

Počítačové systémy

Adam Šmelko

smelko@ksi.mff.cuni.cz

NSWI170 - cvičení

LS 2020/2021

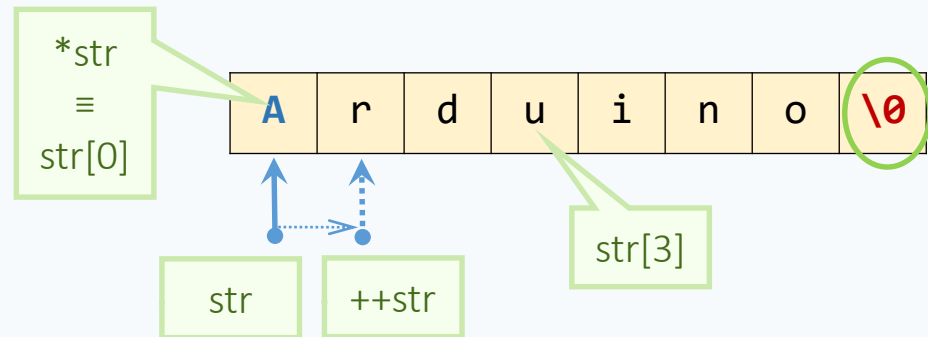
Arduino

Segmentový displej

Alfanumerický displej

Řetězce

- řetězec $\approx$ `char[]` zakončený `'\0'`



- ➡ funkce pro délku řetězce
 - pointrova aritmetika
 - `++str` dokud `*str != '\0'`
 - `*str++` postinkrementace ukazatele
 - výsledek: hodnota referencovaná před inkrementací
 - typický obrat při zpracování řetězců
- automatická konverze `*str` ➡ `bool`
 - `true` $\equiv$ `hodnota != 0`

```
size_t fnc( const char *str) {  
    size_t i = 0;  
    while( str[i]) ++i;  
    return i;  
}
```

```
size_t fnc( const char *str) {  
    size_t i = 0;  
    while( *str++) ++i;  
    return i;  
}
```

A blue arrow points from the `str[i]` expression in the first code block to the `*str++` expression in the second code block, illustrating the equivalence between indexing and pointer arithmetic.

Alfanumerický displej

- ➡ glyphs pro písmena
 - V poznámkách na stránce cvičenia
- ➡ 6.1 zobrazení znaku
 - číslice, znaky a-z (ignore-case), ...
 -  char $\rightsquigarrow$ glyph
 - nezapomeňte na znaky, pro které nemáte glyph
 - inicializovaná pole
 - pozor na platnost indexů!
 - funkce pro vlastnosti znaků
 - isDigit, isAlpha, isLowerCase, isUpperCase
- ➡ 6.2 zobrazení krátkého řetězce
 - multiplex
 - const char *
 - zjistit velikost
 - zobrazit max 4 znaky
 -  kratší - doplnit mezerami

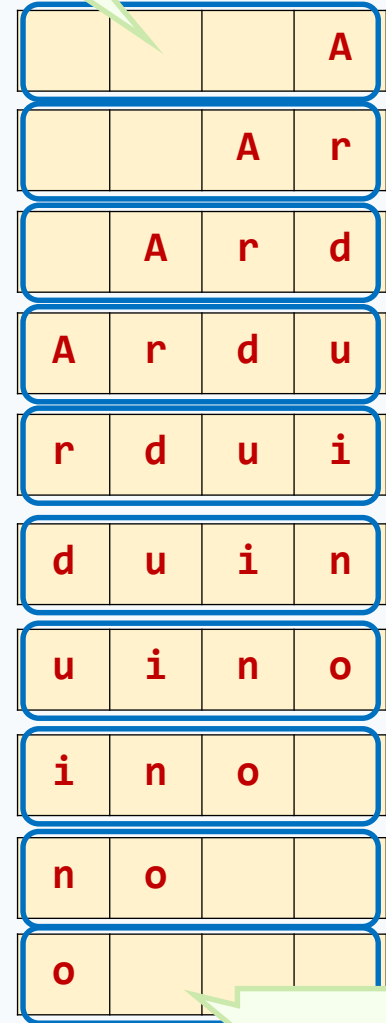
```
constexpr char letter_glyph[] {..};  
void displayChar( char c) {  
    byte glyph;  
    if( isAlpha( c))  
        glyph = ...  
    else  
        glyph = ...  
  
    writeGlyph...;  
}
```

```
constexpr byte letter_glyph [] {..};  
int index = c - (isUpperCase(c) ? 'A' : 'a');
```

Alfanumerický displej

- ➡ 6.3 běžící text
 - každý znak postupně na všech pozicích
 - parametr: rychlost posunu
 - ♥ řetězec opakovat
 - mezi opakováními mezery
 - 👁 začátek a konec
 - dekompozice
 - low-level třída pro multiplex zobrazení
 - hi-level aplikační třída
- ➡ 6.4 ♥ rychlost a směr
 - ➡ 6.4a ovládání rychlosti
 - b1/b2 zpomalení/zrychlení
 - b3 změna směru
 - ➡ 6.4b ruční posun
 - b1 ≈ stop / continue
 - b2/b3 ≈ posun o 1 ←→

začátek



konec

Alfanumerický displej

- ➔ 6.5  text zadaný přes sériovou linku
 - zobrazuje načtené řetězce jako běžící text
 - 👁 čtení ze sériové linky
 - `Serial.available`, `Serial.read`
 - vždy běží pouze poslední načtený řetězec
 - do `'\n'`
 - Recodex - připravený wrapper
 - `#include "input.h"`
 - `SerialInputHandler`
 - kde je uložený načtený řetězec?
 - implementace pomocí ukazatelů
 - pozor na životnost proměnných!
 - ukazatel na řetězec musí přežít do dalšího běhu loopu
 - podobně jako např. glyphy pro zobrazení v multiplexu

potom, co je
dokončeno
zobrazení
předchozí
zprávy

```
SerialInputHandler h;  
  
int myfnc() {  
    const char* msg;  
    ....  
    msg = h.getMessage();  
    ... msg ....  
}  
  
int setup() {  
    h.initialize();  
}  
  
int loop() {  
    h.updateInLoop();  
}
```