

Struktury a uniony

Jan Faigl

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Přednáška 06

BOB36PRP – Procedurální programování

Přehled témat

- Část 1 – Struktury a uniony

Struktury – struct

Proměnné se sdílenou pamětí – union

Příklady

S. G. Kochan: kapitola 9 a 17

- Část 2 – Zadání 6. domácího úkolu (HW06)

Appendix – Kódovací příklady

Struktury – struct

Uniony

Příklady

Část I

Část 1 – Struktury a uniony

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 1 / 35

Struktury – struct Uniony Příklady

Struktura – struct

- Struktura je konečná množina prvků (proměnných), které nemusí být stejného typu.
- Skladba struktury je definovaná uživatelem jako nový typ sestavený z již definovaných typů.
- K prvkům struktury **přístupujeme tečkovou notací**, např. `struct_proměnná.prvek`.
- K prvkům můžeme přistupovat přes ukazatel operátorem `->`, např. `proměnná_typu_ukazatel_na_struct->prvek`.
- **Pro struktury stejného typu je definován operátor přiřazení**, `var_struct1 = var_struct2;`
- Struktury (jako celek) **nelze** porovnávat relačním operátorem `==`.
- Struktura může být funkcí předávána hodnotou i ukazatelem.
- Struktura může být návratovou hodnotou funkce.

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 5 / 35

Struktury – struct Uniony Příklady

Příklad struct – Inicializace

- Struktury:

```
1 struct record {
2     int number;
3     double value;
4 };
1 typedef struct {
2     int n;
3     double v;
4 } item;
```
- Proměnné typu struktura můžeme inicializovat prvek po prvku.

```
1 struct record r;
2 r.value = 21.4;
3 r.number = 7;
```
- Podobně jako pole lze inicializovat přímo při definici

```
1 item i = { 1, 2.3 };
```
- nebo pouze konkrétní položky (ostatní jsou nulovány).

```
1 struct record r2 = { .value = 10.4 };
```

lec06/struct.c

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 8 / 35

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 2 / 35

Struktury – struct Uniony Příklady

Příklad struct – Definice

- Bez zavedení nového typu (`typedef`) je nutné před identifikátor jména struktury uvádět klíčové slovo `struct`.
- Jméno struktury je ve jmenném prostoru složených typů (struktur).

```
1 struct record {
2     int number;
3     double value;
4 };
1 typedef struct {
2     int n;
3     double v;
4 } item;
```

```
1 record r; /* IT IS NOT ALLOWED! */
2           /* Type record is not known */
4 struct record r; /* Keyword struct is required */
5 item i; /* type item defined using typedef */
```
- Zavedením nového typu `typedef` používáme definovaný typ a nemusíme používat (a ani definovat) jméno struktury. `lec06/struct.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 6 / 35

Struktury – struct Uniony Příklady

Příklad struct jako parametr funkce

- Struktury můžeme předávat jako parametry funkcí hodnotou.

```
1 void print_record(struct record rec) {
2     printf("record: number(%d), value(%1f)\n",
3           rec.number, rec.value);
4 }
```
- Nebo hodnotou ukazatele

```
1 void print_item(item *v) {
2     printf("item: n(%d), v(%1f)\n", v->n, v->v);
3 }
```
- Při předávání parametru
 - **hodnotou** se vytváří nová proměnná a původní obsah předávané struktury se kopíruje na zásobník (pro složený typ je definován operátor přiřazení);
 - **hodnotou ukazatele** se kopíruje pouze hodnota ukazatele (adresa) a pracujeme tak s původní strukturou. `lec06/struct.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 9 / 35

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 3 / 35

Struktury – struct Uniony Příklady

Definice jména struktury a typu struktury

- Uvedením `struct record` zavádíme nové jméno struktury `record`.

```
1 struct record {
2     int number;
3     double value;
4 };
```

 - Definujeme identifikátor `record` ve jmenném prostoru struktur.
- Definicí typu `typedef` zavádíme nové jméno typu `record`.

```
1 typedef struct record record;
```

 - Definujeme globální identifikátor `record` jako jméno typu `struct record`.
 - Obojí můžeme kombinovat v jediné definici jména a typu struktury.

```
1 typedef struct record {
2     int number;
3     double value;
4 } record;
1 typedef struct record_struct_name {
2     int number;
3     double value;
4 } record_type;
```

