**Stiva – implementare secvențială (statică, cu vectori)**

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

//dimensiunea maxima a vectorului
const int MAX_SIZE=100;

//declaram stiva
int STIVA [MAX_SIZE];

//crearea unei stive vide: pentru a crea o stiva vida initializam varful stivei cu -1
//varful stivei indica intotdeauna pozitia ultimului element introdus in stiva
//elementele sunt memorate in vector incepand cu pozitia 0
int varf=-1;

//inserare element in stiva
void push (int valoare)
{
   if (varf==MAX_SIZE-1) //stiva este plina
      cout<<"Stiva este plina!"<<endl;
   else //inseram elementul in stiva
   {
      varf++; //marim varful stivei
      STIVA[varf] = valoare; //plasam la varf noul element
   }
}

// extragere element din stiva
void pop ()
{
   if (varf<0)  //stiva este vida
      cout<<"Stiva este vida!"<<endl;
   else //eliminam elementul de la varf
      varf--;
}

//accesarea elementului de la varful stivei
int top ()
{
   if (varf<0)
      {
      cout<<"Stiva este vida!";
      return -1; // trebuie returnata o valoare diferita de cea a elementelor
      }
   return STIVA[varf];
}
```

```
//afisare stiva
void afisare()
{
   cout<<"Stiva ["<<varf+1<<"]: ";
   for (int i=0; i<=varf; i++)
       {
       cout<<STIVA[i]<<" ";
       }
   cout<<endl;
}

int main()
{
//am declarat deja global stiva vida

push(10);
push(20);
push(30);
push(40);
push(50);
afisare(); // Stiva[5]: 10 20 30 40 50
pop();
pop();
pop();
afisare(); // Stiva[2]: 10 20
cout<<top(); // 20
return 0;
}
```

**Coada – implementare secvențială (statică, cu vectori)**

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

//dimensiunea maxima a vectorului
const int MAX_SIZE=100;

//declaram coada
int COADA [MAX_SIZE];

//crearea unei cozi vide
//elementele sunt memorate in vector incepand cu pozitia 0
//numarul de elemente din coada este 0
//pozitia pe care va fi plasat primul element din coada este 0
//elementele sunt memorate în vector de la pozitia inc (inceput) pana la pozitia sf (sfarsit)
//numarul lor este sf-(inc-1)=sf-inc+1
int inc=0, sf=-1;

//inserare element in coada
void push (int valoare)
{
   if (sf==MAX_SIZE-1) //nu mai avem loc
      cout<<"Nu mai putem adauga elemente in coada!"<<endl;
   else //inseram elementul in coada
   {
      sf++;
      COADA[sf]=valoare; //plasam noul element la sfarsitul cozii
   }
}

// extragere element din coada
void pop ()
{
   if (inc>sf) //coada este vida
      cout<<"Coada este vida!"<<endl;
   else //eliminam primul element
      inc++; //marim cu o unitate inceputul cozii
}

//accesarea primului element din coada
int Front ()
{
   if (inc>sf)
      {
      cout<<"Coada este vida!"<<endl;
      return -1; // trebuie returnata o valoare diferita de cea a elementelor
      }
   return COADA[inc];
}
```

```
//accesarea ultimului element din coada
int Back ()
{
   if (inc>sf)
      {
      cout<<"Coada este vida!"<<endl;
      return -1; // trebuie returnata o valoare diferita de cea a elementelor
      }
   return COADA[sf];
}

//afisare coada
void afisare()
{
   cout<<"Coada ["<<sf-inc+1<<"]: ";
   for (int i=inc; i<=sf; i++)
      {
      cout<<COADA[i]<<" ";
      }
   cout<<endl;
}

int main() {
//am declarat deja global coada vida
push(10);
push(20);
push(30);
push(40);
push(50);
afisare(); // Coada[5]: 10 20 30 40 50
cout<<Front()<<endl; //10
cout<<Back()<<endl; //50
pop();
pop();
afisare(); // Coada[3]: 30 40 50
cout<<Front()<<endl; // 30
return 0;
}
```

Observație!
În acest mod de alocare statică a unei cozi observați că pe măsură ce inserăm şi extragem elemente, atât sfârşitul, cât şi începutul cozii „migrează” către limita maximă a vectorului. Dezavantajul este că în vector rămân poziții neutilizate (de la 0 până la inc-1). De exemplu, ar putea să apară o situație în care coada conține un element, dar operația de inserare să nu fie posibilă pentru că inc=sf= MAX_SIZE -1. O soluție mai bună ar fi de a reutiliza circular spațiul de memorie alocat cozii. Pentru aceasta, următoarea poziție după poziția i poate fi:

- i+1, dacă i< MAX_SIZE -1
- 0, dacă i= MAX_SIZE -1

Acest lucru se poate scrie concis: (i+1)% MAX_SIZE.