

Typy

Skalární typy

void triviální typ, s jehož jedinou hodnotou se nepracuje (a proto není ani pojmenovaná). Označuje například

- prázdný seznam parametrů bezparametrické funkce
`double rand(void)`
- nebo typ výsledku procedury, která nestanoví svůj výsledek
`void printresult(int res)`
- nebo je použit v typu ukazatele na data nespecifikovaného typu
`void* ptr;`

Skalární typy

`int`, `long int`, `unsigned int` typy celých čísel

`float`, `double` typy desetinných v pohyblivé řádové čárce (racionální aproximace reálných čísel).

`char` typ znaků. Bez knihovních rozšíření pracuje C jen se znakovou sadou ASCII.

`bool` typ logických hodnot (`false` a `true`) z knihovny `stdbool`. Původní C s explicitně pojmenovaným typem nepracovalo; nepravda/pravda byly hodnoty 0 a 1 typu `short int`.

`int *`, `char *`, `double *`, `myListItem *` ... typy ukazatelů / adres se rozlišují podle typu dat, na něž adresy ukazují. Pouze `void *` nemá určen cílový typ.

Výčty

```
enum { ... }
```

podrobněji později

Uniony

```
union { ... }
```

podrobněji později

Struktury

```
struct { ... }
```

podrobněji později

Pole

<code>float t[20];</code>	typ, jméno, velikost	definice s alokací paměti
<code>int a[3] = {6,5,4};</code>	typ, jméno, velikost, hodnoty	s inicializací
<code>double w[]</code>	typ, jméno	deklarace bez alokace (např. formálního parametru)

Funkce

<pre>int round(double x) { ... return e; }</pre>	typ výsledku, jméno, typ a jméno parametru, tělo, výsledek (funkční hodnota)
<pre>void putline(int n) { ... return; }</pre>	„funkce“ bez výsledku (<i>procedura</i>)
<pre>float rand(void) { ... return e; }</pre>	funkce „bez parametrů“

Funkce s hlavičkou `int round(double x)` má typ `int(double)`.
(`round` je jméno funkce, `x` je jméno jejího formálního parametru.)

Zápis obvyklý v matematice: $\text{round} : \text{double} \rightarrow \text{int}$

Typy funkcí nepatří mezi *datové* typy.

Řetězce

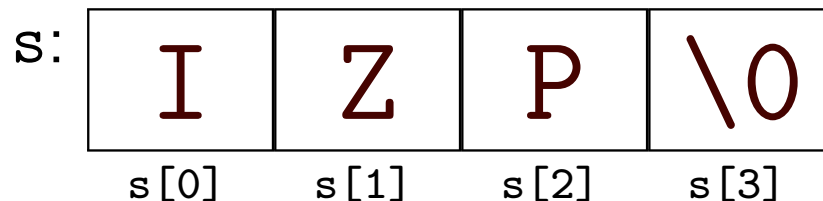
Znakové řetězce jsou konečné posloupnosti znaků.

V C je řetězec konečná n -prvková posloupnost znaků ($n \geq 0$), uvnitř které se nesmí vyskytnout nulový znak `'\0'`. Je reprezentován polem délky $n + 1$, které obsahuje znaky řetězce a na svém konci má nulový znak `'\0'`.

```
char s[] = "IZP";
```

nebo

```
char * s = "IZP";
```



Čtení řetězců ze standardního vstupu

<code>scanf ("%s", s);</code>	řetězec neobsahující bílé znaky
<code>scanf ("%100s", s);</code>	řetězec neobsahující bílé znaky, s omezením délky na 99 znaků
<code>fgets (s, 100, stdin);</code>	celý řádek, ale nejvýše 98 znaků

Počet znaků v řetězci

Napište funkci, která vrátí délku slova (řetězce neobsahujícího mezery).
Do délky se nezapočítává ukončovací nulový znak.

```
#include <stdio.h>
int stringlength(char s[]) {

    /*  DOPLŇTE  */

}

int main (void) {
    char s[100];
    printf("Word: "); scanf("%100s", s);
    printf("Length: %d\n", stringlength(s));
    return 0;
}
```


Převod na malá písmena

Cvičení: Napište proceduru, která v řetězci vymění všechna velká písmena za malá. Ostatní znaky ponechá. Řetězec má nejvýše 100 znaků.

```
#include <stdio.h>
const int offset = 'a' - 'A'; // V ASCII offset = 32

void makeLowercase (char s[]) {

    /* DOPLŇTE */

    return;
}

int main (void) {
    char s[101];           // nejvýše 100znakový řetězec
    fgets(s, 101, stdin);  // načtení s
    makeLowercase(s);      // převod na malá
    printf("%s\n", s);     // výpis výsledku
    return 0;
}
```

Palindromy

Palindrom (česky *ráček*) je věta, která je stejná, je-li čtena odzadu. Například věta „*Jelenovi pivo nelej.*“ je palindrom, ale věta „*Pije jen kofolu.*“ palindrom není.

Napište program, který přečte řetězec a napíše, zda je to palindrom.
Program

1. převede všechna písmena na malá,
2. odfiltruje nepísmenné znaky,
3. otestuje symetrii,
4. oznámí výsledek.

```

#include <stdio.h>
#include <stdbool.h>

void makeLowercase(char s[]) {
    /* Převeďte s na malá písmena */
}

void filterLcLetters(char s[]) {
    /* Ponechá jen malá písmena */
}

int stringlength(char s[]) {
    /* Délka řetězce */
}

bool palindrom(char s[]) {
    /* Test symetrie */
}

int main(void) {
    char s[101];           // nejvýše 100znakový řetězec
    fgets(s, 101, stdin);  // načtení s
    makeLowercase(s);      // převod na malá
    filterLcLetters(s);    // vyfiltrování písmen
    printf("%s palindrom.\n", palindrom(s) ? "Je" : "Neni");
    return 0;
}

```

Caesarova šifra

Historická šifra použitá ve starověku. Spočívá v cyklickém posunutí abecedy o daný počet písmen.

Napište program `caesar`, který

- spustíte s číslem n na příkazovém řádku, např. `./caesar 12`
- přečte standardní vstup, např. `Fakulta Informacnich Technologii`
- posune text v abecedě o n písmen, zvlášť malá a zvlášť velká písmena
- nepísmenné znaky ponechá tak, jak jsou
- výsledek napíše na `stdout`, např. `Rmwgxfm Uzradymozuot Fqotzaxasuu`
- program napište jako filtr, tj. znaky se neukládají, ale hned konvertují

```

#include <stdio.h>
#include <stdlib.h>

char caesarShift (int n, char c) {    // Posune 1 znak
    /* DOPLŇTE */
}

int main(int argc, char* argv[]) {
    int n;
    char c;
    if (argc != 2) {
        fprintf(stderr, "%s SHIFT\n", argv[0]); return 1;
    }
    n = atoi(argv[1]);
    while ((c=getchar()) != EOF) { putchar(caesarShift(n,c)); }
    return 0;
}

```

Napište cyklus z hlavního programu bez přiřazování uvnitř podmínky.