

Desáté počítačové cvičení

Základy programování (IZP)

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole
Petr Veigend, veigend@fit.vut.cz



- **Modulární programování – ukázka**
- **Jednosměrně vázaný lineární seznam**
 - **(VELMI PRAVDĚPODOBNĚ) BUDE NA ZKOUŠCE**

- Definuje předpis pro program **make**

```
CC=gcc
CFLAGS=-std=c11 -Wall -Wextra -g

all: sll_main

sll_main: sll.o
    $(CC) $(CFLAGS) sll_main.c sll.o -o sll_main

sll.o: sll.c
    $(CC) $(CFLAGS) -c sll.c

clean:
    rm *.o sll_main

dep:
    $(CC) -MM *.c
```

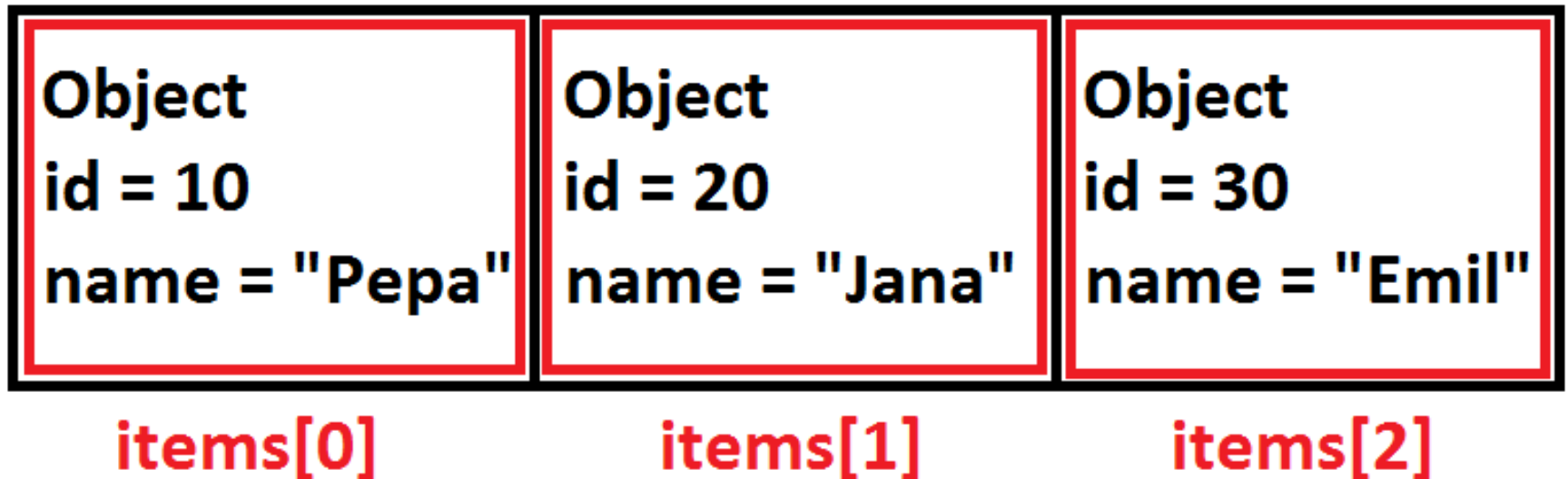
```
typedef struct {  
    int id;  
    char *name;  
} Object;
```

```
typedef struct {  
    unsigned size; // velikost pole  
    Object* items; // pole objektu  
} Array;
```

- Pokud si budete chtít zkusit modulární verzi dnešního příkladu, použijte online prostředí (odkaz na profilu)

