Куп (stack) и основни операции кај куп – час 1

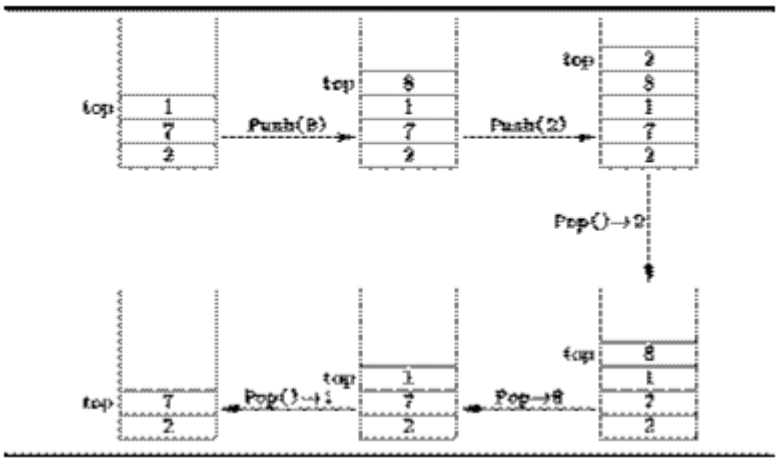
Куп (анг. stack) е линеарна листа која се карактеризира со операциите кои можат да се извршуваат со неа. Елементите од купот се додаваат според стратегијата на работа наречена "кој последен влегува прв излегува" (анг. Last In Frst Out- LIFO). Тоа значи дека секогаш нов податок врз купот се става врз последниот податок, додека вадењето податоци се врши во спротивен редослед од ставањето т.е прв се зема последниот ставен податок.

Податок - м
.....
.....
Податок – к
.....
.....
Податок – 4
Податок – 3
Податок – 2
Податок - 1

На сликата е даден графички приказ на куп со m елементи. Значи елементот кој што прв влегол во купот е податок 1, а елемент т.е податок кој што прв ќе излезе од купот е податок m односно ова е и последниот податок кој што влегол во купот. Купот исто така може да се дефинира како структура која зафаќа посебна меморија со одредена големина. Доколку капацитетот на купот е n, тогаш во него може да се сместат само n податоци. Со купот се извршуваат следните основни операции:

- иницијализација на празен куп
- проверка дали е купот полн
- ставање податок во купот
- проверка дали е купот празен
- земање податок од куп

Предноста на купот е дека сместувањето и отстранувањето на податок од врвот на купот се врши во O (1) време. Додека голем недостаток е тоа што освен првиот, има спор пристап до останатите податоци што се сместени во купот. За да се разбере идејата за куп, ќе разгледаме пример за Македонска пошта. Многу луѓе, кога ја добиле поштата, ја фрлаат во купот (корпа за пошта). Потоа, кога тие имаат слободно време ја процесираат акумулираната пошта од горе па надолу. Прво тие го отвараат писмото на врвот од купот, и превземаат соодветно дејство - ја плаќаат сметката, потоа ја фрлаат итн. Кога првото писмо е распоредено, тие го прегледуваат наредното писмо, кое е сега на врвот на купот, и го проследуваат. Тие работат по ред до долното писмо на купот (кое е сега на врвот).



Како што може да се забележи на сликата купот е базиран на низа. Па така на него гледаме како низа од податоци. Иако е базирано на низа, купот има ограничен пристап, па така не можеме да пристапиме на него како во низа, т.е немаме пристап до било кој податок. Купот на сликата започнува со веќе внесени податоци. Доколку сакаме да започнеме со празен куп, креираме метод **new** што ќе конструира куп без податоци. За се внесе податок во купот, креираме метод **push**. Како што гледаме на сликата се врши внесување на елементите 8, па потоа 2. Внесувањето податоци во купот е на таков начин што се оди по редослед од дното па нагоре до врвот. Доколку сакаме да отстраниме податок од врвот на купот, креираме метод **pop**(). Како што гледаме на сликата се врши отстранување на елементите 2,8 и 1. При пишување на програмата важно е да се напише код кој нема да ги дозволи следните два случаи: - кога купот е празен, а ние сакаме да отстраниме податоци од празен куп - коге купот е полн, а ние сакаме да сместиме уште податоци.

