IB001 Úvod do programování skrze C
Cvičení 8

Petr Velan
velan@mail.muni.cz

Fakulta informatiky
Masarykova univerzita

5. 11. 2012

Písemka

- Zadání písemky je v ISu
  `https://is.muni.cz/auth/el/1433/podzim2012/IB001/cviceni/velan/zadani_vnitro1.c`
- Na vypracování máte 50 minut
- Řešení odevzdejte do příslušné odevzdávárny

Pole - dokončení

- Vytvoření statického pole dynamické velikosti
- Není možná inicializace

```c
int main(void)
{
    int delka = 0;
    scanf("%i", &delka);

    int pole[delka];
    printf("Velikost pole v bajtech: %i\n", sizeof(pole));
    printf("Pocet prvku pole: %i\n",
        sizeof(pole)/sizeof(pole[0]));

    return 0;
}
```

Načítaní řetězců ze vstupu

- Načtení řetězce ze vstupu do pole
- Problematické, potřebujeme znát délku před načítáním
- getline(3) - pouze pro GNU C
- fgets(3) - čtení nejvýše daného počtu znaků

```
int main(void)
{
    int delka = 10;
    char pole[delka];
    fgets(pole, delka, stdin);

    printf("Nacetl jsem: %s", pole);

    return 0;
}
```

Pole a funkce

```
void tisk(int *pole, int velikost) {
    for (int i=0; i<velikost; i++) {
        printf("%i ", pole[i]);
    }
}

void tisk2(int pole[], int velikost) {
    for (int i=0; i<velikost; i++) {
        printf("%i ", pole[i]);
    }
}

int main(void)
{
    int pole[] = {1,2,3,4,5,6,7,8,9,10};
    tisk2(pole, sizeof(pole)/sizeof(int));
    return 0;
}
```

Vícerozměrná pole

- Jazyk C umožňuje používat vícerozměrná pole
- Pro pole se vyhradí jednolitý prostor v paměti
- Do paměti se přistupuje v závislosti na typu pole
- První je počet řádků, pak sloupců - počet řádků je možné nechat odvodit z inicializace

```
#define N 2
int pole[][N] = {{1,2},{3,4},{5,6},{7,8},{9,10}};

for (unsigned int i=0; i<sizeof(pole)/sizeof(int)/N; i++) {
    for (int j=0; j < N; j++) {
        printf("%i ", pole[i][j]);
    }
    printf("\n");
}
```

Vícerozměrná pole

- Je možné k paměti přistupovat přes ukazatele
- Pro pole se vyhradí jednolitý prostor v paměti
- Do paměti se přistupuje v závislosti na typu pole
- Jedná se o pole polí - prvek dvourozměrného pole je jednorozměrné. Tomu odpovídá paměťová reprezentace
- Automatická konverze mezi polem a ukazatelem je jednoúrovňová, nelze int **ptr = pole;

```
#define N 2
int pole[][N] = {{1,2},{3,4},{5,6},{7,8},{9,10}};

int *ptr = pole[0];
int i = *(*(pole+2) + 1);
int j = pole[2][1];
printf("%i %i %i\n", ptr[4], i, j);
```

Vícerozměrná pole a funkce

```
void tisk(int pole[][N], int x)
{
    for (int i = 0; i < x; i++) {
        for (int j = 0; j < N; j++) {
            printf("%i ",pole[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}

int main(void)
{
    int pole[][N] = {{1,2},{8,2},{4,5},{7,2},{1,3}};
    tisk(pole, sizeof(pole)/sizeof(int)/N);
    return 0;
}
```

Bubble Sort

- Prochází pole a postupně přehazuje (přebublává) sousedící prvky
- Lze optimalizovat přidáním ukončující podmínky pokud v rámci vnějšího cyklu nedojde k žádnému prohození

```
void bubbleSort(int pole[], int velikost)
{
    int tmp;
    for (int i = 0; i < velikost; i++) {
        for (int j = i+1; j < velikost; j++) {
            if (pole[i] > pole[j]) {
                tmp = pole[i];
                pole[i] = pole[j];
                pole[j] = tmp;
            }
        }
    }
}
```

Insert Sort

- Nové prvky přidává na správné místo do již seřazené části
- Od místa kde se zrovna nachází posouvá prvek postupně doleva, až na jeho místo

```
void insertSort(int pole[], int velikost)
{
    for (int i = 0; i < velikost; i++) {
        int prvek = pole[i], index = i;
        for (; index > 0 && pole[index-1] > prvek;
            pole[index] = pole[index-1], index--);
        pole[index] = prvek;
    }
}
```

Domácí úloha 2

- Zadání je součástí zdrojového kódu
- `https://is.muni.cz/auth/el/1433/podzim2012/IB001/cviceni/velan/zadani_ukol2.c`
- Odevzdat do půlnoci 11.11.2012 (neděle)
- Úkol má vlastní odevzdávárnu
- Opravy úkolu mají vlastní odevzdávárnu