Structura de date de tip coadă(Queue)

Această structură de date este un caz particular de listă care funcționează pe principiul **FIFO** (first in – first out, primul intrat este primul servit). Evident, modelul unui şir la un ghișeu este cea mai nimerită reprezentare pentru o coadă.

Prin urmare principalele prelucrări care se referă la această structură de date vor fi:

- creare coada
- listare coada (parcurgere)
- adăugare la sfârșit
- ștergere prim element
- prelucrare prim element

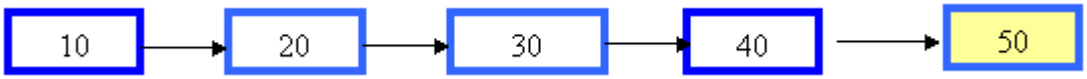
Specific acestei structuri de date este faptul ca adăugarea se va face întotdeauna la un capăt în timp ce prelucrarea (ștergerea) se va face la celălalt capăt.

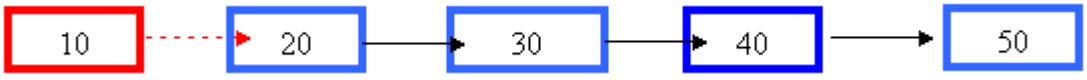
Deoarece intrările se fac la un capăt şi ieşirile la celălalt capăt, se observă că este bine să utilizăm doi pointeri: unul către adresa primului nod şi al doilea pentru adresa ultimului nod, astfel încât să prelucrăm mai uşor coada.

Funcția ***pune( )*** sau ***push()***, creează coada când aceasta este vidă sau adaugă un nou element la sfârșit în caz contrar.
Funcția ***scoate( )*** sau ***pop()***, elimină elementul din vârful cozii.

Fie o coadă care reţine 4 numere întregi: 10,20,30,40

În urma adăugării valorii 50 , prin apelul funcției ***push()*** se va obține:





In urma eliminării primului element,10, reținut de vârful cozii(prim), prin apelul funcției ***pop()*** se va obține:

Implementarea statică

Implementare 1

Coada prelucrează numere întregi.
Vom folosi un vector pentru reprezentarea unei cozi şi reţinem ultimul element din coada.

Declarare:

int c[100], ultim; //c este vectorul in care retinem elementele

Coada vida

ultim=0; //nu exista nici un element in coada

PUSH – adaugarea unui element (la sfarsit)

ultim++; cin>>c[ultim]; //facem loc unui noi valori si citim valoarea dorita

POP – eliminarea primului element

Urmând modelul real (al unei cozi la un ghișeu), toate elementele, începând cu poziția a 2-a, vor fi mutate la stânga. Coada va avea acum un element în minus.

for(i=2;i<=ultim;i++) c[i-1]=c[i];
ultim--;

Un exemplu de implementare:

```
#include <iostream>

using namespace std;

int c[100],ultim;

void push()
{
   ultim++;
   cout<<"Introdu valoarea: ";
   cin>>c[ultim];
}

void pop()
{
   int i;
   //prim++;
   for(i=2;i<=ultim;i++)
      c[i-1]=c[i];
   ultim--;
}
```

```
void top()//prelucarea primului element din coada
{
   cout<<c[1]<<" ";
}

int main()
{
   int n,i;
   cout << "Numar de elemente in coada= " ;
   cin>>n;
   for(i=1;i<=n;i++)
      push();
   cout<<"Elementele din coada= ";
   while(ultim!=0)
   {
      top();
      pop();
   }

   return 0;
}
```

Practic, oricare operație tip POP este de complexitate N (numărul de elemente din coadă), ceea ce este destul de mult.

O alternativă ar fi să nu respectăm modelul real, în care se mută cei care stau la rând, ci să îl punem pe funcționar să se miște cu biroul lui.

Avem nevoie de un indice **prim** care sa arate cine este persoana/elementul din fața funcționarului.

Implementare 2

Declarare:
int c[100], prim, ultim; //sc este sfârșitul cozii; ic este începutul cozii

Coada vida
Inițial: *prim=0; ultim =0*//nu exista nici un element in coada
Pe parcurs coada va fi vida când *prim>ultim*

PUSH – adăugarea unui element (la sfârșit)
if(ultim==0) prim++; //marchez inceputul cozii
ultim++; cin>>c[ultim]; //facem loc unui noi valori si citim valoarea dorita

POP – eliminarea primului element
Eliminarea primului element se reduce la deplasarea începutului cozii la dreapta.

if(prim<=ultim) prim++;
Se observă ca operația se rezolvă într-un pas, nu în N ca în cazul anterior

