# Základy programování (IZP)

## Deváté počítačové cvičení

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole
Petr Veigend, iveigend@fit.vut.cz, Alena Tesařová, atesarova@fit.vutbr.cz

BRNO UNIVERSITY OF TECHNOLOGY FACULTY OF INFORMATION TECHNOLOGY

11. týden

# Organizační okénko

- **Odevzdání druhého projektu**
  - Odevzdávají všichni členové týmu
  - Porovnávají se výsledky pomocí diff
  - Každý test bude spouštěný s valgrindem
- Obhajoba

# Opakování z minulého cvičení

- Co je tady špatně?

```
Vector *vector_ctor(){
  Vector v;
  v->size = 0;
  v->items = NULL;
  return &v;
}
```

# Opakování z minulého cvičení

- Co je tady špatně?

```
Vector *vector_ctor(){
  Vector v;
  v->size = 0;
  v->items = NULL;
  return &v;
}
```

**v** je pouze lokální proměnná alokovaná na zásobníku!
V momentě ukončení funkce zanikne.

# Opakování z minulého cvičení

- Co je tady špatně?

```
int vector_add(Vector *vec, int value){
   vec->items = realloc(vec->items, (v->size + 1)*sizeof(int));
   if (vec->items == NULL) {
        return 0;
   }
   vec->items[vec->size] = value;
   vec->size += 1;
   return 1;
}
```

# Opakování z minulého cvičení

- Co je tady špatně?

```
int vector_add(Vector *vec, int value){
   vec->items = realloc(vec->items, (v->size+1)*sizeof(int));
   if (vec->items == NULL) {
        return 0;
   }
   vec->items[vec->size] = value;
   vec->size += 1;
   return 1;
}
```

V případě, že by **items** obsahovalo nějaké položky, tak ztratíme na ně ukazatel (bude **NULL**) a nebude možné položky uvolnit pomocí free -> **memory leak!**

# Opakování z minulého cvičení

- Hledali jste někdo chyby, co pro vás byly nachystané na wiki?

```
// Initial string content
const char *str_init = "Hello World!";

void main(){
   char *str = (char *) malloc(strlen(str_init)+1);
   strcpy(str, str_init);
   printf("%s\n", str);
   free(str);
   free(str); // double free !
}
```

- **Double free způsobuje nedefinované chování!!**
- Může poškodit správu paměti a způsobit poškození stávajících bloků paměti nebo selhání budoucích alokacím paměti.

# Náplň cvičení

- ovládat základní metody vyhledávání a řazení
- vyjádřit řešení problémů pomocí rekurze

# Algoritmus Selection Sort

1) Pole je rozděleno na **seřazenou část** a **neseřazenou část**

2) V neseřazené části se nalezne minimum a vymění se s prvkem bezprostředně následujícím za seřazenou částí a tento prvek se do ní zahrne

3) Na začátku má seřazená část délku nula, na konci má délku pole

# Algoritmus Selection sort – příklad

**Neseřazená část** **Seřazená část** **Nejmenší prvek**

| | | | | | |
|---|---|---|---|---|---|
| 6 | 60 | 42 | 200 | 4 | 3 |
| 3 | 60 | 42 | 200 | 4 | 6 |
| 3 | 4 | 42 | 200 | 60 | 6 |
| 3 | 4 | 6 | 200 | 60 | 42 |
| 3 | 4 | 6 | 42 | 60 | 200 |
| 3 | 4 | 6 | 42 | 60 | 200 |
| 3 | 4 | 6 | 42 | 60 | 200 |

