

Nama: Achmad soewardi

Kelas: RPL 2A

NIM: 2301508

Alur Algoritma dari soal No. 1, 2, dan 4

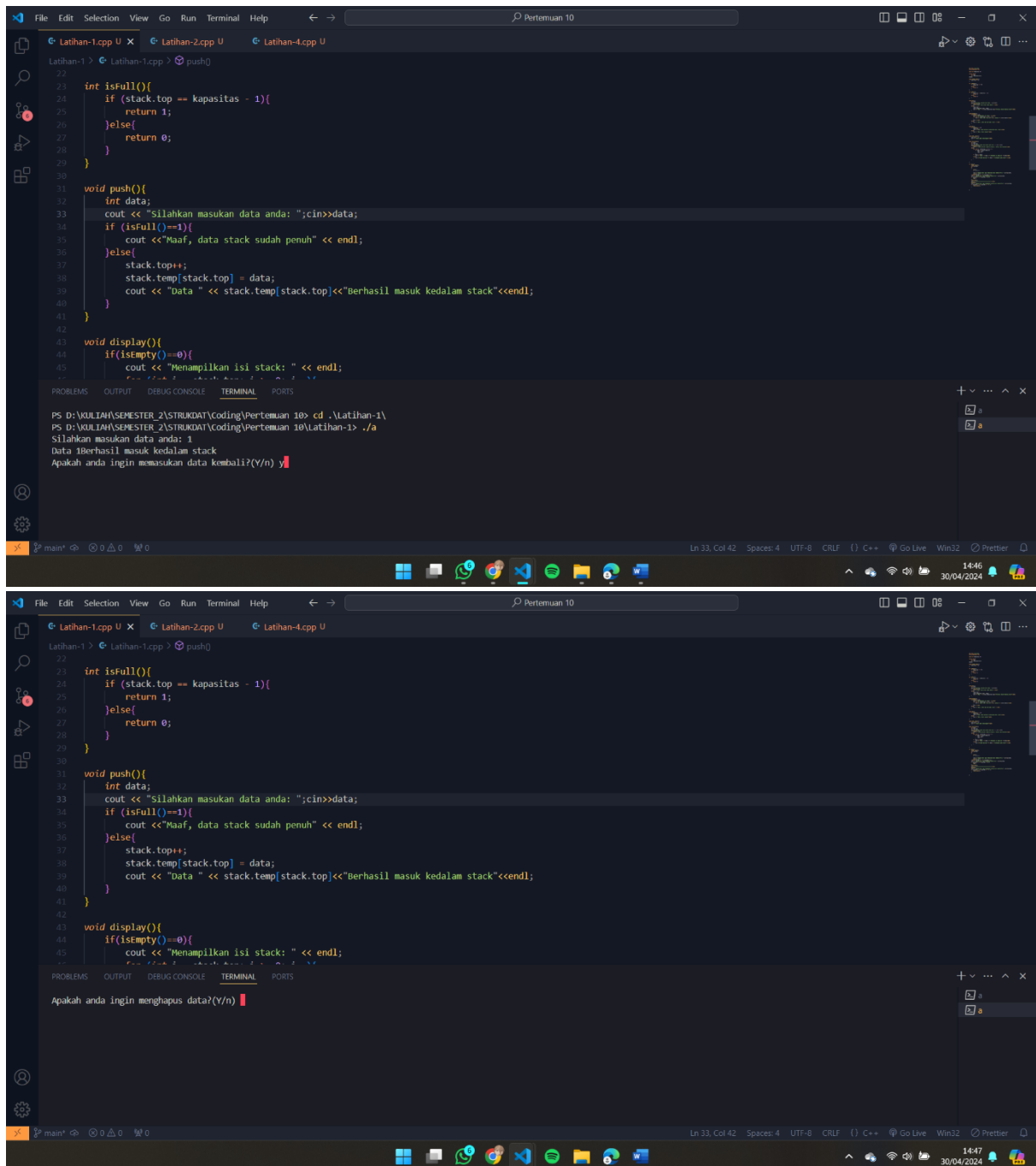
1. Algoritma dari soal No. 1
 - a) Pertama, dilakukan penggunaan header file `<iostream>` yang berisi deklarasi input-output stream dan menggunakan namespace `std`.
 - b) Kemudian, didefinisikan konstanta kapasitas yang memiliki nilai 5. Konstanta ini menentukan kapasitas maksimum dari stack yang akan dibuat.
 - c) Selanjutnya, dideklarasikan sebuah struct stack yang memiliki dua anggota, yaitu `top` yang merupakan indeks dari elemen teratas stack, dan `temp` yang merupakan array untuk menyimpan elemen-elemen stack.
 - d) Fungsi `create_stack()` digunakan untuk menginisialisasi stack dengan mengatur nilai `top` menjadi -1, menandakan bahwa stack kosong.
 - e) Fungsi `isEmpty()` digunakan untuk memeriksa apakah stack kosong atau tidak. Jika `top` sama dengan -1, maka stack dikatakan kosong dan fungsi ini mengembalikan nilai 1, jika tidak, mengembalikan nilai 0.
 - f) Fungsi `isFull()` digunakan untuk memeriksa apakah stack penuh atau tidak. Jika `top` sama dengan kapasitas - 1, maka stack dikatakan penuh dan fungsi ini mengembalikan nilai 1, jika tidak, mengembalikan nilai 0.
 - g) Fungsi `push()` digunakan untuk menambahkan elemen ke dalam stack. Pengguna diminta untuk memasukkan data, kemudian fungsi ini memeriksa apakah stack penuh atau tidak. Jika penuh, akan muncul pesan kesalahan. Jika tidak, elemen akan dimasukkan ke dalam stack dan nilai `top` akan dinaikkan.
 - h) Fungsi `display()` digunakan untuk menampilkan seluruh elemen dalam stack. Fungsi ini memeriksa apakah stack kosong atau tidak. Jika tidak kosong, elemen-elemen dalam stack akan ditampilkan dari atas ke bawah.
 - i) Fungsi `pop()` digunakan untuk menghapus elemen teratas dari stack. Fungsi ini memeriksa apakah stack kosong atau tidak. Jika tidak kosong, elemen teratas akan dihapus dengan menurunkan nilai `top`.
 - j) Fungsi `clear_stack()` digunakan untuk mengosongkan seluruh isi stack dengan mengatur nilai `top` menjadi -1.
 - k) Fungsi `find_stack()` digunakan untuk mencari suatu data dalam stack. Pengguna diminta untuk memasukkan data yang ingin dicari. Fungsi ini akan melakukan pencarian dari atas ke bawah dalam stack. Jika data ditemukan, akan ditampilkan indeksnya, jika tidak, akan muncul pesan bahwa data tidak ditemukan.

