Procedurálne programovanie

slido.com
2726923
PrPr – P9

Ján Zelenka

Ústav Informatiky

Slovenská akadémia vied

Oznamy

slido slido.com
2726923
PrPr – P9

Termín odovzdania druhého projektu (Spájaný zoznam štruktúr): odovzdanie v 11. týždni (3.12. do 23:59)

- neskoré odovzdanie 12. týždeň (10.12.do 23:59), penalizácia - uznáva sa 80% zo získaného počtu bodov
- za projekt musí získať študent **min. 6 bodov** (akceptovateľný),bez bodov za prezentáciu (2 bodov)

IDstudenta_Rok_projekt_2.c

Obsah prednášky

slido slido.com
2726923
PrPr – P9

1. **Opakovanie**
2. **Spájaný zoznam**

Spätná väzba: *https://forms.gle/6q5D2G6UwrtimXEx9*

Štruktúry

slido.com
2726923
PrPr – P9

Štruktúry a ukazovatele

```
typedef struct {
    char meno[30];
    int rocnik;
} STUDENT;
```

```
typedef struct {
    char meno[30];
    int rocnik;
} STUDENT, *P_STUDENT;
```

definícia typu ukzovateľa na štruktúru

```
STUDENT s;
P_STUDENT  p_s;
p_s = (P_STUDENT) malloc(sizeof(STUDENT));
```

```
s.rocnik = 2;
(*p_s).rocnik = 3;
p_s->rocnik = 4;
```

prístup k štruktúre pomocou ukazovateľa

Štruktúra v inej štruktúre

- štruktúra, ktorá je v inej štruktúre musí byť definovaná skôr ako je do inej štruktúry uložená (vhniezdená štruktúra)
- predtým sme mali len odkaz na štruktúru

```
typedef struct {
    char ulica[30];
    int cislo;
} ADRESA;
```

```
typedef struct  {
    char meno[30];
    ADRESA adresa;
    float plat;
} OSOBA;
```

Spájaný zoznam

slido.com
2726923
PrPr – P9

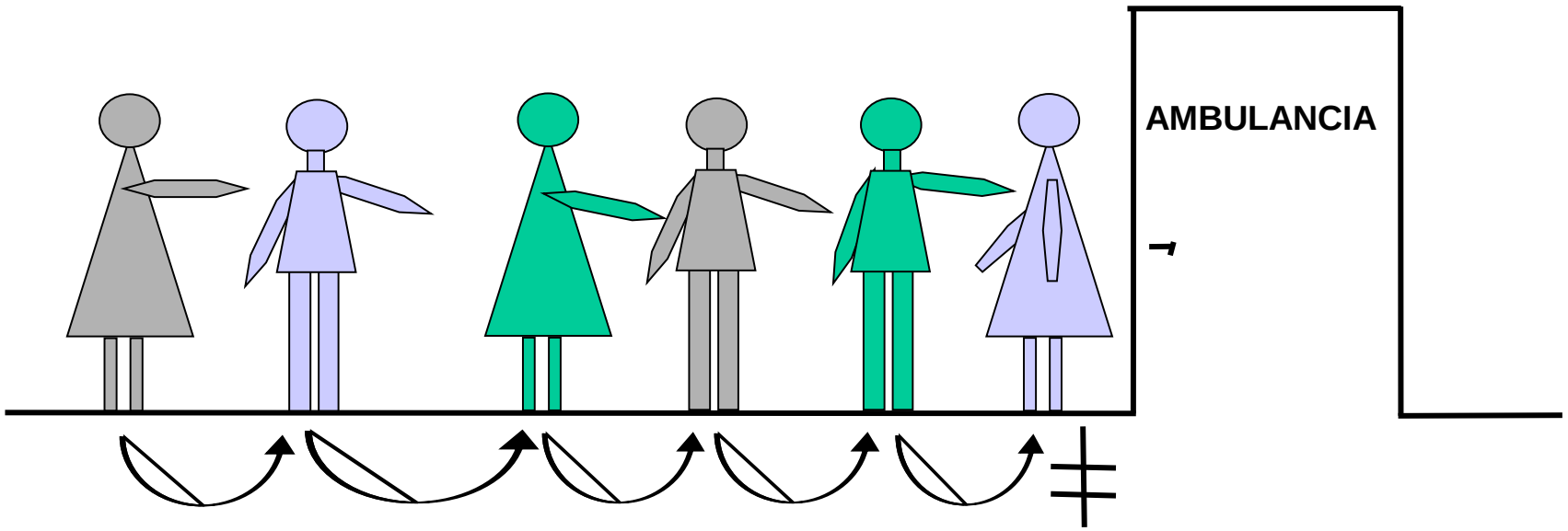
Štruktúry ukazujúce samy na seba

slido.com
2726923
PrPr – P9

Pacienti v čakárni u lekára

"Kto je posledný?"

každý človek si pamätá človeka, ktorý je pred ním (ukazovateľ na ten istý typ)



Štruktúry ukazujúce samy na seba

slido slido.com
2726923
PrPr – P9

```
typedef struct prvok {
    int hodnota;
    struct prvok *p_dalsi;
} PRVOK;
```

odkaz na samého seba (na takú istú štruktúru)

