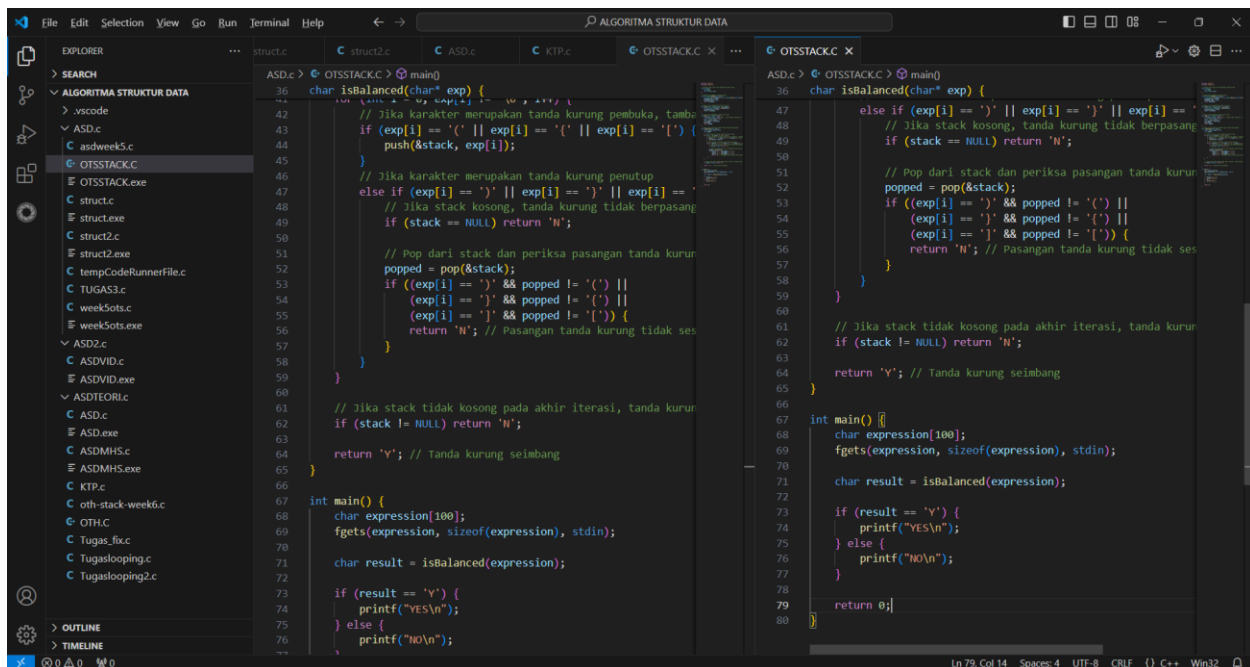
NIM:1203230019

KELAS:IF 03-02



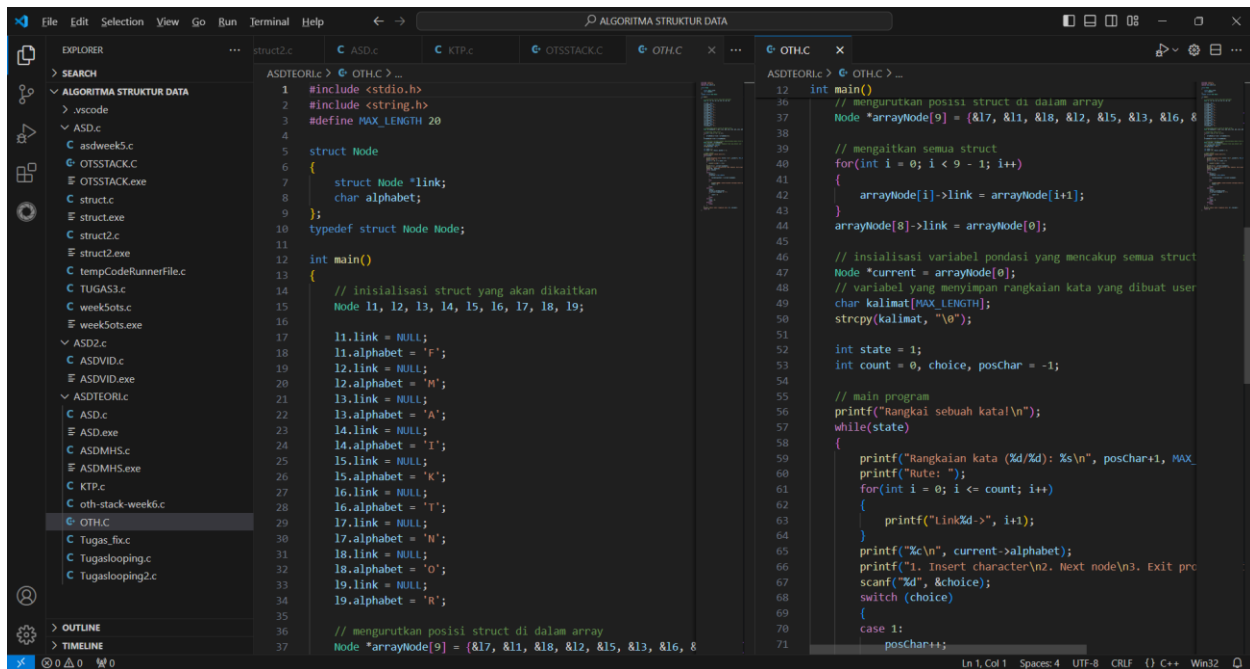
```
ASD.c > OTSSTACK.C > main()
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 // Struktur data stack untuk menyimpan tanda kurung
5 struct Stack {
6     char data;
7     struct Stack* next;
8 };
9
10 // Fungsi untuk membuat node stack baru
11 struct Stack* newNode(char data) {
12     struct Stack* stackNode = (struct Stack*)malloc(sizeof(struct Stack));
13     stackNode->data = data;
14     stackNode->next = NULL;
15     return stackNode;
16 }
17
18 // Fungsi untuk menambahkan elemen ke stack
19 void push(struct Stack** root, char data) {
20     struct Stack* stackNode = newNode(data);
21     stackNode->next = *root;
22     *root = stackNode;
23 }
24
25 // Fungsi untuk menghapus elemen dari stack
26 char pop(struct Stack** root) {
27     if (*root == NULL) return '\0'; // Return NULL character
28     struct Stack* temp = *root;
29     *root = (*root)->next;
30     char popped = temp->data;
31     free(temp);
32     return popped;
33 }
34
35 // Fungsi untuk memeriksa apakah tanda kurung berpasangan
36 char isBalanced(char* exp) {
37     struct Stack* stack = NULL;
```

```
ASD.c > OTSSTACK.C > main()
36 char isBalanced(char* exp) {
37
