MINGGU 5
Stack & Queue

DESKRIPSI TEMA

Pada pertemuan kali ini, mahasiswa akan belajar mengenai *Stack* dan *Queue*. *Stack* adalah sebuah struktur data yang meletakan elemen baru di atas elemen sebelumnya. Dengan demikian, elemen terakhir yang dimasukkan ke dalam *Stack* merupakan elemen pertama yang akan diambil. Struktur data ini menggunakan konsep LIFO (*Last In First Out*).

Queue adalah sebuah adalah sebuah struktur data yang menerapkan konsep FIFO (*First In First* Out*).* Dengan demikian, elemen yang pertama kali masuk ke dalam *Queue* merupakan elemen yang pertama kali akan keluar dari dalam *Queue*.

CAPAIAN PEMBELAJARAN MINGGUAN (SUB-CAPAIAN PEMBELAJARAN)

1. Mahasiswa mampu menerapkan *Stack* menggunakan bahasa pemrograman C.
2. Mahasiswa mampu menerapkan *Queue* menggunakan bahasa pemrograman C.

PENUNJANG PRAKTIKUM

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

LANGKAH-LANGKAH PRAKTIKUM

A. Stack.

Secara umum, operasi dalam struktur data *Stuck* terdiri dari 4 operasi, yaitu:

a. push : Memasukkan data ke dalam *Stack*. Tidak me-*return* apapun.
b. pop : Mengeluarkan data dari dalam *Stack*. Tidak me-*return* apapun.
c. isEmpty : Mengecek apakah *Stack* memiliki isi atau tidak. Melakukan *return TRUE/FALSE*.
d. top : Mengakses elemen paling atas dari sebuah *Stack*. Melakukan *return data*.

- Tutorial 1.1 – Stack
  1. Buatlah sebuah *file* dengan nama **W05_NIM_Stack.c**
  2. Salinlah *code* berikut ke dalam *file* tersebut. Komentar dalam *code* tidak wajib disalin, namun disarankan untuk dibaca selagi menyalin.

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

// #1 Siapin tipe data Struct yang akan digunakan
// Sebagai dasar stack
typedef struct Mahasiswa{
  char nim[13];
  char nama[50];
  char jurusan[30];

  // Jangan lupa menyiapkan sebuah pointer untuk mengakses
  // data dalam Stack
  struct Mahasiswa *next;
} Mahasiswa;
```

```c
// #2 Membuat keempat function untuk operasi dasar Stack
// Urutan pembuatan adalah: isEmpty, push, pop, top
// #2.1 Membuat function isEmpty
int isEmpty(Mahasiswa *stack){
  // Cek apakah Stack memiliki isi.
  if (stack == NULL)
    return 1; // 1 Jika Stack kosong
  return 0;   // 0 Jika Stack berisi
}

// #2.2 Membuat function push. Digunakan untuk memasukkan data ke dalam Stack
void push(Mahasiswa **stack, char nim[], char nama[], char jurusan[]){
  Mahasiswa *data = (Mahasiswa*) malloc(sizeof(Mahasiswa));
  strcpy(data->nim, nim);
  strcpy(data->nama, nama);
  strcpy(data->jurusan, jurusan);
  data->next = NULL;

  if (!isEmpty(*stack)) data->next = *stack;
  *stack = data;
  printf("Adding %s to Stack\n", nama);
}

// #2.3 Membuat function pop. Digunakan untuk mengeluarkan data dari dalam Stack
void pop(Mahasiswa **stack){
  printf("Removing Stack's top element\n");
  if(*stack == NULL){ // Cek apakah stack sudah kosong atau belum
    // Jika sudah kosong, maka tidak ada lagi yang bisa di pop
    printf("Nothing to pop. Stack already empty\n");
    return;
  }
  // trash digunakan untuk menampung data yang akan dihapus
  Mahasiswa *trash = *stack;
  *stack = trash->next;
  free(trash);
  printf("Popping Stack success\n");
//  if (!isEmpty(*stack))
}

// #2.4 Membuat function top. Mengembalikan data teratas dalam Stack
Mahasiswa *top(Mahasiswa *stack){
  if (stack == NULL) return NULL; // Jika Stack kosong
  return stack; // Jika Stack tidak kosong
}
```