Dalam fungsi `main()`, program pertama-tama memanggil fungsi `create_stack()` untuk menginisialisasi stack. Kemudian, program akan melakukan penambahan data ke dalam stack dengan menggunakan perulangan `do-while`. Setelah itu, pengguna ditanyai apakah ingin

menghapus data dari stack. Jika iya, fungsi pop() akan dipanggil. Selanjutnya, program akan mencari suatu data dalam stack dengan memanggil fungsi find_stack(), menampilkan isi stack dengan memanggil fungsi display(), dan mengosongkan seluruh isi stack jika pengguna menginginkannya dengan memanggil fungsi clear_stack().

Algoritma ini mengimplementasikan operasi dasar dari struktur data stack, yaitu push (menambahkan elemen), pop (menghapus elemen), dan beberapa operasi lain seperti menampilkan isi stack dan mencari elemen dalam stack.

Hasil Output:



```

22
23 int isFull(){
24     if (stack.top == kapasitas - 1){
25         return 1;
26     }else{
27         return 0;
28     }
29 }
30
31 void push(){
32     int data;
33     cout << "Silahkan masukan data anda: ";cin>>data;
34     if (isFull()==1){
35         cout <<"Maaf, data stack sudah penuh" << endl;
36     }else{
37         stack.top++;
38         stack.temp[stack.top] = data;
39         cout << "Data " << stack.temp[stack.top]<<"Berhasil masuk kedalam stack"<<endl;
40     }
41 }
42
43 void display(){
44     if(isEmpty()==0){
45         cout << "Menampilkan isi stack: " << endl;
46     }
47 }
48
49 int main(){
50     int pilihan;
51     while (1){
52         cout << "Pilih operasi yang akan dilakukan: ";
53         for (int i = 1; i <= 5; i++){
54             cout << i << " ";
55             if (i % 5 == 0) cout << endl;
56         }
57         if (pilihan < 1 || pilihan > 5) continue;
58         switch (pilihan){
59             case 1: push(); break;
60             case 2: pop(); break;
61             case 3: find_stack(); break;
62             case 4: display(); break;
63             case 5: clear_stack(); break;
64         }
65         cout << "Apakah anda ingin melakukan operasi lain? (y/n) ";
66         if (getchar() != 'y') break;
67     }
68     return 0;
69 }
```

PS D:\KULIAH\SEMESTER_2\STRUKTUR\coding\Pertemuan 10> cd .\Latihan-1
PS D:\KULIAH\SEMESTER_2\STRUKTUR\coding\Pertemuan 10\Latihan-1> ./a
Silahkan masukan data anda: 1
Data 1Berhasil masuk kedalam stack
Apakah anda ingin memasukan data kembali?(y/n) y

