Reťazce v jazyku C

Textový reťazec je v jazyku C štandardne uložený v poli ako postupnosť znakov (`char`) ukončená znakom s kódom 0.

- Nemusíme teda ukladať zvlášť dĺžku, reťazec ide po prvú nulu
- Pozor, rozdiel medzi znakom s kódom 0 a znakom '0' s kódom 48
- Reťazce teda nemôžu obsahovať vo vnútri znak s kódom 0, ten je rezervovaný na ukončovanie
- Na reťazec s *n* znakmi potrebujeme pole dĺžky aspoň *n*+1, lebo jeden znak sa minie na ukončovací symbol

Teraz by sme vedeli prepísať `nacitajRiadok` a `vypisRetazec`, ale netreba, lebo existujú v knižniciach

Inicializácia reťazcov

- Chceme vytvoriť premennú str obsahujúce reťazec `Ahoj` spolu s koncom riadku
- Prvý spôsob je zdĺhavý:

```
char str[10];
str[0] = 'A';
str[1] = 'h';
str[2] = 'o';
str[3] = 'j';
str[4] = '\n'; // znak pre koniec riadku
str[5] = 0; '\0'
```

- Alebo ako inicializácia poľa: `char str[10]={'A','h','o','j','\n',0};`
- Špeciálna skratka: `char str[10]="Ahoj\n";`

- Ako vytvoríme prázdny reťazec?

Reťazec je naozaj pole

Znaky reťazca môžeme meniť

```
char a[100] = "vlk";
char ch = a[0]; // ch obsahuje hodnotu 'v'
char b[100] = "pes";

b[0] = ch;     // priradíme do jedného prvku reťazca premmennú typu char.
Výsledkom je 'ves'.
b[0] = 'd';    // priradíme do jedného prvku reťazca konštantný znak.
Výsledkom je 'des'.
b[0] = a[1];   // priradíme do jedného prvku reťazca prvok iného reťazca.
Výsledkom je 'les'.
```

Reťazec sa nedá kopírovať jednoduchým priradením, nemôžeme teda spraviť

```
char a[100];
a = "Ahoj";           // chyba
char b[100] = "Ahoj"; // ok - inicializacia
```

```
a = b;                  // chyba
```

Reťazce sa nedajú ani porovnávať pomocou ==, !=, < atď

Kopírovanie a porovnávanie si musíme naprogramovať cez cykly, alebo použiť hotové funkcie z knižníc.

Knižnica cstring

Obsahuje mnohé funkcie na prácu s reťazcami, napríklad tieto:

- `strlen(retazec)`: vráti dĺžku reťazca
- `strcpy(kam, co)`: skopíruje reťazec `co` do reťazca `kam` (pole `kam` musí byť dosť dlhé)
- `strcat(kam, co)`: za koniec reťazca `kam` pridá reťazec `co` (pole `kam` musí byť dosť dlhé)
- `strcmp(retazec1, retazec2)`: vráti nulu ak sa reťazce rovnajú, kladné číslo keď je prvý neskôr v abecednom poradí, záporné číslo, ak je skôr. Pozor, to či je skôr alebo neskôr sa berie podľa kódov znakov, takže napr. `'Z'` je skôr ako `'a'`.

Všetky tieto funkcie by sme si však vedeli naprogramovať aj sami. Tu je napríklad výpočet dĺžky:

```
int myStrLen(char a[]) {
    int n=0;
    while(a[n]!=0) {  n++; }
    return n;
}
```

- čo bude funkcia robiť ak reťazcu chýba na konci 0?

Dve verzie kopírovania:

```
void myStrCpy(char a[], char b[]) {
    /* Skopiruj obsah retazca b do retazca a.
     * Pole a musi mat dost miesta. */
    int n = 0;
    while (b[n] != 0) {
        a[n] = b[n];
        n++;
    }
    a[n] = 0; // reťazec musí končiť 0
}

void myStrCpy2(char a[], char b[]) {
    /* Skopiruj obsah retazca b do retazca a.
     * Pole a musi mat dost miesta. */
    for (int i = 0; i <= strlen(b); i++) {
        a[i] = b[i];
    }
}
```

Namiesto `strcmp` naprogramujeme len test na rovnosť:

```
bool rovnostRetazcov(char a[], char b[]) {
    /* vrati true ak su retazce a, b rovnake, inak vrati false */

    for (int i = 0; a[i] != 0 || b[i] != 0; i++) {
        if (a[i] != b[i]) return false;
    }
    return true;
}
```

- Ako bude prebiehať funkcia, ak jeden reťazec je začiatkom druhého?

