Typ ukazatel (1)

- Datový typ umožňující vytváření a zpřístupnění tzv. dynamických proměnných
- Dynamické proměnné - proměnné jejichž počet a (nebo) velikost paměti využívané těmito proměnnými se v průběhu programu mění
- Ukazatel (pointer) je datový typ jehož hodnoty „ukazují“ na (obvykle) dynamické proměnné

Typ ukazatel (2)

- Přístup k dynamickým proměnným se provádí zprostředkovaně přes hodnotu typu ukazatel
- Proměnná typu ukazatel obsahuje adresu paměti, kde se nachází hodnota dynamické proměnné

Typ ukazatel (3)

- Zprostředkovaný přístup k dynamickým proměnným tedy spočívá v tom, že je nezbytné se na ně odkazovat nepřímo, a to identifikací proměnné P typu ukazatel, jejíž hodnota právě ukazuje na zpřístupňovanou dynamickou proměnnou
- Na jednu dynamickou proměnnou může obecně ukazovat více ukazatelů, pomocí nichž je možné provést její zpřístupnění

Typ ukazatel (4)

- Datový typ ukazatel na dynamickou proměnnou jistého (bázového) typu T se definuje následujícím zápisem:

  type U = ^T;

  kde U je identifikátor typu ukazatel
  T je identifikátor bázového typu

- Proměnná typu ukazatel bude ukazovat vždy jen na proměnnou takového datového typu, jenž je jako bázový typ uveden v definici typu ukazatel

Typ ukazatel (5)

- Typ ukazatel je tedy svázán s určitým bázovým typem
- Příklad:

```
type PInteger = ^Integer;
var   PInt: PInteger;
```

- PInt je nyní proměnná, která ukazuje na dynamické proměnné typu integer
- Pozor!
  Touto deklarací ještě nevzniká vlastní dynamická proměnná

Typ ukazatel (6)

- Dynamické proměnné mohou vznikat a zanikat v průběhu činnosti programu, což je zabezpečeno pomocí dvou standardních procedur:
  - new (P):
    - vytvoří dynamickou proměnnou bázového typu (alokuje pro ni v paměti místo) s níž je svázána proměnná P typu ukazatel
    - proměnné P se přiřadí hodnota (adresa paměti), která ukazuje na nově vzniklou dynamickou proměnnou

Typ ukazatel (7)

- dispose (P):
  - zruší dynamickou proměnnou (uvolní dříve alokované paměťové místo), na kterou ukazuje ukazatel P
  - po provedení procedury dispose je hodnota ukazatele P a všech ukazatelů ukazujících na rušenou dynamickou proměnnou nedefinována

- Příklad:

  new (PInt)

  vytvoří dynamickou proměnnou typu integer a ukazatel na ni vloží do proměnné PInt

Typ ukazatel (8)

- Zpřístupnění dynamické proměnné je možné provést pomocí zápisu V^, kde V je proměnná typu ukazatel
- Například:

```
PInt^ := 0;
PInt^ := PInt^ + 1;
```

- Poznámka:
  - dynamické proměnné nemají vlastní identifikátor a tudíž se ně odkazujeme pomocí ukazatele

Typ ukazatel (9)

- Hodnoty typu ukazatel (téhož bázového typu) je možné porovnávat na rovnost (resp. nerovnost), což lze interpretovat jako test, zda-li dva ukazatele ukazují (resp. neukazují) na stejnou dynamickou proměnnou
- Hodnoty typu ukazatel není možné zapisovat jako konstanty; lze je vytvářet pouze pomocí procedury new

Typ ukazatel (10)

- Jedinou výjimkou je použití konstanty nil, která vyjadřuje skutečnost, že ukazatel ne-ukazuje na žádnou dynamickou proměnnou
- Příklad:

```
PInt := nil;
```

Ukazatel PI nyní neukazuje na žádnou dynamickou proměnnou. Následný odkaz pomocí PI^ povede k chybě.

