

Základy programovania (IZP)

Ukazovatele (6. cvičenie)

Ing. Pavol Dubovec

Vysoké Učení Technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno- Královo Pole
pdubovec@fit.vutbr.cz



Cvičenia

- Možnosť získať **1 bod** za **aktívnu** účasť na cvičení.
Bod získate za správnu odpoveď na praktickú alebo teoretickú otázku.
- Cvičení je spolu 10. Pre zápočet je nutné získať aspoň **6 z 10** bodov.

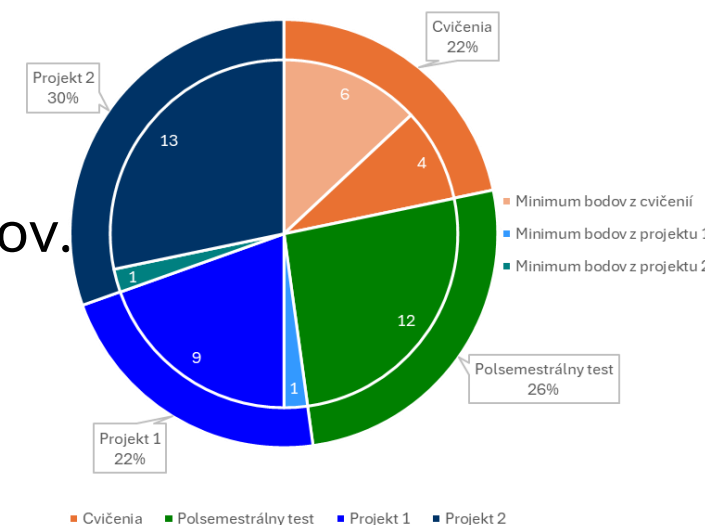
Polsemestrálny test

- Prebieha v **7. týždni** semestra (30.10.2024 a 31.10.2024).
 - Ide teda o **budúci týždeň**.
 - Z tohto dôvodu taktiež **nebude** cvičenie.
- Je možné získať až **12** bodov.

Projekty

- Práca s textom – odovzdanie do **25.10.2024 23:59:59**. Je možné získať až **10 bodov**.
 - Práca s dátovými štruktúrami – odovzdanie v **29.11.2024 23:59:59**. Až **14 bodov**.
Obhajoba v 12 týždni (cca. 5 minútové vysvetlenie funkčnosti s otázkami).
- Pre získanie zápočtu je nutné získať aspoň **1 z 12 bodov z každého z projektov**.

Distribúcia bodov



Akým operandom získame adresu premennej?
Do akého dátového typu ukladáme adresy?

- **Adresu** z operačnej pamäti počítača získame pomocou operátoru referencie (&)

```
int var;  
scanf("%d", &var);  
printf("x = %i\n", var);
```

- Ukazovatele nám umožňujú ukladať tieto **adresy** do premenných dátového typu ukazovateľ. Ukazovateľ vytvoríme pridaním hviezdičky (*) **za** názov dátového typu.

```
int var = 10;  
int *ptr = &var;  
printf("Hodnota var: %d\n", var); // Výstup: 10  
printf("Adresa var: %p\n", &var); // Výstup: Adresa var  
printf("Hodnota na ptr: %d\n", *ptr); // Výstup: 10
```

- Ukazovateľ **musí** byť inicializovaný pred použitím.
- **Dereferencia** – použitie * na získanie hodnoty na adrese, na ktorú ukazovateľ ukazuje.

- Ukazovatele môžu byť **inkrementované** alebo **dekrementované**, aby ukazovali na **dálšiu** alebo **predchádzajúcu** pamäťovú adresu.

