1. 8.3 Stack dengan Linked List

Pada bagian kita akan menggunakan linked list dengan metode stack

Berikut adalah stuktur node awal dari stack nya yang sudah terdapat function untuk menaruh value pada stack dengan function node()

struct node{

int data;

node \*next;

node(int v, node \*x): data(v), next(x) {}

Lalu kode akan membuat head menjadi null dan membuat int dengan nama ukr yang bernilai 0 unuk menghitung isi stack

node \*top = NULL; int ukr = 0;

Lalu berikutnya adalah fungsi push yang akan menambahkan node ke dalam stack sebagai top dan akan menambahkan value ukr karena stack bertambah

void push(int v){

top = new node(v, top);

ukr++;

Lalu berikutnya adalah fugsi pop yang akan mengembalikan value dari top dan memindahkan top kepada node sebelumnya lalu di hapus dan ukr akan di kurangi valuenya

int pop(){

if(top == NULL){

throw runtime\_error("Stack kosong");

} else{

int r = top->data;

node \*temp = top;

top = top->next;

delete temp;

ukr--;

return r;

Berikutnya adalah fungsi print yang akan menampilkan semua isi stack dengan cara membuat node temp di top lalu menampilkan datanya sampai temp menjadi null

void print(){

node \*temp = top; cout <<"isi stack:";

while(temp != NULL){

cout << "<-" << temp->data;

temp = temp->next;

Selanjutnya adalah fungsicheck size yang akan mengembalikan nilau ukr yang adalah jumlah dari stack sekarang

int checkSize(){

return ukr;

Selanjutnya adalah fungsi boolean kosong yang akan menjadi true jika ukr adalah 0

bool kosong(){

return ukr == 0;

Selanjutnya adalah fungsi cek Head yang akan mengecek top dari stack dan mengembalikan value dari top dan jika top nya null kode akan berakhir error dengan kalimat stack kosong

int cekHead(){

if(top == NULL){

throw runtime\_error("Stack kosong");

} return top->data;

Terakhir adalah fungsi add tail dimana kode ini akan menambahkan data pada bagian bawah stack, kode ini bekerja dengan mengecek apakah stack kosong jika iya maka node barunya akan di push, tetapi jika tidak maka kode akan membuat int temp dengan value pop lalu memanggil dirinya lagi secara rekursif sampai stack menjadi kosong dan ketika stack kosong maka nilai baru akan di push dan semua fungsi yang sebelumnya berjalan tadi akan mengembalikan temp dengan cara di push

void pushtail(int v){

if(kosong()){

push(v);

}else{

int temp = pop();

pushtail(v);

push(temp);

Sisanya adalah menu dari cara menggunakan semua fungsi tadi

int main(){

int pilihan, x;

do{print(); if(kosong()){} else {int s = checkSize(); cout << "\tJumlah data: " << s << endl;}

cout << "\nMenu Operasi Stack:\n";

cout << "1. Push\n";

cout << "2. Pop\n";

cout << "3. Peek\n";

cout << "4. PushAtBottom\n";

cout << "5. Exit\n";

cin >> pilihan;

cin.ignore();

switch (pilihan){

case 1: cout << "Masukkan data yang ingin dimasukan: ";

cin >> x;

push(x);

break;

case 2: cout << "Data yang di pop: " << pop() << endl; break;

case 3: cout << "Data yang di peek: " << cekHead() << endl; break;

case 4: cout << "Masukkan data yang ingin dimasukan: ";

cin >> x;

pushtail(x);

break;

default:

if(pilihan>5){

cout<<"Masukan pilihan yang benar\n";

}break;

}

} while (pilihan != 5); return 0;