38     // Iterasi melalui setiap karakter dalam string
39     for (int i = 0; exp[i] != '\0'; i++) {
40         // Jika karakter merupakan tanda kurung pembuka, tamba
41         if (exp[i] == '(' || exp[i] == '[' || exp[i] == '{') {
42             push(&stack, exp[i]);
43         }
44         // Jika karakter merupakan tanda kurung penutup
45         else if (exp[i] == ')' || exp[i] == ']' || exp[i] == '}') {
46             // Jika stack kosong, tanda kurung tidak berpasang
47             if (stack == NULL) return 'N';
48
49             // Pop dari stack dan periksa pasangan tanda kurung
50             popped = pop(&stack);
51             if ((exp[i] == ')' && popped != '(') ||
52                 (exp[i] == ']' && popped != '[') ||
53                 (exp[i] == '}' && popped != '{')) {
54                 return 'N'; // Pasangan tanda kurung tidak ses
55             }
56         }
57     }
58
59     // Jika stack tidak kosong pada akhir iterasi, tanda kurung
60     if (stack != NULL) return 'N';
61
62     return 'Y'; // Tanda kurung seimbang
63 }
64
65 int main() {
66     char expression[100];
67     fgets(expression, sizeof(expression), stdin);
68
69     char result = isBalanced(expression);
70
71     if (result == 'Y') {
72         printf("YES\n");
73     }
74 }
```

