**UREJENO POVEZAN ENOSTRANSKI LINEARNI SEZNAM**

Vstavljanje novega elementa:

- Dinamična rezervacija za hranjenje novega elementa

- Določi mesto vstavljanja:

- Vstavljanje novega elementa v seznam (med prevPtr in currPtr)

struct element

{

intdata**;**

struct element \*nasl;

}\*Start, \*tmp;

void insert(int a)

{

struct element \*currptr,\*prevptr;

tmp=(struct element \*)malloc(sizeof(struct element));

tmp->data=a

currptr=Start;

prevptr=NULL;

//doloci pozicijo vstavljanja

while ((currptr!=NULL)&&(item>(currptr->data)))

{

prevptr=currptr;

currptr=currptr->nasl;

}

//vstavljanje novega elementa

if (prevptr==NULL) //vstavi na zacetek; velja tudi, če je seznam prazen

{

tmp->nasl=Start;

Start=tmp;

}

else /\*vstavi med currptr in prevptr\*/

{

tmp->nasl=currptr;

prevptr->nasl=tmp;

}

}

Primer:

#include <iostream>

#include <cstdio>

using namespace std;

struct **element**

{

int x;

struct element \*nasl;

}\*Start,\*tmp;

void **insert**(int a)

{

struct element \*currptr,\*prevptr;

tmp=(struct element \*)malloc(sizeof(struct element));

tmp->x=a;

currptr=Start;

prevptr=NULL;

while ((currptr!=NULL)&&(a>(currptr->x)))

{

prevptr=currptr;

currptr=currptr->nasl;

}

if (prevptr==NULL)

{

tmp->nasl=Start;

Start=tmp;

}

else

{

tmp->nasl=currptr;

prevptr->nasl=tmp;

}

}

void **Izpis** (struct element \*p)

{

if (p!=NULL)

{ cout << p->x;

Izpis(p->nasl);}

}

int **main** ()

{

int i,st;

Start=NULL;

for(i=0;i<5;i++)

{

cin >> st;

insert(st);

}

cout<<"Seznam pred brisanjem:\n";

Izpis(Start);

return 0;

}

**Brisanje elementa seznama:**

Pazimo na to ali podatek za brisanje ustreza data v prvem elementu seznama, drugače se podamo na iskanje elementa(vozlišča) za brisanje v seznamu:

void **Delete** (int a)

{

struct element \*prevPtr, \*currPtr, \*tempPtr;

if (Start==NULL)

cout<< "Seznam je prazen";

else

{

//a preveri, ce brises prvega

if (a == (Start->x))

{

tempPtr=Start;

Start=Start->nasl;

free (tempPtr);

}

else

{ // isci element (vozlisce) za brisanje

prevPtr = Start; // zacetek-prvi

currPtr = Start->nasl; // drugi

while ((currPtr!=NULL) && ((currPtr->x)!=a))

{

prevPtr = currPtr;

currPtr = currPtr->nasl;

}

if (currPtr == NULL)

cout<< "elementa ni v seznamu ";

else

{ // brisi element

tempPtr = currPtr;

prevPtr->nasl = currPtr->nasl;

free (tempPtr);

}

}

}

}