STU FEI

Programovacie techniky

1. Úvod, Bjarne, zreťazený zoznam.

Martin Drozda
martin.drozda@stuba.sk
C601

Podmienky absolvovania

Test 1: 5. týždeň (10 bodov)

Test 2: 10. týždeň (10 bodov)

Test 1 + Test 2: min. 10 bodov

Test 3 (predtermín): 12. týždeň (80 bodov)

Riadny termín / Opravný termín

Bonusové body na cvičeniach a na prednáške

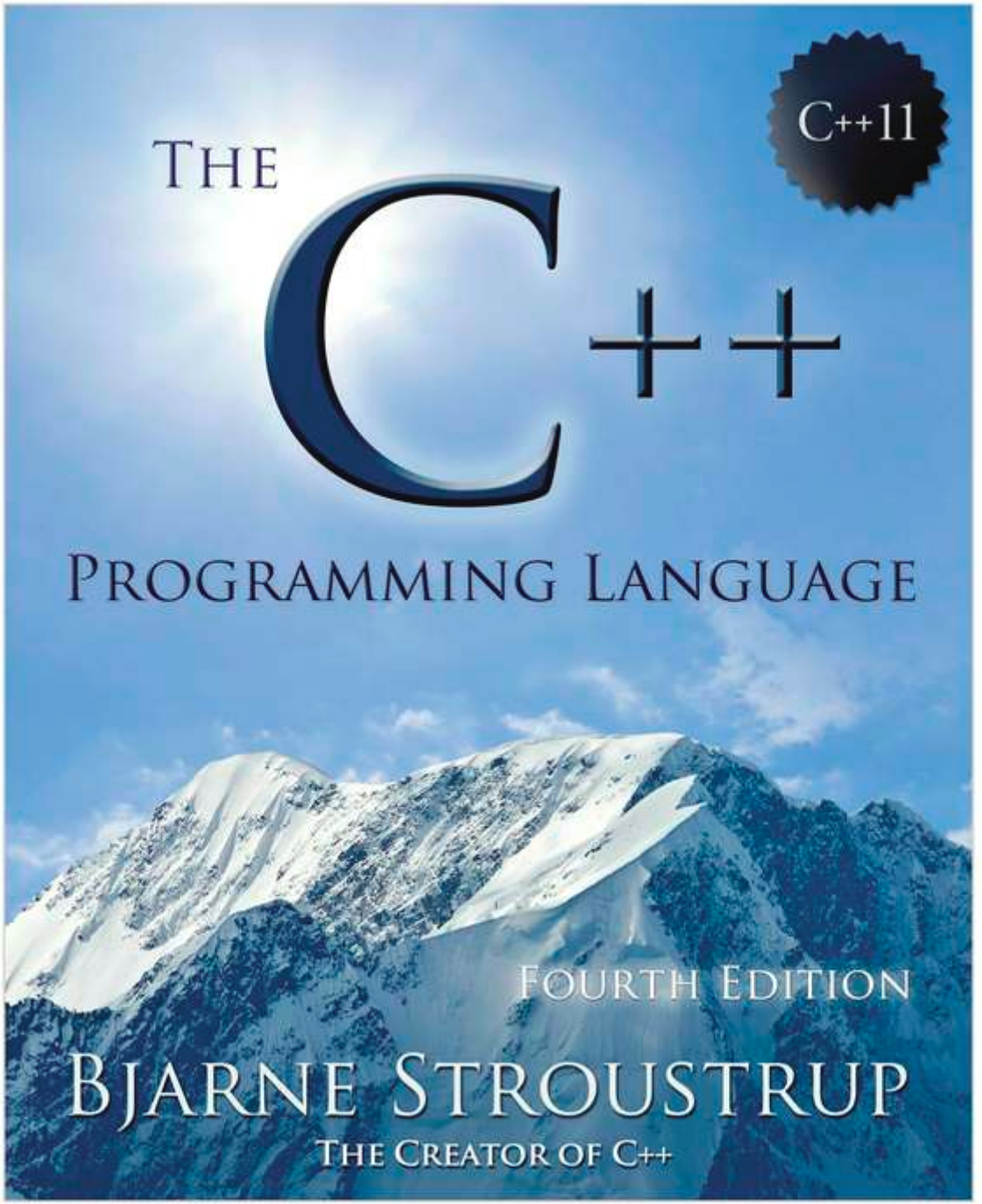
Programming challenge v druhej polovici semestra