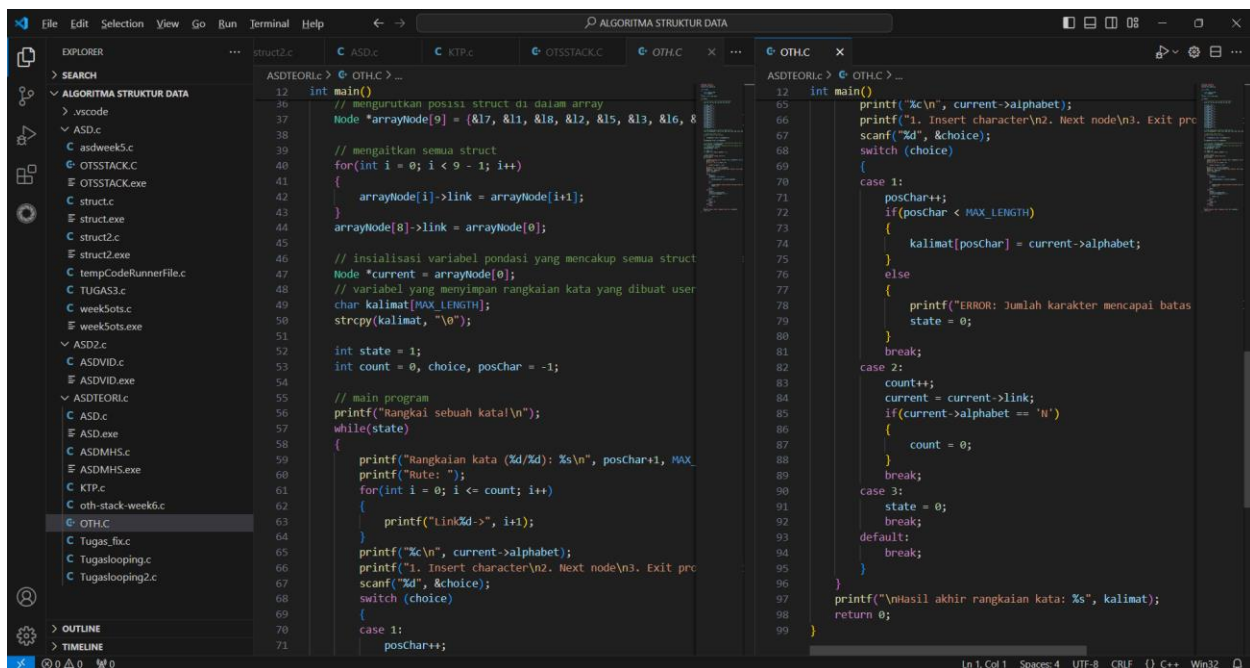


```
ASD.c > OTSSTACK.C > main()
36 char isBalanced(char* exp) {
37
38     // Iterasi melalui setiap karakter dalam string
39     for (int i = 0; exp[i] != '\0'; i++) {
40         // Jika karakter merupakan tanda kurung pembuka, tamba
41         if (exp[i] == '(' || exp[i] == '[' || exp[i] == '{') {
42             push(&stack, exp[i]);
43         }
44         // Jika karakter merupakan tanda kurung penutup
45         else if (exp[i] == ')' || exp[i] == ']' || exp[i] == '}') {
46             // Jika stack kosong, tanda kurung tidak berpasang
47             if (stack == NULL) return 'N';
48
49             // Pop dari stack dan periksa pasangan tanda kurung
50             popped = pop(&stack);
51             if ((exp[i] == ')' && popped != '(') ||
52                 (exp[i] == ']' && popped != '[') ||
53                 (exp[i] == '}' && popped != '{')) {
54                 return 'N'; // Pasangan tanda kurung tidak ses
55             }
56         }
57     }
58
59     // Jika stack tidak kosong pada akhir iterasi, tanda kurung
60     if (stack != NULL) return 'N';
61
62     return 'Y'; // Tanda kurung seimbang
63 }
64
65 int main() {
66     char expression[100];
67     fgets(expression, sizeof(expression), stdin);
68
69     char result = isBalanced(expression);
70
71     if (result == 'Y') {
72         printf("YES\n");
73     } else {
74         printf("NO\n");
75     }
76 }
```

