Vaje 16

Matjaž Mastnak Dalibor Šabič

23.2, 24.2 in 3.3

LIFO in FIFO metoda

- LIFO metoda, lahko bi tudi rekli, *last in first out* oz. zadnji ki ga damo noter pride prvi ven.
- FIFO metoda, *first in first out* oz. prvi noter prvi ven.

Primer lifo metode smo si že ogledali pri skladih, kjer stvari skladamo eno na drugo. Primer fifo metode pa si bomo ogledali danes.

`Vrsta.h`:

```
#include "../../code/bool/bool.h"
#include "../../code/sklad/sklad.h"
class Vrsta {
private:

  Sklad vhodni;            /* nedostopno ostalim */
  Sklad izhodni;           /* nedostopno ostalim */

public:

  Vrsta();
/* Desc:   Ustvari prazno vrsto.
   Parms:  /
   Result: vrsta
   Pre:    /
   Post:   VRSTA_Prazna()
*/
  ~Vrsta();
/* Desc:   Unici vrsto.
   Parms:  /
   Result: /
   Pre:    Vrsta je bila ustvarjena.
   Post:   Vrste ne moremo vec uporabiti.
*/
//-------------------------SPREMINJANJA

  void Dodaj( int element);
/* Desc:   Doda element v vrsto.
```

```
   Parms:  element = element, ki ga dodamo v vrsto ( po vrednosti )
   Result: /
   Pre:    not(Polna()) and [vrsta]
   Post:   vrsta'= [element|vrsta]
*/
  int Izloci();
/* Desc:   Izloci element iz vrste.
   Parms:  /
   Result: element iz vrste
   Pre:    not(Prazna()) and [ostalo|element]
   Post:   vrsta'= [ostalo]
*/
//-------------------------POIZVEDOVANJA

  bool Prazna();
/* Desc:   Preveri, ali je vrsta prazna.
   Parms:  /
   Result: vrsta = []
   Pre:    Vrsta ustvarjena.
   Post:   /
 */
  bool Polna();
/* Desc:   Preveri, ali je vrsta polna.
   Parms:  /
   Result: ce ne moremo vec dodati then TRUE else FALSE
   Pre:    Vrsta je bila ustvarjena.
   Post:   /
*/
};  /* Vrsta */
```

In še izvedbeni modul za vrsto (`Vrsta.cpp`):

```
#include "Vrsta.h"
Vrsta::Vrsta(){};

Vrsta::~Vrsta(){};

//-------------------------SPREMINJANJA

                 // ce je vhodni sklad poln, potem ga prestavimo
                 // v izhodni sklad

                 // naloži na vhodni sklad element
void Vrsta::Dodaj(int element) {
  // ce je vrednost dane funkcije TRUE, potem prestavimo -zlozi
  // dokler ni prazen
  if ( vhodni.Poln())
    while (!vhodni.Prazen())  izhodni.Nalozi(vhodni.Zlozi());

  vhodni.Nalozi(element);
}   /* Dodaj */
```

```
int Vrsta::Izloci() {
  // ce je izhodni sklad prazen, potem prestavi vse iz vhodnega
  // v izhodnega

  if ( izhodni.Prazen())
    while (!vhodni.Prazen()) izhodni.Nalozi(vhodni.Zlozi());

  return (izhodni.Zlozi());
}  /* Izloci */

//-------------------------POIZVEDOVANJA

bool Vrsta::Prazna() {
  // ce vhodni izhodni sklad prazen je vrsta prazna

  return (vhodni.Prazen() && izhodni.Prazen());
}  /* Prazna */

bool Vrsta::Polna() {
  // vrsta je polna, ce vhodni sklad poln in izhodni ni prazen

  return (vhodni.Poln() && !izhodni.Prazen());
} /* Polna */
```

Še testni program (`TestVrsta.cpp`):

- Vstavi v vrsto
- Preberi znak:
  - V - postavi element v vrsto
  - I - iz vrste izloci element
  - K - konec

```
#include <stdio.h>
#include "../../code/vrsta/vrsta.h"

main () {
  Vrsta A;
  int element;
  char znak;

  znak = getchar();      // getchar - pomeni, da bo znak prebral
  while (znak != 'K') {
    if (znak == 'V') {
      scanf("%d", &element);
      if (A.Polna()) printf("Ni vec prostora.\n");
      else A.Dodaj(element);
    } //if
    if (znak == 'I') {
```

```
      if (A.Prazna()) printf("V vrsti ni vec elementov.\n");
      else printf("%d", A.Izloci());
    } //if
    znak = getchar();
  } //while
} //main
```