```c
int main(){
    printf("STACK TUTORIAL\n");
    printf("--------------\n\n");

    printf("Initializing Stack\n\n");
    Mahasiswa *stackMhs; // stackMhs sebagai Stack utama
    stackMhs = NULL; // Inisialisasi Stack awal dengen mengosongkan stack

    printf("Is Stack empty? %s\n", isEmpty(stackMhs) ? "Yes" : "No");
    printf("What is the name at the top? %s\n\n",
            top(stackMhs)==NULL ? "No one. Stack is empty" : top(stackMhs)->nama
    );

    push(&stackMhs, "14026", "James Christian Wira", "Informatika");
    printf("What is the name at the top? %s\n",
            top(stackMhs)==NULL ? "No one. Stack is empty" : top(stackMhs)->nama
    );
    printf("Is Stack empty? %s\n\n", isEmpty(stackMhs) ? "Yes" : "No");

    pop(&stackMhs);
    printf("Is Stack empty? %s\n", isEmpty(stackMhs) ? "Yes" : "No");
    printf("What is the name at the top? %s\n\n",
            top(stackMhs)==NULL ? "No one. Stack is empty" : top(stackMhs)->nama
    );

    push(&stackMhs, "17492", "Matthew Evans", "Informatika");
    push(&stackMhs, "14045", "Michael Roni", "Informatika");
    push(&stackMhs, "10000", "Mario Alexander", "Informatika");
    printf("What is the name at the top? %s\n\n",
            top(stackMhs)==NULL ? "No one. Stack is empty" : top(stackMhs)->nama
    );

    pop(&stackMhs);
    printf("What is the name at the top? %s\n\n",
            top(stackMhs)==NULL ? "No one. Stack is empty" : top(stackMhs)->nama
    );

    pop(&stackMhs);
    printf("What is the name at the top? %s\n\n",
            top(stackMhs)==NULL ? "No one. Stack is empty" : top(stackMhs)->nama
    );

    pop(&stackMhs);
    printf("What is the name at the top? %s\n\n",
            top(stackMhs)==NULL ? "No one. Stack is empty" : top(stackMhs)->nama
    );

    pop(&stackMhs);
    printf("\n");
    pop(&stackMhs);