HACKER RANK



```
1 #include <stdio.h>
2 #include <string.h>
3 #define MAX_LENGTH 20
4
5 struct Node
6 {
7     struct Node *link;
8     char alphabet;
9 };
10 typedef struct Node Node;
11
12 int main()
13 {
14     // inialisasi struct yang akan dikaitkan
15     Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
16
17     l1.link = NULL;
18     l1.alphabet = 'F';
19     l2.link = NULL;
20     l2.alphabet = 'M';
21     l3.link = NULL;
22     l3.alphabet = 'A';
23     l4.link = NULL;
24     l4.alphabet = 'I';
25     l5.link = NULL;
26     l5.alphabet = 'K';
27     l6.link = NULL;
28     l6.alphabet = 'T';
29     l7.link = NULL;
30     l7.alphabet = 'N';
31     l8.link = NULL;
32     l8.alphabet = 'O';
33     l9.link = NULL;
34     l9.alphabet = 'R';
35
36     // mengurutkan posisi struct di dalam array
37     Node *arrayNode[9] = {&l7, &l1, &l8, &l2, &l5, &l3, &l6, &l4, &l9};
38
39     // mengurutkan posisi struct di dalam array
40     for(int i = 0; i < 9 - 1; i++)
41     {
42         arrayNode[i]->link = arrayNode[i+1];
43     }
44     arrayNode[8]->link = arrayNode[0];
45
46     // inialisasi variabel pondasi yang mencakup semua struct
47     Node *current = arrayNode[0];
48     // variabel yang menyimpan rangkaian kata yang dibuat user
49     char kalimat[MAX_LENGTH];
50     strcpy(kalimat, "\0");
51
52     int state = 1;
53     int count = 0, choice, posChar = -1;
54
55     // main program
56     printf("Rangkai sebuah kata!\n");
57     while(state)
58     {
59         printf("Rangkaian kata (%d/%d): %s\n", posChar+1, MAX_LENGTH, kalimat);
60         printf("Rute: ");
61         for(int i = 0; i <= count; i++)
62         {
63             printf("Link%d->", i+1);
64         }
65         printf("%c\n", current->alphabet);
66         printf("1. Insert character\n2. Next node\n3. Exit program\n");
67         scanf("%d", &choice);
68         switch (choice)
69         {
70             case 1:
71                 posChar++;
72                 if(posChar < MAX_LENGTH)
73                 {
74                     kalimat[posChar] = current->alphabet;
75                 }
76                 else
77                 {
78                     printf("ERROR: Jumlah karakter mencapai batas\n");
79                     state = 0;
80                 }
81                 break;
82             case 2:
83                 count++;
84                 current = current->link;
85                 if(current->alphabet == 'N')
86                 {
87                     count = 0;
88                 }
89                 break;
90             case 3:
91                 state = 0;
92                 break;
93             default:
94                 break;
95         }
96     }
97     printf("Mhasil akhir rangkaian kata: %s", kalimat);
98     return 0;
99 }
```



```
12 int main()
13 {
14     // mengurutkan posisi struct di dalam array
15     Node *arrayNode[9] = {&l7, &l1, &l8, &l2, &l5, &l3, &l6, &l4, &l9};
16
17     // mengaitkan semua struct
18     for(int i = 0; i < 9 - 1; i++)
19     {
20         arrayNode[i]->link = arrayNode[i+1];
21     }
22     arrayNode[8]->link = arrayNode[0];
23
24     // inialisasi variabel pondasi yang mencakup semua struct
25     Node *current = arrayNode[0];
26     // variabel yang menyimpan rangkaian kata yang dibuat user
27     char kalimat[MAX_LENGTH];
28     strcpy(kalimat, "\0");
29
30     int state = 1;
31     int count = 0, choice, posChar = -1;
32
33     // main program
34     printf("Rangkai sebuah kata!\n");
35     while(state)
36     {
37         printf("Rangkaian kata (%d/%d): %s\n", posChar+1, MAX_LENGTH, kalimat);
38         printf("Rute: ");
39         for(int i = 0; i <= count; i++)
40         {
41             printf("Link%d->", i+1);
42         }
43         printf("%c\n", current->alphabet);
44         printf("1. Insert character\n2. Next node\n3. Exit program\n");
45         scanf("%d", &choice);
46         switch (choice)
47         {
48             case 1:
49                 posChar++;
50                 if(posChar < MAX_LENGTH)
51                 {
52                     kalimat[posChar] = current->alphabet;
53                 }
54                 else
55                 {
56                     printf("ERROR: Jumlah karakter mencapai batas\n");
57                     state = 0;
58                 }
59                 break;
60             case 2:
61                 count++;
62                 current = current->link;
63                 if(current->alphabet == 'N')
64                 {
65                     count = 0;
66                 }
67                 break;
68             case 3:
69                 state = 0;
70                 break;
71             default:
72                 break;
73         }
74     }
75     printf("Mhasil akhir rangkaian kata: %s", kalimat);
76     return 0;
77 }
```

PENJELASAN

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

```
#include <string.h>
```

```
#define MAX_LENGTH 20
```

Baris pertama dan kedua merupakan direktif preprocessor yang mengimpor file header `stdio.h` untuk fungsi-fungsi standar input-output dan `string.h` untuk fungsi-fungsi pemrosesan string.