Apakah anda ingin menghapus data?(y/n)


```
22 int isFull(){
23     if (stack.top == kapasitas - 1){
24         return 1;
25     }else{
26         return 0;
27     }
28 }
29
30 void push(){
31     int data;
32     cout << "Silahkan masukan data anda: ";cin>>data;
33     if (isFull()==1){
34         cout <<"Maaf, data stack sudah penuh" << endl;
35     }else{
36         stack.top++;
37         stack.temp[stack.top] = data;
38         cout << "Data " << stack.temp[stack.top]<<"Berhasil masuk kedalam stack"<<endl;
39     }
40 }
41
42 void display(){
43     if(isEmpty()==0){
44         cout << "Menampilkan isi stack: " << endl;
45         for (int i = stack.top; i >= 0; i--){
46             cout << stack.temp[i] << " ";
47         }
48     }
49 }
```

Menampilkan isi stack:
Data index stack ke-3 Adalah 4
Data index stack ke-2 Adalah 3
Data index stack ke-1 Adalah 2
Data index stack ke-0 Adalah 1

apakah anda ingin menghapus seluruh isi stack?(Y/n)
stack sudah dikosongkan
Maaf, tidak ada data pada stack

2. Algoritma soal No. 2

Deklarasi Struktur Stack: Program mulai dengan mendeklarasikan sebuah struktur stack yang memiliki dua anggota: `top` untuk menunjukkan indeks puncak stack, dan `temp` sebagai array untuk menyimpan data.

a) Fungsi-fungsi Dasar Stack:

- `create_stack()`: Ini adalah fungsi untuk menginisialisasi stack dengan mengatur `top` ke nilai -1, menunjukkan bahwa stack saat ini kosong.
- `isEmpty()`: Fungsi ini memeriksa apakah stack kosong dengan memeriksa apakah nilai `top` sama dengan -1.
- `isFull()`: Fungsi ini memeriksa apakah stack penuh dengan memeriksa apakah nilai `top` sama dengan kapasitas stack dikurangi 1.

b) Fungsi `push()`: Fungsi ini menerima karakter sebagai argumen dan menambahkannya ke stack. Sebelum menambahkan, fungsi ini memeriksa apakah stack sudah penuh menggunakan `isFull()`.


c) Fungsi `pop()`: Fungsi ini menghapus dan mengembalikan karakter teratas dari stack. Sebelum melakukan pop, fungsi ini memeriksa apakah stack kosong menggunakan `isEmpty()`.

d) Fungsi `reverseSentence()`: Fungsi ini menerima sebuah string (kalimat) sebagai argumen. Pertama, panjang kalimat dihitung menggunakan ``strlen()``. Kemudian, setiap karakter dalam kalimat dimasukkan ke stack menggunakan fungsi ``push()``. Setelah itu, karakter-karakter tersebut dipop dari stack satu per satu, sehingga menghasilkan kalimat yang terbalik. Karakter-karakter yang dipop dicetak satu per satu untuk menampilkan kalimat yang sudah dibalik.

e) Fungsi `main()`:

- Pertama, fungsi ``create_stack()`` dipanggil untuk menginisialisasi stack.
- Pengguna diminta untuk memasukkan sebuah kalimat (maksimal 20 karakter) menggunakan `cin.getline()`.
- Kalimat yang dimasukkan oleh pengguna kemudian diberikan ke fungsi `reverseSentence()` untuk dibalikkan.
- Dengan demikian, program menggunakan struktur stack untuk membalikkan kalimat yang dimasukkan oleh pengguna dengan mengambil setiap karakter kalimat dan menempatkannya ke dalam stack terlebih dahulu, dan kemudian mengambil karakter-karakter tersebut dari stack untuk membalikkan urutannya.

Hasil Output:

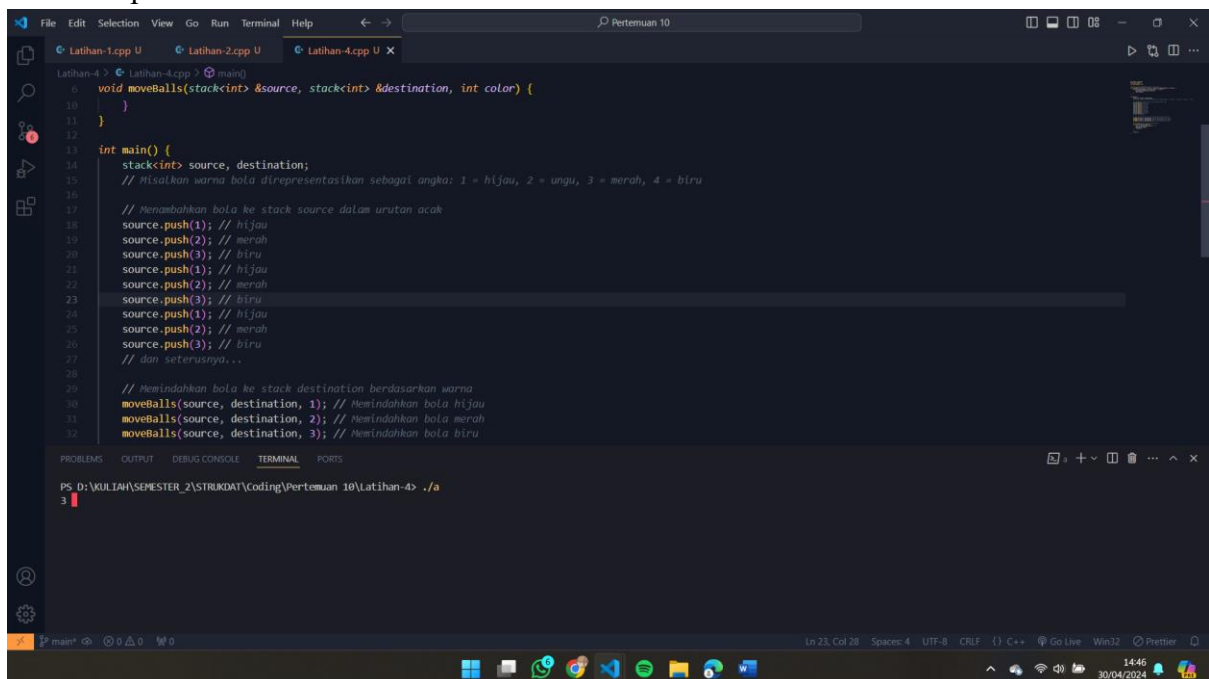


```
PS D:\KULIAH\SEMESTER_2\STRUKDAT\coding\Pertemuan 10\Latihan-1> cd ..
PS D:\KULIAH\SEMESTER_2\STRUKDAT\coding\Pertemuan 10> cd ..\Latihan-2\
PS D:\KULIAH\SEMESTER_2\STRUKDAT\coding\Pertemuan 10\Latihan-2> ./a
Masukan kalimat (Maksimal 20 karakter): Achmad Soewardi
Kalimat Terbalik: idraweoS danhcA
PS D:\KULIAH\SEMESTER_2\STRUKDAT\coding\Pertemuan 10\Latihan-2>
```

3. Algoritma No. 4

- a) Inisialisasi dua stack: source dan destination. Stack source akan digunakan untuk menyimpan bola-bola berwarna dalam urutan acak, sedangkan stack destination akan digunakan untuk menyimpan bola-bola yang telah diurutkan.
- b) Menambahkan bola ke stack source: Bola-bola berwarna ditambahkan ke dalam stack source dalam urutan acak. Misalnya, warna bola direpresentasikan sebagai angka: 1 = hijau, 2 = ungu, 3 = merah, 4 = biru.
- c) Memindahkan bola ke stack destination berdasarkan warna:
Fungsi moveBalls dipanggil untuk setiap warna bola. Fungsi ini memindahkan bola dari stack source ke stack destination berdasarkan warna. Bola-bola dengan warna yang sama akan dipindahkan ke stack destination dalam urutan yang sama seperti mereka muncul di stack source.
- d) Menampilkan bola di stack destination: Setelah semua bola telah dipindahkan ke stack destination, bola-bola di stack destination ditampilkan. Bola-bola akan ditampilkan dalam urutan terbalik dari urutan mereka ditambahkan ke stack (karena sifat LIFO dari stack).

Hasil Output:



```
void moveBalls(stack<int> &source, stack<int> &destination, int color) {
    // ...
}

int main() {
    stack<int> source, destination;
    // Misalkan warna bola direpresentasikan sebagai angka: 1 = hijau, 2 = ungu, 3 = merah, 4 = biru

    // Menambahkan bola ke stack source dalam urutan acak
    source.push(1); // hijau
    source.push(2); // merah
    source.push(3); // biru
    source.push(1); // hijau
    source.push(2); // merah
    source.push(3); // biru
    source.push(1); // hijau
    source.push(2); // merah
    source.push(3); // biru
    // dan seterusnya...

    // Memindahkan bola ke stack destination berdasarkan warna
    moveBalls(source, destination, 1); // Memindahkan bola hijau
    moveBalls(source, destination, 2); // Memindahkan bola merah
    moveBalls(source, destination, 3); // Memindahkan bola biru

    // ...
}
```

PS D:\KULIAH\SEMESTER_2\STRUKTUR\coding\pertemuan 10\Latihan-4> .\a
3