aj štruktúra, aj typ musia byť pomenované

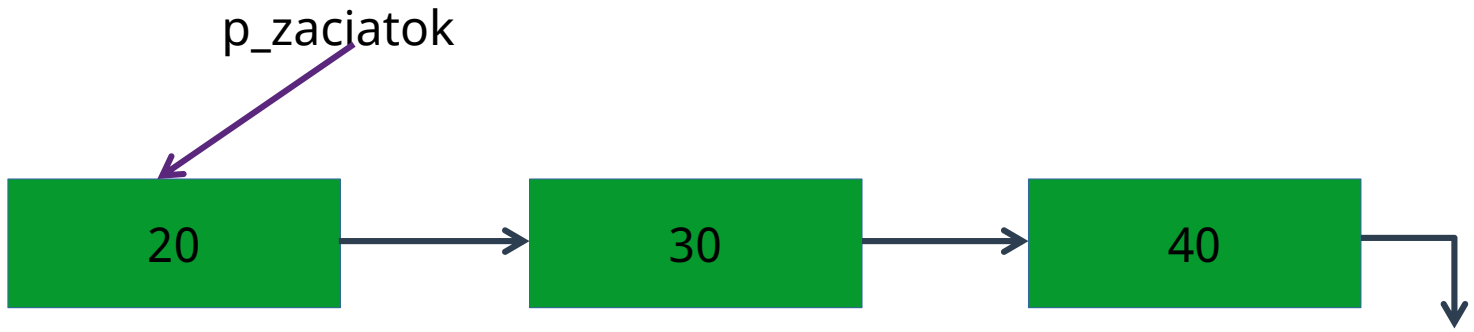
```
typedef struct {
    int hodnota;
    struct PRVOK *p_dalsi;
} PRVOK;
```

chyba: v čase, keď sa definuje **p_dalsi**, položka **PRVOK** ešte nie je známa

Spájaný zoznam

```
typedef struct prvok {
    int hodnota;
    struct prvok *p_dalsi;
} PRVOK;
```

- postupnosť hodnôt
- reprezentovaný ukazovateľom na prvý prvok
- posledný prvok nemá nasledovníka
- ukazovateľ **`p_dalsi`** je nastavený na špeciálnu hodnotu, zvyčajne **`NULL`**



Spájaný zoznam

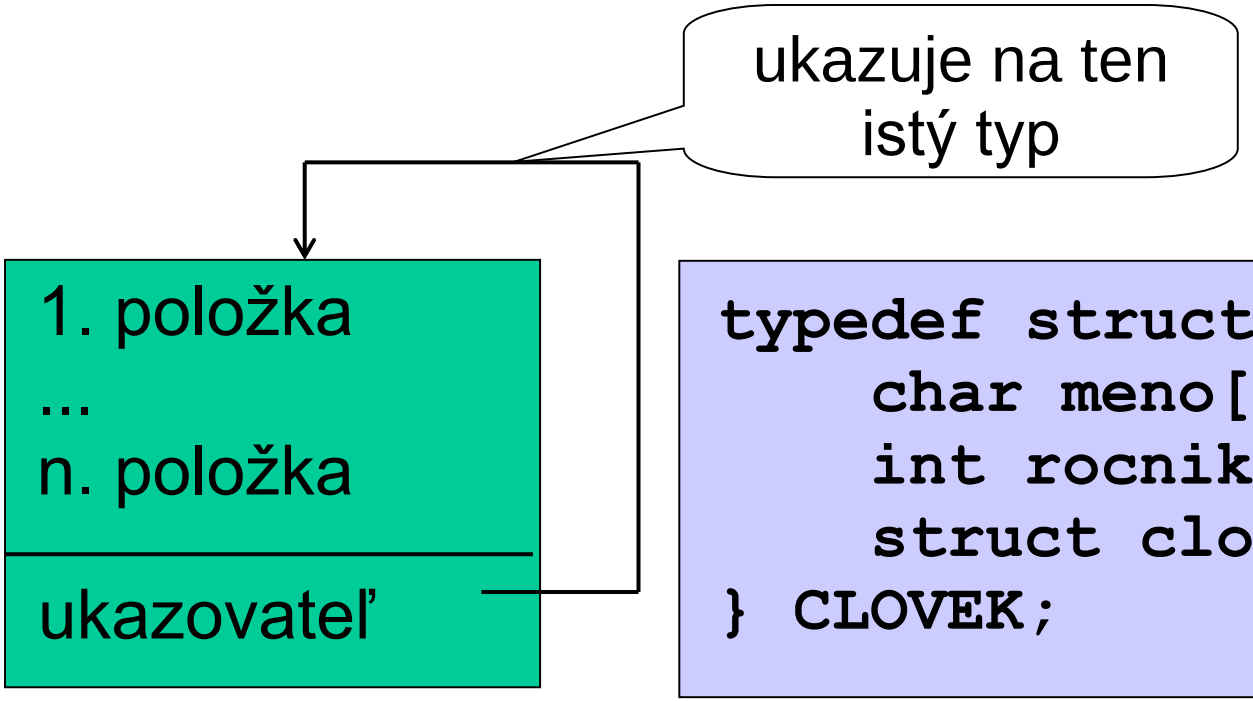
slido slido.com
2726923
PrPr – P9

- Lineárna dátová štruktúra (ako pole)
  - nemá vetvenie (napr. viacero nasledovníkov) ako graf, strom…
  - jeho špeciálnym prípadom je front a zásobník
    - pracujeme iba s koncom poprípade začiatkom
- Veľké dáta s premenlivou dĺžkou
- Často meniace sa dáta
  - priebežne vkladáme a mažeme prvky
- "Hodnota" môže byť zložitejšia
  - struct student
- Prvky nie sú uložené bezprostredne za sebou v pamäti
  - prepojené sú pomocou ukazovateľov

Spájaný zoznam

Dynamický zoznam prvkov:

- v pamäti je práve toľko prvkov, koľko je potreba
- dá sa pridávať na ktorékoľvek miesto v zozname




```
typedef struct clovek {
    char meno[30];
    int rocnik;
    struct clovek *dalsi;
} CLOVEK;
```