Obsah predmetu

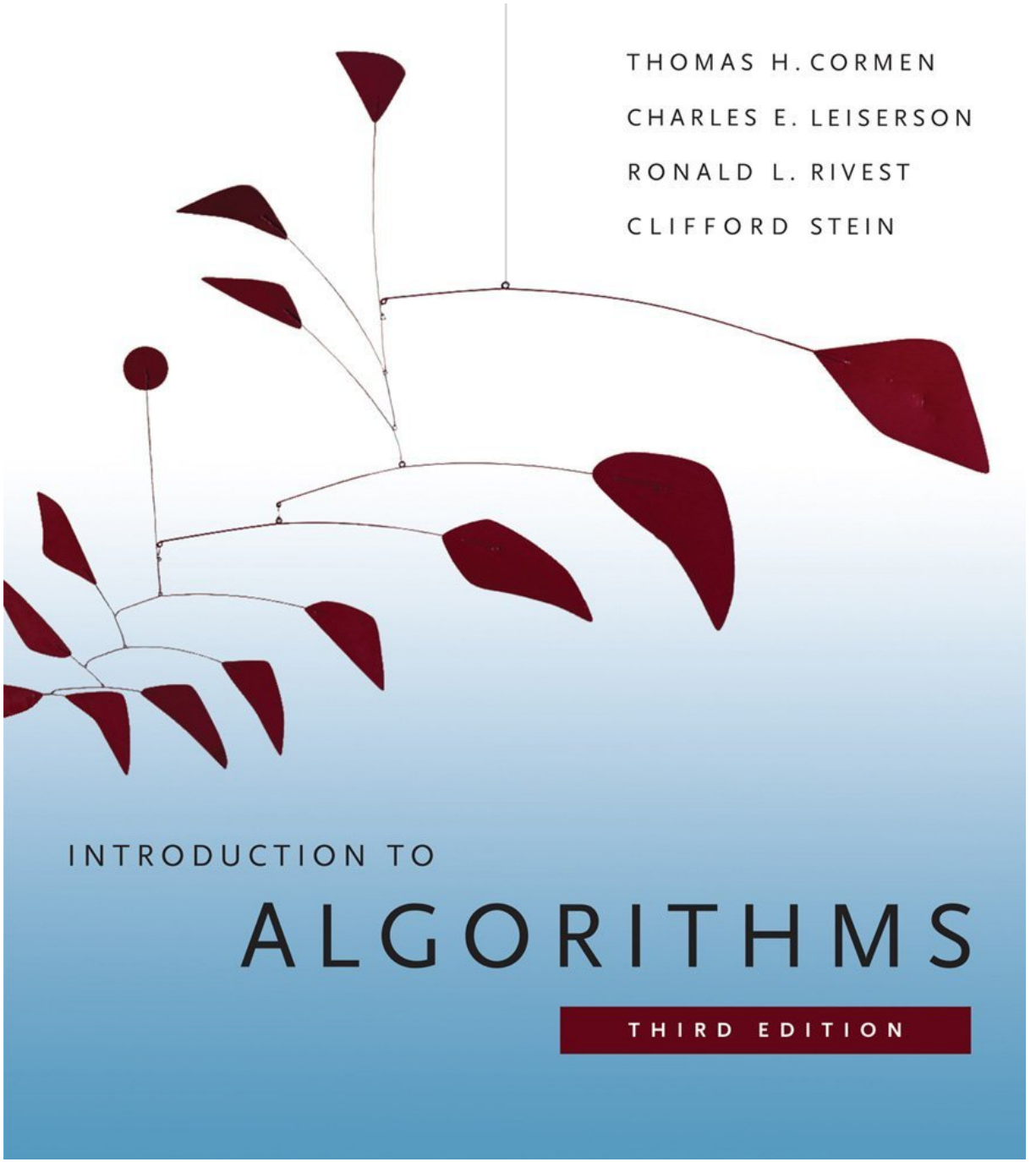
Programovacie techniky: dátové štruktúry údajov a techniky práce s nimi.

- Hľadanie a triedenie
- Grafové algoritmy

Programovací jazyk C++ (základy)


C++11
THE
C++
PROGRAMMING LANGUAGE
FOURTH EDITION
BJARNE STROUSTRUP
THE CREATOR OF C++

B. Stroustrup


THOMAS H. CORMEN
CHARLES E. LEISERSON
RONALD L. RIVEST
CLIFFORD STEIN
INTRODUCTION TO
ALGORITHMS
THIRD EDITION

Programátorské výzvy

Smerníky (C, C++)

Rekurzia (funkcionálne jazyky: Haskell, ML)

Objekty (C++)

Algoritmy (hľadanie, triedenie, ...)

Dátové štruktúry (zásobník, zreťazený zoznam...)

