Štruktúra pole, reťazce a práca s textom

1. Štruktúra pole
   - štruktúry pole a jej reprezentácia
   - príklady použitia
2. Reťazce
   - čo je reťazec a reprezentácia reťazca
   - základné operácie s reťazcami
   - štandardné funkcie v jazyku C pre prácu s reťazcami
3. Práca s textom
   - použitie štandardného vstupu a výstupu
   - programové schémy pre čítanie a zápis textu
   - štandardné funkcie v jazyku C pre prácu s textom

Štruktúra pole

- **Pole** je homogénna štruktúra, ktorá pozostáva z prvkov jediného typu, ktorý nazývame **základný typ**
- Štruktúra s **náhodným prístupom**, kde všetky prvky môžu byť vybraté náhodne a sú rovnako sprístupniteľné
- Na každý prvok poľa musíme pristupovať individuálne
- Referencia individuálneho prvku poľa je prístupná cez meno poľa a **index**, určujúci vybratý prvok
- Index nadobúda hodnoty typu, definovaného ako **typ indexu** poľa
  - v jazyku C sa môže použiť ako index iba celé nezáporné číslo (0, 1, 2, 3, ...)

Štruktúra pole

- Definícia poľa

typ identifikátor[počet];

```
int a[10];   // pole 10 celých čísel
```

- Inicializácia prvkov poľa

typ identifikátor[počet] = {p_0, p_1, p_2, ..., $p_{počet-1}$};

```
int a[10] = {2, 5, 23, 4, 3};   // pole 10 celých čísel s prvými piatimi
                                // prvkami inicializovanými
int b[] = {3, 5, 2};   // pole 3 celých čísel s inicializovanými prvkami
```

- Výraz pre výber prvku z poľa (selektor)

identifikátor[index]

```
int x = a[3];   // výber 4. prvku z poľa a a jeho použitie v
                // priraďovacom príkaze
a[i + 3] = 1;   // uloženie hodnoty do prvku poľa s výpočtom jeho indexu
```


Viacrozmerné pole

- Polia, ktorých prvkami sú polia nazývame **viacrozmerné polia**
- Definícia viacrozmerného poľa

typ identifikátor[počet$_1$][počet$_2$]...[počet$_n$];

```
int matica[3][5];  // dvojrozmerné pole
```

- Inicializácia viacrozmerného poľa

typ identifikátok[počet$_1$][počet$_2$]...[počet$_n$] = {...};

```
int matica[2][3] = {{1, 3, 5}, {2, 3, 1}};
```

- Selektor pre viacrozmerné pole (zreťazenie indexov)

identifikátor[index$_1$][index$_2$]...[index$_n$]

```
matica[i][j] = 3;  //priradenie čísla do prvku poľa
```

Reprezentácia poľa

- Reprezentácia poľa je zobrazenie poľa do pamäti počítača
- Výpočet adresy *a* prvku poľa s indexom *i*

$$a = a_0 + i \cdot s$$

- Označenie a_0 predstavuje adresu prvého prvku poľa a *s* je počet slov, ktoré zaberá jeden prvok poľa
  - **v jazyku C budeme pri výpočte adresy prvkov poľa uvažovať vždy *s* = 1**
- Príklad:

```
int p[4];
```

(v jazyku C pre adresu prvého prvku poľa použijeme výraz **`&p[0]`** alebo tiež iba **`p`**)

adresa	hodnota v pamäti
`&p[0] + 0`	`p[0]`
`&p[0] + 1`	`p[1]`
`&p[0] + 2`	`p[2]`
`&p[0] + 3`	`p[3]`

Reprezentácia viacrozmerného poľa

- Viacrozmerné polia si môžeme predstaviť ako **polia polí** a s každým prvkom poľa môžeme pracovať ako s poľom
- Pre *n*-rozmerné pole *a* určíme adresu prvku s indexmi $i_1, i_2, ..., i_n$ nasledovne

$$a = a_0 + ((...(i_1 \cdot r_2 + i_2) \cdot r_3 + ... + i_{n-1}) \cdot r_n + i_n) \cdot s$$

- Platí to isté ako v predošlom prípade, pričom $r_2, ..., r_n$ sú rozmery poľa
- Príklad dvojrozmerného poľa:

```
int p[2][3];
```

(adresa prvého prvku je daná výrazom `&p[0][0]`)

adresa	hodnota v pamäti
`&p[0][0] + 0 * 3 + 0`	`p[0][0]`
`&p[0][0] + 0 * 3 + 1`	`p[0][1]`
`&p[0][0] + 0 * 3 + 2`	`p[0][2]`
`&p[0][0] + 1 * 3 + 0`	`p[1][0]`
`&p[0][0] + 1 * 3 + 1`	`p[1][1]`
`&p[0][0] + 1 * 3 + 2`	`p[1][2]`