Typ ukazatel (11)

- Poznámky:
  - v BP (TP) je možné, v případě potřeby, pracovat i s adresou statické proměnné, tj. ukazatelem na statickou proměnnou. Na tuto adresu je možné se odkázat pomocí zápisu @V, kde V značí statickou proměnnou
  - BP (TP) zavádí rovněž datový typ Pointer, který reprezentuje tzv. netypový ukazatel. Jedná se o ukazatel, který ukazuje na blíže nespecifikované paměťové místo

Typ ukazatel (12)

- datový typ Pointer lze využít například při alokaci (dealokaci) paměťových bloků pomocí procedury GetMem (FreeMem)

- Hlavním využitím datového typu ukazatel a dynamických proměnných je tvorba dynamických datových struktur

Dynamické datové struktury (1)

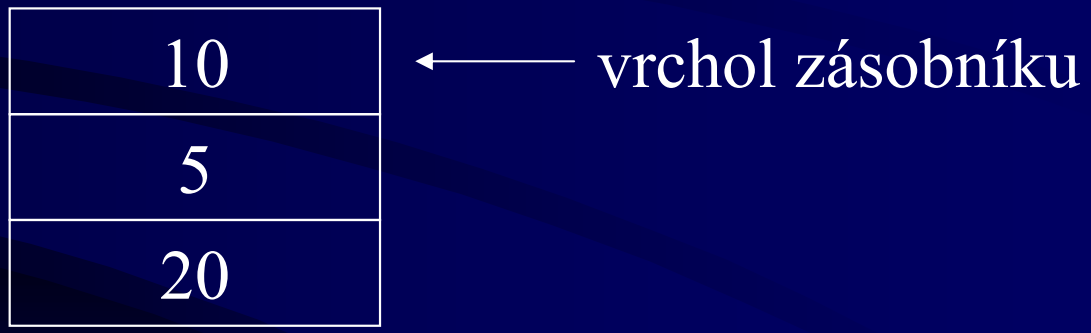
- Datové struktury, jejichž velikost (počet položek) není známa v době překladu programu
- Jednotlivé položky těchto struktur mohou vznikat a zanikat během činnosti programu
- Bývají velmi často realizovány pomocí datového typu record, který obsahuje:
  - část nesoucí vlastní datové informace

Dynamické datové struktury (2)

  - ukazatel(e), dovolující zpřístupnit další položku (resp. položky) dané struktury
- Příklady dynamických datových struktur:
  - zásobník (stack)
  - fronta (queue)
  - lineární seznam (linear list)
  - různé typy stromů
  - obecné grafy

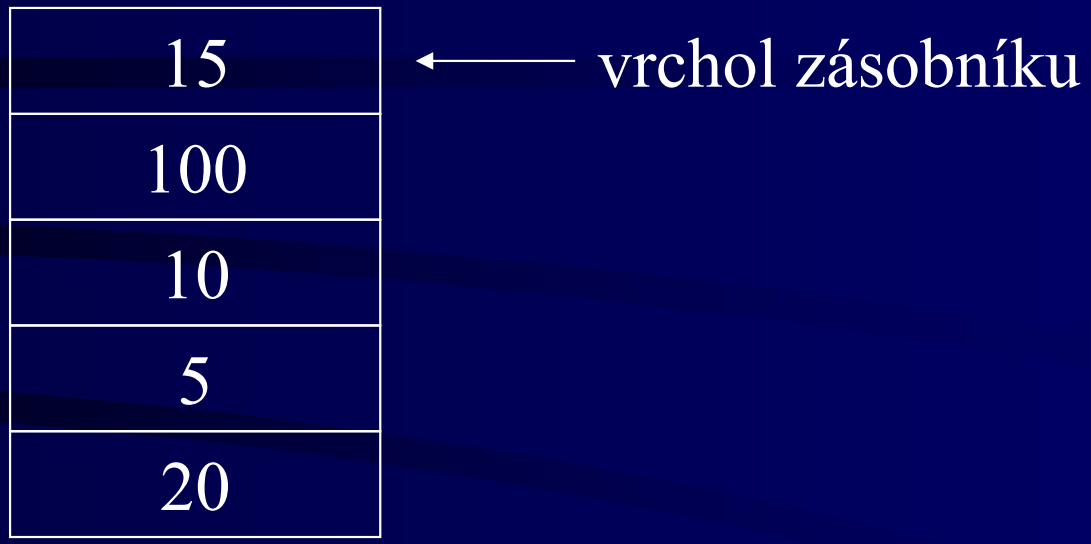
Zásobník (1)

- Dynamická datová struktura typu LIFO (Last In First Out)
- Dovoluje přidávat (odebírat) data pouze na jednom svém místě - vrcholu zásobníku