Smerníky

```
int k;
int* smernik;
int** smernik2;

printf(„%d“, **smernik2);
printf(„%p“, smernik);
scanf(„%d“, &k);
```

Pole

int pole[10]; //statická alokácia

pole[0] = 1;

pole[1] = 8;

pole[9] = -9;

1, 8, 8, 0, 100, 77, 66, 1, 0, -9

Zmeniť veľkosť poľa je problém.

C90 / C99

```
int pole[10];

int k = 10;
int pole[k]; //C99
```

Dynamická alokácia

```
int *pole;

pole = (int*) malloc(10*sizeof(int)); //stdlib.h
pole[6] = 7;

free((void*) pole);
```

Zmeniť veľkosť poľa ostáva problém.

malloc, free, realloc

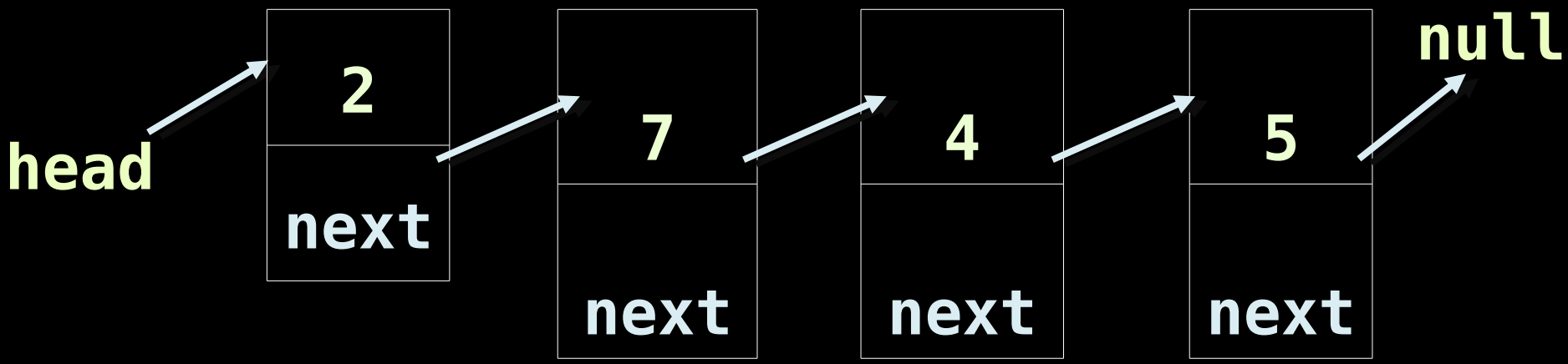
```
void* malloc (size_t size);

void free (void* ptr);

void* realloc (void* ptr, size_t size);
```

NULL

```
int *pole;


pole = (int*) malloc(10*sizeof(int));
if(pole==NULL) {
  fprintf(stderr, „Nedala sa alokovat pamat“);
  return EXIT_FAILURE; //stdlib.h
}
free(pole);
return EXIT_SUCCESS; //stdlib.h
```