Spájaný zoznam

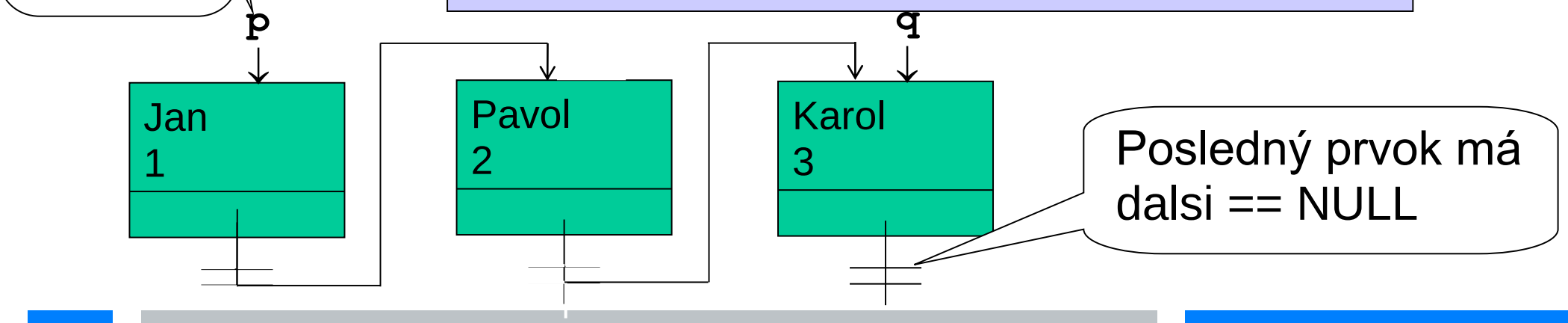
```
typedef struct clovek {
    char meno[30];
    int rocnik;
    struct clovek *dalsi;
} CLOVEK;

CLOVEK *p, *q;
```

dáta

ukazovateľ na nasledujúci prvok

```
q = (CLOVEK *) malloc(sizeof(CLOVEK));
q->meno = Karol;
q->rocnik =  3;
q->dalsi = NULL;

p->dalsi->dalsi = q;
```

musíme si zapamätať začiatok zoznamu



Prístup k prvkom

- Pamätáme si (cez ukazovateľ) začiatok spájaného zoznamu
  - k ostatným položkám sa dostaneme z neho
- V mnohých prípadoch si pamätáme aj koniec spájaného zoznamu
  - vloženie prvku na koniec zoznamu je častá operácia
- Prechádzame zoznam pomocou cyklu
  - zvyčajne používame while cyklus
  - nemeniť `p_začiatok`

```
p_akt = p_zaciatok;
while(p_akt != NULL){
    printf("%d",p_akt->hodnota);
    p_akt = p_akt->dalsi;}
```

Spájaný zoznam a funkcie

slido slido.com
2726923
PrPr – P9

Ak vo funkcii meníme zoznam (vkladáme, mažeme, preusporadúvame), môže sa stať, že:

- Pridávame na začiatok zoznamu
- Zmazávame zo začiatku zoznamu
- Zmazávame celý zoznam

- Je potrebné vrátiť ukazovateľ na začiatok zoznamu

```
CLOVEK *zmena(CLOVEK *p_prvy, ...) {
    ...
    return prvy;
}
```

Návratová hodnota

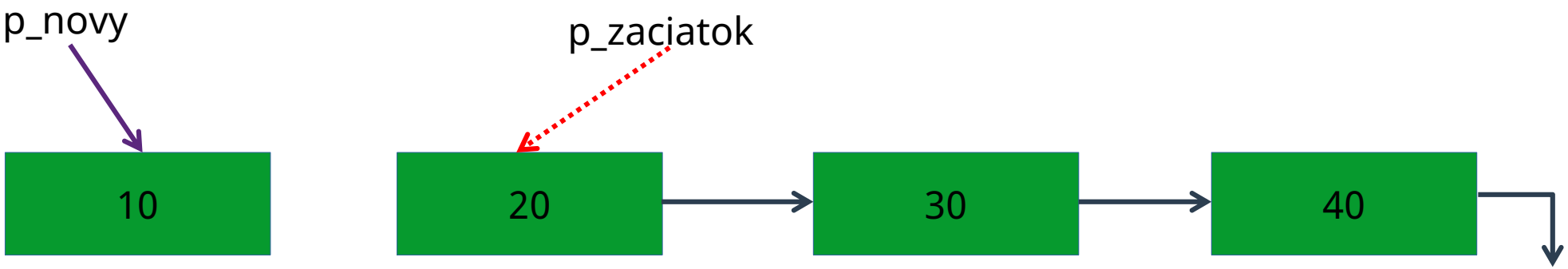
```
void zmena(CLOVEK **p_prvy, ...) {
    ...
}
```

Pomocou argumentov funkcie (ukazovateľ na ukazovateľ)

Pridanie prvku na začiatok zoznamu


```
typedef struct prvok {
    int data;
    struct prvok* dalsi;
} PRVOK;
```

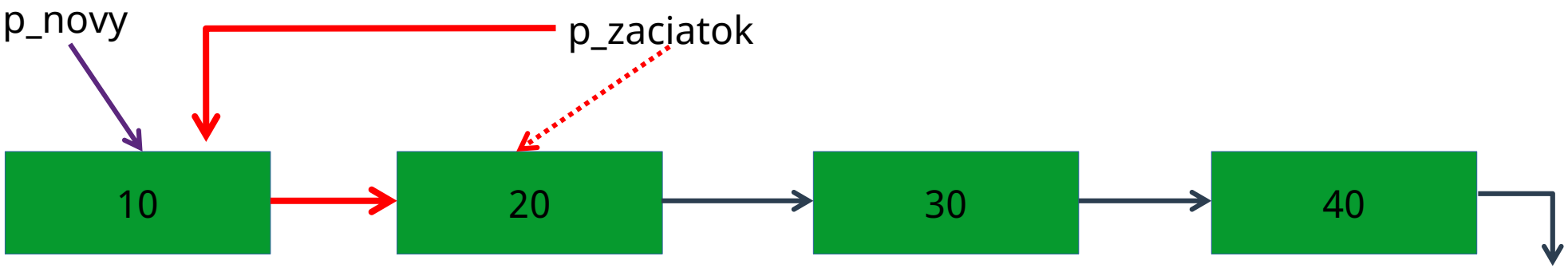
Vloženie prvku 10 na začiatok zoznamu:



Pridanie prvku na začiatok zoznamu

```
typedef struct prvok {
    int data;
    struct prvok* dalsi;
} PRVOK;
```

Vloženie prvku 10 na začiatok zoznamu:



Pridanie prvku na začiatok zoznamu

slido slido.com
2726923
PrPr – P9

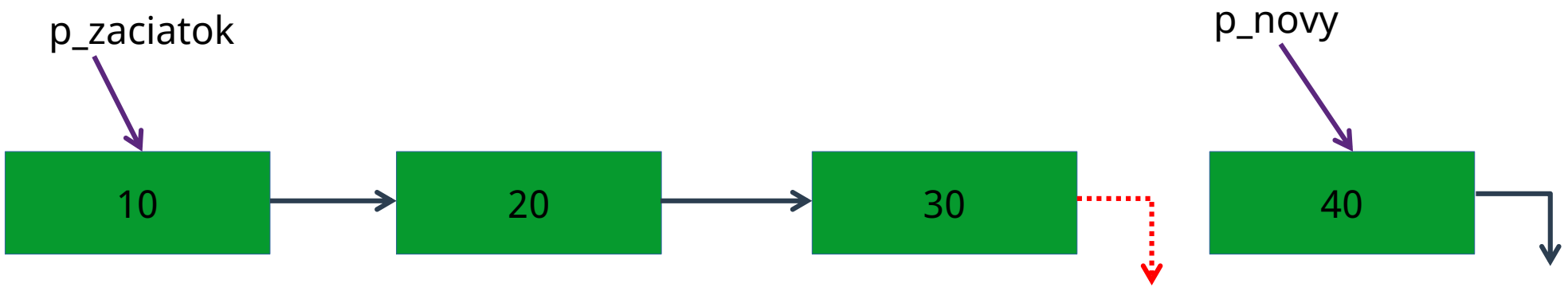
1. vytvoríme prvok so vstupnými dátami
2. nasledovníka nového prvku nastavíme na začiatok zoznamu
3. začiatok zoznamu nastavíme na nový prvok

```
void priadaj_na_zaciatok(PRVOK** p_p_zaciatok, int hodnota)
{
    PRVOK* novy_prvok = (PRVOK*) malloc(sizeof(PRVOK));
    novy_prvok->data  = hodnota;
    novy_prvok->dalsi = (*p_p_zaciatok);
    (*p_p_zaciatok)    = novy_prvok;
}
```

Pridanie prvku na koniec zoznamu

Vloženie prvku 40
na koniec zoznamu:

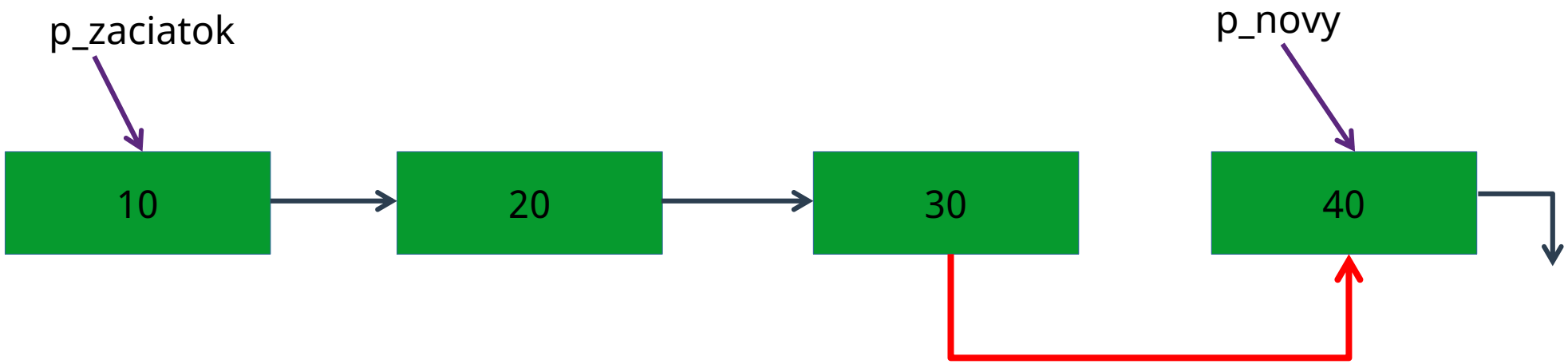
```
typedef struct prvok {
    int data;
    struct prvok* dalsi;
} PRVOK;
```



Pridanie prvku na koniec zoznamu

Vloženie prvku 40
na koniec zoznamu:

```
typedef struct prvok {
    int data;
    struct prvok* dalsi;
} PRVOK;
```