Príklad: Nájdenie prvočísiel

- **Formulácia úlohy**
  - Na štandardný výstup vypíšte všetky prvočísla po zadanú hornú hranicu.
- **Návrh riešenia**
  - Jedna z možností je použitie Erathostenovho sita. Základnou myšlienkou je fakt, že násobky prvočísiel nie sú prvočísla. Takže na pomyselnom výpise celého intervalu začíname od najmenšieho prvočísla a škrtáme jeho násobky. Potom sa posunieme na najbližšie nepreškrtnuté číslo, ktoré musí byť prvočíslo a ďalej vyradzujeme jeho násobky na celom intervale. Takto postupujeme, až nedosiahneme hornú hranicu intervalu. Maximálna horná hranica bude obmedzená konštantou *MAX*.
- **Návrh údajových štruktúr**
  - Pre hornú hranicu použijeme premennú typu celé číslo *n*. Pre zobrazenie výpisu všetkých čísel intervalu, na ktorom hľadáme prvočísla, použijeme pole krátkych celých čísel s názvom *sito*, pričom prípustné hodnoty budú 1 a 0 v zmysle nepreškrtnutého a preškrtnutého čísla.

Reťazce

- **Reťazcom** nazývame postupnosť znakov (prvkov typu char) reprezentujúcu textový údaj
- Reťazce reprezentujeme v jazyku C ako jednorozmerné polia znakov
- Posledný znak ukončujúci postupnosť znakov v reťazci je vždy **nulový znak** (znak '\0')
- Pri definícii poľa znakov musíme preto vždy uvažovať o jednom (koncovom) znaku naviac

Reťazce

- Definícia reťazca

```
char identifikátor[počet + 1];

char a[10];   // pole pre uloženie reťazca dĺžky 9
```

- Inicializácia reťazca

```
char identifikátor[] = reťazcová_konštanta;

char a[] = {'p', 'r', 'o', 'g', 'r', 'a', 'm', '\0'};   // vymenovaním znakov
char b[] = "program";   // reťazcovou konštantou
```

pri inicializácii poľa reťazcovou konštantou sa vyhradí miesto pre zadaný počet znakov a jeden koncový znak automaticky

- Na reťazec sa často odkazujeme ako na celok tzn. adresou prvého prvku poľa (používame samotný identifikátor poľa), čo vyžaduje prácu so smerníkmi
- Pre prístup k jednotlivým znakom používame selektor ako pri poliach s iným základným typom

Základné operácie s reťazcami

- Niektoré základné operácie s reťazcami:
  - zistenie dĺžky reťazca – `strlen()`
  - kopírovanie reťazca – `strcpy()`, `strncpy()`
  - konkatenácia (spojenie) reťazcov – `strcat()`, `strncat()`
  - porovnanie reťazcov – `strcmp()`, `strncmp()`
  - nájdenie prvého/posledného výskytu znaku v reťazci – `strchr()`, `strrchr()`
  - prevod znakv na veľké/malé písmená – `strlwr()`, `strupr()`
- Uvedené operácie s reťazcami sú implementované funkciami v štandardnej knižnici prekladača jazyka C
  - k zdrojovému textu je potrebné pripojiť hlavičkový súbor `string.h`

Príklad: Nájdenie reťazca v texte

- **Formulácia úlohy**
  - Pre zadaný text a reťazec nájdite pozíciu prvého výskytu reťazca v texte.
- **Návrh riešenia**
  - Vezmeme prvý znak reťazca a nájdeme jeho výskyt v texte. Potom od tohto miesta skúsime porovnať aj zvyšok reťazca s textom. Ak sa zhodujú, skončili sme, ak nie hľadáme ďalší výskyt prvého znaku reťazca v texte. Maximálna dĺžka textu a reťazca bude obmedzená konštantou *MAX*.
- **Návrh údajových štruktúr**
  - Pre implementáciu textu aj hľadaného reťazca použijeme polia znakov *text* a *ret*.