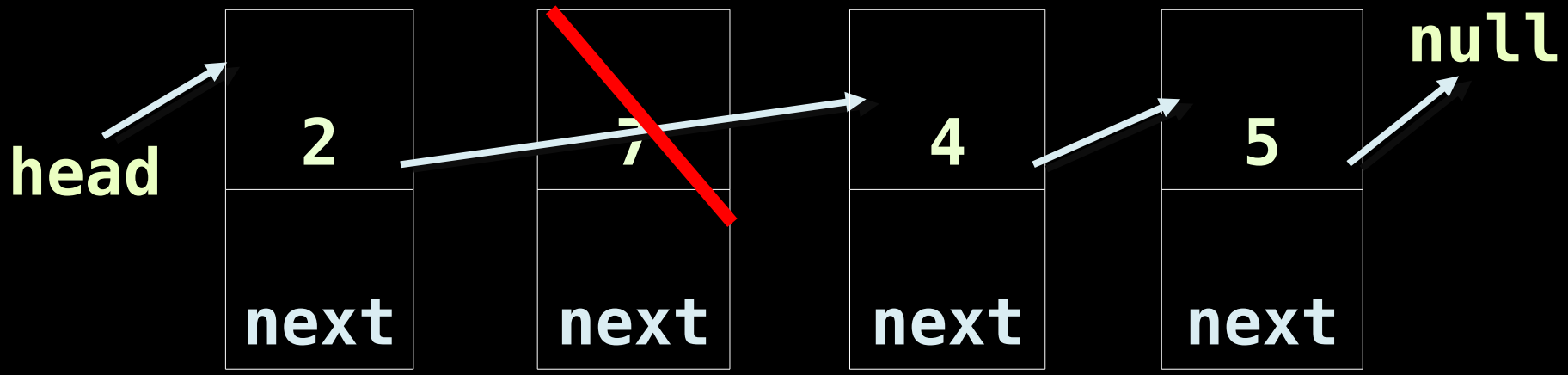
The stack

Zreťazený zoznam



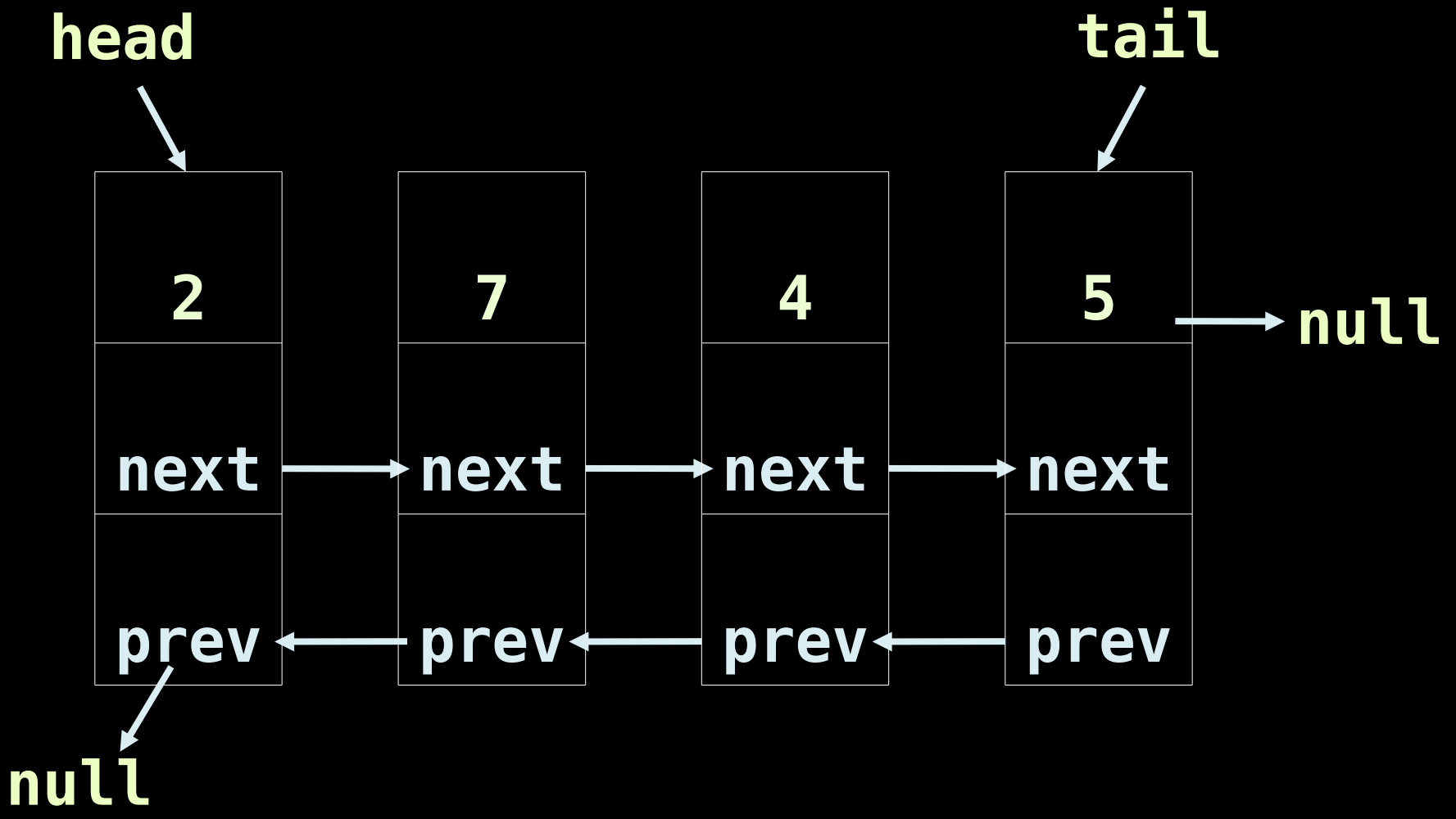
Každá položka má hodnotu a smerník na nasledujúcu položku