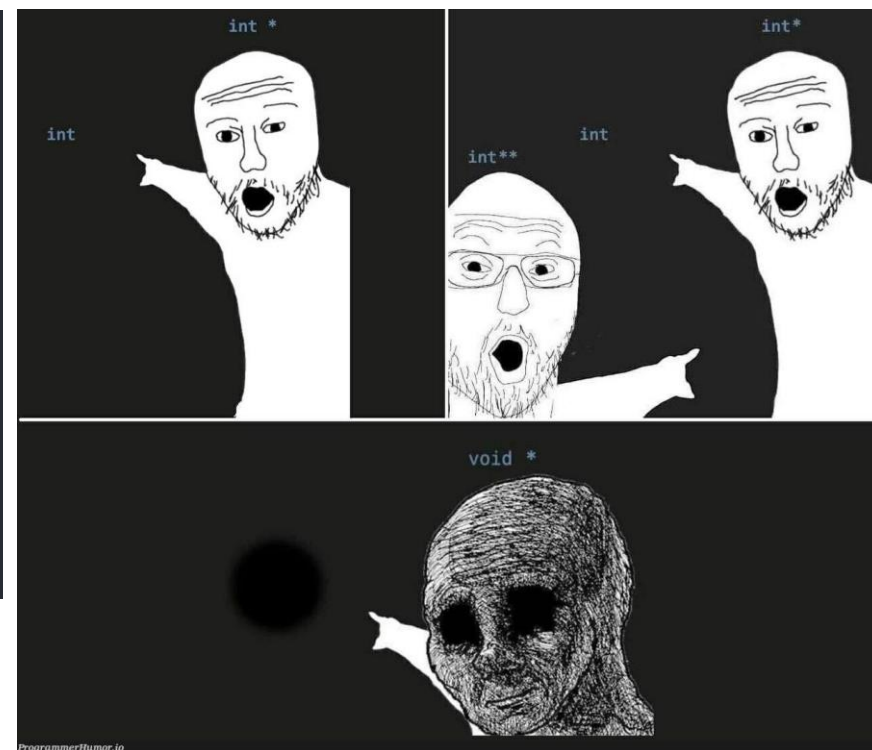
```
int arr[3] = {100, 200, 300};  
int *ptr = arr;  
printf("Prvý prvok: %d\n", *ptr); // Výstup: 100  
ptr++;  
printf("Druhý prvok: %d\n", *ptr); // Výstup: 200
```

- Rozdiel **medzi dvoma ukazovateľmi** môže byť použitý na určenie počtu prvkov medzi nimi.

```
int *start = arr;  
int *end = arr + 2;  
printf("Počet prvkov medzi start a end: %lld\n", end - start);  
// Výstup: 2
```

- Ukazovatele je možné **vrstviť** – ukazovateľ na ukazovateľ na ukazovateľ ...
- Jedným z bežných príkladov, kde sa používa **ukazovateľ na ukazovateľ**, je napríklad **dynamická alokácia pamäte pre dvojrozmerné polia** – matice alebo tabuľky.
- Ilustračný príklad – dynamická alokácia pamäte je preberaná až v 9. cvičení

```
int main() {  
    int rows = 3, cols = 4;  
    // Alokácia pamäte pre riadky (ukazovatele na riadky)  
    int **matrix = (int **) malloc(rows * sizeof(int *));  
    // Alokácia pamäte pre každý riadok (ukazovatele na  
    // prvky riadkov)  
    for (int i = 0; i < rows; i++) {  
        matrix[i] = (int *) malloc(cols * sizeof(int));  
    }  
    free(matrix);  
}
```



ProgrammerHumor.io

<https://programmerhumor.io/backend-memes/how-to-c-pointers/>

Ktorá z nasledujúcich funkcií vykoná operáciu swap ?

```
void swap(int a, int b)
{
    int temp = a;
    a = b;
    b = temp;
}
```

```
void swap(int *a, int *b)
{
    int temp = *a;
    *a = *b;
    *b = temp;
}
```