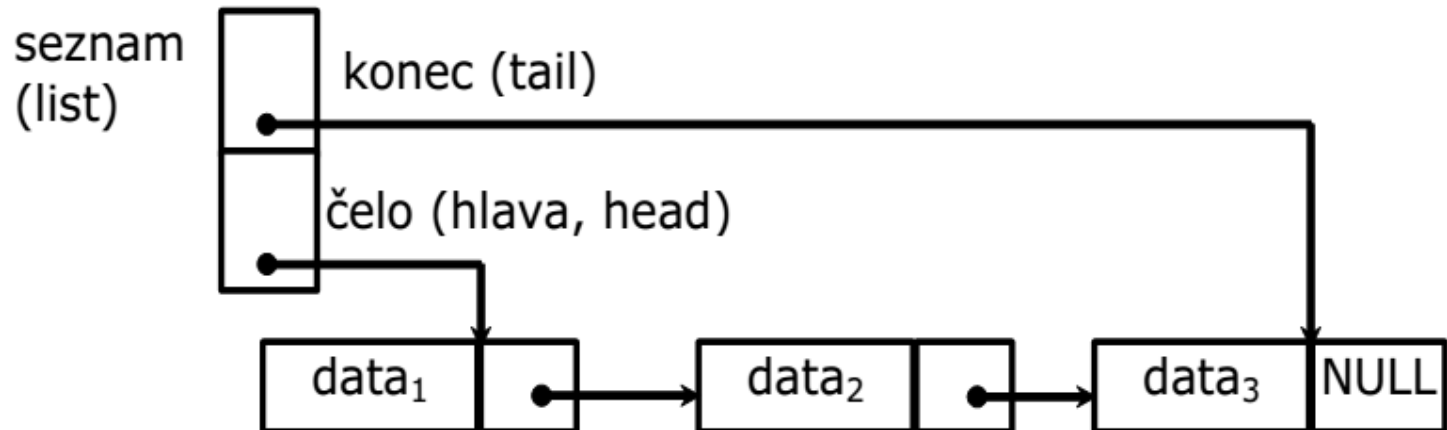
```
typedef struct {  
    unsigned size; // velikost pole  
    Object* items; // pole objektu  
} Array;
```

- Pole objektů: `Object* items;`
 - Na každém indexu pole je struktura `Object`



JEDNOSMĚRNĚ VÁZANÝ SEZNAM

- Jedná se o **dynamickou datovou strukturu**
 - Skládá se z jednotlivých položek, které jsou svázány **ukazateli**
 - V datovém typu jsou mimo **užitečných dat** ještě **ukazatele na stejný typ**, které zajišťují provázání
 - Dynamicky vytvořené proměnné můžeme snadno spojovat pomocí ukazatelů



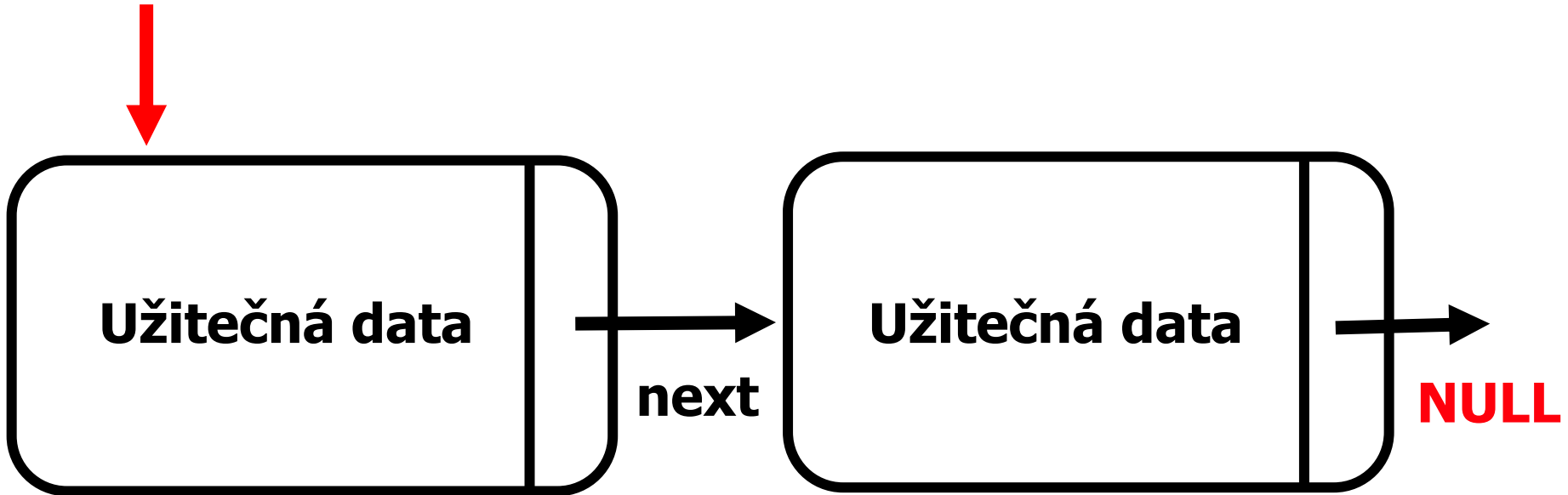
- Jak bylo řečeno, datový typ musí obsahovat
 - **Užitečná data**
(reprezentovaná datovým typem **Object**)
 - **Ukazatel na další prvek**
- Jak bude vypadat?