Práca s textom

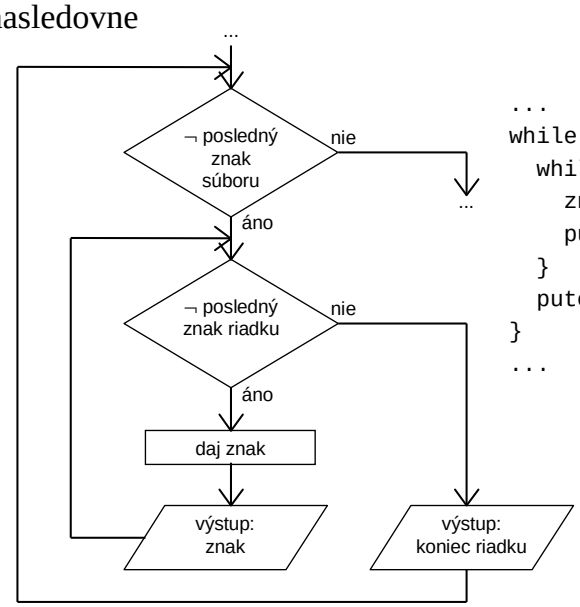
- Súbory s prvkami typu znak (char) sú dôležité pri výpočtoch a spracovaní údajov, pretože predstavujú **prepojenie medzi počítačom a používateľom**
- Čitateľný vstup resp. výstup programu predstavuje znakovú postupnosť
- Komunikácia medzi výpočtovým procesom a používateľom je realizovaná dvoma súbormi (vstupný a výstupný)
- Tieto dva súbory označujeme ako **štandardný vstup** a **štandardný výstup**
- Tieto súbory reprezentujú štandardné vstupné a výstupné médium počítačového systému (klávesnica, displej a pod.)
- Štandardné knižnice prekladača jazyka C obsahujú realizácie týchto súborov ako aj operácie so vstupom a výstupom vo forme funkcií
  - k zdrojovému textu je potrebné pripojiť hlavičkový súbor `stdio.h`

Práca s textom

- Text predstavuje postupnosť znakov, ktorá je ukončená znakom konca súboru `EOF`
- Text môžeme deliť na podštruktúry, napr. kapitoly, články, odseky, riadky, slová a pod.
- Jednotlivé podštruktúry textu sú oddelené rôznymi oddeľovacími znakmi (tiež nazývané biele znaky)
- Najčastejšie uvažujeme text ako postupnosť riadkov, pričom riadok je postupnosť znakov ukončená znakom konca riadku `'\n'`
- Uvažujme najjednoduchšie operácie pre prácu s textom
  - čítanie znaku zo vstupného súboru
  - zápis znaku do výstupného súboru
  - test konca súboru
  - test konca riadku

Zápis do textového súboru

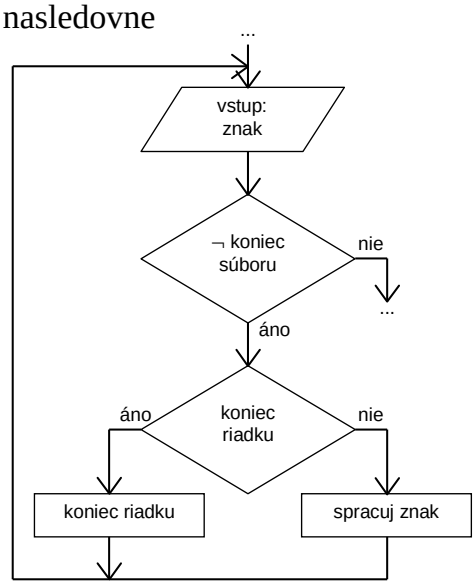
- Základná programová schéma pre zápis do textového súboru môže vyzerať nasledovne




```
...
while (!posledznyZnakSuboru()) {
  while (!posledznyZnakRiadku()) {
    znak = dajZnak();
    putchar(znak);
  }
  putchar('\n');
}
...
```

Čítanie textového súboru

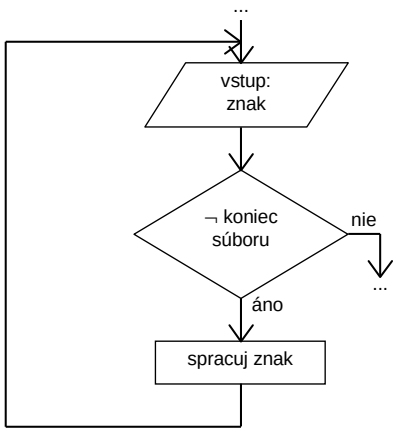
- Základná programová schéma pre čítanie textového súboru môže vyzerať nasledovne




```
...
while ((znak = getchar()) != EOF) {
  if (znak == '\n')
    koniecRiadku();
  else
    spracujZnak(znak);
}
...
```


Čítanie textového súboru

- Niekedy nie je riadková štruktúra textu podstatná
- Zjednodušená programová schéma pre čítanie textového súboru potom vyzerá nasledovne




```
...
while ((znak = getchar()) != EOF) {
  spracujZnak(znak);
}
...
```

Štandardné vstupno-výstupné funkcie

- Väčšina programovacích jazykov dovoľuje v operáciách s textom pracovať aj s inými typmi údajov ako znakmi (int, float a pod.)
- Vstupno-výstupné operácie preto zvyčajne obsahujú konverziu textu (postupnosti znakov) na tieto typy údajov
- V jazyku C môžeme využívať nasledovné funkcie pre prácu so štandardným vstupom a výstupom:
  - načítanie znaku zo vstupu – `getchar()`
  - zápis znaku na výstup – `putchar()`
  - formátovaný vstup – `scanf()`
  - formátovaný výstup – `printf()`
  - načítanie riadku zo vstupu – `gets()`
  - zápis riadku na výstup – `puts()`