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 7 / 35

Struktury – struct Uniony Příklady

Složený typ, operátor přiřazení a pole jako prvek složeného typu 1/2

- Velikost složeného typu musí být známa během překladu, proto můžeme mít definovaný operátor přiřazení. *Nebo naopak, abychom mohli jednoduše přiřazovat, tak potřebujeme znát velikost typu.*
- Prvek složeného typu může být pole (definované velikosti) nebo ukazatel.

```
1 void print(const char *str, int n, int *a);
2 #define N 10 // We need named literal.
3 int main(void)
4 {
5     const int n = N;
6     struct { // Anonymous struct
7         int a[N]; // Defined size, no VLA
8     } s1, s2; // Two struct variables
9     printf("s1 %p; s2 %p\n", &s1, &s2);
10    for (int i = 0; i < n; ++i) {
11        s1.a[i] = i;
12    }
13    print("s1.a", n, s1.a);
14    s2 = s1; // Assignment
15    print("s2.a", n, s2.a);
16    for (int i = 0; i < n; ++i) {
17        s1.a[i] = n - i;
18    }
19    print("s1.a", n, s1.a);
20    print("s2.a", n, s2.a);
21    return 0;
22 } // end main()
23 void print(const char *str, int n, int *a) {
24     printf("%s %p ", str, a);
25     for (int i = 0; i < n; ++i) {
26         printf("%dX", a[i], i < (n-1) ? ", " : "\n");
27     }
28 }
```

lec06/demo-struct_array.c

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 10 / 35

Struktury – struct Uniony Příklady

Složený typ, operátor přiřazení a pole jako prvek složeného typu 2/2

Příklad `lec06/demo-struct_array.c`

- Používáme anonymní složený typ - definice strukturu přímo v definici proměnných `s1` a `s2`.
- Musíme použít textový literál pro definici velikosti položky `a` jako pole definované délky.
- Ve funkci `print()` tiskneme hodnotu adresy, kde je alokované pole.
V našem případě se shoduje s adresou, kde je struktura uložena. Struktura je „organizovaný“ pohled na blok paměti důležitý zejména pro zpráhlední programu. Při běhu programu vlastně není nutné mít v paměti dílčí jména prvků složeného typu.

```
s1 0x7fffffff0840; s2 0x7fffffff0818
s1.a 0x7fffffff0840: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
s2.a 0x7fffffff0818: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
s1.a 0x7fffffff0840: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1
s2.a 0x7fffffff0818: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

- V příkladu si vyzkoušejte chování překladač a programu v případě použití VLA nebo konstantní proměnné definující velikost pole.
- Pole definované velikosti nahraďte dynamicky alokovaným polem.

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 11 / 35

Struktury – struct Uniony Příklady

Příklad struct – Přiřazení

- Hodnoty proměnné stejného typu struktury můžeme přiřadit operátorem `=`.

```
1 struct record {
2     int number;
3     double value;
4 };
1 typedef struct {
2     int n;
3     double v;
4 } item;
1 struct record rec1 = { 10, 7.12 };
2 struct record rec2 = { 5, 13.1 };
3 item i;
4 print_record(rec1); /* number(10), value(7.120000) */
5 print_record(rec2); /* number(5), value(13.100000) */
6 rec1 = rec2;
7 i = rec1; /* IT IS NOT ALLOWED! */
8 // Different types, albeit with the same memory representation.
9 print_record(rec1); /* number(5), value(13.100000) */
```

`lec06/struct.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 12 / 35

Struktury – struct Uniony Příklady

Příklad struct – Přímá kopie paměti

- Jsou-li dvě struktury stejně veliké, můžeme přímo kopírovat obsah příslušné paměťové oblasti.
Například funkci `mempcy()` z knihovny `string.h`

```
1 struct record r = { 7, 21.4 };
2 item i = { 1, 2.3 };
3 print_record(r); /* number(7), value(21.400000) */
4 print_item(&i); /* n(1), v(2.300000) */
5 if (sizeof(i) == sizeof(r)) {
6     printf("i and r are of the same size\n");
7     memcpy(&i, &r, sizeof(i));
8     print_item(&i); /* n(7), v(21.400000) */
9 }
```

- V tomto případě je interpretace hodnot v obou strukturách identická, obecně tomu však být nemusí. Například v případě změny pořadí prvků typu `int` a `double`.

`lec06/struct.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 13 / 35

Struktury – struct Uniony Příklady

Struktura struct a velikost

- Vnitřní reprezentace struktury nutně nemusí odpovídat součtu velikostí jednotlivých prvků.