- Jak bylo řečeno, datový typ musí obsahovat
 - **Užitečná data**
(reprezentovaná datovým typem `Object`)
 - **Ukazatel na další prvek**
- Jak bude vypadat?

```
typedef struct item { // prvek seznamu
    Object data;      // užitečná data
    struct item* next; // ukazatel na další prvek
}Item;
```

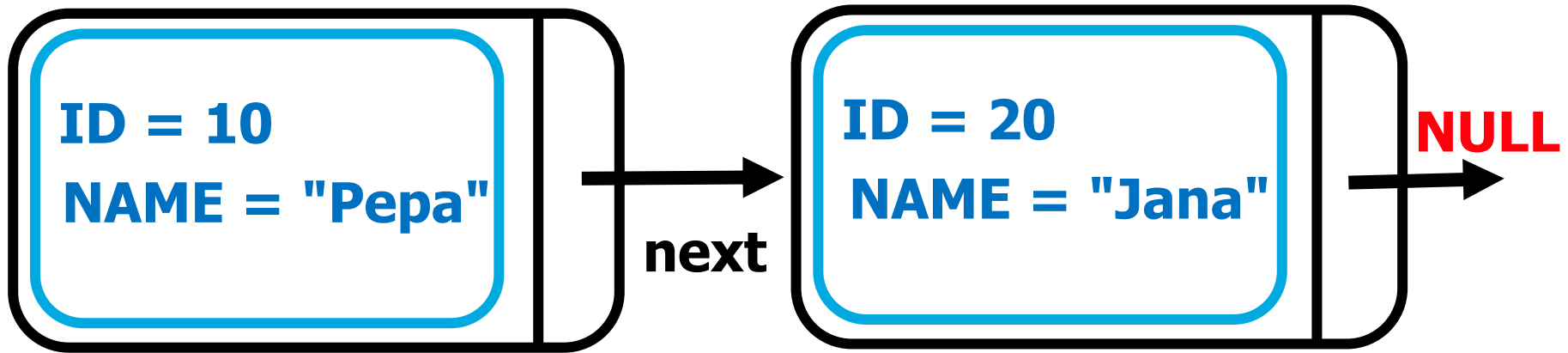
```
typedef struct { // seznam
    Item* first; // ukazatel na první prvek
}List;
```

