Základy programování v C++ 18. cvičení Lineární spojový seznam I.

Zuzana Petříčková

27. listopadu 2019

Přehled

Dynamické datové struktury

Již známe:

- dynamické pole
- dynamická matice
- ,,chytré" pole

Dnes:

- spojový seznam

Spojový seznam

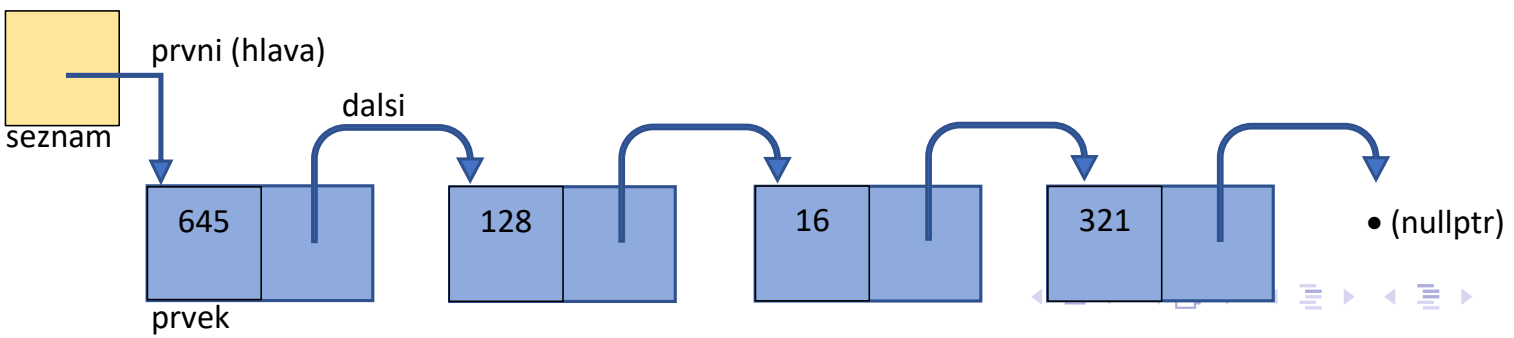
Jednosměrně zřetězený (lineární) spojový seznam

- dynamická datová struktura, jejíž prvky jsou stejného typu, ale na rozdíl od pole nejsou nutně v paměti umístěny za sebou

Narozdíl od dynamického pole:

- **„rychlejší"** vkládání nového prvku „na" požadovanou pozici
- **„pomalejší"** přístup k požadovanému (např. i-tému) prvku

Spojový seznam je velmi často používaná datová struktura, je součástí standardních knihoven většiny moderních programovacích jazyků (C++, Java,...)



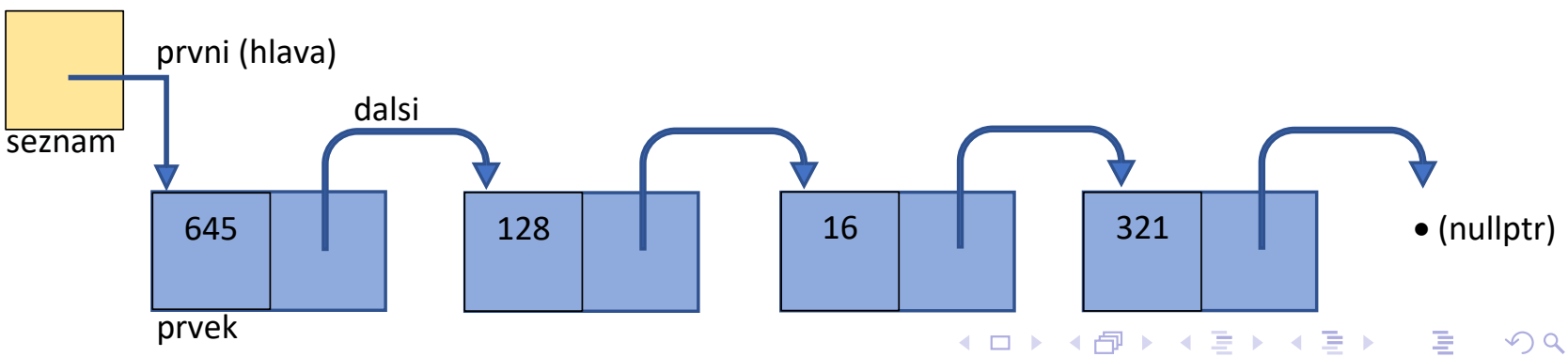
Spojový seznam (základní varianta)