1. 8.4 Mengecek keseimbangan kurung

Pada bagian ini kita akan di berikan kode tentang mengecek setiap huruf dalam string dengan membuat stack yang isinya mengambil char dari string tersebut lalu jika bertemu dengan kurung terbuka kode akan berjalan dengan melakukan push dan jika bertemu kurung tutup kode akan mengecek apakah head (char sebelumnya) bukan kurung bagian terbuka atau kosong jika iya maka value nya akan menjadi false, jika bukan maka head sebelumnya akan di pop yang membuat headnya mundur dan mengecek char selanjutnya

#include <iostream>

#include <stack>

#include <string>

using namespace std;

bool isBalancedParenthesis(string expn) {

stack<char> stk;

for (char ch : expn) {

switch (ch) {

case '{':

case '[':

case '(':

stk.push(ch);

break;

case '}':

if (stk.empty() || stk.top() != '{') {

return false;

}

stk.pop();

break;

case ']':

if (stk.empty() || stk.top() != '[') {

return false;

}

stk.pop();

break;

case ')':

if (stk.empty() || stk.top() != '(') {

return false;

}

stk.pop();

break;

}

}

return stk.empty();

}

int main() {

char x = 'y';

while(x == 'y') {

string expn;

cout << "\tpengecek keseimbangan kurung\nMasukan tanda kurung: ";

cin >> expn;

bool value = isBalancedParenthesis(expn);

cout << "Input: " << expn << endl;

cout << "Hasil setelah pengecekan: " << (value ? "Seimbang" : "Tidak seimbang") << endl;

cout << "Apakah anda ingin mengulang program? (y/n) ";

cin >> x;

cin.ignore();

}

}

1. 8-5 Merubah infix menjadi postfix

Pada bagian ini user akan di minta untuk memasukan ekspresis matematika dan lalu ekspresi tersebut akan dirubah menjadi postfix dengan metode shunting yard dengan cara mengecek isi dari string tersebut dengan membuat stack dari setiap char string tersebut dan melakuak setiap aksi tergantung dari isi string tersebut  
#include <iostream>

#include <stack>

#include <string>

using namespace std;

int precedence(char op) {

if (op == '^')

return 3;

else if (op == '\*' || op == '/' || op == '%')

return 2;

else if (op == '+' || op == '-')

return 1;

else

return -1;

}

string infixToPostfix(string expn) {

stack<char> stk;

string output = "";

char temp;

for (char ch : expn) {

if (ch >= '0' && ch <= '9') {

output += ch;

} else {

switch (ch) {

case '+':

case '-':

case '\*':

case '/':

case '%':

case '^':

while (!stk.empty() && precedence(ch) <= precedence(stk.top())) {

temp = stk.top();

stk.pop();

output += " ";

output += temp;

}

stk.push(ch);

output += " ";

break;

case '(':

stk.push(ch);

break;

case ')':

while (!stk.empty() && (temp = stk.top()) != '(') {

stk.pop();

output += " ";

output += temp;

output += " ";

}

stk.pop();

break;

}

}

}

while (!stk.empty()) {

temp = stk.top();

stk.pop();

output += " ";

output += temp;

}

return output;

}

int main() {

char x = 'y'; string expn;

while(x == 'y') {

cout << "\tMerubah infix dengan Shunting Yard \nMasukkan ekspresi matematika: ";

cin >> expn;

string value = infixToPostfix(expn);

cout << "Infix Expn: " << expn << endl;

cout << "Postfix Expn: " << value << endl;

cout << "Apakah anda ingin mengulang program? (y/n) ";

cin >> x;

cin.ignore();

}

}

1. 8-6 Mengubah infix menjadi prefix

Pada bagian ini user akan di minta memasukan ekspresi matematika dan ekspresi tersebut akan di cek terlebih dahulu dan ketika di dalam ekspresi tersbesut terdapat kurung maka kurung tersebut akan di ganti dengan kurung kebalikannya dan di tampilkan lagi  
#include <iostream>

#include <cstring>

#include <algorithm>

using namespace std;

string InfixToPrefix(string expn);

void ReplaceParenthesis(char a[], int length)