FIRST – ukazatel na první položku seznamu



- Užitečná data jsou v našem případě **struktura Object**

FIRST



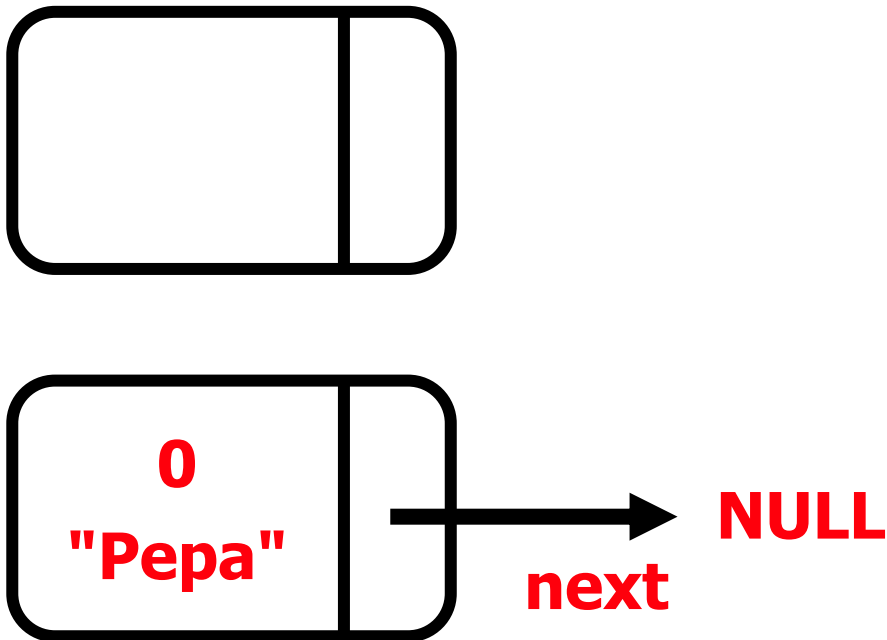
- **Základní funkce pro práci se seznamem**
 - Inicializace seznamu
 - Vytvoření nové položky
 - Vložení nové položky na začátek seznamu
 - Odstranění první položky ze seznamu
 - Dealokace seznamu