Načítavanie a vypisovanie reťazcov

- Bežné načítanie z konzoly do reťazca (`cin >> str`) načíta jedno slovo
    - Preskočí biele znaky (medzery, konce riadkov, tabulátory), potom prečíta všetko po ďalší biely znak (alebo koniec vstupu) a uloží do premennej.
    - Pri čítaní je vhodné nastaviť maximálny počet znakov na načítanie, aby sme nevyšli z poľa

- Na načítanie jedného riadku je možné použiť funkciu `getline`. Načíta až po koniec riadku, ten zahodí.
- Vypisovanie funguje normálne pomocou `cout << str`

```
#include <iostream>
using namespace std;

int main(void) {
    const int maxN = 100;
    char str[maxN], str2[maxN], str3[maxN];

    cin.getline(str, maxN); // cely riadok, ale najviac maxN-1 znakov

    cin.width(maxN); // najviac maxN-1 znakov pri najbližšom načítaní
    cin >> str2;     // nacita jedno slovo
    cin.width(maxN); // najviac maxN-1 znakov pri najbližšom načítaní
    cin >> str3;     // nacita dalsie slovo

    cout << "str: \"" << str << "\"" << endl;
    cout << "str2: \"" << str2 << "\"" << endl;
    cout << "str3: \"" << str3 << "\"" << endl;
}
```

Príklad behu programu (prvé dva riadky zadal užívateľ, na začiatku a konci každého je medzera)

```
 a b c
 g h i
str: " a b c "
str2: "g"
str3: "h"
```

Algoritmy s reťazcami

Prácu s reťazcami si precvičíme na niekoľkých menších príkladoch.

Vyhľadávanie podreťazca

Chceme zistiť, či a kde sa v reťazci nachádza určité slovo alebo iná vzorka.

```
#include <iostream>
#include <cstring>
using namespace std;

int find(char text[], char pattern[]) {
    /* Vráti -1 ak sa reťazec pattern nevyskytuje v reťazci text,
     * inak vráti polohu jeho prvého výskytu. */

    int n = strlen(text);
    int m = strlen(pattern);
    for (int i = 0; i < n - m + 1; i++) {
        int j = 0;
        while (j < m && text[i + j] == pattern[j]) {
            j++;
        }
        if (j == m) {
            return i;
        }
    }
    return -1;
}

int main(void) {
    const int maxN = 2000;
    char A[maxN], B[maxN];

    cout << "Zadaj text: ";
    cin.getline(A,maxN);
    cout << "Zadaj vzorku: ";
    cin.getline(B,maxN);
    cout << find(A,B) << endl;
}
```

- Predpočítame si dĺžky a uložíme do premenných, aby sa zbytočne nerátali znova a znova
- Vedeli by sme do poľa uložiť polohy všetkých výskytov?

Prevod čísla na reťazec

Máme danú premennú *x* typu int, chceme ju uložiť v desiatkovej sústave do reťazca.

- Zvyšok po delení 10 je posledná cifra, uložíme si ju do reťazca, vydelíme *x* desiatimi
- Opakujeme, kým nespracujeme celé číslo.
- Prevod z čísla *c* (0..9) na cifru: '0'+*c*
- Nezabudneme na ukončovací znak 0
- Dostaneme číslo v opačnom poradí, napr pre x=12 budeme mať reťazec {'2', '1', 0}
- Preto ešte celé číslo otočíme naopak.

```
void reverse(char a[]) {
    int n = strlen(a);
    int i = 0;
    int j = n - 1;
    while (i < j) {
        char tmp = a[i];
        a[i] = a[j];
        a[j] = tmp;
        i++; j--;
    }
}

void int2str(int x, char a[]) {
    /* prevedie kladne cele cislo x na retazec,
     * vysledok ulozi do retazca a, ktory musi mat dost miesta. */
    assert(x > 0);

    int n = 0;
    while(x > 0) {
        a[n] = '0' + x % 10;
        x /= 10;
        n++;
    }
    a[n] = 0;

    /* teraz je cislo naopak, treba otocit */
    reverse(a);
}
```

- Ako upravíme funkciu, aby fungovala aj pre x=0, prípadne záporné x?
- Pozor na rozdiel medzi znakom 0 a '0' (a medzi reťazcom "0")

Formátovanie čísla

- Chceme číslo zapísať do reťazca a doplniť naľavo medzerami na šírku width.

```
const int maxN = 100;

void formatInt(int x, char A[], int width) {
    /* číslo x konvertujeme na reťazec
     * a uložíme do poľa A zarované doprava na šírku width */

    /* najprv x uložíme do pomocného reťazca B  a zrátame jeho dĺžku n */
    char B[maxN];
    int2str(x, B);
    int n = strlen(B);

    /* do A dáme width-n medzier a ukončovaciu 0 */
    assert(n <= width);
    int i;
    for (i = 0; i < width - n; i++) {
        A[i] = ' ';
    }
    A[i] = 0;

    /* za A prikopírujeme B */
    strcat(A, B);
}
```

- Čo by sa stalo, ak by sme nedali do *A* ukončovaciu 0?
- Vedeli by ste prepísať program, aby pracoval priamo v poli *A* (bez poľa *B*)?