- Po postupném přidání hodnot 100 a 15 bude zásobník vypadat takto:

Zásobník (2)



- Odebírání hodnot je možné opět pouze z vrcholu zásobníku
- Není možné zpřístupnit jinou hodnotu než tu, která je právě na vrcholu zásobníku

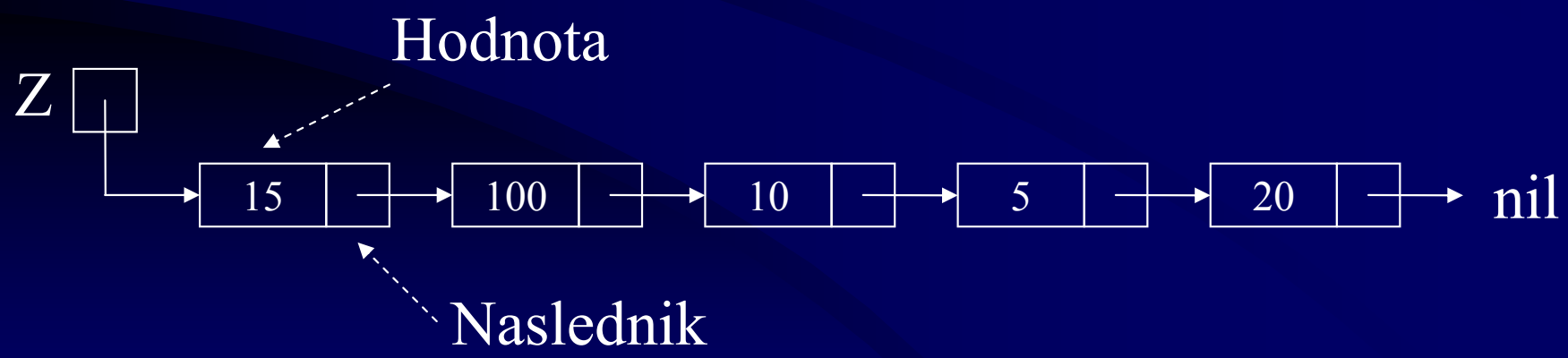
Zásobník (3)

- Se zásobníkem je obecně možné provádět následující operace:
  - init (Z): provede prvotní inicializaci zásobníku Z (před jeho prvním použitím)
  - push (Z, X): uloží hodnotu X na vrchol zásobníku Z
  - pop (Z, X): odebere hodnotu z vrcholu zásobníku Z a vloží ji do proměnné X
  - empty (Z): testuje, zda-li je zásobník Z prázdný či nikoliv

Zásobník (4)

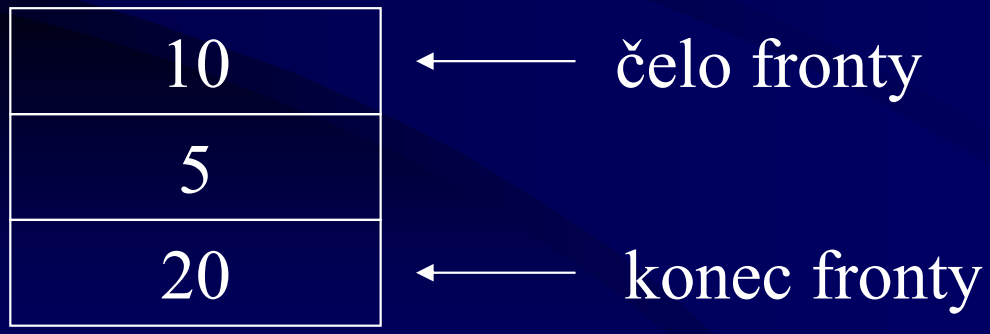
- Implementace zásobníku pomocí dynamických proměnných:

```
type PZasobnik = ^TZasobnik;
     TZasobnik = record
         Hodnota: typ_hodnoty;      (* např. integer *)
         Naslednik: PZasobnik;
     end;
var Z: PZasobnik;
```