{

for (int i = 0; i < length; i++)

{

if (a[i] == '(')

a[i] = ')';

else if (a[i] == ')')

a[i] = '(';

}

}

void ReverseString(char expn[], int length)

{

int lower = 0;

int upper = length - 1;

char tempChar;

while (lower < upper)

{

tempChar = expn[lower];

expn[lower++] = expn[upper];

expn[upper--] = tempChar;

}

}

char\* InfixToPostfix(char expn[])

{

return expn;

}

string InfixToPrefix(string expn)

{

char\* arr = new char[expn.length() + 1];

strcpy(arr, expn.c\_str());

ReverseString(arr, expn.length());

ReplaceParenthesis(arr, expn.length());

arr = InfixToPostfix(arr);

ReverseString(arr, expn.length());

string result(arr);

delete[] arr;

return result;

}

int main() {

char x = 'y'; string expn;

while(x == 'y') {

cout <<"\tMengubah infix menajdi Prefix\nMasukkan ekspresi matematika: ";

cin >> expn ;

string value = InfixToPrefix(expn);

cout << "Infix Expn: " << expn << endl;

cout << "Prefix Expn: " << value << endl;

cout << "Apakah anda ingin mengulang program? (y/n) ";

cin >> x;

cin.ignore();

}

return 0;

}

1. 8.7 Mengubah postfix menjadi hasil

Pada bagian ini kode nya berfungsi untuk mengubah postfix menjadi hasil dari operasi matematika awalnya (in fix) jadi kode ini seperti kebalikan dari infix to postfix lalu setelah menjadi infix kode akan di jalankan operasi matematika  
#include <iostream>

#include <stack>

#include <string>

#include <sstream>

using namespace std;

int postfixEvaluate(string expn) {

stack<int> stk;

stringstream ss(expn);

string token;

while (ss >> token) {

if (token == "+" || token == "-" || token == "\*" || token == "/") {

int num2 = stk.top(); stk.pop();

int num1 = stk.top(); stk.pop();

switch (token[0]) {

case '+':

stk.push(num1 + num2);

break;

case '-':

stk.push(num1 - num2);

break;

case '\*':

stk.push(num1 \* num2);

break;

case '/':

stk.push(num1 / num2);

break;

}

} else {

stk.push(stoi(token));

}

}

return stk.top();

}

int main() {

string expn = "6 5 2 3 + 8 \* + 3 + \*";

int value = postfixEvaluate(expn);

cout << "Given Postfix Expn: " << expn << endl;

cout << "Result after Evaluation: " << value << endl;

return 0;

}

1. Stack Palindrom

Kode ini bekerja dengan cara melakukan membuat stack baru lalu melakukan push ke dalam stack tersebut hingga setengah dari string tersebut masuk

bool Palindrome(string str) {

stack<char> charStack;

int length = str.length();

for (int i = 0; i < length / 2; ++i) {

charStack.push(str[i]);

}

lalu kode akan mengecek berdasarkan apakah panjang dari string tersebut ganjil atau genap lalu akan di iterasi dari tengah string sampai terakhir dan jika top dari char tidak sama dengan data di string maka return akan menjadi false tetapi jika tidak maka fungsi tersebut akan menjadi true

int i = (length % 2 == 0) ? length / 2 : length / 2 + 1;

while (i < length) {

if (charStack.empty() || charStack.top() != str[i]) {

return false;

}

charStack.pop();

++i;

} return true;

Setelahnya hanya kode menu untuk memasukan palindrom dan mennunjukan hasil

int main() {

string v; char x = 'y'; bool y;

while (x == 'y'){

cout << "Masukan kata: ";

getline(cin, v);

y = Palindrome(v);

cout << "Kata " << v << " Adalah " << (y ? "Palindrom" : "Bukan Palindrom") << endl;

cout << "Apakah anda ingin mengulang program? (y/n) ";

cin >> x;

cin.ignore();

}

return 0;