Zmazanie položky



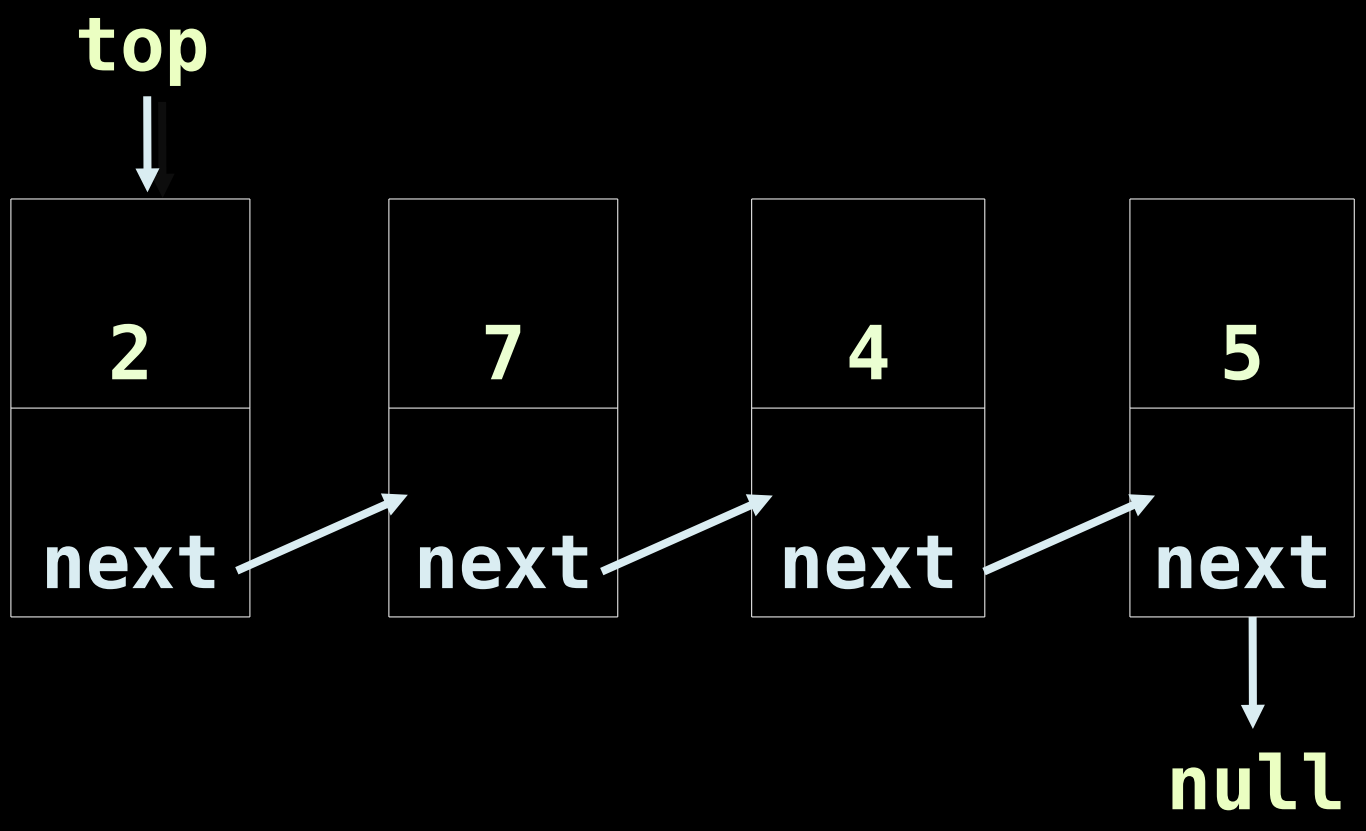
Ako pridať novú položku?