```
1 struct record {
2     int number;
3     double value;
4 };
1 typedef struct {
2     int n;
3     double v;
4 } item;
```

```
1 printf("Size of int: %lu size of double: %lu\n", sizeof(int), sizeof(double));
2 printf("Size of record: %lu\n", sizeof(struct record));
3 printf("Size of item: %lu\n", sizeof(item));
```

Size of int: 4 size of double: 8
Size of record: 16
Size of item: 16

`lec06/struct.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 14 / 35

Struktury – struct Uniony Příklady

Struktura struct a velikost 1/2

- Při kompilaci zpravidla dochází k zarovnání prvků na velikost slova příslušné architektury.
Např. 8 bytů v případě 64-bitové architektury.
- Můžeme explicitně předspsat kompaktní paměťovou reprezentaci, např. direktivou `__attribute__((packed))` překladačů `clang` a `gcc`.

```
1 struct record_packed {
2     int n;
3     double v;
4 } __attribute__((packed));
```

`lec06/struct.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 15 / 35

Struktury – struct Uniony Příklady

Struktura struct a velikost 2/2

- Nebo

```
1 typedef struct __attribute__((packed)) {
2     int n;
3     double v;
4 } item_packed;
```

- Příklad výstupu:

```
1 printf("Size of int: %lu size of double: %lu\n", sizeof(int), sizeof(double));
2 printf("record_packed: %lu\n", sizeof(struct record_packed));
3 printf("item_packed: %lu\n", sizeof(item_packed));
```

Size of int: 4 size of double: 8
Size of record_packed: 12
Size of item_packed: 12

- Zarovnání zpravidla přináší rychlejší přístup do paměti, ale zvyšuje paměťové nároky.
<http://www.catb.org/esr/structure-packing>

<https://stackoverflow.com/questions/4306186/structure-padding-and-packing>

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 16 / 35

Struktury – struct Uniony Příklady

Proměnné se sdílenou pamětí – union

- **Union** je uživatelsky definovaný typ, který sdružuje více již existujících typů do jedné paměťové oblasti
- Prvky unionu sdílejí společně stejná paměťová místa. *Překrývají se*
- Velikost unionu je dána velikostí největšího z jeho prvků.
- K prvkům unionu se přistupuje tečkovou notací.
- Pokud nedefinujeme nový typ, je nutné k identifikátoru proměnné unionu uvádět klíčové slovo `union`. *Podobně jako u struktury `struct`.*

```
1 union Nums {
2     char c;
3     int i;
4 };
5 Nums nums; /* THIS IS NOT ALLOWED! Type Nums is not known! */
6 union Nums nums;
```

- Používá se v návrhovém vzoru **tagged union**.
Např. měření senzorických dat, která mohou mít různý význam, počet průchodů nebo teplota, ale pořád se jedná o senzorická měření.

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 18 / 35

Struktury – struct Uniony Příklady

Příklad union 1/2

- Union složený z proměnných typu: `char`, `int` a `double`.

```
1 int main(int argc, char *argv[])
2 {
3     union Numbers {
4         char c;
5         int i;
6         double d;
7     };
8     printf("size of char %lu\n", sizeof(char));
9     printf("size of int %lu\n", sizeof(int));
10    printf("size of double %lu\n", sizeof(double));
11    printf("size of Numbers %lu\n", sizeof(union Numbers));
12
13    union Numbers numbers;
14
15    printf("Numbers c: %d i: %d d: %lf\n", numbers.c, numbers.i, numbers.d);
```

- Příklad výstupu.

```
size of char 1
size of int 4
size of double 8
size of Numbers 8
Numbers c: 48 i: 740313136 d: 0.000000
```

`lec06/union.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 19 / 35

Struktury – struct Uniony Příklady

Příklad union 2/2

- Proměnné sdílejí paměťový prostor.

```
1 numbers.c = 'a';
2 printf("\nSet the numbers.c to 'a'\n");
3 printf("Numbers c: %d i: %d d: %lf\n", numbers.c, numbers.i, numbers.d);
4
5 numbers.i = 5;
6 printf("\nSet the numbers.i to 5\n");
7 printf("Numbers c: %d i: %d d: %lf\n", numbers.c, numbers.i, numbers.d);
8
9 numbers.d = 3.14;
10 printf("\nSet the numbers.d to 3.14\n");
11 printf("Numbers c: %d i: %d d: %lf\n", numbers.c, numbers.i, numbers.d);
```

- Příklad výstupu

```
Set the numbers.c to 'a'
Numbers c: 97 i: 1374389601 d: 3.140000
Set the numbers.i to 5
Numbers c: 5 i: 5 d: 3.139999
Set the numbers.d to 3.14
Numbers c: 31 i: 1374389535 d: 3.140000
```

`lec06/union.c`

Jan Faigl, 2025 BOB36PRP – Přednáška 06: Struktury a uniony 20 / 35

Příklad IPv4 adresy jako číslo nebo čtveřice bajtů

- IPv4 adresa je čtveřice bajtů, ale také **unsigned** 32-bitové číslo.

Jedná se o příklad naznačující použití, v tom případě je také nutné zajistit správné pořadí bajtů.

