#include "stack.h"

#include<iostream>

#include<string>

using namespace std;

int main()

{

stack<string> ss;

stack<char> sc;

stack<int> si;

stack<float> sf;

int choice;

char option;

cout << "which type of date type you want to use for entering a value?" << endl

<< "a: string" << endl

<< "b: character" << endl

<< "c: integer" << endl

<< "d: floating point" << endl

<< "enter your choice: ";

cin >> option;

string str, str1, str2;

char ch, ch1, ch2;

int integ, integ1, integ2;

float flo, flo1, flo2;

if ((option == 'a') || (option == 'b') || (option == 'c') || (option == 'd'))

{

do

{

cout << endl

<< "## MENU ##" << endl

<< "1: push" << endl

<< "2: pop" << endl

<< "3: isEmpty" << endl

<< "4: isfull" << endl

<< "5: total of stack" << endl

<< "6: exit" << endl

<< "enter your choice: ";

cin >> choice;

switch (choice)

{

case 1:

{

if (option == 'a')

{

ss.create();

cout << "enter string value: ";

getline(cin, str, '.');

ss.push(str);

}

else if (option == 'b')

{

sc.create();

cout << "enter a character: ";

cin >> ch;

sc.push(ch);

}

else if (option == 'c')

{

si.create();

cout << "enter an integer: ";

cin >> integ;

si.push(integ);

}

else if (option == 'd')

{

sf.create();

cout << "enter a floating point: ";

cin >> flo;

sf.push(flo);

}

else

{

cout << "error" << endl;

}

break;

}

case 2:

{

if (option == 'a')

{

str1 = ss.pop();

if (str1 != "0")

{

cout << "now we are on last current node" << endl;

cout << "now the string is: " << str1 << endl;

}

}

else if (option == 'b')

{

ch1 = sc.pop();

if (ch1 != '0')

{

cout << "now we are on last current node" << endl;

cout << "now the character is: " << ch1 << endl;

}

}

else if (option == 'c')

{

integ1 = si.pop();

if (integ1 != 0)

{

cout << "now we are on last current node" << endl;

cout << "now the integer is: " << integ1 << endl;

}

}

else if (option == 'd')

{

flo1 = sf.pop();

if (flo1 != 0)

{

cout << "now we are on last current node" << endl;

cout << "now the floating point is: " << flo1 << endl;

}

}

else

{

cout << "error" << endl;

}

break;

}

case 3:

{

if (option == 'a')

{

if (ss.isempty())

{

cout << "list is empty " << endl;

}

else if (!ss.isempty())

{

cout << "list is not empty" << endl;

}

}

else if (option == 'b')

{

if (sc.isempty())

{

cout << "list is empty " << endl;

}

else if (!sc.isempty())

{

cout << "list is not empty" << endl;

}

}

else if (option == 'c')

{

if (si.isempty())

{

cout << "list is empty " << endl;

}

else if (!si.isempty())

{

cout << "list is not empty" << endl;

}

}

else if (option == 'd')

{

if (sf.isempty())

{

cout << "list is empty " << endl;

}

else if (!sf.isempty())

{

cout << "list is not empty" << endl;

}

}

else

{

cout << "error" << endl;

}

break;

}

case 4:

{

if (option == 'a')

{

if (ss.isfull())

{

cout << "list is full" << endl;

}

else if (!ss.isfull())

{

cout << "list is not full" << endl;

}

}

else if (option == 'b')

{

if (sc.isfull())

{

cout << "list is full" << endl;

}

else if (!sc.isfull())

{

cout << "list is not full" << endl;

}

}

else if (option == 'c')

{

if (si.isfull())

{

cout << "list is full" << endl;

}

else if(!si.isfull())

{

cout << "list is not full" << endl;

}

}

else if (option == 'd')

{

if (sf.isfull())

{

cout << "list is full" << endl;

}

else if(!sf.isfull())

{

cout << "list is not full" << endl;

}

}

else

{

cout << "error" << endl;

}

break;

}

case 5:

{

if (option == 'a')

{

str2 = ss.tos();

if (str2 != "0")

{

cout << "the top of stack is :" << str2 << endl;

}

}

else if (option == 'b')

{

ch2 = sc.tos();

if (ch2 != '0')

{

cout << "the top of stack is :" << ch2 << endl;

}

}

else if (option == 'c')

{

integ2 = si.tos();

if (integ2 != 0)

{

cout << "the top of stack is :" << integ2 << endl;

}

}

else if (option == 'd')

{

flo2 = sf.tos();

if (flo2 != 0)

{

cout << "the top of stack is :" << flo2 << endl;

}

}

else

{

cout << "error" << endl;

}

break;

}

case 6:

{

if (option == 'a')

{

sc.exit();

}

else if (option == 'b')

{

ss.exit();

}

else if (option == 'c')

{

si.exit();

}

else if (option == 'd')

{

sf.exit();

}

else

{

cout << "error" << endl;

}

break;

}

default:cout << "enter correct choice" << endl;

}

} while (choice != 6);

}

else

{

cout << "invalid choice so you cannot continue with the program" << endl;

}

system("pause");

return 0;

}

template <class T>

void stack<T>::create()

{

if (max == 0)

{

cout << "enter the maximum values you want to enter: ";

cin >> max;

a = new T[max];

}

}

template <class T>

void stack<T>::push(T &c)

{

if (top < (max - 1))

{

a[++top] = c;

}

else

{

cout << "stack overflow in push" << endl << "so this value will not be considered" << endl;

}

}

template <class T>

T stack<T>::pop()

{

if (max != 0)

{

if (top != -1)

{

if (top <= max)

{

return a[top--];

}

else

{

cout << "list is empty so you cannot deal with more values" << endl;

return 0;

}

}

else

{

cout << "list is empty" << endl;

return 0;

}

}

else

{

cout << "list has not been created" << endl;

return 0;

}

}

template <class T>

bool stack<T>::isempty()

{

if (max != 0)

{

if (top == -1)

{

return true;

}

else

{

return false;

}

}

else

{

cout << "list has not been created" << endl;

return 0;

}

}

template <class T>

bool stack<T>::isfull()

{

if (max != 0)

{

if (top != -1)

{

if (top == (max - 1))

{

return true;

}

else

{

return false;

}

}

else

{

cout << "list is empty" << endl;

return 0;

}

}

else

{

cout << "list has not been created" << endl;

return 0;

}

}

template <class T>

T stack<T>::tos()

{

if (max != 0)

{

if (top != -1)

{

cout << "total of stack is: " << top << endl;

return a[top];

}

else

{

cout << "list is empty" << endl;

return 0;

}

}

else

{

cout << "list has not been created" << endl;

return 0;

}

}

template <class T>

void stack<T>::deallocate()

{

if (max != 0)

{

delete[]a;

cout << "array has been deallocated" << endl;

}

else

{

cout << "list has not been created" << endl;

}

}

template <class T>

void stack<T>::exit()

{

deallocate();

cout << "program ended" << endl;

}

#pragma once

#include<iostream>

#include<string>

using namespace std;

template <class T>

class stack

{

int top;

int max;

T \*a;

public:

stack()

{

top = -1;

max = 0;

a = nullptr;

}

~stack()

{

}

void push(T &c);

T pop();

bool isempty();

bool isfull();

T tos();

void exit();

void create();

void deallocate();

};