# Hledání chyb pomocí nástroje valgrind

1) Stáhněte si soubor array.c na wiki
2) Postupujte podle wiki

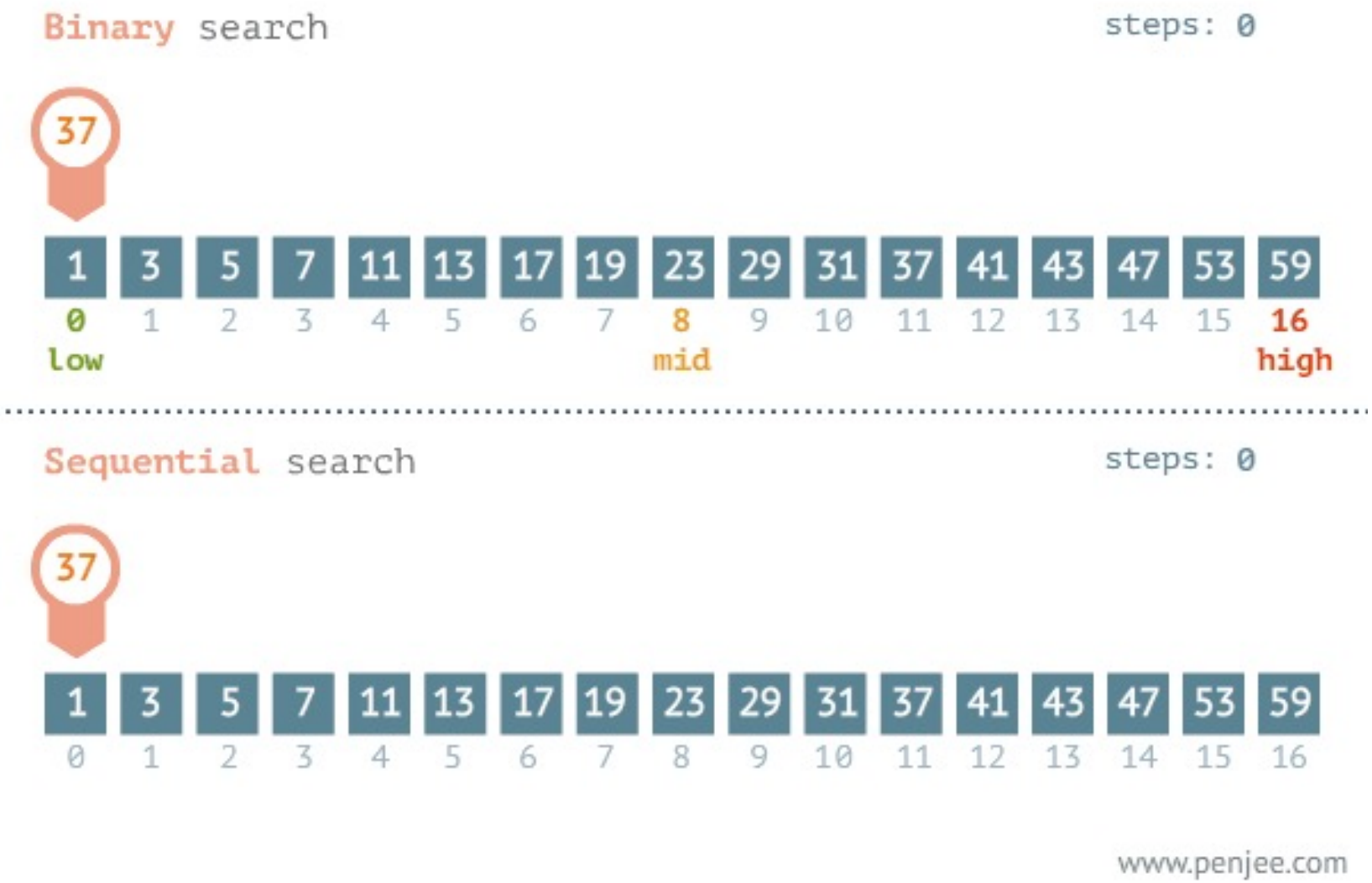
# Rekurze

- Implementujte program pro výpočet Fibbonaciho čísel (kostra na wiki)
  - Algoritmus v komentáři zdrojového kódu
- Zjistěte, kolikrát byla funkce volána (výjimečně můžete použít globální proměnnou)
- Pokuste se zrychlit výpočet s použitím globálního pole již vypočítaných výsledků

| $F_0$ | $F_1$ | $F_2$ | $F_3$ | $F_4$ | $F_5$ | $F_6$ | $F_7$ | $F_8$ | $F_9$ | $F_{10}$ | $F_{11}$ | $F_{12}$ | $F_{13}$ | $F_{14}$ | $F_{15}$ | $F_{16}$ | $F_{17}$ | $F_{18}$ | $F_{19}$ | $F_{20}$ |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 1 | 2 | 3 | 5 | 8 | 13 | 21 | 34 | 55 | 89 | 144 | 233 | 377 | 610 | 987 | 1597 | 2584 | 4181 | 6765 |

$$F(n) = \begin{cases} 0\,, & \text{pro } n = 0\,; \\ 1, & \text{pro } n = 1; \\ F(n-1) + F(n-2) & \text{jinak.} \end{cases}$$

# Úkol – binární vyhledávání (pole je seřazené)



# Úkol – binární vyhledávání (hint)

- Implementujte rekurzivní a nerekurzivní variantu

```
int binary_search_recursive(int items[], int leftIndex, int
   rightIndex, int value){
   if (rightIndex < leftIndex){
        return -1;
   }
   // middleIndex = ??
   …
}
```

```
int binary_search(int items[], int value, int length){
   // leftIndex = ??
   // rightIndex = ??
   // middle = ??
   while(..){
        …
   }
}
```

Děkuji za pozornost