Основни операции со куп

push – додава елемент на врвот на стекот
Синтакса: ime_na_stek.push(vrednost)

Vrednost се пренесува како параметар и како резултат се додава елемент во стекот.
Пр. stek1.push(77)
stek1.push(88)
Излез
77, 88

pop – го отстранува најгорниот елемент од стекот
Синтакса: ime_na_stek.pop()
Пр.
stek1 = 10,20,30;
stek1.pop();
Излез
10, 20

peek- го зема најгорниот елемент од стекот но не го отстранува

isFull – проверува дали стекот е полн

swap – се корист за замана на содржините на два стека

Empty- проверува дали стекот е празен

Синтакса: ime_na_stek.empty()

Враќа вредност true или false т.е. дали стекот е празен или не е.

size – број на елементи во купот

Синтакса: ime_na_stek.size()

Враќа вредност број на елементи на стекот

top – пристап до најгорниот елемент на купот

Синтакса: ime_na_stek.top()

Враќа вредност на најгорниот елемент на стекот

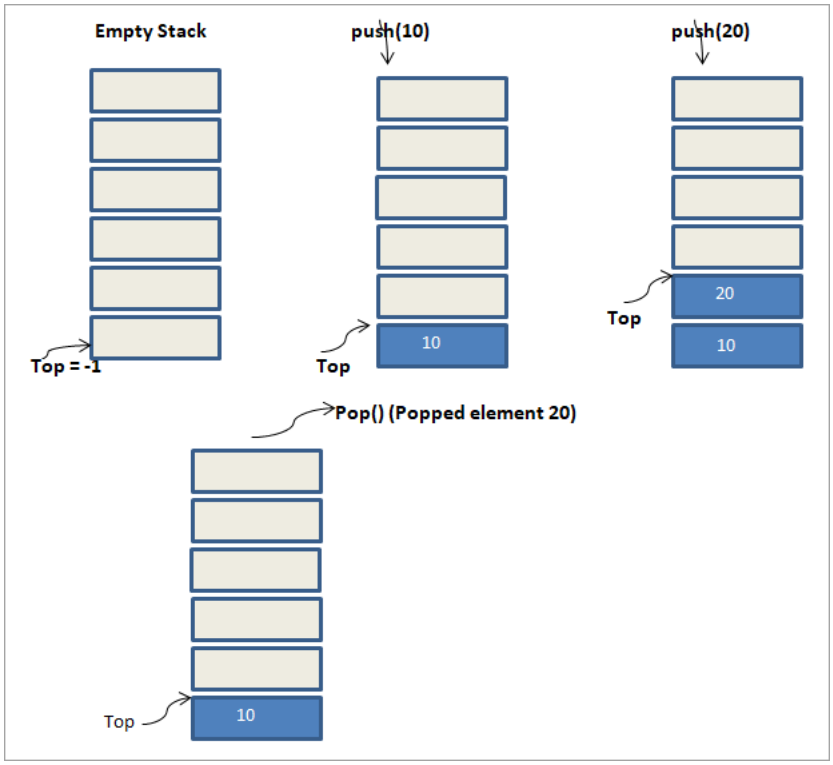
Пр.

```
ime_na_stek.push(5);
ime_na_stek (6);
ime_na_stek.top();
```

Излез:

6

Сликава ги прикажува операциите што се извршуваат со стек.
На почеток стекот е празен. За празен стек, top на стекот е поставен на -1.
Потоа со push се внесува елементот 10 во стекот. Top сега покажува на 10.



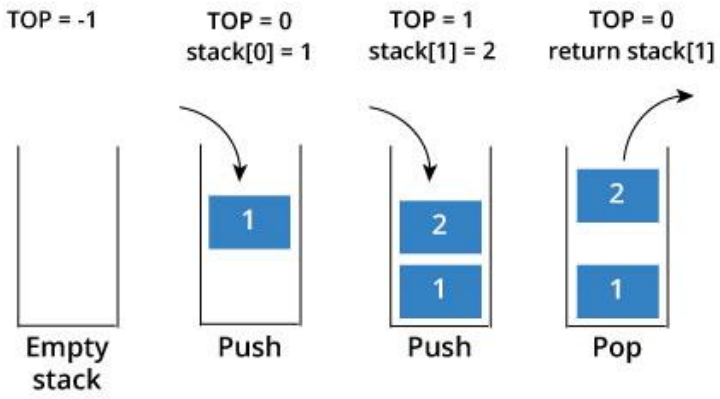
Следно се додава 20, а top покажува на 20. На последната фигура од сликата е прикажана операцијата pop. Како резултат на оваа операција елементот кој е на врвот од стекот се (вади) бриши од стекот. Во примерот, 20 е избришан. Сега на врвот на стекот е 10.

Покажувачот TOP се користи за да го означува горниот елемент на стекот. Кога се иницијализира

стек, неговата вредност е -1, па за да се провери дали стекот е празен, може (top ==-1) да се спореди со -1. Со додавањето на елемент, вредноста на top се зголемува и новиот елемент се поставува на позиција на која покажува top.