```
1 #include <stdio.h>
2 #include <stdint.h>
3
4 union IPv4Address {
5     uint32_t addr; // adresa jako celé číslo (32 bitů)
6     uint8_t bytes[4]; // adresa po jednotlivých bytech
7 };
8
9 int main(void)
10 {
11     union IPv4Address ip;
12
13     // nastavení adresy "192.168.0.1"
14     ip.bytes[0] = 192;
15     ip.bytes[1] = 168;
16     ip.bytes[2] = 0;
17     ip.bytes[3] = 1;
18
19     printf("IPv4 address: %u.%u.%u.%u\n",
20         ip.bytes[0], ip.bytes[1], ip.bytes[2], ip.bytes
21         [3]);
22     printf("IPv4 as a number: %u\n", ip.addr);
23     return 0;
24 }
```

IPv4 address: 192.168.0.1
IPv4 as a number: 16820416

lec06/demo-union.c

Příklad struktura, pole a výtčový typ 2/3

- Připravíme si pole struktur pro konkrétní jazyk (angličtina a čeština).
- Program vytiskne jméno a zkratku dne v týdnu dle čísla dne v týdnu.

V programu používáme jednotné číslo dne bez ohledu na jazykovou mutaci.

```
19 const week_day_s days_cs[] = {
20     [MONDAY] = { "Pondělí", "po" },
21     [TUESDAY] = { "Úterý", "ut" },
22     [WEDNESDAY] = { "Středa", "st" },
23     [THURSDAY] = { "Čtvrtek", "ct" },
24     [FRIDAY] = { "Pátek", "pa" },
25 };
26
27 int main(int argc, char *argv[], char **envp)
28 {
29     int day_of_week = argc > 1 ? atoi(argv[1]) : 1;
30     if (day_of_week < 1 || day_of_week > 5) {
31         fprintf(stderr, "(EE) File: '%s' Line: %d -- Given day of week out of range\n",
32             __FILE__, __LINE__);
33         return 101;
34     }
35     day_of_week -= 1; // start from 0
36 }
```

lec06/demo-struct.c

Kódovací příklad union – is_big_endian()

Tisk hodnot v šestnáctkové soustavě

- Reprezentace float hodnot.
 - Hodnota 85.125 je **0x42aa4000**.
 - Hodnota 0.1 je sice **0x3dcccccd**, ale je kódována **0x3dcccccd**. *Protože chyba je absolutně menší.*
- Implementujeme funkci pro tisk paměťové reprezentace hodnoty typu float jako posloupnosti hodnot bajtů v šestnáctkové soustavě.
- Přístup k float jako posloupnosti bajtů a tisk hex hodnot "%02x" funkcí printf().
 - Adresním operátorem & získáme adresu proměnné.
 - Přetypujeme adresu jako ukazatel na hodnotu char.
 - Použijeme nepřímý adresní operátor * k přístupu k hodnotě na adrese uložené v ukazateli.
- Datový typ float má vnitřní reprezentaci dle IEEE 754 definující necelocíselnou reprezentaci ve 32-bitech, které jsou v paměti uloženy jako 4 bajty dle architektury.

```
1 #include <stdio.h>
2 void print_float_hex(float v);
3 int main(void)
4 {
5     print_float_hex(85.125);
6     print_float_hex(0.1);
7     return 0;
8 }
9
10 void print_float_hex(float v)
11 {
12     ...
13     ...
14     ...
15 }
```

Inicializace union

- Proměnnou typu union můžeme inicializovat při definici.

```
1 union {
2     char c;
3     int i;
4     double d;
5 } numbers = { 'a' };
```

Pouze první položka (proměnná) může být inicializována.

- V C99 můžeme inicializovat konkrétní položku (proměnnou).

```
1 union {
2     char c;
3     int i;
4     double d;
5 } numbers = { .d = 10.3 };
```

Příklad struktura, pole a výtčový typ 3/3

- Detekci národního prostředí provedeme podle hodnoty proměnné prostředí.

Pro jednoduchost detekujeme češtinu na základě výskytu řetězce "cs" v hodnotě proměnné prostředí LC_CTYPE.

