



PROGRAMOVACIE JAZYKY PRE VSTAVANÉ SYSTÉMY

Cvičenie 6

NÁPLŇ CVIČENIA

1. Práca s pamäťou na úrovni bajtov.
2. Dynamicky alokovaná pamäť.
3. Porovnanie rôznych typov polí v jazyku C.
4. Implementácia dynamického poľa.
5. Jednostranne zreťazený zoznam.



PRÁCA S PAMÄŤOU NA ÚROVNI BAJTOV

- Jazyk C poskytuje niekoľko funkcií, pomocou ktorých je možné pracovať s celými blokmi operačnej pamäte.
- Prototypy funkcií sú v hlavičkovom súbore `<string.h>` (<http://en.cppreference.com/w/c/string/byte>):
 - `void* memcpy(void *dest, const void *src, size_t n);`
 - `void* memmove(void *dest, const void *src, size_t n);`
 - `void* memset(void *ptr, int c, size_t n);`
 - `int memcmp(const void *ptr1, const void *ptr2, size_t n);`
 - `void* memchr(const void *ptr, int c, size_t n);`
- `size_t` – nezáporný celočíselný typ, návratový typ operátora `sizeof`.



DYNAMICKY ALOKOVANÁ PAMÄŤ

- Pamäť **explicitne** alokovaná a uvoľňovaná za behu programu.
- Funkcie pre prácu s dynamickou pamäťou – hlavičkový súbor `<stdlib.h>` (<http://en.cppreference.com/w/c/memory>):
 - `void* malloc(size_t size);`
 - `void* calloc(size_t nitems, size_t size);`
 - `void* realloc(void *ptr, size_t new_size);`
 - `void free(void *ptr);`
 - `void* aligned_alloc(size_t alignment, size_t size) (C11).`



UKÁŽKA PRÁCE S DYNAMICKOU PAMĚTÍ

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  void naplnPole(int n, int pole[]) {
5      for (int i = 0; i < n; i++) {
6          pole[i] = i;
7      }
8  }
9
10 void vypisPole(int n, int pole[]) {
11     for (int i = 0; i < n; i++) {
12         printf("%d ", pole[i]);
13     }
14     printf("\n");
15 }
16
17 int main(int argc, char* argv[]) {
18     int* pole; //pole - "wild pointer"
19
20     pole = calloc(5, sizeof(int));
21     naplnPole(5, pole);
22
23     pole = realloc(pole, sizeof(int[10])); //pri takomto zapise existuju rizika
24     naplnPole(5, pole + 5);
25
26     vypisPole(10, pole);
27     free(pole); //pole - "dangling pointer"
28     pole = NULL; //pole - "NULL pointer"
29
30     return 0;
31 }
```



3 TYPY POLÍ V JAZYKU C

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4  int main(int argc, char* argv[]) {
5      int pole1[5];
6      int n = 5;
7      int pole2[n];
8      int *pole3 = (int *)malloc(5 * sizeof(int));
9
10     printf("sizeof(pole1) = %u\n", sizeof(pole1));
11     printf("pole1 = %p\n", pole1);
12     printf("&pole1 = %p\n", &pole1);
13     printf("&(*pole1) = %p\n\n", &(*pole1));
14
15     printf("sizeof(pole2) = %u\n", sizeof(pole2));
16     printf("pole2 = %p\n", pole2);
17     printf("&pole2 = %p\n", &pole2);
18     printf("&(*pole2) = %p\n\n", &(*pole2));
19
20     printf("sizeof(pole3) = %u\n", sizeof(pole3));
21     printf("pole3 = %p\n", pole3);
22     printf("&pole3 = %p\n", &pole3);
23     printf("&(*pole3) = %p\n\n", &(*pole3));
24
25     free(pole3);
26     pole3 = NULL;
27     return 0;
28 }
```



ÚLOHY – DYNARRAY

- Vytvorte štruktúru DynArray, ktorá bude predstavovať zoznam implementovaný poľom dynamickej veľkosti, t.j. pri vkladaní a odoberaní prvkov sa bude automaticky meniť množstvo alokovanej pamäte.
- Predpokladajte, že jednotlivé položky v štruktúre majú typ double.
- So štruktúrou budú pracovať nasledujúce funkcie:
 - void init(DynArray *array);
 - void dispose(DynArray *array);
 - void print(DynArray *array);
 - void add(DynArray *array, double data);
 - _Bool tryInsert(DynArray *array, double data, int pos);
 - _Bool trySet(DynArray *array, int pos, double data);
 - _Bool tryGet(DynArray *array, int pos, double *data);
 - _Bool tryRemove(DynArray *array, int pos, double *data);
 - _Bool tryCopy(DynArray *src, DynArray *dest); //využite funkciu memcpy
 - void readFromTxt(DynArray *array, FILE *txtFile);
 - void writeToTxt(DynArray *array, FILE *txtFile).



ÚLOHY – LINZOZ

- Vytvorte štruktúru LinZoz, ktorá bude predstavovať jednostranne zreťazený zoznam položiek typu double.
- So štruktúrou budú pracovať nasledujúce funkcie:
 - void init(LinZoz *linZoz);
 - void dispose(LinZoz *linZoz);
 - void print(LinZoz *linZoz);
 - void add(LinZoz *linZoz, double data);
 - _Bool tryInsert(LinZoz *linZoz, double data, int pos);
 - _Bool trySet(LinZoz *linZoz, int pos, double data);
 - _Bool tryGet(LinZoz *linZoz, int pos, double *data);
 - _Bool tryRemove(LinZoz *linZoz, int pos, double *data);
 - _Bool tryCopy(LinZoz *src, LinZoz *dest);
 - void readFromTxt(LinZoz *linZoz, FILE *txtFile);
 - void writeToTxt(LinZoz *linZoz, FILE *txtFile).



ÚLOHY – BONUSOVÉ

- Prerobte štruktúru DynZoz a všetky metódy tak, aby štruktúra uchovávala reťazce (pozor na správnu alokáciu a dealokáciu pamäte):
 - a) s maximálnou dĺžkou 30 znakov (0.5 bodu);
 - b) ľubovoľnej dĺžky (0.5 bodu).
- Vytvorte štruktúru LinZoz2 (a implementujte všetky funkcie, ktoré má štruktúra LinZoz), ktorá bude predstavovať obojstranne zreťazený zoznam:
 - a) položiek typu double (0.5 bodu);
 - b) reťazcov s maximálnou dĺžkou 30 znakov (0.5 bodu).

