Reťazce v jazyku C

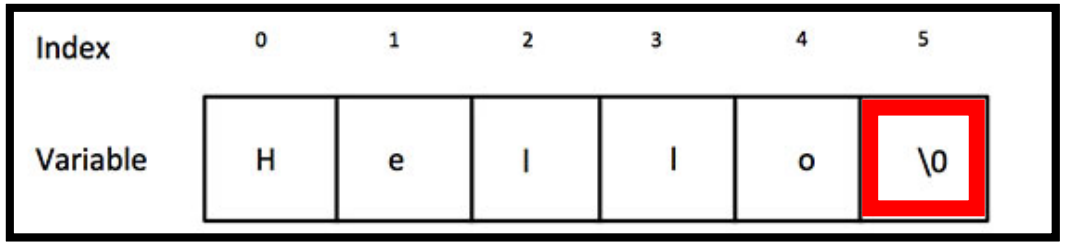
Reťazce v jazyku C

Je to jednorozmerné pole znakov ukončené znakom '\0' (NULL Terminator).

Načo pri písaní kódu pamätať ?
Veľkosť poľa musí byť vždy o jeden znak väčšia, ako je dĺžka reťazca, ktorý v ňom chceme uchovať.

Ak sa reťazec neukončí terminátorom, tak sa za reťazec **považuje celá nasledujúca pamäť RAM** až po najbližší znak **'\0'**.

(Teda problém!)



Reťazec je možné inicializovať podobne ako jednorozmerné pole:

char str[] = "Hello"; // veľkosť alokovanej pamäte **5 + terminátor = 6**
char str[20] = "Hello"; // veľkosť alokovanej pamäte **20 --> môžem použiť 19 znakov**
char* string = "Hello";// veľkosť alokovanej pamäte **5 + terminátor = 6** ale nemožno
// meniť obsah „teda **len na čítanie!**“.

Reťazce v jazyku C

Čo by sme dnes ešte mali spomenúť ?

- Načítanie formátovaného textu
- Zobrazovanie formátovaného textu

- Zistenie dĺžky reťazca

- Zistenie či je znak veľké alebo malé písmeno
- Prevod znaku na veľké alebo malé písmeno

Reťazce v jazyku C

Čo by sme dnes ešte mali spomenúť ?

- Načítanie formátovaného textu
- Zobrazovanie formátovaného textu

#include <stdio.h>

- Zistenie dĺžky reťazca

#include <string.h>

- Zistenie či je znak veľké alebo malé písmeno
- Prevod znaku na veľké alebo malé písmeno

#include <ctype.h>

Reťazce v jazyku C - Načítanie reťazca z klávesnice

Reťazec môžeme načítať pomocou funkcie **scanf()**

scanf(“%<údajový _typ> ”, &<názov_premennej>)

& (hovorí o adrese) - používame iba ak načítavame jednu premennú! Pri reťazcoch sa nepoužije

Pre načítanie reťazca do premennej string môžeme postupovať napr. takto:

```
char string[50] ;
scanf(“%s ”, string);
```

Ak ale používateľ zadá text obsahujúci medzeru načíta sa len časť pred medzerou!

" Hello world! " bude len *"Hello"*!

Pre načítanie celého riadku je **scanf()** potrebné zapísať v tvare:

scanf(“%[^\n] ”, string);

Reťazce v jazyku C - Vypísanie reťazca na monitor

Už by sme túto funkciu mali veľmi dobre poznať ale pre istotu ...

Ako ju použiť ?

printf(“Ľubovoľný text %<údajový _typ> iný text ... ”, <názov_premennej>);

printf(“Zadal si takýto text: %s ”, string);

Reťazce v jazyku C - Zistenie dĺžky reťazca

Ak chceme zistiť dĺžku reťazca, môžeme na to využiť funkciu **strlen()**

- Predpokladane, že reťazec je zadaný korektne a obsahuje terminátor!

```
int dlzka = strlen(string);
```

Ak chcete použiť túto funkciu v slučke **for()**, nikdy to nerobte takto!

```
for (int i = 0; i < strlen(string); i++)
```

Program pôjde ale príkaz **strlen(string)** sa vykoná v každej iterácii.

Lepšie je definovať ukončovaciu podmienku pred samotným for-om.