```
35 _Bool cz = 0;
36 while (*envp != NULL) {
37     if (strstr(*envp, "LC_CTYPE") && strstr(*envp, "cs")) {
38         cz = 1;
39         break;
40     }
41     envp++;
42 }
43 const week_day_s *days = cz ? days_cs : days_en;
44 printf("%d %s %s\n", day_of_week,
45     days[day_of_week].name,
46     days[day_of_week].abbr
47 );
48 return 0;
49 }
```

lec06/demo-struct.c

V programu jsme využili koncept definování datových struktur, které následně programově přepínáme a využíváme. Alternativně můžeme data načítat ze souboru. V programu se snažíme obecně pracovat s datovými strukturami.

Příklad – Tisk hodnot v šestnáctkové soustavě 1/3

- Získáme adresu proměnné float v operátorem &v.
- K hodnotám na adrese &v budeme přistupovat jako k bajtům, proto přetypujeme adresu na ukazatel (adresu) na hodnoty typu char.
 - unsigned char *p = (unsigned char*)&v;
- Hodnotu uloženou na adrese p získáme operátorem nepřímého adresování *p.
- Adresu následujícího bajtů za adresou uloženou v p získáme p = p + 1;
 - Protože se jedná o ukazatel na char, probíhá inkrementace o sizeof(char), tj. o 1 (ukazatelová aritmetika).
- Vytíštěné hodnoty jsou v opačném než očekávaném pořadí **0x42aa4000** a **0x3dcccccd**.

```
1 int main(void)
2 {
3     print_float_hex(85.125);
4     print_float_hex(0.1);
5     ...
6     void print_float_hex(float v)
7     {
8         unsigned char *p = (unsigned char*)&v;
9         printf("Value %13.10f is 0x", v);
10        for (int i = 0; i < sizeof(float); ++i,
11            p = p + 1) {
12            printf("%02x", *p); // or use [i]
13        }
14        putchar('\n');
15    }
```

1 \$ clang floats.c -o floats && ./floats
2 Value 85.1250000000 is 0x0040aa42
3 Value 0.1000000015 is 0xcdcccc3d

Příklad struktura, pole a výtčový typ 1/3

- Hodnoty (konstanty) výtčového typu jsou celá čísla, která mohou být použita jako indexy (pole).
- Také je můžeme použít pro inicializaci pole struktur.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 enum weekdays { MONDAY, TUESDAY, WEDNESDAY, THURSDAY, FRIDAY };
5
6 typedef struct {
7     char *name;
8     char *abbr; // abbreviation
9 } week_day_s;
10
11 const week_day_s days_en[] = {
12     [MONDAY] = { "Monday", "mon" },
13     [TUESDAY] = { "Tuesday", "tue" },
14     [WEDNESDAY] = { "Wednesday", "wed" },
15     [THURSDAY] = { "Thursday", "thr" },
16     [FRIDAY] = { "Friday", "fri" },
17 };
18 }
```

lec06/demo-struct.c

Příklad – Zprávy různých typů v komunikaci

- Komunikace probíhá definovanými zprávami s hlavičkou (typ zprávy) a daty, která mohou mít různou podobu. union umožňuje uložit různá data do stejného bufferu.

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #define MSG_LEN 32
5 typedef enum {
6     MSG_TEXT,
7     MSG_SENSOR,
8     MSG_COMMAND
9 } msg_type_t;
10
11 struct {
12     char text[MSG_LEN];
13     int temperature;
14     int humidity;
15     int commandId;
16 } data;
17
18 int main(void)
19 {
20     Message m1 = { .type = MSG_TEXT };
21     Message m2 = { .type = MSG_SENSOR };
22     Message m3 = { .type = MSG_COMMAND };
23
24     strcpy(m1.data.text, "Hello PRP!");
25     m2.data.sensor.temperature = 23;
26     m2.data.sensor.humidity = 45;
27     m3.data.commandId = 42;
28
29     if (m1.type == MSG_TEXT) {
30         printf("Text: %s\n", m1.data.text);
31     }
32     if (m2.type == MSG_SENSOR) {
33         printf("Sensor: %d°C, %d%%\n",
34             m2.data.sensor.temperature,
35             m2.data.sensor.humidity);
36     }
37     if (m3.type == MSG_COMMAND) {
38         printf("Command ID: %d\n",
39             m3.data.commandId);
40     }
41     return EXIT_SUCCESS;
42 }
```

lec06/union-msg.c

Příklad – Tisk hodnot v šestnáctkové soustavě 2/3

- Očekávaná reprezentace v šestnáctkové soustavě je pro 85.125 výstup **0x42aa4000** a pro 0.1 výstup **0x3dcccccd**. Namísto toho dostáváme **0x0040aa42** a **0xcdcccc3d**.
- Výstup je závislý na reprezentaci více bajtových hodnot v paměti. Pro architekturu (amd64) je to tzv. little endian.
 - <https://en.wikipedia.org/wiki/Endianness>
- Proto potřebujeme detekovat, jak jsou hodnoty uloženy, například funkcí
 - _Bool is_big_endian(void);
- a případně vytiskneme hodnoty v opačném pořadí.