Dvojito zreťazený zoznam



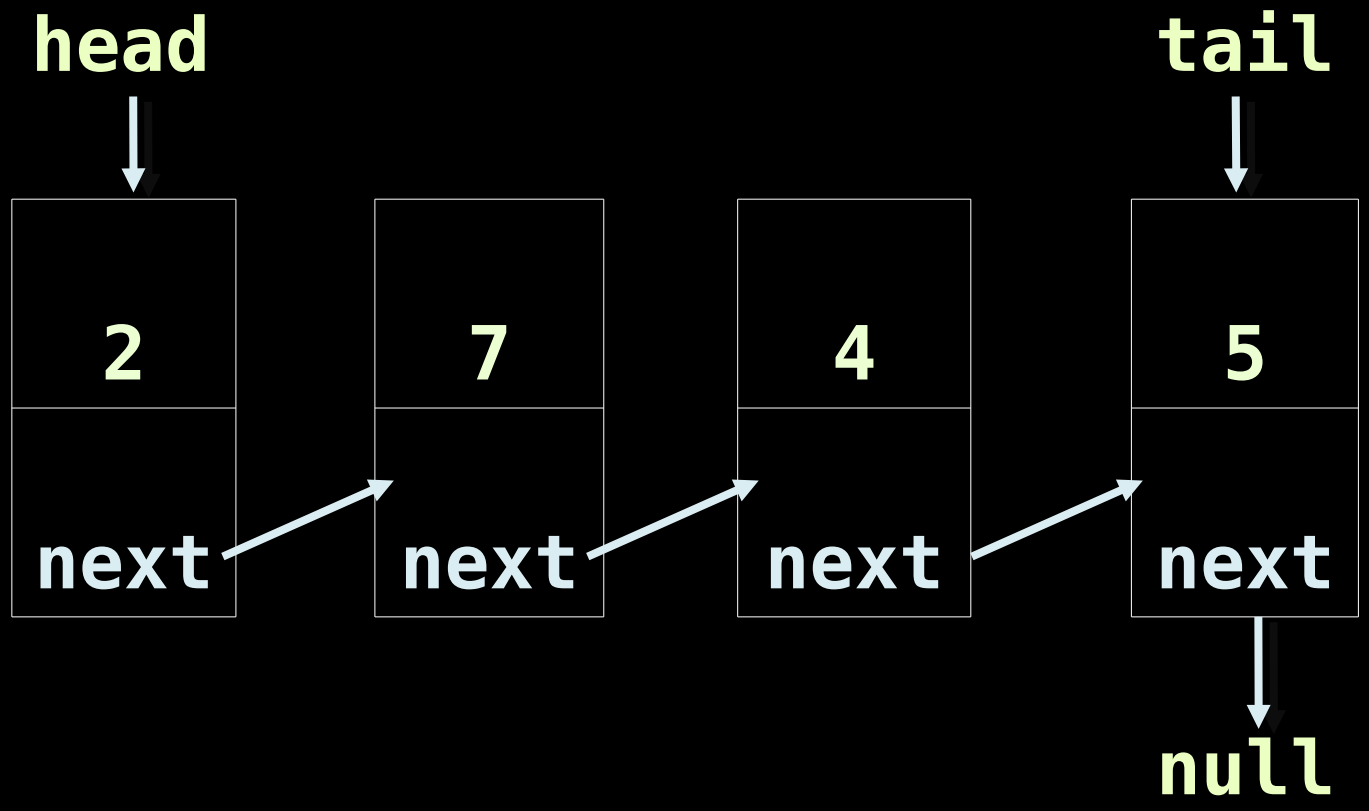
Každá položka má hodnotu a smerník na nasledujúcu a predchádzajúcu položku

Zásobník (stack)



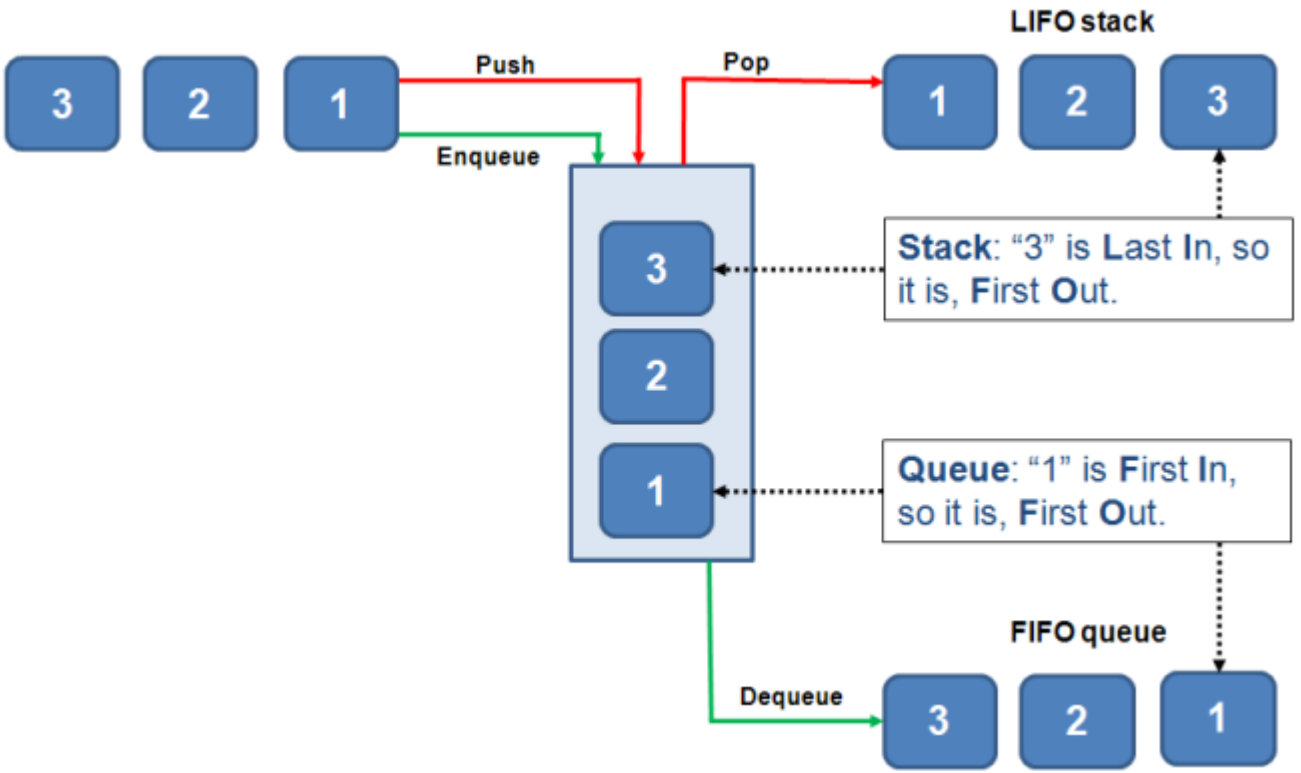
top je smerník na položku, ktorá bola vložená ako posledná