Bežné operácie so znakmi

Zistenie či je veľké písmeno:

int isupper(znak)

Zistenie či je malé písmeno:

int islower(znak)

Prevod na malé písmeno:

int tolower(znak)

Prevod na veľké písmeno:

int toupper(znak)

Bežné operácie so znakmi

```
#include <stdio.h>
#include <string.h>
#include <ctype.h>
int main()
{
 char text[100];                        // Vytvoríme dostatočne veľké pole
 int dlzka;
 printf("Zadaj text\n >>");
 scanf("%[^\n]", text);                 //Načítame od používateľa text (až po stlačenie ENTER)

 dlzka = strlen(text);                  //Zistenie dĺžky reťazca
 char Velke[dlzka+1], Male[dlzka+1];    // Deklarácia ďalších polí, teraz už s konkrétnou dĺžkou, nezabudnúť na „Arniho“

 for(int i=0;i<=dlzka;i++)              // POZOR! Aj pri kopírovaní textu musím mať na pamäti ukončenie reťazca
 {
     Male[i] = text[i];
     Velke[i] = text[i];
     if(isupper(Male[i]))
             Male[i]=tolower(Male[i]);
     if(islower(Velke[i]))
             Velke[i]=toupper(Velke[i]);
  }
 printf(">>> %s \n>>> %s \n>>> %s\n",text,Velke,Male);
 return 0;
}
```

Ternárny operátor

Je to veľmi zaujímavá konštrukcia, ktorá môže nahradiť príkaz if-else

Podmienka ako ju už poznáme:

```
if(podmienka)
        Procedura_1;
else
        Procedura_2;
```

```
int a=1, b=2, c;
if(a>b)
        c=1;
else
        c=0;
```

Tento zápis môžeme nahradiť **ternárným operátorom** nasledovne:

```
podmienka ? (Procedura_1): (Procedura_2)
```

```
a>b ? (c=1) :  (c=0) ;   alebo aj takto   c = a>b ? 1 : 0;
```

2R polia v jazyku C

Dvojrozmerné pole si môžeme predstaviť ako dvojrozmernú tabuľku (maticu) s istým číslom riadku a stĺpca, v bunkách tabuľky sú samotné hodnoty.

V jazyku C 2R pole deklarujeme tak, že uvedieme jeho typ, názov a veľkosť:

<typ> **<názov>**[**<veľkosť_v>**][**<veľkosť_h>**];

int **Matica**[**3**][**3**];

Tak ako to bolo pre 1R pole aj pre 2R pole je vhodné vykonať inicializáciu. Takto predídeme situácii, že z pamäte budeme čítať náhodné hodnoty, ktoré tam ostali po iných programoch. Inicializáciu je možné vykonať dvojako.

Malé 2R pole

```
int Matica[3][3] ={
                    {0, 0, 0 },
                    {0, 0, 0 },
                    {0, 0, 0 },
                   };
```

Veľké 2R pole

```
int size = 256;
int OBR[size][size];

for(int i=0; i<size; i++)
  for(int j=0; j<size; j++)
  {
     OBR[i][j] = 0;
  }
```

K prvkom 2R poľa pristupujeme rovnako ako v prípade 1R poľa s tým rozdielom, že najprv určíme súradnicu vo vertikálnom smere a potom horizontálnom smere (resp. prv riadok a potom stĺpec).

Čítane z bunky poľa:

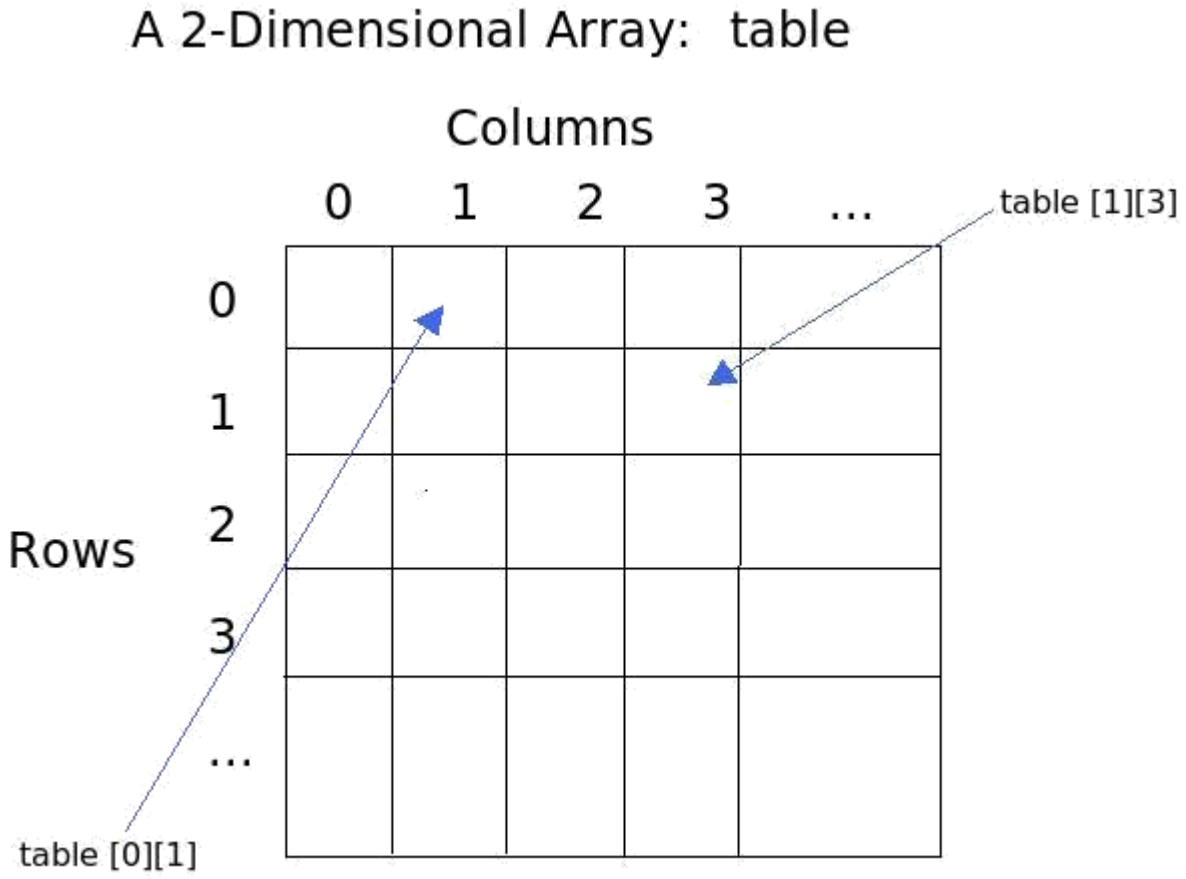
```
int a = pole[0][0]; // v a bude bunka poľa som súradnicou (0 0)
int b = pole[1][10]; // v b  bude bunka poľa som súradnicou (1 10)
```

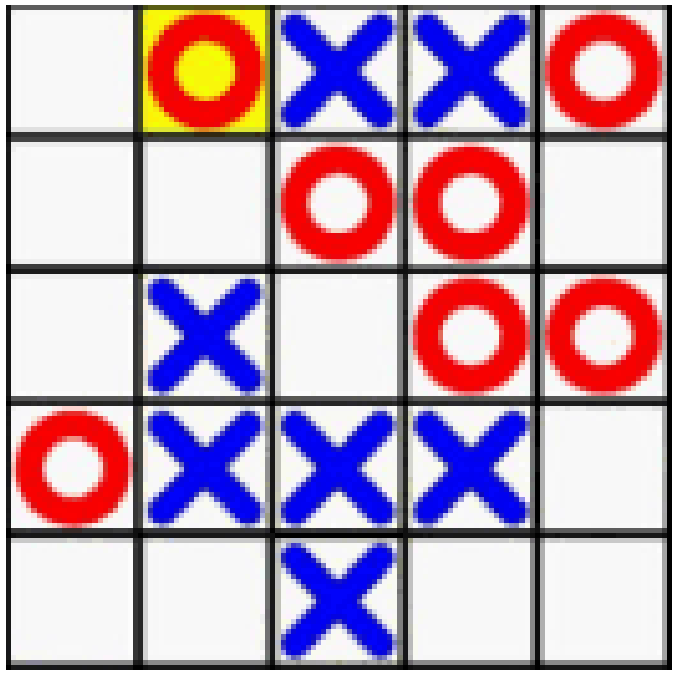
Zápis do bunky poľa:

```
pole[4][1] = 10;
pole[3][2]= a; // premenná a bola niekde predtým deklarovaná a
               // inicializovaná
```

Načítanie zo štandardného vstupu:

```
scanf("%d", &pole[1][2]);
```





Samostatná práca