O implementare

```
#include <iostream>
```

```
using namespace std;

int c[100],prim,ultim;

void push()
{
   cout<<"Introdu valoarea: ";
   if(ultim==0)
      prim++; //marchez inceputul cozii
   ultim++;
   cin>>c[ultim];
}

void pop()
{
   if(prim<=ultim)
      prim++;
}


void top()//prelucarea primului element din coada
{
   cout<<c[prim]<<" ";
}

int main()
{
   int n,i;
   cout << "Numar de elemente in coada= " ;
   cin>>n;
   for(i=1;i<=n;i++)
      push();
   cout<<"Elementele din coada= ";
   while(prim<=ultim)
   {
      top();
      pop();
   }

   return 0;
}
```

Implementarea dinamică

In continuare se prezintă o soluție pentru implementarea funcțiilor. Coada prelucrează numere întregi. Vom reține adresa primului nod și al ultimului.

```
#include<iostream>

using namespace std;

struct nod
{
   int info;
   nod *urm;
};

void push(nod* &prim, nod* &ultim, int x)
{
   nod *c;
   if(!prim)
      {
         prim=new nod;
         prim->info=x;
         prim->urm=NULL;
         ultim=prim;
      }
   else
      {
         c=new nod;
         ultim->urm=c;
         c->info=x;
         ultim=c;
         ultim->urm=NULL;
      }
}

void afisare(nod *prim)
{
   nod *c;
   c=prim;
   while(c)
   {
      cout<<c->info<<" ";
      c=c->urm;
   }
}
```

```
void pop(nod* &prim)
{
   nod* c;
   if(!prim)
      cout<<"coada este vida si nu mai ai ce elimina!!!";
   else
   {
      c=prim;
      prim=prim->urm;
      delete c;
   }
 }


int main()
{
   int n, a, i;
   nod *prim=0,*ultim=0;//varful si ultimul element al cozii

   cout<<"numarul initial de noduri ";
   cin>>n;
   for(i=1;i<=n;i++)
   {
      cout<<"valoarea de adaugat in coada ";
      cin>>a;
      push(prim, ultim, a);
   }
   cout<<endl;
   afisare(prim);

   int nre, nra;
   cout<<endl<<"cate adaugari ?";
   cin>>nra;
   for(i=1;i<=nra; i++)
   {
      cout<<"valoarea de adaugat ";
      cin>>a;
      push(prim, ultim, a);
   }
   cout<<endl<<"dupa adaugare"<<endl;
   n=n+nra;
   cout<<"coada are  "<<n<<" elemente"<<endl;
   afisare(prim);
   cout<<endl<<"cate eliminari ?";
   cin>>nre;
   for(i=1; i<=nre; i++)
```

```
      pop(prim);
   cout<<endl<<"dupa eliminare"<<endl;
   n=n-nre;
   cout<<"coada are  "<<n<<" elemente"<<endl;
   afisare(prim);
   //prelucrez varful cozii: de exemplu se poate dubla continutul:
   prim->info=2*prim->info;
   cout<<endl<<"dupa dublarea valorii varfului "<<endl;
   afisare(prim);
   return 0;
}
```

Se observa că în main se realizează prelucrarea (modificarea) informației primului nod.

Exercitiu propus:

1) Sa se prelucreze datele celor n angajati ai unei firme (nume, profesia, salariu). Datele se citesc din fisier si sunt inregistrate in ordinea angajarii (pentru fiecare persoana datele sunt scrise pe o linie).

a) Sa se afiseze datele dupa ce acestea au fost memorate intr-o structura de tip date.
b) Cel mai vechi angajat se pensioneaza. Sa se afiseze si sa modifice datele din structura si apoi din fisier
c) Cel mai vechi angajat primeste o prima de 2000 000 de lei. . Sa se afiseze si sa modifice datele din structura si apoi din fisier

2) Sa se memoreze n numere intrefi intr-o structura de tip coada . Sa se stearga elementele neprime din varful cozii pana se intalneste un numar prim.

3) La o intrare intr-o sala de spectacole sunt 2 randuri de persoane pastrate in doua structuri de tip coada. Pentru ca exista doar un singur angajat care verifica biletele, pe poarta poate sa intre la un moment dat o singura persoana. Stiind ca angajatul este corect si permite sa intre alternativ cate o persoana din fiecare rand, se cere sa se afiseze ordinea de intrare a persoanelor in sala de spectacol.

4) La un cabinet stomatologic sunt asezate n persoane la rand dintre care m urgente. Asistenta duce lista pacientilor medicului de serviciu astfel incat acesta va trebui sa rezolve mai intai urgentele.