Struktura prvku

- vlastní data
- ukazatel na další prvek (**dalsi** / **next**) nebo **nullptr**

Struktura seznamu

- ukazatel na první prvek (**první** / **hlava** / **head**)
- prvky seznamu nacházející se za hlavou se někdy souhrnně nazývají jako **ocas** / **tail**



Spojový seznam

Typy spojových seznamů podle struktury seznamu

struktura seznamu obsahuje:

1. jen ukazatel na první prvek
2. ukazatel na první prvek a ukazatel na poslední prvek
3. ukazatel na první prvek a ukazatel na tzv. **zarážku** (speciální poslední prvek, který nenese data)

Spojový seznam

Typy spojových seznamů podle zřetězení

1. jednosměrně zřetězený ... každý prvek obsahuje:
   - vlastní data
   - ukazatel na další prvek
2. obousměrně zřetězený ... každý prvek obsahuje:
   - vlastní data
   - ukazatel na další prvek
   - ukazatel na předchozí prvek

Spojový seznam

Příklad: Vytvoříme jednosměrně zřetězený spojový seznam se zarážkou. Implementujeme některé z následujících operací:

- vytvoření prázdného seznamu
- smazání všech prvků seznamu
- zrušení seznamu
- test prázdnosti seznamu
- výpis obsahu seznamu
- přidání prvku
  - na začátek seznamu
  - na konec seznamu
  - za zadaný prvek
- nalezení prvku s požadovanými daty
- smazání (vyjmutí)
  - prvního prvku seznamu
  - posledního prvku seznamu
  - prvku zadaného hodnotou / indexem

Spojový seznam

Další operace nad seznamem (pokud každý prvek má klíč a hodnotu)

- nalezení prvku s maximálním klíčem
- setřídění prvků seznamu podle klíče
- ...

Spojový seznam ... definice datové struktury

```
// definuji typ hodnot ukladanych do seznamu:
using T = int;
// using T =  Zamestnanec;
// using T =  float;

struct Prvek
{
    T data;
    Prvek *dalsi=nullptr;
};

struct Seznam
{
    Prvek *hlava = nullptr;
    Prvek *zarazka = nullptr;
};
...
```

Spojový seznam ... prvni funkce

```
void vytvor(Seznam &s);
void smazVsechny(Seznam &s);
void zrus(Seznam &s);
bool jePrazdny(Seznam &s);

void vlozNaZacatek(Seznam &s, T co);
void vlozNaKonec(Seznam &s, T co);

void vypis(Seznam &s);

T vyjmiPrvni(Seznam &s);
void smazPrvni(Seznam &s);

T vyjmiPosledni(Seznam &s);
void smazPosledni(Seznam &s);
```

Spojový seznam ... funkce main()

```
/* pro seznam prvku typu int */
int main()
{
    Seznam s;
    vytvor(s);

    vlozNaZacatek(s, 4); vlozNaZacatek(s, 5);
    vypis(s);

    smazVsechny(s);
    if (jePrazdny(s))
        cout << "seznam je prazdny\n";

    vlozNaKonec(s, 5); vlozNaKonec(s, 3);
    vlozNaZacatek(s, 8); vlozNaZacatek(s, 2);
    vypis(s);
    if(!jePrazdny(s))
        cout << "mazu " << vyjmiPosledni(s);
    if(!jePrazdny(s))
        cout << "mazu " << vyjmiPrvni(s);
    zrus(s);
    return 0;
```