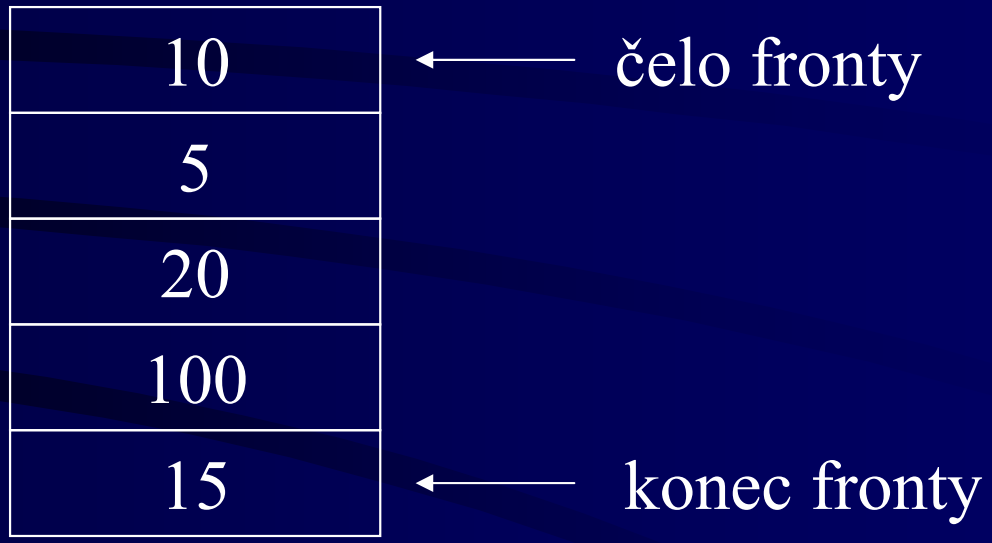
Fronta (1)

- Dynamická datová struktura typu FIFO (First In First Out)
- Dovoluje data:
  - přidávat pouze na jednom svém místě - konci fronty
  - odebírat pouze na jednom svém místě - čele fronty



Fronta (2)

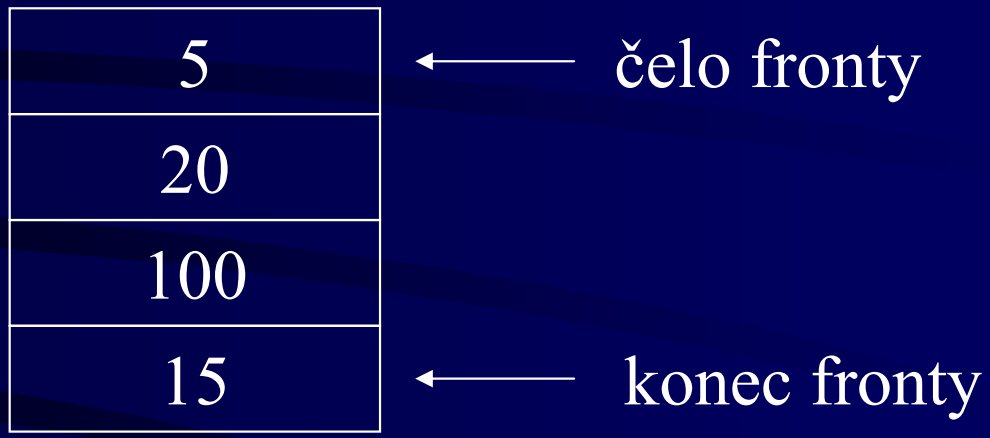
- Po postupném přidání hodnot 100 a 15 bude fronta vypadat takto:



- Odebírání hodnot je možné pouze z čela fronty, tj. první odebíraný prvek bude 10

Fronta (3)

- Stav fronty po odebrání jednoho prvku:



- Podobně jako u zásobníku, tak i u fronty není možné zpřístupnit libovolnou hodnotu, která je v ní uložená

Fronta (4)

- S frontou je obecně možné provádět následující operace:
  - init (F): provede prvotní inicializaci fronty F (před jejím prvním použitím)
  - enqueue (F, X): přidá hodnotu X na konec fronty F
  - dequeue (F, X): odebere hodnotu z čela fronty F a vloží ji do proměnné X
  - empty (F): testuje, zda-li je fronta F prázdná či nikoliv

Fronta (5)

- Implementace fronty pomocí dynamických proměnných:

```
type PPrvek = ^TPrvek;
        TPrvek = record
            Hodnota: typ_hodnoty;      (* např. integer *)
            Naslednik: PPrvek;
        end;
        TFronta = record
            Celo, Konec: PPrvek;
        end;
var F: TFronta;
```

Fronta (6)