`#define MAX_LENGTH 20` mendefinisikan konstanta `MAX_LENGTH` dengan nilai 20, yang digunakan sebagai batas maksimum panjang kata yang dapat dibuat oleh pengguna.

```
struct Node
```

```
{
```

```
    struct Node *link;
```

```
    char alphabet;
```

```
};
```

```
typedef struct Node Node;
```

Mendefinisikan struktur `Node` yang memiliki dua anggota: pointer `link` untuk menghubungkan node dengan node lainnya, dan karakter `alphabet` untuk menyimpan huruf pada node tersebut.

`typedef struct Node Node;` membuat alias `Node` untuk `struct Node`, sehingga kita bisa menggunakan `Node` langsung tanpa menulis `struct` setiap kali.

```
    int main()
```

```
{
```

```
    // inisialisasi struct yang akan dikaitkan
```

```
    Node l1, l2, l3, l4, l5, l6, l7, l8, l9;
```

`// Inisialisasi setiap node dengan pointer link NULL dan karakter alphabet sesuai dengan urutan batu-batu yang telah ditentukan.`

Fungsi main dimulai di sini.

Mendeklarasikan dan menginisialisasi 9 node (`l1` hingga `l9`) yang akan digunakan untuk merepresentasikan batu-batu.

Setiap node diinisialisasi dengan link `NULL` dan karakter `alphabet` sesuai dengan urutan batu-batu yang telah ditentukan.

```
Node *arrayNode[9] = {&l7, &l1, &l8, &l2, &l5, &l3, &l6, &l9, &l4};
```

Membuat sebuah array arrayNode yang berisi pointer ke setiap node, sesuai dengan urutan batu-batu yang telah ditentukan.

```
for(int i = 0; i < 9 - 1; i++)  
{  
    arrayNode[i]->link = arrayNode[i+1];  
}
```

```
arrayNode[8]->link = arrayNode[0];
```

Mengaitkan setiap node dalam array dengan node berikutnya, kecuali node terakhir yang dihubungkan kembali ke node pertama untuk membentuk lingkaran. Node *current = arrayNode[0];

```
char kalimat[MAX_LENGTH];
```

```
strcpy(kalimat, "\0");
```

```
int state = 1;
```

```
int count = 0, choice, posChar = -1;
```

Mendeklarasikan variabel current yang merupakan pointer ke node saat ini yang digunakan dalam proses pembuatan kata.

Mendeklarasikan array kalimat untuk menyimpan kata yang sedang dibuat oleh pengguna.

Mendeklarasikan variabel state, count, choice, dan posChar yang digunakan dalam logika program.

```
printf("Rangkai sebuah kata!\n");  
while(state)  
{  
    // Menampilkan rangkaian kata yang sedang dibuat, urutan node yang telah dipilih, dan pilihan menu  
    kepada pengguna.  
  
    // Menerima input dari pengguna dan menjalankan perintah sesuai pilihan.  
  
    // ...  
}
```

Menampilkan pesan untuk meminta pengguna untuk merangkai sebuah kata.

Memulai loop while yang akan terus berjalan selama state bernilai 1 (true), yang berarti program masih berjalan.

```

switch (choice)
{
    case 1:
        // Memasukkan karakter dari node saat ini ke dalam array kalimat sesuai dengan posisi karakter yang
        // dimasukkan.
        // ...
        break;
    case 2:
        // Pindah ke node berikutnya dalam urutan dan menambahkan karakter dari node tersebut ke dalam
        // array kalimat.
        // ...
        break;
    case 3:
        // Menghentikan program dengan mengubah nilai state menjadi 0 (false).
        // ...
        break;
    default:
        break;
}

```

Switch-case untuk memproses pilihan menu yang dimasukkan oleh pengguna.

Pilihan 1: Memasukkan karakter dari node saat ini ke dalam array kalimat.

Pilihan 2: Pindah ke node berikutnya dalam urutan dan menambahkan karakter dari node tersebut ke dalam array kalimat.

Pilihan 3: Menghentikan program dengan mengubah nilai state menjadi 0. `printf("\nHasil akhir rangkaian kata: %s", kalimat);`

`return 0;`

Menampilkan hasil akhir dari rangkaian kata yang telah dibuat oleh pengguna.

Mengembalikan nilai 0 sebagai tanda bahwa program telah berakhir dengan sukses.