Predávanie parametrov hodnotou

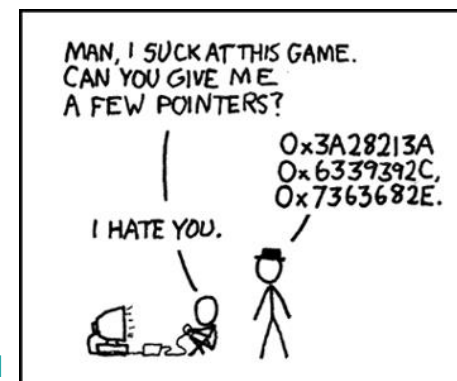
- Funkcia pracuje s **kópiou** premennej.
- Originálna premenná zostáva **nezmenená**.
- Vhodné pre malé a jednoduché dátové typy.
- Tento prístup je **bezpečný**, avšak **spotrebúva viac pamäte**.

```
struct Pair {  
    int a;  
    int b;  
};  
  
struct Pair swapByValue(int a, int b)  
{  
    int temp = a;  
    a = b;  
    b = temp;  
    struct Pair result = {a, b};  
    return result;  
}
```

Predávanie parametrov odkazom

- Funkcia pracuje s **adresou** premennej.
- Originálna premenná môže byť **zmenená**.
- Vhodné pre veľké dátové typy a štruktúry.
- Tento prístup je **efektívny**, avšak **nie vždy chceme meniť originálnu premennú**.

```
void swapByReference(int *a, int *b)  
{  
    int temp = *a;  
    *a = *b;  
    *b = temp;  
}
```



<https://knowyourmeme.com/photos/2162233-xkcd>

Polia

```
void function(int arr[], int size);
```

- Polia sú predvolene prenášané pomocou referencie.
- Keď deklarujeme parameter funkcie ako `int arr[]`, znamená to, že funkcia prijíma ukazovateľ na prvý prvok poľa. `int *arr`, by znamenalo to isté.

Štruktúry

- Štruktúry môžu byť prenášané podľa hodnoty alebo podľa referencie.

```
struct Point {  
    int x, y;  
};  
void printPoint(struct Point p) {  
    printf("Point(%d, %d)\n", p.x, p.y);  
}
```

```
void printPointByRef(struct Point *p) {  
    printf("Point(%d, %d)\n", p->x, p->y);  
}
```

```
int main() {  
    struct Point point = {10, 20}; // Deklarácia a inicializácia štruktúry  
    printPoint(point); // Volanie funkcie s predávaním parametrov pomocou hodnoty  
    printPointByRef(&point); // Volanie funkcie s predávaním parametrov pomocou referencie  
    return 0;  
}
```

Deklarujeme premenné

- `int *p;`
- `int a = 0;`
- `int b = 99;`

Q.1 Priradíte ukazovateľ p na premennú **b**: `p = &b;`

Čo obsahuje **p**, ***p**, **&p**, **b**?

A.1

Q.2 Priradíte ukazovateľ p na premennú **a**: `p = &a;`

Čo obsahuje **p**, ***p**, **&p**, **a**?

A.2

Q.3 Inkrementujte hodnotu v adrese, na ktorú ukazuje **p**: `(*p)++;`

Čo obsahuje **p**, ***p**, **&p**, **a**?

A.3

1. **Zadanie:** Implementujte funkciu, ktorá zistí dĺžku reťazca využitím ukazovateľov.
2. **Zadanie:** Implementujte funkciu, ktorá vráti N čísel danej parity s daným počiatočným prvkom.
3. **Zadanie:** Implementujte funkciu, ktorá zistí počet písmen, slov a viet v reťazci pomocou ukazovateľov. Predpokladajme, že veta končí bodkou a sekvencia bodiek znamená 1 bodku.
4. **Zadanie:** Implementujte funkciu, ktorá vráti adresu hodnoty mediánu v načítanom poli zoradených celých čísel nepárnej dĺžky alebo NULL, ak je pole prázdne, má párnú dĺžku alebo nie je zoradené.
5. **Zadanie:** Implementujte funkciu, ktorá vráti adresu prvku matice, ktorý je najbližší priemeru všetkých prvkov matice.
6. **Zadanie:** Implementujte funkciu, ktorá vráti pole prvkov väčších ako daná hodnota.
7. **Zadanie:** Implementujte funkciu, ktorá spočíta priemernú vzdialenosť všetkých bodov množiny.
8. **Zadanie** 🧠: Implementujte funkciu, ktorá zistí, či zadaná množina tvorí štvorec.