Pridanie prvku na koniec zoznamu

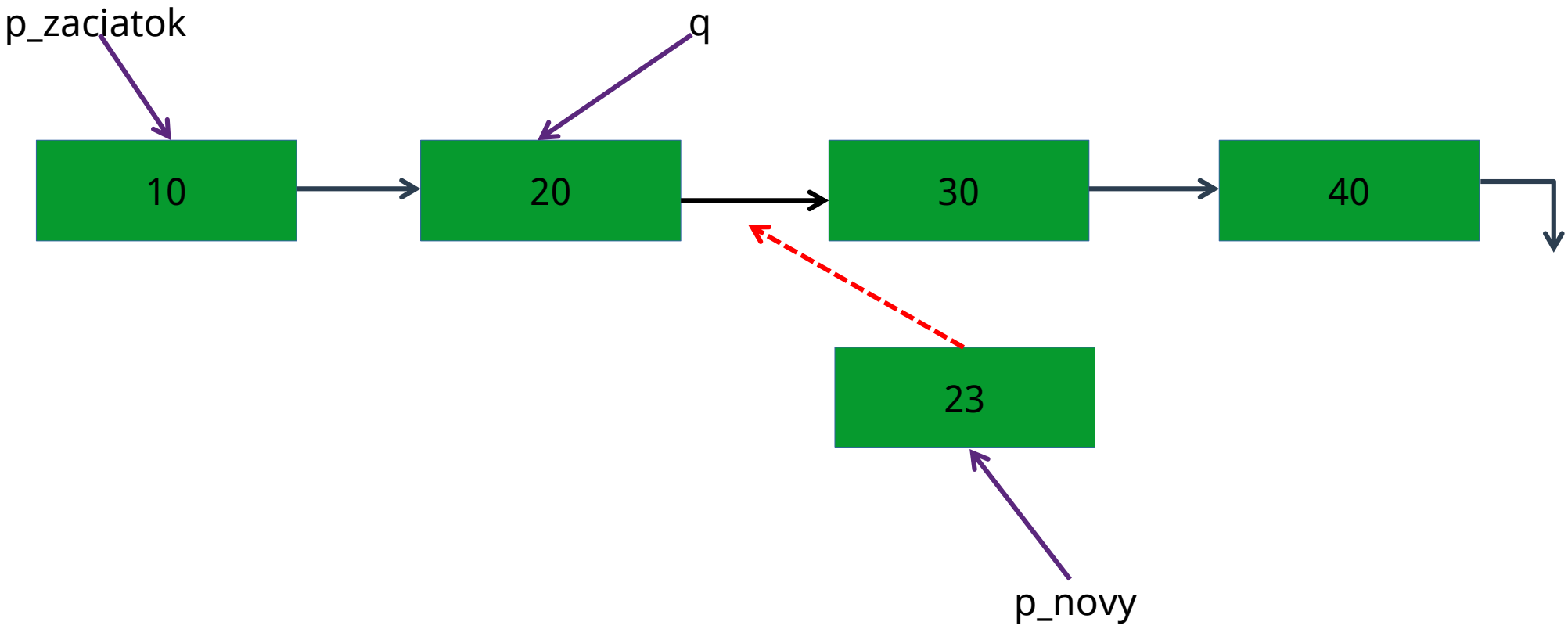
1. vytvoríme prvok so vstupnými dátami
2. nasledovníka nového prvku nastavíme na NULL
3. nájdeme posledný prvok
4. nasledovník posledného prvku bude nový prvok

```
void priadaj_na_koniec(PRVOK** p_p_zaciatok, int hodnota) {
    PRVOK *novy_prvok = (PRVOK*) malloc(sizeof(PRVOK));
    PRVOK *posledny = *p_p_zaciatok;

    novy_prvok->data  = hodnota;
    novy_prvok->dalsi = NULL;

    if (*p_p_zaciatok == NULL) {
       *p_p_zaciatok = novy_prvok;
       return;}

    while (posledny->dalsi != NULL)
        posledny = posledny->dalsi;

    posledny->dalsi = novy_prvok;
    return;}
```

čo ak je spájaný zoznam prázdny?

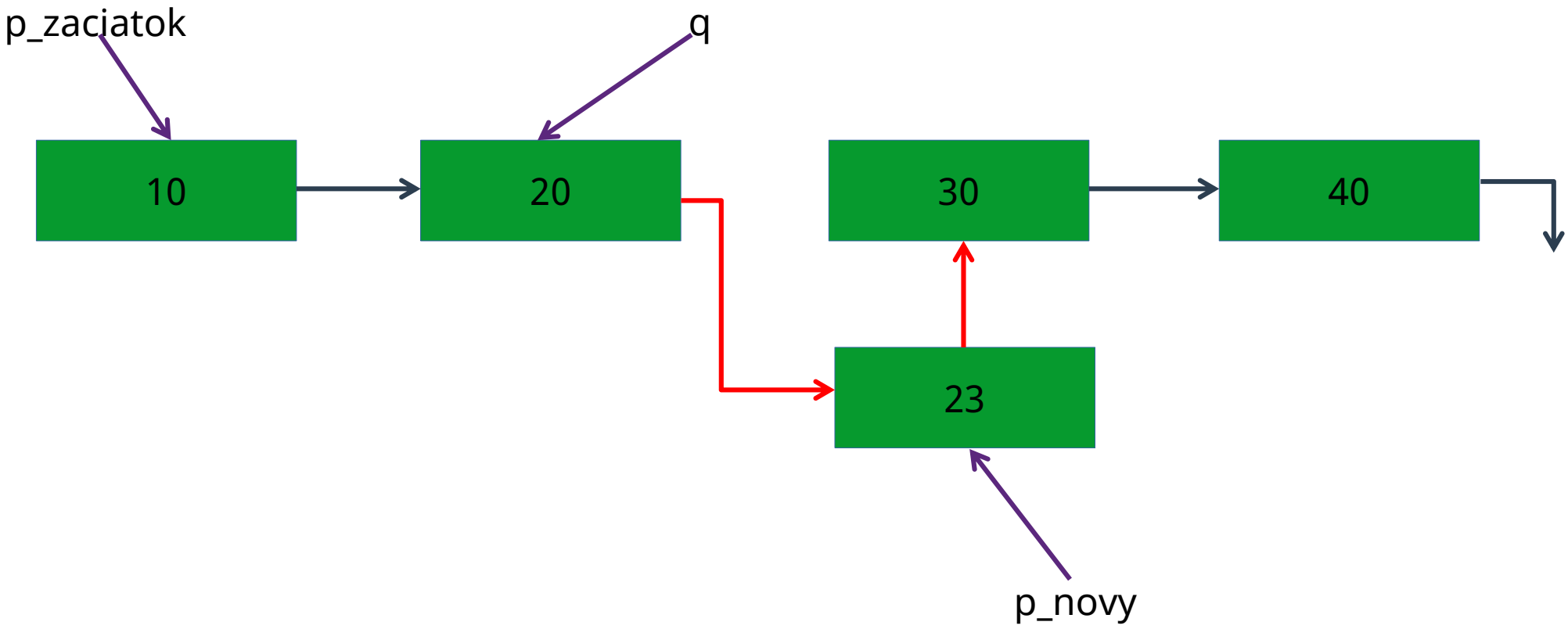
Pridanie prvku do zoznamu

Vloženie prvku 23 za prvok 20:



Pridanie prvku do zoznamu

Vloženie prvku 23 za prvok 20:



Pridanie prvku do zoznamu

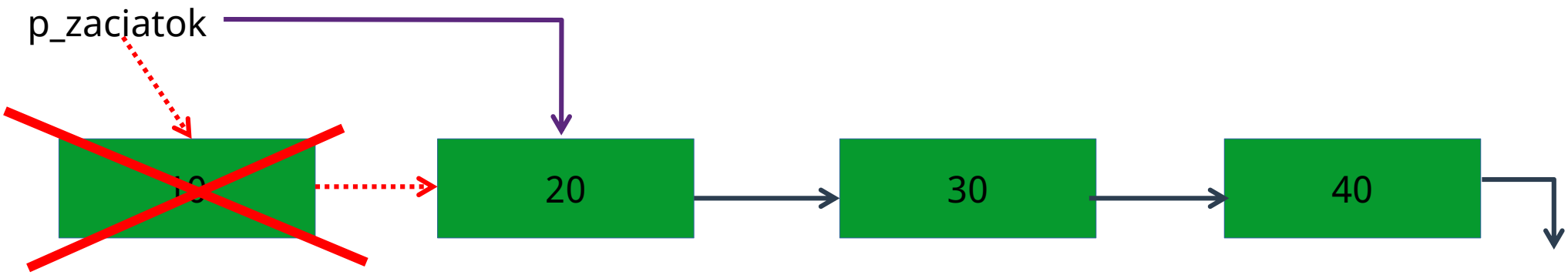
Pridanie nového prvku za prvok zo zoznamu:

1. vytvoríme prvok so vstupnými dátami
2. nájdeme prvok v zozname
3. nasledovníka nového prvku nastavíme na nasledovníka nájdeného prvku zo zoznamu
4. nasledovníka nájdeného prvku zoznamu nastavíme na nový prvok

Odstránenie prvého prvku zo zoznamu

slido.com
2726923
PrPr – P9

Odstránenie prvku 10:



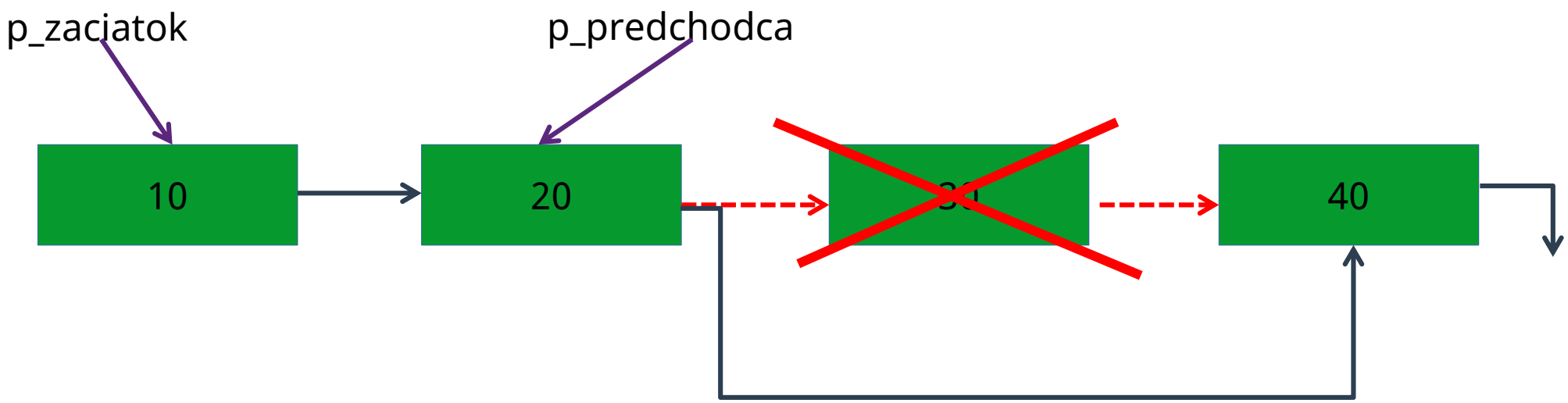
Odstránenie prvého prvku zo zoznamu

1. zapamätáme si prvý prvok zoznamu v pomocnej premennej
2. posunieme začiatok zoznamu na jeho nasledovníka
3. uvoľníme pamäť vyhradenú pre pôvodný prvý prvok zoznamu

```
void zmaz_zaciatok(PRVOK **p_p_zaciatok) {
    PRVOK *p_akt;

    if(p_p_zaciatok == NULL || *p_p_zaciatok == NULL)
        return;
    p_akt = *p_p_zaciatok;
    *p_p_zaciatok = (*p_p_zaciatok)->dalsi;
    free(p_akt);}
```

Odstránie prvku do zoznamu

Odstránenie prvku 30:



Odstránenie prvku zo zoznamu

slido slido.com
2726923
PrPr – P9

1. nájdeme prvok, ktorý v zozname predchádza označenému prvku
2. v ňom zmeníme hodnotu ukazovateľa **`dalsi`**
3. označený prvok zrušíme (uvoľníme pamäť, ktorá mu bola pridelená)