```
1 void print_float_hex(float v)
2 {
3     const _Bool big_endian = is_big_endian();
4     // cast pointer to float to pointer to char
5     unsigned char *p = (unsigned char*)&v
6     + (big_endian ? 0 : (sizeof(float) - 1));
7     printf("Value %13.10f is 0x", v);
8     for (int i = 0; i < sizeof(float); ++i) {
9         printf("%02x",
10             *(big_endian ? p++ : p--));
11     }
12     printf("\n");
13 }
14 }
```

\$ clang floats.c -o floats && ./floats
Value 85.1250000000 is 0x42aa0000
Value 0.1000000015 is 0x3dcccccd

Příklad – Tisk hodnot v šestnáctkové soustavě 3/3

- Detekce uložení můžete být založena na různých principech.
- Intuitivně můžeme uložit definovanou hodnotu, která má pouze jeden bajt nenulový a ostatní nulové.
- Využijeme složeného typu `union`, ve kterém položky sdílejí paměť a umožňuje nám tak různý pohled na konkrétní block paměti.
 1. Definujeme celočíselnou proměnnou o čtyřech bajtech, např., `uint32_t` z knihovny `stdint.h`.
 2. Nastavíme hodnotu na `0x01 00 00 00`.
 3. Otestujeme první bajt paměťové reprezentace.

```

1 #include <stdint.h>
2 _Bool is_big_endian(void)
3 {
4     union {
5         uint32_t i;
6         char c[4]; // 4 is fine here as we use
7             uint32_t
8     } e = { 0x01000000 };
9     return e.c[0];
10 }

```

toupper()	strrev()	strvc()	strsplit()	Knihovna strings.h	„String objekt“
-----------	----------	---------	------------	--------------------	-----------------

Shrnutí přednášky

Část 2 – Zadání 6. domácího úkolu (HW06)

Část II

Část 2 – Zadání 6. domácího úkolu (HW06)

toupper()	strrev()	strvc()	strsplit()	Knihovna strings.h	„String objekt“
-----------	----------	---------	------------	--------------------	-----------------

Diskutovaná témata

- Struktury, způsoby definování, inicializace a paměťové reprezentace
- Uniony
- Přístě: Paměť programu. Standardní knihovny C. Rekurze.

toupper()	strrev()	strvc()	strsplit()	Knihovna strings.h	„String objekt“
-----------	----------	---------	------------	--------------------	-----------------

Kódovací příklad – Textové řetězce – toupper() 2/2

```

1 #include <stdio.h>
2 #include <string.h>
3 #include "my_malloc.h"
4 char* strtoupper(const char *str);
5 int main(void)
6 {
7     const char *str = "I like prp!";
8     char *stru = strtoupper(str);
9     printf("stru: %s\n", stru);
10    free(stru);
11    return EXIT_SUCCESS;
12 }

```

- Volání funkce `strtoupper()` může být předán neplatný ukazatel `NULL`.
- Explicitně ošetřujeme, ikdyž například funkce `strlen()` předpokládá validní vstup a volání `strlen(NULL)` může skončit chybou programu.
- V našem programu, alokujeme ve funkci `strtoupper()` paměť dynamicky a to vždy nejméně pro jeden znak (`'\0'`).

Téma: Maticové počty

- Povinné zadání: **3b**; Volitelné zadání: **5b**; Bonusové zadání: **5b**
- **Motivace:** Získání zkušenosti s dvojrozměrným polem.
 - **Cíl:** Osvojit si práci s polem variabilní délky a předávání ukazatelů.
 - **Zadání:** <https://cw.fel.cvut.cz/wiki/courses/b0b36prp/hw/hw06>
 - Načtení vstupních hodnot dvou matic a znaku operace (`'*'` – násobení).
 - **Volitelné zadání** rozšiřuje úlohu o další operace s maticemi sčítání (`'+'`) a odčítání (`'-'`), které mohou být zapsány ve výrazu.
 - **Bonusové zadání** pak řeší zpracování celého výrazu, ve kterém jsou však jednotlivé matice uvedeny jako symboly, které jsou nejdříve definovány načtením hodnot matic ze standardního vstupu.
 - **Termín odevzdání:** 06.12.2025, 23:59:59 PST.
 - **Bonusová úloha:** 10.01.2026, 23:59:59 CET.

toupper()	strrev()	strvc()	strsplit()	Knihovna strings.h	„String objekt“
-----------	----------	---------	------------	--------------------	-----------------

Část III Appendix

toupper()	strrev()	strvc()	strsplit()	Knihovna strings.h	„String objekt“
-----------	----------	---------	------------	--------------------	-----------------

Kódovací příklad – Textové řetězce – strrev() 1/2

