Základy programování (IZP)

8. počítačové cvičenie

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

Dynamicky alokované pole

1. Alokujte si pamäť pre 5 prvkové pole
2. Zo STDIN načítajte do pola 5 čísiel
3. Prvky pola vypíšte a uvoľnite pamäť

```
#include <stdio.h>
#include <stdlib.h>
#define ARR_SIZE 5

int main() {
    int *p_a;
    // Dynamic memory allocation

    for (int i = 0; i < ARR_SIZE; i++) {
        scanf("%d", &p_a(i));
    }
    for (int i = 0; i < ARR_SIZE; i++) {
        printf("%d ", p_a(i));
    }

    // Free used memory block
    return 0;
}
```

Dynamicky alokované pole

```
#include <stdio.h>
#include <stdlib.h>
#define ARR_SIZE 5

int main() {
    int *p_a = malloc(sizeof(int) * ARR_SIZE);

    if (p_a == NULL) {
        return 1;
    }

    for (int i = 0; i < ARR_SIZE; i++) {
        scanf("%d", &p_a[i]);
    }
    for (int i = 0; i < ARR_SIZE; i++) {
        printf("%d ", p_a[i]);
    }

    free(p_a);

    return 0;
}
```

Zväčšenie dynamicky alokovaného pola

1. Využite predchádzajúci príklad
2. Napíšte funkciu, ktorá upraví veľkosť zadaného dynamického pola
3. Vložte do novej časti pola ľubovoľnú hodnotu

```
#include <stdio.h>
#include <stdlib.h>
#define ARR_SIZE 5

int main() {
    int *p_a = malloc(sizeof(int) * ARR_SIZE);

    if (p_a == NULL) {
        return 1;
    }

    for (int i = 0; i < ARR_SIZE; i++) {
        scanf("%d", &p_a(i));
    }

    // Fuction call
```

Zväčšenie dynamicky alokovaného pola

```
int *resize(int *p, int newSize) {
    int *new_p = realloc(p, sizeof(int) * newSize);
    if (new_p == NULL) {
        free(p);
    }
    return new_p;
}
```

```
    p_a = resize(p_a, 10);

    if (p_a == NULL) {
        return 1;
    }

    p_a(9) = 10;
    for (int i = 0; i < 10; i++) {
        printf("%d ", p_a(i));
    }
    free(p_a);

    return 0;
}
```

Výpis vektora

1. Doplňte na príslušné miesta správnu alokácia a uvoľnenie pamäte
2. Napíšte funkciu, ktorá vypíše hodnoty daného vektoru

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

typedef struct {
    int *items;
    unsigned int size;
} vector_t;

int main() {
    vector_t v = {NULL, 5};
    // Memory allocation
    for (int i = 0; i < v.size; i++) {
        v.items(i) = i * i;
    }
    // Print vector
    // Free memory
    return 0;
}
```

Výpis vektora

```
void printVector(vector_t *v) {
    for (int i = 0; i < v->size; i++) {
        printf("%d ", v->items(i));
    }
}

int main() {
    vector_t v = {NULL, 5};
    v.items = malloc(sizeof(int) * v.size);
    if (v.items == NULL) {
        return 1;
    }

    for (int i = 0; i < v.size; i++) {
        v.items(i) = i * i;
    }

    printVector(&v);

    free(v.items);
    return 0;
}
```

Obsahuje string zadaný podstring?

1. Napíšte funkciu, ktorá vráti index zadaného podstringu

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

const char * str_init  = "Hello World!";

int main() {
    char *str = malloc(strlen( str_init ) + 1);
    if (str == NULL) {
        return 1;
    }
    strcpy(str , str_init );

    int indx = find( str , "ld"); // Function call

    printf("%d\n", indx);

    free(str);
    return 0;
}
```

Obsahuje string zadaný podstring?

```
int find(char *str, char *subStr) {
    int subStrLen = strlen(subStr), j = 0, i = 0;
    while (str[i] != '\0') {
        if (str[i] == subStr[0]) {
            j = 0;
            while (str[i + j] == subStr[j]) {
                j++;
            }

            if (j == subStrLen) {
                return i;
            }
        }
        i++;
    }

    return -1;
}
```

Nahraďte string zadaný podstring

1. Napíšte funkciu, ktorá nahradí časť stringu za rovnako dlhý podstring

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

const char * str_init  = "Hello World!";

int main() {
    char *str = malloc(strlen( str_init ) + 1);
    if (str == NULL) {
        return 1;
    }
    strcpy(str , str_init );

    int indx = find( str , "ld");
    replace(str, indx, "aa");

    printf("%s", str);
    free(str);
    return 0;
}
```

Nahraďte string zadaný podstring

```
void replace(char *str, int indx, char* newStr) {
    int i = 0;
    while (newStr(i) != '\0') {
        str (indx + i) = newStr(i);
        i++;
    }
}
```

Nahraďte string zadaný podstring

1. Napíšte funkciu, ktorá nahradí časť stringu za dlhší podstring

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

const char * str_init  = "Hello World!";

int main() {
    char *str = malloc(strlen( str_init ) + 1);
    if (str == NULL) {
        return 1;
    }
    strcpy(str , str_init );

    str = replace(str, "World", "Eleanora");

    printf("%s\n", str);
    free(str);
    return 0;
}
```

Nahraďte string zadaný podstring

```
char *replace(char *str, char *oldStr, char* newStr) {
    int indx = find( str , oldStr);

    int strLen = strlen( str );
    int oldLen = strlen(oldStr);
    int newLen = strlen(newStr);
    int lenDiff = newLen – oldLen;

    char *newPtr = realloc(str, sizeof(char) * (strLen + lenDiff + 1));
    if (newPtr == NULL) {
        return str ;
    }
    str = newPtr;

    for (int i = strLen + lenDiff ; i >= (indx + oldLen); i --) {
        newPtr(i) = str (i – lenDiff );
    }
    for (int i = 0; i < newLen; i ++) {
        newPtr(indx + i) = newStr(i);
    }
    return newPtr;
}
```

Ladenie programu

- Breakpoints
- Printf
- Valgrind
- gdb

Ladenie programu pri práci s dynamickou pamäťou

```
gcc –std=c99 –Wall –Werror –g –o prog stringReplace.c
valgrind ./prog
```

Stačí vypracovať jednu variantu

- Zo STDIN načítajte 1 celé číslo. Alokujte si pamäť pre pole, ktoré bude mať počet prvkov daný týmto vstupom. Následne načítajte do tohto poľa príslušný počet čísiel zo STDIN a výsledok vypíšte v opačnom poradí. Nezabudnite na korektnú prácu s pamäťou.
- Definujte si 10 prvkové dynamické pole a inicializujte ho ľubovoľnými hodnotami. Potom toto pole rozšírte na 20 prvkov a do hornej polovice pola uložte druhé mocniny dolnej polovice pola. Výsledok vypíšte a nezabudnite na korektnú prácu s pamäťou.
- Zo STDIN načítajte 1 celé číslo. Alokujte si pamäť pre pole, ktoré bude mať počet prvkov daný týmto vstupom a načítajte do tohto pola textový reťazec zo STDIN. Následne pre vami zvolené písmeno spočítajte a vypíšte počet jeho výskytov v danom textovom reťazci.