Spájaný zoznam a dynamické pole

slido.com
2726923
PrPr – P9

Spájaný zoznam	Dynamické pole
Dynamická veľkosť	Zväčšovanie je náročné
Vkladanie a mazanie je efektívne	Vkladanie a mazanie je neefektívne (zvyčajne treba posunúť prvky)
Nie je vhodný pre pristupovanie k prvkom podľa indexu (napr. triedenie)	**Prístup na i-ty prvok**
Pamäť je alokovaná dynamicky podľa potreby	Plytvanie pamäťou pri poloprázdnom poli

Výhody spájaného zoznamu

slido slido.com
2726923
PrPr – P9

- Dynamická veľkosť/štruktúra
- Ľahké pridávanie a mazanie prvkov
  - vloženie nového prvku do poľa je drahé, pretože musíme vytvoriť miesto pre nový prvok, to znamená, že musíme posunúť existujúce prvky
    Napríklad majme pole, kde máme usporiadané ID-čka používateľov
    `id = [12, 28, 34, 50, 78, 91]`
    Ak chceme vložiť nového používateľa s ID 47 a chceme mať usporiadané pole, musíme posunúť všetky prvky za 34
- Mazanie je tiež drahé pri poliach
  -> čo treba spraviť pri mazaní prvku 28?

Nevýhody spájaného zoznamu

slido slido.com
2726923
PrPr – P9

- prístup k i-temu prvku je drahý
  - potrebujeme prejsť sekvenčne cez všetky prvky zoznamu od jeho začiatku (pokročilé vyhľadávacie techniky napr. binárne vyhľadávanie sú časovo náročné, to isté triedenie)
- potrebujeme si navyše pamätať ukazovateľ pri každom prvku zoznamu

Rodostrom

slido slido.com
2726923
PrPr – P9

- Záznam o človeku:
  - Meno
  - Priezvisko
  - Ukazovateľ na záznam o matke
  - Ukazovateľ na záznam o otcovi
  - Počet detí
  - Pole ukazovateľov na záznamy o deťoch

```
typedef struct clovek {
    char meno[20];
    char priezvisko[20];
    struct clovek *matka;
    struct clovek *otec;
    int pocet_deti;
    struct clovek **deti;
} CLOVEK;
```

Štruktúra spájaný zoznam

```
typedef struct prvok {
    int data;
    struct prvok* dalsi;
} PRVOK;
```

- Musíme udržiavať ukazovateľ na prvý prvok
- Môžeme vytvoriť dodatočnú štruktúru `zoznam`, ktorá bude obsahovať ukazovateľ na začiatok/koniec zoznamu
- Argumentom funkcií posielame ukazovateľ na `zoznam` (namiesto ukazovateľa na prvý prvok)
- Do štruktúry `zoznam` si môžeme pridať dodatočné informácie (napr. veľkosť zoznamu)

```
typedef struct zoznam {
    PRVOK *prvy;
    PRVOK *posledny;
} ZOZNAM;
```

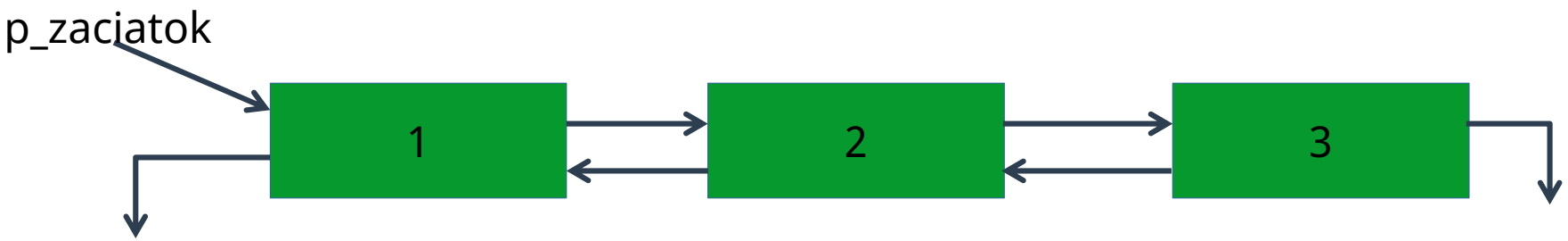
Zhrnutie

slido slido.com
2726923
PrPr – P9

- spájaný zoznam sa skladá z prvkov, ktoré sú prepojené ukazovateľmi
- celý zoznam je dostupný cez ukazovateľ na prvý prvok
- každý prvok zoznamu, okrem posledného má jediného nasledovníka (ukazovateľ `dalsi`)

Obojsmerný spájaný zoznam

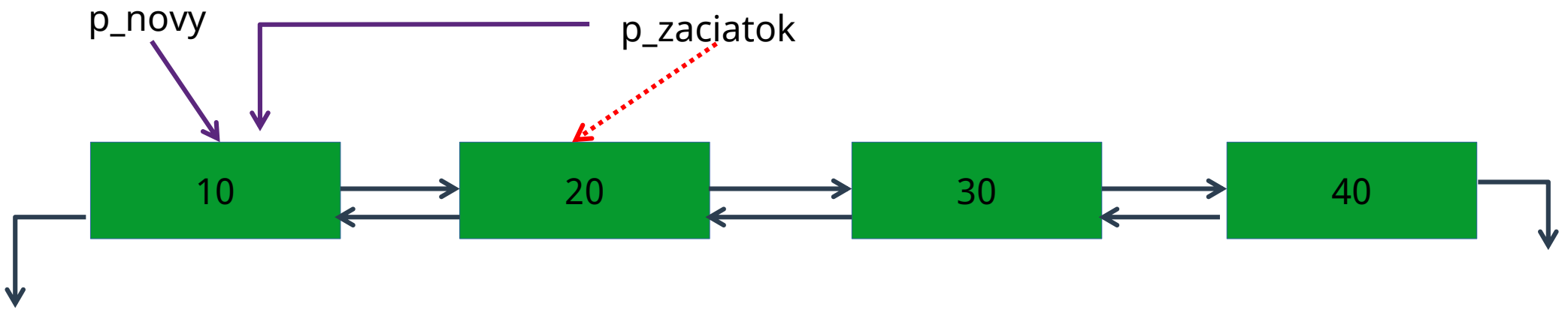
- obsahuje ukazovateľ na nasledovníka a aj na predchodcu
- môžeme ho prechádzať oboma smermi