```

1 #include <stdio.h>
2 #include <string.h>
3 #include "my_malloc.h"
4 int main(void)
5 {
6     char *str = "I like prp!";
7     size_t j, n = strlen(str);
8     char *strr = str;
9     for (size_t i = 0, j = n-1; i < n/2; ++i, --j) {
10        str[i] = str[j];
11        str[j] = str[i];
12    }
13    printf("str: %s\n", str);
14    return EXIT_SUCCESS;
15 }

```

- V cyklu využíváme operátor čárky k inicializaci a dekrementaci proměnné `j`.
- Opět v našem programu je řetězec `str` platný a můžeme tak bezpečně volat funkci `strlen(str)`.
- Nicméně po odladění obrácení řetězce, program přepíšeme s implementací naší nové funkce `strrev()`.

Kódovací příklad – Textové řetězce – strev() 2/2

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include "my_malloc.h"
4 char strev(const char *str);
5 int main(void)
6 {
7     char str = "I like prp!";
8     char str2 = strev(str);
9     printf("YaYa", str);
10    printf("YaYa", str2);
11    free(str);
12    return EXIT_SUCCESS;
13 }

```

- V funkci `strev()` vytváří nový řetězec, proto můžeme bezpečně předat ukazatel na textový literál.
- Volání `strev()` vrátí textový řetězec, nebo končí chybou (volání `my_malloc()`).
- Proměnná `str` tak vždy ukazuje na paměť, ve které je nejméně jeden znak a to '\0'.
- Program tak v rámci `main()` vždy skončí úspěšně `EXIT_SUCCESS`.
- V funkci `main()` tak vlastně ani explicitně neresíme návratové hodnoty volání.
- V funkci explicitně ověřujeme, že vstupní řetězec není NULL.
- V naší implementaci je prázdný (NULL) řetězec ekvivalentní s řetězcem o délce nula.
- Pokud je `str == NULL`, není hodnota `cur` validní.
- Proto ve `while` cyklu explicitně testujeme `str`.
- Z hlediska efektivity bychom mohli volání funkce v případě `str == NULL` ukončit dříve.
- Nicméně volíme přehlednost, menší počet řádků a jediný `return` ve funkci.

Kódovací příklad – Textové řetězce – strvc() 1/2

- Implementujeme funkci, která vrátí počet slov v řetězci.
- Slovo interpretujeme jako souvislou sekvenci znaků vyhovující funkci `isalpha()` z knihovny `ctype.h`.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #include <ctype.h>
5 int main(void)
6 {
7     int c, wc = 0;
8     bool inword = false;
9     while ((c = getchar()) != EOF) {
10        if (isalpha(c)) {
11            if (!inword) {
12                inword = true;
13                wc++;
14            }
15        } else {
16            inword = false;
17        }
18        printf("Input contains %d words.\n", wc);
19        return EXIT_SUCCESS;
20    }
21 }

```

Kódovací příklad – Textové řetězce – strvc() 2/2

- Čtení znaků ze `stdin` funkcí `getchar()` nahradíme voláním `getline()` z `stdlib.h`. Viz `man getline`.
- `ssize_t` `getline(char ** restrict linep, size_t * restrict linecap, FILE * restrict stream);`

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <stdbool.h>
4 #include <ctype.h>
5 int main(void)
6 {
7     int strvc(const char *str);
8     int main(void)
9     {
10        char *line = NULL; // nezbytné k alokaci v getline()
11        size_t cap = 0; // alokovaná kapacita v getline()
12        while (getline(&line, &cap, stdin)) {
13            int wc = strvc(line);
14            printf("Input contains %d words.\n", wc);
15            free(line); // proměnná je alokována dynamicky.
16        }
17        return EXIT_SUCCESS;
18    }
19 }

