Základy programování (IZP)

Osmé počítačové cvičení

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

array.c

LADĚNÍ

Valgrind

- Pro práci s dynamickou alokací paměti doporučuji používat nástroj **`valgrind`**.

```
valgrind ./<nazev programu>
```

- Druhý projekt **je nutné** testovat na Merlinovi, případně lokálním Linuxu (WSL)

- Na vizitce je bonusová prezentace k programování vzdáleně.

Alokace paměti

- Pro alokaci paměti na hromadě se používá funkce **`malloc`**

```
char* first = malloc(<velikost alokované paměti>);
if (first == NULL) // chyba
```

- Pokud použijete **`malloc`**, je nutné před koncem programu volat odpovídající **`free`**, jinak **memory leak**

```
free(first);
```

- Pokud chceme upravovat velikost již alokované paměti, lze použít funkci **`realloc`**

```
char* first = malloc(<velikost alokované paměti>);
if (first == NULL) // chyba
```

- Pokud chceme upravovat velikost již alokované paměti, lze použít funkci **`realloc`**

```
char* new_str = realloc(<puvodni ukazatel>,<nova velikost v B>);
```

```
char* first = malloc(10*sizeof(char));
...
char* new_str = realloc(first,20*sizeof(char));
if (new_str != NULL) {
   first = new_str; // first má nyní kapacitu pro 20
                    //   znaků
}
```

Práce s dynamickou pamětí (01-vector.c)

VEKTORY

Funkce pro práci s vektory

- Implementujte funkce pro práci s datovým typem vector_t
  - Konstruktor, destruktor

```
vector_t* vector_ctor();
void vector_dtor(vector_t* v);
```

  - Tisk vektoru

```
void vector_print(vector_t *v);
```

  - Přidání jednoho prvku na konec vektoru

```
int vector_add(vector_t *v, int value);
```

    - Funkce bude využívat funkci, která upraví velikost alokované paměti pro pole celých čísel na novou velikost

```
int* resize(int* arr, int new_size);
```

- Vektor se bude naplňovat ze stdin.

Práce s dynamickou pamětí (02-string.c)

PRÁCE S PODŘETĚZCI

Podřetězce v řetězci

- Napište funkce, které
  - Najdou podřetězec v řetězci
  - Nahradí tento podřetězec jiným řetězcem stejné délky
  - Nahradí tento podřetězec delším nebo kratším řetězcem.

- Jak na to?

PŘÍKLADY K PROCVIČENÍ

Příklad na rozehřátí

- Implementujte funkci

```
void getMax(int *pole, int len, int *max);
```

která vyhledá v poli maximální hodnotu a vrátí ji přes ukazatel

Příklad na rozehřátí

- Implementujte funkci

```
void getMax(int *pole, int len, int *max);
```

která vyhledá v poli maximální hodnotu a vrátí ji přes ukazatel

- Jak inicializujeme pole?

Příklad na rozehřátí

- Implementujte funkci

```
void getMax(int *pole, int len, int *max);
```

která vyhledá v poli maximální hodnotu a vrátí ji přes ukazatel

- Jak inicializujeme pole?

```
int pole[10]={7,2,3,9,15,20,-1,42,100,-75};
```

Příklad na rozehřátí

- Implementujte funkci

```
void getMax(int *pole, int len, int *max);
```

která vyhledá v poli maximální hodnotu a vrátí ji přes ukazatel

- Jak inicializujeme pole?

```
int pole[10]={7,2,3,9,15,20,-1,42,100,-75};
```

- Jak zavoláme funkci **`getMax()`**?

Příklad na rozehřátí

- Implementujte funkci

```
void getMax(int *pole, int len, int *max);
```

která vyhledá v poli maximální hodnotu a vrátí ji přes ukazatel

- Jak inicializujeme pole?

```
int pole[10]={7,2,3,9,15,20,-1,42,100,-75};
```

- Jak zavoláme funkci **`getMax()`**?

```
int maximum = 0;
getMax(pole, &maximum);
```

Děkuji Vám za pozornost!