Spojový seznam ... vytvor a zrus

```
// vytvor "prazdny" seznam (= seznam s jedinnym prvkem, zarazkou)
void vytvor(Seznam &s)
{
    // 1. vytvor novy prvek (pomoci operatoru new)
    //    a jeho adresu vloz do ,hlavy'
    // 2. do ,zarazky' vloz jeho adresu taktez
    // 2. nastav naslednika noveho prvku (tj. zarazky) na nullptr
}
// smaz vsechny prvky seznamu krome zarazky
void smazVsechny(Seznam &s)
{
    // 1. dokud seznam obsahuje krome zarazky i dalsi prvky:
    // 1.1. adresu prvniho prvku uloz do pomocneho ukazatele
    // 1.2. posun hlavu na dalsi prvek
    // 1.3. puvodni prvni prvek zrus (pomoci operatoru delete)
}
void zrus(Seznam &s)
{
   // 1. smaz prvky seznamu (zavolej predchozi funkci)
   // 2. zrus prvek, na ktery ukazuje zarazka (delete)
   // 3. nastav hlavu i zarazku na nullptr
}
```

Spojový seznam

```
// seznam je prazdny, pokud obsahuje prave jeden prvek  (zarazku)
bool jePrazdny(Seznam &s)
{ ... }

void vlozNaZacatek(Seznam &s, T co)
{
    // 1. vytvor novy prvek (pomoci operatoru new)
    // 2. vloz do nej data (co)
    // 3. jako jeho naslednika nastav prvni prvek seznamu
    // 4. do hlavy seznamu vloz adresu nove vytvoreneho prvku
}

void vlozNaKonec(Seznam &s, T co)
{
    // 1. data (co) vloz do aktualni zarazky
    // 1. vytvor novy prvek jako naslednika zarazky
    //    (bude to nova zarazka)
    // 3. do ukazatele ,zarazka' vloz adresu nove vytvoreneho prvku
    // 4. nastav naslednika nove zarazky na nullptr
}
```

Spojový seznam

```
void vypis(int x) // pro int
{
    cout << x << " ";
}
/* pro Zamestnance
void vypis(const Zamestnanec&z)
{
    cout << z.ID << " " << z.jmeno << " " << z.prijmeni
        << " " << z.plat << endl;
}
*/
void vypis(Seznam &s)
{
    // 1. adresu prvniho prvku vloz do pomocneho ukazatele
    // 1. dokud se s timto ukazatelem nedostanes k zarazce:
    // 1.1. vypis obsah prvku (zavolej pomocnou funkci)
    // 1.2. posun pomocny ukazatel na dalsi prvek
}
```

Spojový seznam ... vyjmutí / smazání prvního prvku

```
T vyjmiPrvni(Seznam &s)
{
    // 1. over, ze seznam neni prazdny
    //      (pokud je, hod vyjimku nebo vrat spec. hodnotu)
    // 2. adresu prvniho prvku vloz do pomocneho ukazatele
    // 3. posun ukazatel hlava na nasledujici prvek
    // 4. data z puvodni hlavy uloz do pomocne promenne
    // 5. puvodni hlavu odstran (pomoci operatoru delete)
    // 6. vrat data
}

void smazPrvni(Seznam &s)
{
    // ...
}
```

Spojový seznam ... vyjmutí / smazání posledního prvku

```
T vyjmiPosledni(Seznam &s)
{
    // 1. over, ze seznam neni prazdny
    //     (pokud je, hod vyjimku nebo vrat spec. hodnotu)
    // 2. adresu prvniho prvku vloz do pomocneho ukazatele
    // 3. dokud naslednikem aktualniho prvku neni zarazka:
    // 3.1. posun pomocny ukazatel na dalsi prvek
    // 4. puvodni zarazku odstran (pomoci operatoru delete)
    // 5. do zarazky vloz adresu akt. prvku
    //   (je v pomocnem ukazateli)
    // 6. nastav nasledika akt. prvku na nullptr
    // 7. vrat data z akt. prvku
}

void smazPosledni(Seznam &s)
{
    // ...
}
```