KATEDRA
INFORMATIKY
UNIVERZITA PALACKÉHO V OLOMOUCI

Struktury a uniony

Základy programování 2

Mgr. Markéta Trnečková, Ph.D.

Struktury

- nový datový typ `struct jmeno`

```
struct jmeno{
 polozka_1;
 polozka_2;
 ...
 polozka_n;
};
```

Struktury

- `struct jmeno{ ... } promenna_1, promenna_2, ..., promenna_n;`
- `struct { ... } promenna_1, promenna_2, ..., promenna_n;`
- `struct jmeno{ ... };`

  `struct jmeno promenna_1, promenna_2, ..., promenna_n;`
- ```
  typedef struct{
   polozka_1;
   polozka_2;
   ...
   polozka_n;
  } jmeno;
  ```
  ```

Struktury
Inicializace

- `struct jmeno promenna = {h1, ..., hn};`
- `struct jmeno promenna = {.polozka_i = hi,`
  `.polozka_j = hj, ...};`

Přístup k prvkům

- Operátor `.`
- `promenna.polozka1`

Příklad (structusecka.c)

```
struct bod{
 float x;
 float y;
 float z;
};

struct usecka{
 struct bod A;
 struct bod B;
};
```

`usecka1.A.x`

Příklad

Příklad

Definujte strukturu pro reprezentaci zlomku. Naprogramujte funkce, které pro dva zadané zlomky vrátí jejich součet. Napište funkci, která zlomky vypíše.

Struktury v paměti

Příklad (structusecka.c)

```
struct bod{
 float x;
 float y;
 float z;
};

struct usecka{
 struct bod A;
 struct bod B;
};
```

A.x	A.y	A.z	B.x	B.y	B.z

Struktury v paměti

- struktury mohou zabírat v paměti více místa – více se dozvíte v OS1

Příklad (structsizeof.c)

```
#include <stdio.h>
struct struktura{
    float a;
    char b;
    int c;
};

int main(){
   printf("Velikost struktury: %i\n", sizeof(struct struktura));
   printf("Velikost polozek: %i\n", sizeof(float) + sizeof(int)
                                  + sizeof(char));
   return 0;
}
```

Struktury

Příklad

Může struktura obsahovat jako svou položku sebe sama?

```
struct data{
    struct data d;  /* chyba */
    ...
}
```

Struktury

Ukazatel

```
typedef struct{
    polozka1
    ...
    polozkan
} JMENO;

JMENO s, *p_s;


p_s = (JMENO *) malloc(sizeof(JMENO));


p_s = &s;
```

Struktury

Ukazatel

```
JMENO s, *p_s = &s;
```

Příklad

```
/* pomoci jmena struktury */
s.polozka1;

/* pomoci pointeru p_s komplikovane */
(*p_s).polozka1;

/* pomoci pointeru p_s jednoduseji */
p_s->polozka1;

/* tento zapis je chybny, protoze operator . ma prednost pred
   operatorem dereference *  */
*p_s.polozka1;
```

Cvičení

Máme sejf definovaný následujícím zdrojovým kódem.

Příklad

```c
#include <stdio.h>

typedef struct{
    char* popis;
    float hodnota;
} lup;

typedef struct{
    lup *lup;
    char *sekvence;
} kombinace;
```

Příklad

```c
typedef struct{
    kombinace cislo;
    char *vyrobce;
} sejf;

int main(){
    lup zlato = {"ZLATO!", 1000000.0};
    kombinace cislo = {&zlato, "1234"};
    sejf s  = {cislo,"SEJF2021"};

    return 0;
}
```

Jakou kombinaci následujících částí je potřeba zadat, abyste získali slovo `"ZLATO!"`?(Z každého sloupce vyberte jednu z možností.)

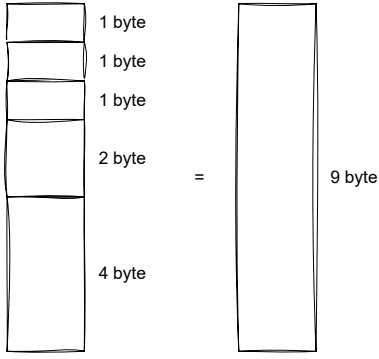
kombinace	.	s	+	lup	.	hodnota
s	->	cislo	.	popis	-	lup
cislo	:	lup	->	hodnota	->	popis
zlato	-	zlato	-	sekvence	+	zlato

Uniony

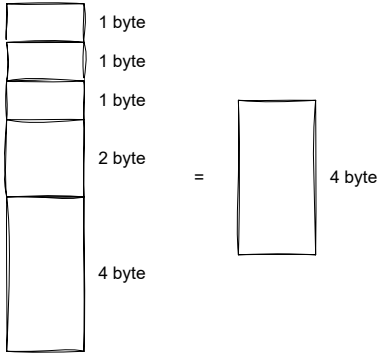
```
typedef union{
    polozka_1;
    polozka_2;
    ...
    polozka_n;
} nazev_union;
```

Struktury, uniony

Struktura



Union



Struktury, uniony

Příklad

Vytvořte union a strukturu se stejnými položkami a pomocí `sizeof()` zjistěte jejich velikost.

Uniony

Příklad

```c
#include <stdio.h>
typedef union{
    short pocet;
    float vaha;
    float objem;
} mnozstvi;

typedef struct{
    char nazev[20];
    mnozstvi mnozstvi;
} polozka;
```

Jak vyřešit problém s nejednoznačností?

Příklad

```c
int main(){
    polozka kosik[3];

    strcpy(kosik[0].nazev,"mrkev");
    kosik[0].mnozstvi.vaha = 0.5;
    strcpy(kosik[1].nazev,"mleko");
    kosik[1].mnozstvi.objem = 1.0;
    strcpy(kosik[2].nazev,"okurka");
    kosik[2].mnozstvi.pocet = 2;

    printf("%s %f\n", kosik[0].nazev,
           kosik[0].mnozstvi.vaha);
    printf("%s %f\n", kosik[1].nazev,
           kosik[1].mnozstvi.objem);
    printf("%s %d\n", kosik[2].nazev,
           kosik[2].mnozstvi.pocet);
    return 0;
}
```

Uniony

```
typedef union{
    polozka_1;
    polozka_2;
    ...
    polozka_n;
} nazev_union;
```

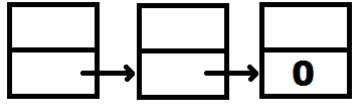
Příklad

```
typedef struct{
    int info;
    nazev_union x;  /* union */
} nazev_struktury;
```

Lineární seznam

Příklad

```
typedef struct _node{
    int data;
    struct _node *next;
} node;
```



Přidání prvku na začátek seznamu

Příklad

```
node *add(node **list, int data)
{
    node *new = malloc(sizeof(node));
    new->data = data;
    new->next = *list;
    *list = new;
    return new;
}
```

Cvičení

1. Funkce pro práci se seznamem. Napište funkci, která:
   1. vypíše všechny prvky seznamu.
   2. zjistí délku seznamu.
   3. přidá prvek na konec seznamu.
   4. smaže prvek na začátku seznamu.
   5. smaže prvek na konci seznamu.
   6. pro zadané i vypíše i-tý prvek seznamu.
   7. pro zadané i vypíše i-tý prvek seznamu od konce.
   8. pro zadané i smaže i-tý prvek seznamu.
   9. vytvoří kopii seznamu. Funkce musí fungovat tak, že pokud změníme kopii seznamu, originální seznam se nezmění.