Pridanie prvku na začiatok zoznamu

Vloženie prvku 10 na začiatok zoznamu:

```
typedef struct prvok {
    int data;

    struct prvok *dalsi;
    struct prvok *pred;
} PRVOK;
```



Pridanie prvku na začiatok zoznamu

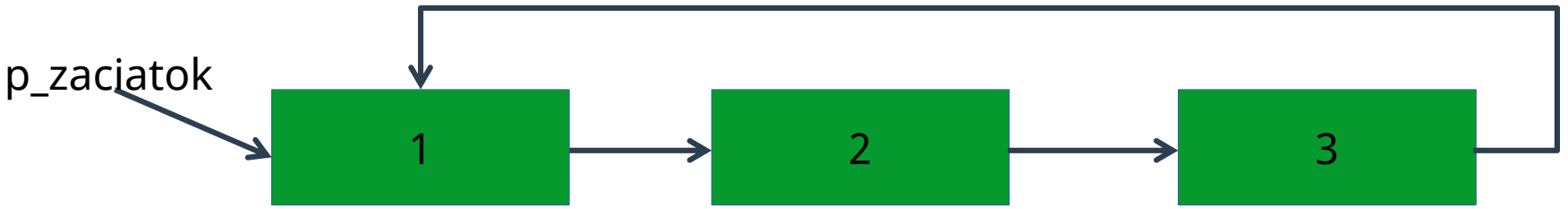
1. vytvoríme prvok so vstupnými dátami
2. nasledovníka nového prvku nastavíme na začiatok zoznamu
3. predchodcu nového prvku nastavíme na NULL
4. začiatok zoznamu nastavíme na nový prvok

```
void priadaj_na_zaciatok(PRVOK** p_p_zaciatok, int hodnota) {
    PRVOK* novy_prvok = (PRVOK*) malloc(sizeof(PRVOK));
    novy_prvok->data  = hodnota;
    novy_prvok->dalsi = (*p_p_zaciatok);
    novy_prvok->pred = NULL;

    if ((*p_p_zaciatok) != NULL)
        (*p_p_zaciatok)->pred = novy_prvok;

    (*p_p_zaciatok) = novy_prvok;}
```

Kruhový spájaný zoznam

- ukazovateľ na nasledovníka posledného prvku ukazuje na prvý prvok zoznamu
- prechádzanie zoznamu môžeme začať v ľubovoľnom prvku (celý zoznam sme spracovali vtedy, ak sme druhýkrát navštívili prvý/vstupný prvok zoznamu)

```
typedef struct prvok {
    int hodnota;

    struct polozka *p_dalsi;
} PRVOK;
```



Hľadanie chýb

slido.com
2726923
PrPr – P9

Typické chyby

slido slido.com
2726923
PrPr – P9

- Nekorektné prepojenie všetkých ukazovateľov
  - treba aktualizovať ukazovatele troch prvkov (vľavo, aktuálny, vpravo)
- Chybné prepojenie počas manipulácie s prvým alebo posledným prvkom
  - snažíme sa pristupovať na NULL
  - poškodenie alebo neuloženie aktuálneho ukazovateľa ako prvý alebo posledný prvok
- Prepísanie aktuálneho ukazovateľa adresou nového prvku
  - stratíme aktuálny ukazovateľ (memory leak)
- Nekorektné uvoľnenie pamäte

Riešenia

slido slido.com
2726923
PrPr – P9

- nakreslenie štruktúry na papier
  - http://pythontutor.com/c.html#mode=display
- výpis obsahu štruktúry/prvku
  - skontrolujte adresu a obsah
  - podmienené výpisy
- kontrola základných parametrov zoznamu (počet prvkov, NULL na konci, korektný prvý prvok…)
- krokovanie programu
  - niektoré vývojové prostredia umožňujú postupne prechádzať prvky zoznamu (klikanie na "next")
  - podmienený breakpoint (napr. veľkých zoznamoch)
- valgrind (memory leaks, neinicializované premenné, zápis mimo alokovanej pamäti...)

Sledovanie pamäti

slido slido.com
2726923
PrPr – P9

- Visual Studio 2010
  - Debug→Windows→Memory
- Code::Blocks
  - Debug→Debugging windows→Examine memory
- QTCreator
  - Windows→View→Memory
- Pamäťový breakpoint (memory breakpoint)
  - nepodporujú ho všetky vývojové prostredia (napr. VS2010)
  - podmienený beakpoint na zmenu pamäte (nutná podpora v CPU)
  - Debug→Breakpoint→New data breakpoint
- Kontrola zle uvoľnenej pamäte

Hodnoty ukazovateľov

slido slido.com
2726923
PrPr – P9

- Debugger zobrazuje aj hodnoty ukazovateľov (t.j. adresy)
  - môžu sa líšiť medzi spusteniami toho istého programu
  - prvá vložné položka nemusí byť na tom istom mieste v halde
- Ukazovateľ na nasledujúcu položku je nastavený na korektnú adresu
  - ak nevidíte adresu, získate ju pomocou operátora &
- V debuggovacom režime môže prekladač nastaviť neinicializované hodnoty ukazovateľov na špeciálne hodnoty
  - napr. **`0xBaadF00d`**
  - IBA v debug režime
    - pri Release neporiadok z pamäte

Ďakujem vám za pozornosť!

Spätná väzba: *https://forms.gle/6q5D2G6UwrtimXEx9*