Fronta (queue)



head je smerník na položku, ktorá bola vložená ako prvá, **tail** smerník na položku, ktorá bola vložená ako posledná

Stack, queue


LIFO stack
3
2
1
Push
Pop
1
2
3
Enqueue
3
2
1
Stack: "3" is Last In, so it is, First Out.
Queue: "1" is First In, so it is, First Out.
FIFO queue
Dequeue
3
2
1

položka

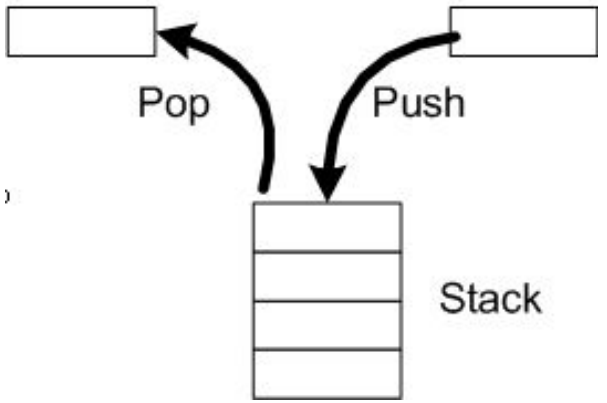
```
struct polozka {
int value;
void *next;
//void *prev;
};

typedef struct polozka polozka;
//polozka je to iste ako struct polozka
```

Úloha

Naprogramuj zreťazený zoznam!

(cvičenia)

Zásobník

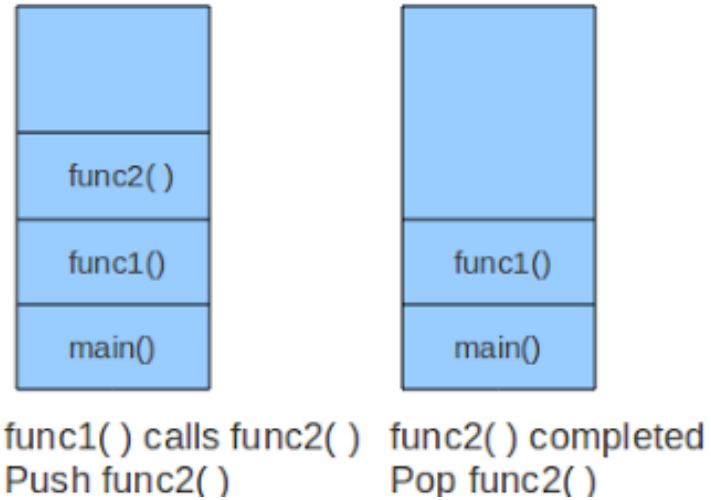


Dve operácie:

Vlož (push)

Vyber (pop)