FIRST

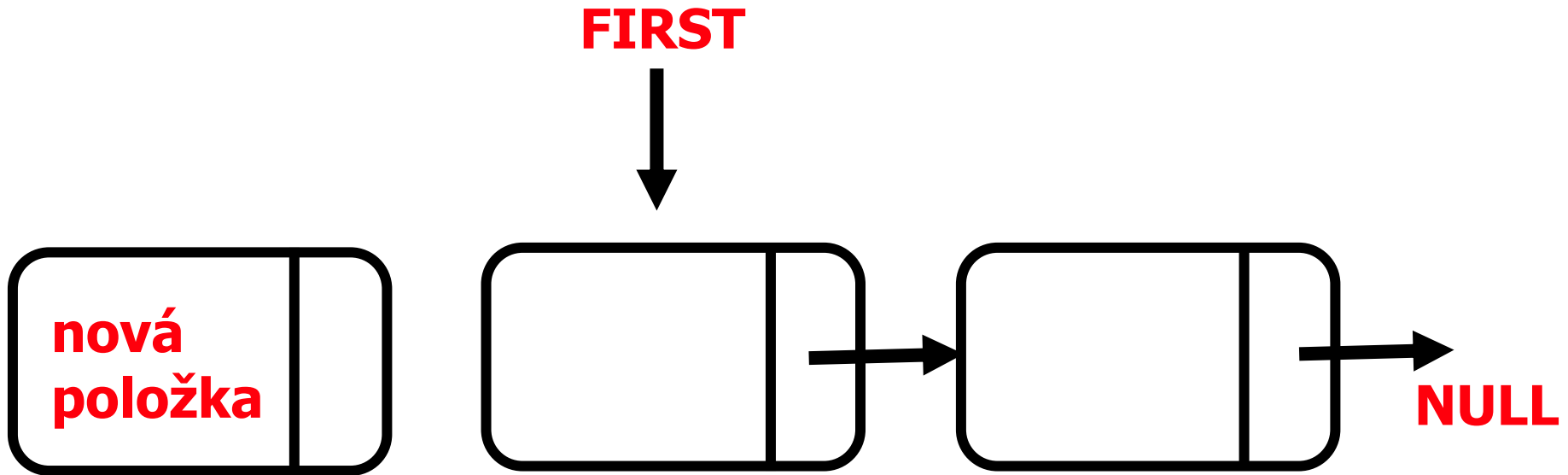


NULL

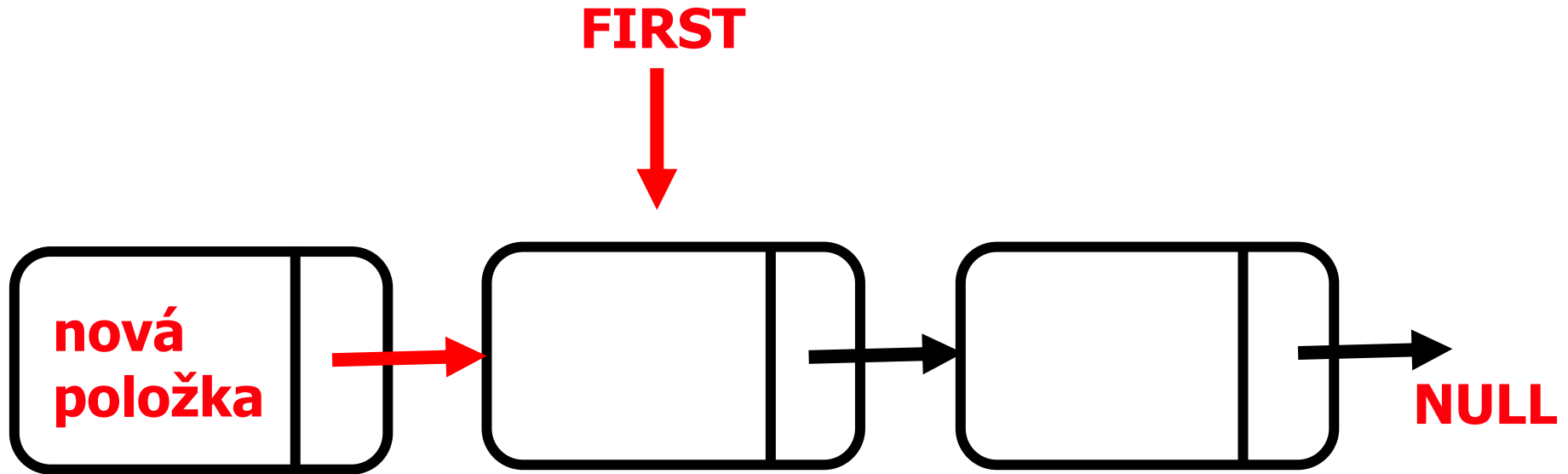
Inicializace seznamu je snadná – pouze nastavíme ukazatel FIRST (hlavičku seznamu) na NULL.



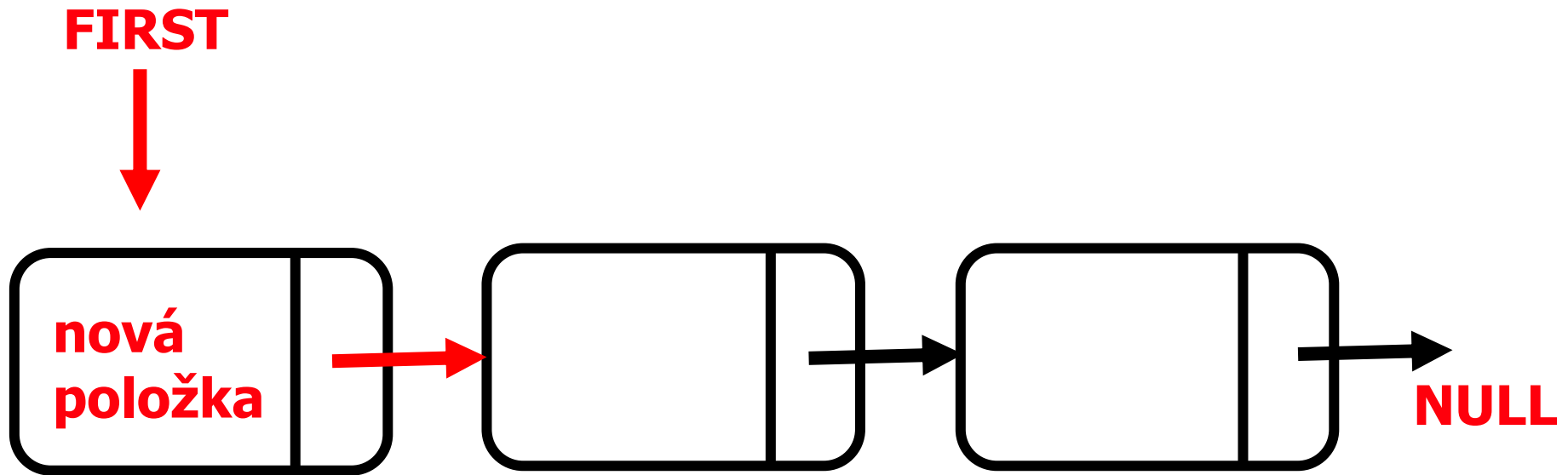
**Nejprve alokujeme paměť pro novou položku.
Potom ji naplníme daty (Object).**