```

Kódovací příklad – Textové řetězce – strsplit() 1/2

- Implementujeme funkci, která rozdělí daný řetězec na dva díle zadaného řetězce. **Všimněte si rozdílů ukazatelů!**

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include "my_malloc.h"
5 int main(void)
6 {
7     const char *str = "I like programming and PRP especially!";
8     char *s1, *s2;
9     char *delim = "and";
10    char *s = strstr(str, delim);
11    s1 = s2 = NULL;
12    if (s) { // poděříme (little) lexikon (v big)
13        fprintf(stderr, "D: str %s\n", str);
14        fprintf(stderr, "D: delin %s\n", delim);
15        fprintf(stderr, "D: s %s\n", s);
16        // rozdíl ukazatelů. Oba odkazují do identického
17        // souvislého bloku paměti.
18        size_t s1 = strlen(str) - strlen(s);
19        size_t s2 = strlen(s);
20    }
21 }

```

- Začátek řetězce v řetězci najdeme funkcí `strstr()`.
- `char* strstr(const char *big, const char *little)` **Viz man strstr.**

Kódovací příklad – Textové řetězce – strsplit() 2/2

- Začátek řetězce v řetězci najdeme funkcí `strstr()`.
- `char* strstr(const char *big, const char *little)` **Viz man strstr.**

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include "my_malloc.h"
5 bool strsplit(const char *str, const char *delim, char **s1, char **s2)
6 {
7     char *s = NULL;
8     if (
9         strstr(str, delim) || strstr(str, delim) // Poděříme lexikon.
10    ) {
11        return false;
12    }
13    size_t l2 = strlen(s); // Předpokládáme null-terminated řetězec.
14    size_t l1 = strlen(str) - l2; // strlen(str) >= l2
15    *s1 = my_malloc(l1 + 1) * sizeof(char); //...FILE..._LINE...;
16    *s2 = my_malloc(l2 + 1) * sizeof(char); //...FILE..._LINE...;
17    strncpy(*s1, str, l1);
18    printf("s1: %s\n", *s1);
19    printf("s2: %s\n", *s2);
20    free(s); // to is ok to call free(NULL);
21    return EXIT_SUCCESS;
22 }

```

- Při implementaci použijeme ladící výstupy na `stderr`.
- Program odladíme a přepíšeme do funkce.
- Při implementaci použijeme ladící program `valgrind`.
- Nicméně ne vždy detekuje možné problémy správně.
- Funkci `strsplit()` můžeme dále doplnit, např. o rozdělení bez `delim`.

Kódovací příklad – Knihovna – strings.h

- Implementované funkce `toupper()`, `strev()`, `strvc()`, `strsplit()` vložíme do knihovny `strings.h` a `strings.c`.
- Do knihovny vložíme lokální verzi funkce `my_malloc()`, kterou definujeme jako `static` v souboru `strings.c`.

```

1 #ifndef __STRINGS_H_
2 #define __STRINGS_H_
3 #include <stdbool.h> // Protože bool v strsplit()
4 int main(void)
5 {
6     char *str = "I like programming and PRP especially!";
7     char *delim = "and";
8     int strvc(const char *str);
9     bool strsplit(const char *str, const char *delim, char **s1,
10    char **s2);
11 #endif

```

- `static void my_malloc(size_t size, const char *filename, int line) { ... } // folded`
- `char* strtoupper(const char *str) { ... } // folded`
- `int strvc(const char *str) { ... } // folded`

Kódovací příklad – „String objekt“

- S využitím složeného typu a ukazatele na funkci implementujeme variantu objektu textového řetězce.

```

1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <string.h>
4 #include <stdbool.h>
5 #include "my_malloc.h"
6 typedef struct string {
7     char *str;
8     ssize_t len; // size_t vs. ssize_t (-1 might indicate error)
9     ssize_t (getlength)(const char *);
10 } string;
11 bool string_create(struct string *s, const char *v);
12 void string_destroy(struct string *s);
13 int main(void)
14 {
15     string str = { .str = NULL, .len = -1, .getlength = &strlen };
16     string_create(&str, "I like PRP!");
17     printf("String str: %s\n", str.str);
18     printf("String length is %d\n", string.getlength(str.str));
19     printf("String length is %d\n", strlen(str.str));
20     string_destroy(&str);
21     return EXIT_SUCCESS;
22 }

```

- `§ clang strobj.c my_malloc.c && ./a.out`
- `String str: "I like PRP!"`
- `String length is 11`
- `String length is 11`