2. Primer

Naša nova vrsta je sestavljena iz členov, ki si med seboj sledijo v krogu. Imenujemo jo hodnik `Vrsta2.h`:

```
#include "../../code/bool/bool.h"
#define max_vrsta 10      // najvecja mozna dolzina vrste

class Vrsta {
private:                  // ZASEBNO - NE GLEJ!

  int hodnik[max_vrsta];  // krozna vrsta s shranjenimi elementi
  int prvi,               // polozaj prvega elementa
              zadnji;     // v resnici polozaj prvega prostega mesta
                          // v vrsti
  bool zakljucena;        // vsa mesta v vrsti so zapolnjena

public:                   // JAVNO
//--------------------------USTVARJANJE IN UNICEVANJE

  Vrsta();
/* Desc:   Ustvari prazno vrsto.
   Parms:  /
   Result: vrsta
   Pre:    /
   Post:   Prazna()
*/

  ~Vrsta();
/* Desc:   Unici vrsto.
   Parms:  /
   Result: /
   Pre:    Vrsta je bila ustvarjena.
   Post:   Vrste ne moremo vec uporabiti.
*/
//-------------------------SPREMINJANJA

  void Dodaj(int element);
/* Desc:   Doda element v vrsto.
   Parms:  element - element, ki ga dodamo v vrsto ( po vrednosti )
   Result: /
   Pre:    not(Polna()) and [vrsta]
   Post:   vrsta'= [element|vrsta]
```

```
*/
  int Izloci();
/* Desc:   Izloci element iz vrste.
   Parms:  /
   Result: element iz vrste
   Pre:    not(Prazna()) and [ostalo|element]
   Post:   vrsta'= [ostalo]
*/
//-------------------------POIZVEDOVANJA

  bool Prazna();
/* Desc:   Preveri, ali je vrsta prazna.
   Parms:  /
   Result: vrsta = []
   Pre:    Vrsta ustvarjena.
   Post:   /
 */
  bool Polna();
/* Desc:   Preveri, ali je vrsta polna.
   Parms:  /
   Result: ce ne moremo vec dodati then TRUE else FALSE
   Pre:    Vrsta je bila ustvarjena.
   Post:   /
*/
};  /* Vrsta */
```

Vrsta2.cpp:

```
#include "Vrsta2.h"

Vrsta::Vrsta(){
  prvi = 0;
  zadnji = 0;
  zakljucena = false;
} // Vrsta

Vrsta::~Vrsta(){};

//-------------------------SPREMINJANJA

void Vrsta::Dodaj(int element) {
  hodnik[zadnji] = element;
  zadnji = ((zadnji+1) % max_vrsta);
  zakljucena = (prvi == zadnji);
}   /* Dodaj */

int Vrsta::Izloci() {
  int zac_elt;                    // zacasni element
  zac_elt = hodnik[prvi];
  prvi = ((prvi+1) % max_vrsta);
  zakljucena = false;
```

```
  return zac_elt;
}  // Izloci

//-------------------------POIZVEDOVANJA

bool Vrsta::Polna() {
  return (zakljucena);
}  // Polna

bool Vrsta::Prazna() {
  return ((prvi == zadnji) && (!zakljucena));
} // Prazna
```

Testni program `TestKrog.cpp`:

```
#include <stdio.h>
#include "../../code/Vrsta/Vrsta2.h"

int main () {
  Vrsta A;
  int element;
  char ukaz[256];

  scanf("%s", ukaz);
  while (ukaz[0] != 'K') {
    switch (ukaz[0]) {
    case 'V':
    case 'v':
      scanf("%d", &element);
      if (A.Polna()) printf("Ni vec prostora.\n");
      else A.Dodaj(element);
    break;
    case 'I':
    case 'i':
      if (A.Prazna()) printf("V vrsti ni vec elementov.\n");
      else printf("%d\n", A.Izloci());
    break;
    } // switch
    scanf("%s", ukaz);
  } //while

  return 0;

} // main
```