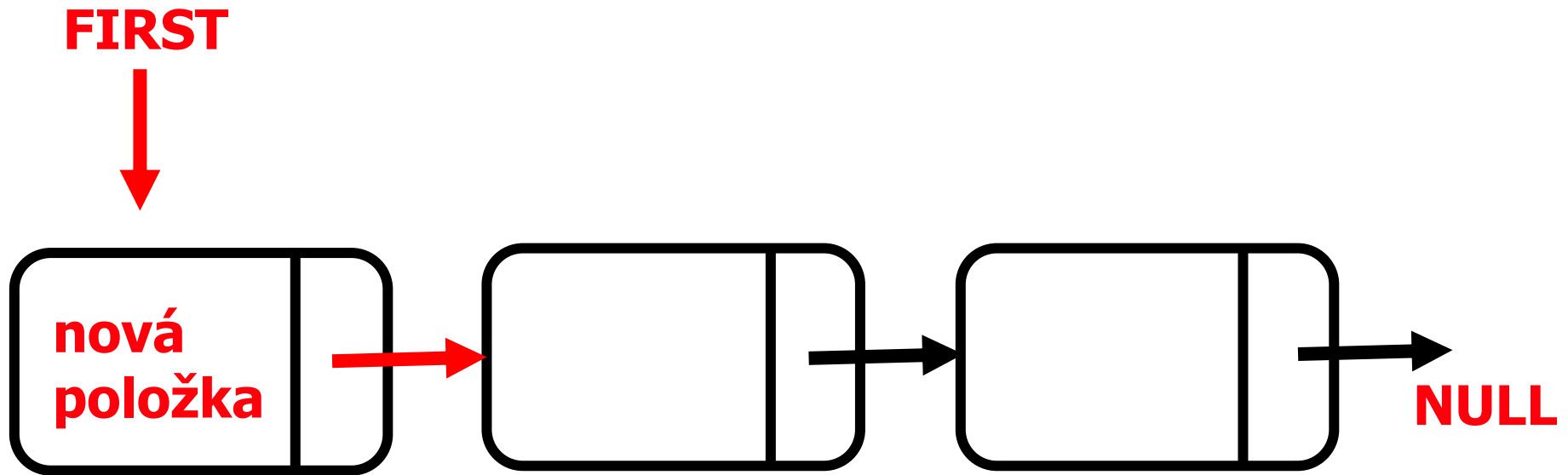
Máme vytvořenou novou položku.
Nyní ji potřebujeme navázat do seznamu.



Nová položka bude ukazovat na hlavičku seznamu.



Nyní posuneme hlavičku seznamu na nově vloženou položku.



Nyní posuneme hlavičku seznamu na nově vloženou položku.

HOTOVO! 😊

- **Další funkce**
 - **Vyhledání položky s nejmenším ID**
 - **Vyhledání položky s odpovídajícím názvem**
 - Počet položek seznamu
- Lze implementovat také
 - Test prázdnoty seznamu
 - Seznam je prázdný, pokud ukazatel **first** ukazuje na **NULL**
 - Výpis prvků seznamu
- Nápořádě
 - Pro průchod seznamem se využívá cyklus

- **Obhajoba projektu v čase cvičení**
- Deadline odevzdání: 6.12.2025, 23:59
- Pro zápočet je potřeba **minimálně 1b**

Děkuji Vám za pozornost!