При (вадење) бришење на елемент, се зема елементот на кој покажува top и се намалува вредноста на top.

Пред да се внесе елемент, треба да се провери дали стекот е полн.



Пред (вадење) бришење на елемент треба да се провери дали стекот е празен.

Иако стекот е едноставна податочна структура за применување, има големи можности:

- за наоѓање на обратен збор – се внесуваат сите букви и потоа се вадат една по една и бидејќи редоследот на стекот е LIFO, се добиваат буквите во обратен редослед.
- Во компајлери – компајлерите користат стек за да пресматаат вредности на изрази како 2+4/5*(7-9) со претворање ма изразот во префиксна или постфиксна форма.
- Во пребарувачи – копчето „назад“ ги зачувува сите урл адреси на претходно посетени страни во стек. Секогаш кога се посетува нова страна, нејзината адреса се додава на врввот на стекот. Кога ќе се притисне копчето „назад“ тековната урл адреса е извадена од стекот и се пристапува до претходната урл адреса.

Пример програма за креирање на стек и примена на основните операции со стек

```
#include <iostream>
#include <stack>

using namespace std;

int main()
{
stack<int> nStack;

cout<<"sega imame stek !!!"<<endl;
cout<<endl;
cout<<"goleminata e="<<nStack.size()<<endl;
if (nStack.empty()==true) cout<<"stekot e prazen ";
else cout<<" stekot ne e prazen ";
cout<<endl;
cout<<"se vnesuvaat dva elementi vo stekot!!!"<<endl;
nStack.push(1);
nStack.push(2);
cout<<"goleminata e="<<nStack.size()<<endl;
if (nStack.empty()==true) cout<<"stekot e prazen "<<endl;
else cout<<" stekot ne e prazen "<<endl;
```

```
cout<<endl;
cout<<"se vnesuva uste eden element vo stekot!!!"<<endl;
nStack.push(3);
cout<<"goleminata e="<<nStack.size()<<endl;
int nElement =nStack.top();
if (nStack.empty()==true) cout<<"stekot e prazen "<<endl;
else cout<<" stekot ne e prazen "<<endl;
cout<<endl;
cout<<"se vadi gorniot element!!!"<<endl;
nStack.pop();
cout<<"goleminata e="<<nStack.size()<<endl;
if (nStack.empty()==true) cout<<"stekot e prazen "<<endl;
else cout<<" stekot ne e prazen "<<endl;
return 0;
}
```

Пр. програма со која се полни стек со n елементи цели броеви и се пресметува збир на внесените броеви

```
#include <iostream>
#include <stack>
using namespace std;

int main()
{
   int sum = 0,n,x,i;
   stack<int> mstack;
   cout<<"vnesi broj na elementi vo stekot"<<endl;
   cin>>n;
   for(i=1;i<=n;i++)
   {
      cout<<"vnesi broj"<<endl;
      cin>>x;
      mstack.push(x);
   }
   while (!mstack.empty()) {
      sum+= mstack.top();
      mstack.pop();
   }
   cout << "zbirot na elementite e "<<sum;
   return 0;
}
```

Пр. програма со која се полни стек со n елементи од тип string, а потоа се печатат елементите

```
#include <iostream>
#include <stack>
using namespace std;

int main()
{
```

```
    int sum = 0,n,i;
    string x;
    stack<string> mstack;
    cout<<"vnesi broj na elementi vo stekot"<<endl;
    cin>>n;
    for(i=1;i<=n;i++)
    {
       cout<<"vnesi string"<<endl;
       cin>>x;
       mstack.push(x);
    }
    while (!mstack.empty()) {
       cout<< mstack.top()<<endl;
       mstack.pop();
    }
      return 0;
}
```

Задачи за домашна работа:

1. Програма со која се полни стек со n елементи цели броеви и се пресметува збир на внесените броеви кои припаѓаат на втората десетка
2. Програма со која се полни стек со n елементи цели броеви и се пресметува производ на елементите кои се делат со 3.
3. Програма со која се полни стек со n елементи цели броеви и се пресметува аритметичка средина на елементите кои се непарни броеви
4. Програма со која се полни стек со n елементи реални броеви и се пресметува збир на позитивните вредности на внесените броеви.

Прашања поврзани со наставните единици може да се испраќаат на email: anetastojceska@gmail.com
Решенијата на задачите за домашна работа да се испратат најдоцна до 15.04.2020 година на email: anetastojceska@gmail.com