    return 0;
}
```

B. Queue

Operasi dasar dalam struktur data ini secara umum dibagi menjadi:

a. enqueue : Memasukkan data ke dalam *queue*. Tidak me-*return* apapun.
b. dequeue : Mengeluarkan data dari dalam *queue*. Tidak me-*return* apapun.
c. isEmpty : Mengecek apakah *queue* memiliki isi atau tidak. Me-*return* *TRUE/FALSE*.
d. front : Mengakses elemen terdepan dalam *queue*. Me-*return* data terdepan.

- Tutorial 2.1 – Queue
  1. Buatlah sebuah file dengan nama **W05_NIM_Queue.c**
  2. Salinlah *code* berikut ke dalam *file* tersebut. Komentar dalam *code* tidak wajib disalin, namun disarankan untuk dibaca selagi menyalin.

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

// #1 Siapin tipe data Struct yang akan digunakan
// Sebagai dasar queue
typedef struct Mahasiswa{
  char nim[13];
  char nama[50];
  char jurusan[30];

  // Jangan lupa menyiapkan sebuah pointer untuk mengakses
  // data dalam queue
  struct Mahasiswa *next;
} Mahasiswa;
```

```c
// #2 Membuat keempat function untuk operasi dasar queue
// Urutan pembuatan adalah: isEmpty, enqueue, dequeue, front
// #2.1 Membuat function isEmpty
int isEmpty(Mahasiswa *queue){
  // Cek apakah queue memiliki isi.
  if (queue == NULL) return 1; // 1 Jika queue kosong
  return 0;   // 0 Jika queue berisi
}

// #2.2 Membuat function enqueue. Digunakan untuk memasukkan data ke dalam queue
void enqueue(Mahasiswa **head, Mahasiswa **tail, char nim[], char nama[], char jurusan[]){
  Mahasiswa *data = (Mahasiswa*) malloc(sizeof(Mahasiswa));
  strcpy(data->nim, nim);
  strcpy(data->nama, nama);
  strcpy(data->jurusan, jurusan);
  data->next = NULL;

  if (isEmpty(*head)) *head = data;
  else (*tail)->next = data;
  *tail = data;
  printf("Adding %s to queue\n", nama);
}
```

```c
// #2.3 Membuat function dequeue. Digunakan untuk mengeluarkan data dari dalam queue
void dequeue(Mahasiswa **head){
    printf("Removing queue's front element\n");
    if(isEmpty(*head)){ // Cek apakah queue sudah kosong atau belum
        // Jika sudah kosong, maka tidak ada lagi yang bisa di dequeue
        printf("Nothing to dequeue. Queue already empty\n");
        return;
    }
    // trash digunakan untuk menampung data yang akan dihapus
    Mahasiswa *trash = *head;
    *head = trash->next;
    free(trash);
    printf("Dequeuing success\n");
}

// #2.4 Membuat function front. Mengembalikan data terdepan dalam queue
Mahasiswa *front(Mahasiswa *queue){
    if (queue == NULL) return NULL; // Jika queue kosong
    return queue; // Jika queue tidak kosong
}
```

```c
int main(){
    printf("QUEUE TUTORIAL\n");
    printf("--------------\n\n");

    printf("Initializing Queue\n\n");
    Mahasiswa *headQueue, *tailQueue; // headQueue = awal Queue; tailQueue = akhir Queue
    headQueue = tailQueue = NULL; // Inisialisasi Queue awal dengen mengosongkan Queue

    printf("Is Queue empty? %s\n", isEmpty(headQueue) ? "Yes" : "No");
    printf("What is the name at the front? %s\n\n",
            front(headQueue)==NULL ? "No one. Queue is empty" : front(headQueue)->nama
    );

    enqueue(&headQueue, &tailQueue, "14026", "James Christian Wira", "Informatika");
    printf("What is the name at the front? %s\n",
            front(headQueue)==NULL ? "No one. Queue is empty" : front(headQueue)->nama
    );
    printf("Is Queue empty? %s\n\n", isEmpty(headQueue) ? "Yes" : "No");

    dequeue(&headQueue);
    printf("Is Queue empty? %s\n", isEmpty(headQueue) ? "Yes" : "No");
    printf("What is the name at the front? %s\n\n",
            front(headQueue)==NULL ? "No one. Queue is empty" : front(headQueue)->nama
    );
```

```
87    enqueue(&headQueue, &tailQueue, "17492", "Matthew Evans", "Informatika");
88    enqueue(&headQueue, &tailQueue, "14045", "Michael Roni", "Informatika");
89    enqueue(&headQueue, &tailQueue, "10000", "Mario Alexander", "Informatika");
90    printf("What is the name at the front? %s\n\n",
91        front(headQueue)==NULL ? "No one. Queue is empty" : front(headQueue)->nama
92    );
93
94    dequeue(&headQueue);
95    printf("What is the name at the front? %s\n\n",
96        front(headQueue)==NULL ? "No one. Queue is empty" : front(headQueue)->nama
97    );
98
99    dequeue(&headQueue);
100   printf("What is the name at the front? %s\n\n",
101       front(headQueue)==NULL ? "No one. Queue is empty" : front(headQueue)->nama
102   );
103
104   dequeue(&headQueue);
105   printf("\n");
106   dequeue(&headQueue);
107   return 0;
108 }
```

C. Tugas

1. Buatlah sebuah program *Parenthesis Checking* yang digunakan untuk memeriksa apakah *input* yang dimasukkan oleh *user* merupakan *parenthesis expression* yang valid.
   Ketentuan program adalah sebagai berikut:
   a. Jenis kurung yang digunakan adalah: [ ], ( ), { }.
   b. Program akan berhenti jika diberi *input* -1.
   c. Simpan program dengan nama **W05_[NIM] _ChallengeStack.c**

```
Masukan pola: {}()[]
Valid parenthesis expression

Masukan pola: { ( ) [ ( ) ] }
Valid parenthesis expression

Masukan pola: ( [ } )
Invalid parenthesis expression

Masukan pola: ( ( ) ) ) )
Invalid parenthesis expression

Masukan pola: a
Invalid parenthesis expression

Masukan pola: 2
Invalid parenthesis expression

Masukan pola: -
Invalid parenthesis expression

Masukan pola:
```

2. Buatlah sebuah program *queue* yang menerima *input* angka secara terus menerus dengan ketentuan:
   a. Jika *queue* kosong, *input* angka dan *enqueue* angka tersebut sebanyak angka tersebut.
   b. Jika *queue* tidak kosong, saat *input* angka, lakukan pengecekan, bila angka yang di *input* lebih kecil dari front, maka lakukan *dequeue* sebanyak *input*-an tersebut (jangan lupa *error handling* ketika *queue* kosong saat sedang melakukan *dequeue*), bila angka yang di *input* lebih besar atau sama dengan front, lakukan *enqueue* sebanyak front tersebut.
   c. Kumpulkan dengan nama file: **W05_[NIM] _ChallengeQueue.c**

REFERENSI