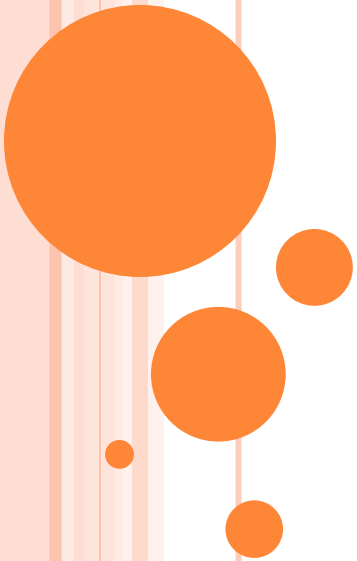

Programovacie jazyky pre vstavané systémy

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 pamäťou

```c
#include <stdio.h>
#include <stdlib.h>

void naplnPole(int n, int pole[]) {
    for (int i = 0; i < n; i++) {
        pole[i] = i;
    }
}
void vypisPole(int n, int pole[]) {
    for (int i = 0; i < n; i++) {
        printf("%d ", pole[i]);
    }
    printf("\n");
}
int main(int argc, char* argv[]) {
    int* pole; //pole - "wild pointer"

    pole = calloc(5, sizeof(int));
    naplnPole(5, pole);

    pole = realloc(pole, sizeof(int[10])); //pri takomto zapise existuju rizika
    naplnPole(5, pole + 5);

    vypisPole(10, pole);
    free(pole); //pole - "dangling pointer"
    pole = NULL; //pole - "NULL pointer"

    return 0;
}
```

3 TYPY POLÍ V JAZYKU C

```c
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[]) {
    int pole1[5];
    int n = 5;
    int pole2[n];
    int *pole3 = (int *)malloc(5 * sizeof(int));

    printf("sizeof(pole1) = %u\n", sizeof(pole1));
    printf("pole1 = %p\n", pole1);
    printf("&pole1 = %p\n", &pole1);
    printf("&(*pole1) = %p\n\n", &(*pole1));

    printf("sizeof(pole2) = %u\n", sizeof(pole2));
    printf("pole2 = %p\n", pole2);
    printf("&pole2 = %p\n", &pole2);
    printf("&(*pole2) = %p\n\n", &(*pole2));

    printf("sizeof(pole3) = %u\n", sizeof(pole3));
    printf("pole3 = %p\n", pole3);
    printf("&pole3 = %p\n", &pole3);
    printf("&(*pole3) = %p\n\n", &(*pole3));

    free(pole3);
    pole3 = NULL;
    return 0;
}
```

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(const 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(const DynArray *src, DynArray *dest); //využite funkciu memcpy
  - void readFromTxt(DynArray *array, FILE *txtFile);
  - void writeToTxt(const 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(const 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(const LinZoz *src, LinZoz *dest);
  - void readFromTxt(LinZoz *linZoz, FILE *txtFile);
  - void writeToTxt(const 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);
  c) položiek ľubovoľného typu (2 body).