Využijeme na vypísanie pekne zarovnanej tabuľky faktoriálov:

```
int factorial(int n) {
    int result = 1;
    for (int i = 1; i <= n; i++) {
        result *= i;
    }
    return result;
}

int main(void) {
    char A[maxN];
    int n = 12;
    for (int i = 1; i <= n; i++) {
        int x = factorial(i);
        formatInt(i, A, 2);
        cout << A << "! = ";
        formatInt(x, A, 10);
        cout << A << endl;
    }
}
 1! =          1
 2! =          2
 3! =          6
 4! =         24
 5! =        120
 6! =        720
 7! =       5040
 8! =      40320
 9! =     362880
10! =    3628800
11! =   39916800
12! =  479001600
```

Dalo by sa aj jednoduchšie pomocou nastavenia `width` v `cin`:

```
int main(void) {
    int n = 12;
    for (int i = 1; i <= n; i++) {
        int x = factorial(i);
        cout.width(2);
        cout << i << "! = ";
        cout.width(10);
        cout << x << endl;
    }
}
```

Zalamovanie riadkov

- Máme reťazec s nejakým textom, v ktorom sa vyskytujú rôzne biele znaky, napríklad medzery a konce riadkov. Máme danú šírku riadku *W*, napr. 80 znakov. Úlohou je ho upraviť tak:

- aby na každom riadku bolo najviac *W* znakov, pričom nový riadok začína tam, kde by už ďalšie slovo presahovalo cez *W*
- medzi dvoma slovami má byť vždy buď jedna medzera alebo jeden koniec riadku
- predpokladáme, že žiadne slovo nemá viac ako *W* znakov

```
Zadavaj text ukonceny prazdnym riadkom.
A AA A AAA
A  A AA AAA
AAA A AA AA  A

Zadaj sirku riadku:
5
Sformatovany odstavec:
A AA
A AAA
A A
AA
AAA
AAA A
AA AA
A
Zadavaj text ukonceny prazdnym riadkom.
Martin Kukucin: Do skoly.    Vakacie sa koncia. Ondrej Rybar sa vse zamysli
nad marnostou sveta i vsetkeho, co je v nom.
Predstupuje mu tu i tu pred oci profesor, ako stoji pred ciernou tabulou,
drziac kruzidlo v ruke a demonstruje pamatnu poucku Pytagorovu.
A zimomriavky naskakuju na chrbat, lebo s geometriou stoji od pociatku na
nohe valecnej.
Ani matematika nenie lepsia, menovite odvtedy, co sa do nej vplichtili
miesto cisel vsakove litery.
Neraz hutal, naco ich ucenci vpustili do matematiky - ved i bez nich je
dost strapata: ci sa im malilo cisel a tak preto vsantrocili medzi ne a a b
a ci fantazia sa im tak rozihrala,
ze prekrocila hranice cisel celych, zlomkov obycajnych i desatinnych i
bohvieakych, a zabludila na nivy, kde rastu nestastne litery?
 ,Uz akokolvek,' huta Ondro, ,litery tam nemaju co hladat. Tazko je uverit,
ze a/b = c, lebo nevies, co je a, alebo b.'

Zadaj sirku riadku:
50
Sformatovany odstavec:
Martin Kukucin: Do skoly. Vakacie sa koncia.
Ondrej Rybar sa vse zamysli nad marnostou sveta i
vsetkeho, co je v nom. Predstupuje mu tu i tu pred
oci profesor, ako stoji pred ciernou tabulou,
drziac kruzidlo v ruke a demonstruje pamatnu
poucku Pytagorovu. A zimomriavky naskakuju na
chrbat, lebo s geometriou stoji od pociatku na
nohe valecnej. Ani matematika nenie lepsia,
menovite odvtedy, co sa do nej vplichtili miesto
cisel vsakove litery. Neraz hutal, naco ich ucenci
vpustili do matematiky - ved i bez nich je dost
strapata: ci sa im malilo cisel a tak preto
vsantrocili medzi ne a a b a ci fantazia sa im tak
rozihrala, ze prekrocila hranice cisel celych,
zlomkov obycajnych i desatinnych i bohvieakych, a
zabludila na nivy, kde rastu nestastne litery? ,Uz
akokolvek,' huta Ondro, ,litery tam nemaju co
hladat. Tazko je uverit, ze a/b = c, lebo nevies,
```

```
co je a, alebo b.'
```

