

ELEMI ALKALMAZÁSOK FEJLESZTÉSE II. Veremtípus osztálysablonja

Készítette: Gregorics Tibor
Steingart Ferenc

Tartalom

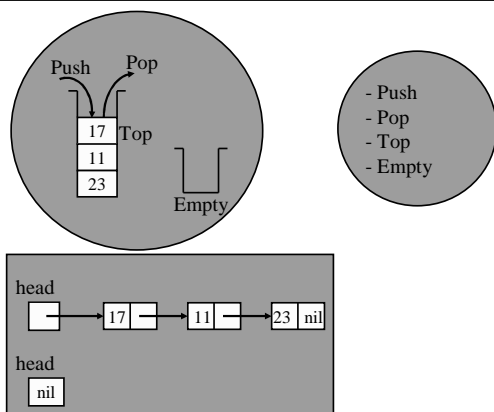
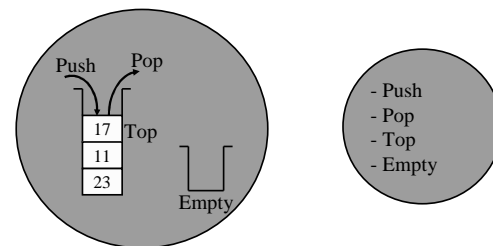


- Verem-típus megvalósítása
láncolt adatszerkezettel
- Verem-típus osztálysablonja (template)

1. Feladat

Olvassuk be a standard inputról érkező számokat, és írjuk ki őket fordított sorrendben a standard outputra!

A feladat megoldásához készítsünk egy egész számokat tartalmazó verem típust. A vermet egy egyirányú, fejelem nélküli láncolt listával reprezentáljuk.



stack.h : public

```
class Stack{
public:
    enum Exceptions{EMPTYSTACK};

    Stack();
    ~Stack();

    void Push(int e);
    int Pop();
    int Top();
    bool Empty();
};
```

stack.h : private

```
private:
Stack(const Stack&);
Stack& operator=(const Stack&);
struct Node{
    int cont;
    Node *next;
    Node(int e, Node *n):
        cont(e), next(n){};
};
Node *head;
};
```

stack.h : konstruktor, destruktor

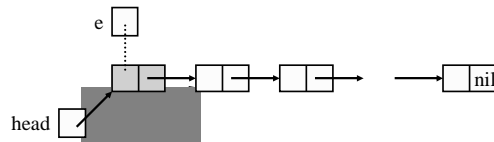
```
#define nil 0

Stack::Stack(): head(nil){}

Stack::~Stack()
{
    Node *p;
    while(head!=nil){
        p = head;
        head = head->next;
        delete p;
    }
}
```

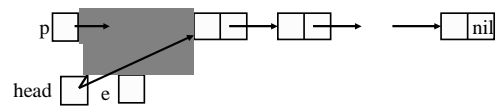
stack.h : Push

```
void Stack::Push(int e)
{
    head = new Node(e,head);
}
```



stack.h : Pop

```
int Stack::Pop()
{
    if(head==nil) throw EMPTYSTACK;
    int e = head->cont;
    Node *p = head;
    head = head->next;
    delete p;
    return e;
}
```



stack.h : Top, Empty

```
int Stack::Top()
{
    if(head==nil) throw EMPTYSTACK;
    return head->cont;
}

bool Stack::Empty()
{
    return head==nil;
}
```

fo.cpp

```
#include <iostream>
#include "stack.h"
int main()
{
    Stack s;
    int i;

    while(cin >> i){
        s.Push(i);
    }

    while(!s.Empty()){
        cout << s.Pop() << endl;
    }
    return 0;
}
```

2. Feladat

Olvassuk be a standard inputról érkező szavakat, és írjuk ki őket fordított sorrendben a standard outputra!

A feladat megoldásához készítsünk egy szavakat tartalmazó verem típust.
A vermet egy egyirányú, fejelem nélküli láncolt listával reprezentáljuk.

Fo.cpp

```
#include <iostream>
#include <string>
#include "stack.h"
int main()
{
    Stack<string> s;
    string i;

    cin >> i;
    while(i.compare("q")){
        s.Push(i);
        cin >> i;
    }
    while(!s.Empty()){
        cout << s.Pop() << endl;
    }
    return 0;
}
```

stack.h : public

```
template <class Element>
class Stack{
public:
    enum Exceptions{EMPTYSTACK};

    Stack();
    ~Stack();
    Stack(const Stack& s);
    Stack& operator=(const Stack& s);

    void Push(const Element& e);
    Element Pop();
    Element Top();
    bool Empty();
}
```

stack.h : private

```
private:
    struct Node{
        Element cont;
        Node *next;
        Node(const Element& e, Node *n):
            cont(e), next(n){};
    };
    Node *head;
};
```

Az osztálysablon műveleteinek megvalósítása is a header fájlba kerül.

stack.h : konstruktor, destruktor

```
template <class Element>
Stack<Element>::~Stack(): head(nil){}

template <class Element>
Stack<Element>::~Stack()
{
    Node *p;
    while(head!=nil){
        p = head;
        head = head->next;
        delete p;
    }
}
```

stack.h : Push, Pop

```
template <class Element>
void Stack<Element>::Push(const Element& e)
{
    head = new Node(e,head);
}

template <class Element>
Element Stack<Element>::Pop()
{
    if(head==nil) throw EMPTYSTACK;
    Element e = head->cont;
    Node *p = head;
    head = head->next;
    delete p;
    return e;
}
```

stack.h : Top, Empty

```
template <class Element>
Element Stack<Element>::Top()
{
    if(head==nil) throw EMPTYSTACK;
    return head->cont;
}

template <class Element>
bool Stack<Element>::Empty()
{
    return head==nil;
}
```

stack.h : copy konstruktor

```
template <class Element>
Stack<Element>::Stack(const Stack<Element>& s)
{
    if(s.head==nil) head = nil;
    else {
        head = new Node(s.head->cont,nil);
        Node *q = head;
        Node *p = s.head->next;
        while(p!=nil){
            q->next = new Node(p->cont,nil);
            q = q->next;
            p = p->next;
        }
    }
}
```

stack.h : értékadás

```
template <class Element>
Stack<Element>& Stack<Element>::operator=
(const Stack<Element>& s)
{
    if(this==&s) return *this;

    // ld destruktor

    // ld copy konstruktor

    return *this;
}
```

Fo.cpp

```
Stack<int>      si;
Stack<char>     sa;
Stack<string>   ss;

si.Push(4);
sa.Push('w');
ss.Push("alma");

try{
    cout << si.Pop();
    cout << sa.Pop();
    cout << ss.Pop();
}catch(Stack<int>::Exceptions e){
    if(e==Stack<int>::EMPTYSTACK){
        cout << "üres verem" << endl;
    }
}catch(Stack<string>::Exceptions e){
    if(e==Stack<string>::EMPTYSTACK){
        cout << "üres verem" << endl;
    }
}
```

VÉGE