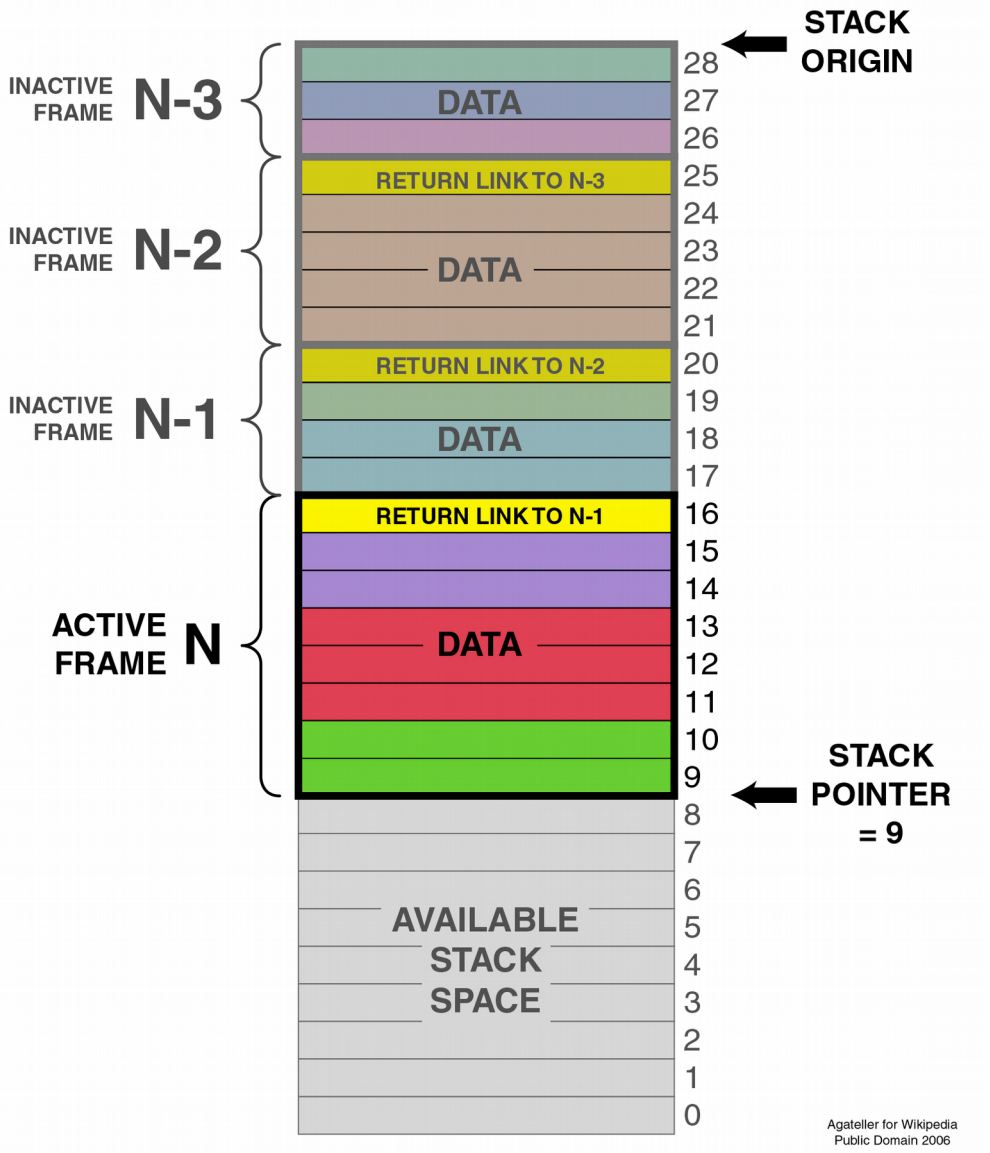
Zásobník



Vľavo: main() zavolala func1() a func1() zavolala func2()

Vpravo: func2() skončila, pop func2()

Zásobník



DATA = lokálne premenné

Zásobník

Veľkosť zásobníka:

1-32 MB podľa architektúry

Keď sa zásobník naplní, dôjde k pretečeniu.

http://www.cs.nyu.edu/exact/core/doc/stackOverflow.txt

Úloha: vieš napísať kód, ktorý zapríčiní pretečenie zásobníka?

gcc (Linux)

gcc file.c -o file //skompiluje file.c do file (bez prepínača -o bude skompilované do súboru a.out)

gcc file.c -Wall -o file //všetky warnings

gcc file.c -Wall -O2 -o file //optimalizácia kódu

gcc file.c -ggdb -o file //kód pre debugger gdb

Chvostová rekurzia

```
#include <stdio.h>
void rekurzia() {

  static int i = 1;
  printf("%d\t", i);
  int pole[1000];
  ++i;

  rekurzia();  //chvostová rekurzia, tail recursion
}
```

insert, delete, merge

Pole: ak je potrebný prístup cez index

Zreťazený zoznam: ak je potrebný insert a delete

Ak chcem nájsť konkrétnu položku, je lepší zreťazený zoznam alebo pole?

Čo sa zmení, keď usporiadam prvky v poli (od najmenšieho po najväčšie)?

Merge: zreťazený zoznam alebo pole?