Plán: úlohu si rozdelíme na viac častí

- Prerobíme reťazec tak, aby sme všetky biele znaky nahradili medzerami. Na rozpoznanie bielych znakov použijeme funkciu `isspace` z knižnice `cctype`.
- Každý súvislý úsek medzier nahradíme práve jednou medzerou, zmažeme medzery na začiatku a na konci.
    - Podobá sa na príklad s vyhadzovaním núl z poľa z cvičení
    - Viac možností na riešenie, napríklad znaky presýpame do nového poľa. My ale použijeme len jedno pole
- Niektoré medzery nahradíme koncom riadku, aby každý riadok mal šírku najviac *W*.
- Spravíme načítanie a vypísanie.

```
#include <iostream>
#include <cstring>
#include <cctype>
#include <cassert>
using namespace std;

void simplify(char A[]) {
    /* V retazci A nahradi kazdy suvisly usek bielych znakov prave jednou
medzerou.
     * Na zaciatku a konci retazca nebudu medzery. */

    int i;

    /* prepis hocijake biele znaky na medzeru */
    for (int i = 0; A[i] != 0; i++) {
        if (isspace(A[i])) {
            A[i] = ' ';
        }
    }

    int kam = 0; /* prve este neobsadene miesto */
    char prev = ' '; /* predchadzajuci znak */

    for (int i = 0; A[i] != 0; i++) {
        /* ak nemame viac medzier po sebe, skopirujeme znak */
        if (A[i] != ' ' || prev != ' ') {
            A[kam] = A[i];
            kam++;
        }
        /* zapamatame si posledny znak */
        prev = A[i];
    }

    /* zrusime pripadnu medzeru na konci */
    if (kam > 0 && A[kam - 1] == ' ') {
        kam--;
    }

    /* retazec ukoncime nulou */
    A[kam] = 0;
}

bool breakLines(char A[], int width) {
```

```
    /* Preformatuje odstavec na sirku riadku width, vyhodi zbytocne
medzery.
     * Dlzka kazdeho slova musi byt najviac width, inak funkcia vrati false
*/

    simplify(A);
    int n = strlen(A);

    int zac = 0;  /* index prveho pismena v riadku */
    while (zac < n) {
        int kon = zac + width;  /* potencialny koniec riadku */
        /* ak uz nemame dost pismen na cely riadok */
        if (kon > n) {
            kon = n;
        }
        /* ak sme na konci, pridame koniec riadku za koniec retazca */
        if (kon == n) {
            A[kon] = '\n';
            A[kon + 1] = 0;
            n++;
        } else {
            /* ideme späť, kým nenájdeme medzeru */
            while (kon > zac && A[kon] != ' ') {
                kon--;
            }
            /* nenašli sme medzeru: slovo bolo príliš dlhé. */
            if (kon == zac) {
                return false;
            }
            /* medzeru prepíšeme na koniec riadku */
            assert(A[kon]==' ');
            A[kon] = '\n';
        }
        /* za koncom riadku bude novy zaciatok */
        zac = kon + 1;
    }
    return true;
}

int main(void) {
    const int maxN = 2000;
    char A[maxN];

    cout << "Zadavaj text ukonceny prazdnym riadkom." << endl;
    while (true) {
        /* nacitame jeden riadok */
        char tmp[maxN];
        cin.getline(tmp, maxN);
        /* ak je prazdny, koncime nacitavanie */
        if (strcmp(tmp, "") == 0) {
            break;
        }
        /* ak je miesto v poli A, pridame do neho novy riadok */
        if (strlen(A) + strlen(tmp) + 2 < maxN) {
            strcat(A, tmp);
            strcat(A, "\n");
        } else {
            cout << "Text je prilis dlhy." << endl;
            return 1;
        }
    }
```

```
    cout << "Zadaj sirku riadku:" << endl;
    int width;
    cin >> width;

    breakLines(A, width);
    cout << "Sformatovany odstavec:" << endl;
    cout << A;
}
```

- Akú zložitosť má načítanie vzhľadom na celkový počet načítaných písmen? Dalo by sa zlepšiť?

Zhrnutie

- Reťazec je pole znakov, za posledným znakom reťazca dáme špeciálny znak s kódom 0
- V knižnici `cstring` sú funkcie na porovnávanie a kopírovanie reťazcov atď. a pomocou `cin` a `cout` môžeme reťazce načítavať a vypisovať.
- Ďalšie funkcie si vieme naprogramovať aj sami, zvyčajne jednoduchá práca s poľom