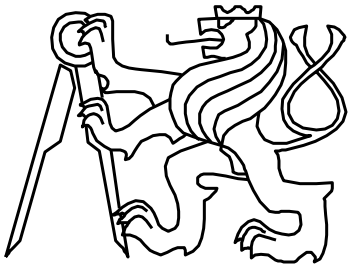
Příprava studijního programu Informatika je podporována projektem financovaným z Evropského sociálního fondu a rozpočtu hlavního města Prahy.
Praha & EU: Investujeme do vaší budoucnosti

Spojové struktury



BI-PA1 Programování a algoritmizace 1, ZS 2012-2013
Katedra teoretické informatiky
© Miroslav Balík
Fakulta informačních technologií
České vysoké učení technické

Obsah

- Téma
  - shrnutí typu struktura
  - spojové seznamy
    - vložení nového prvku na začátek spojového seznamu
    - průchod spojovým seznamem
    - vložení nového prvku na konec spojového seznamu
  - stromy
    - příklad binárního stromu

Shrnutí typu struktura

- Syntaxe popisu struktury:

  struct *značka* **{** *seznam popisů položek* **}**

- Popisy položek mají podobný tvar, jako deklarace proměnných
  - položka nesmí být typu stejná struktura
  - položka může být ukazatelem na stejnou strukturu

```
struct osoba {
    char jmeno[20];
    char prijmeni[25];
    int rok_narozeni;
}; /* deklarace struktury */
struct osoba Jan; /* deklarace proměnné typu struct osoba */
```

Značka struktury není identifikátorem typu

```
    osoba Petr;      /* chyba */
```

Shrnutí typu struktura

- Identifikátor typu lze pro strukturu zavést deklarací typu

```
typedef struct Complex {float Real, Imag;} Complex;
Complex a, b;                    /* O.K. */
```

- Zpřístupnění položky struktury:
  - Přímá selekce:

    X . *položka* kde **X** je výraz typu **struct**
  - Nepřímá selekce (přes ukazatel):

    X -> *položka* kde **X** je výraz typu ukazatel na **struct**
- **X ->** *položka* je zkratkou za **(*X) .** *položka*

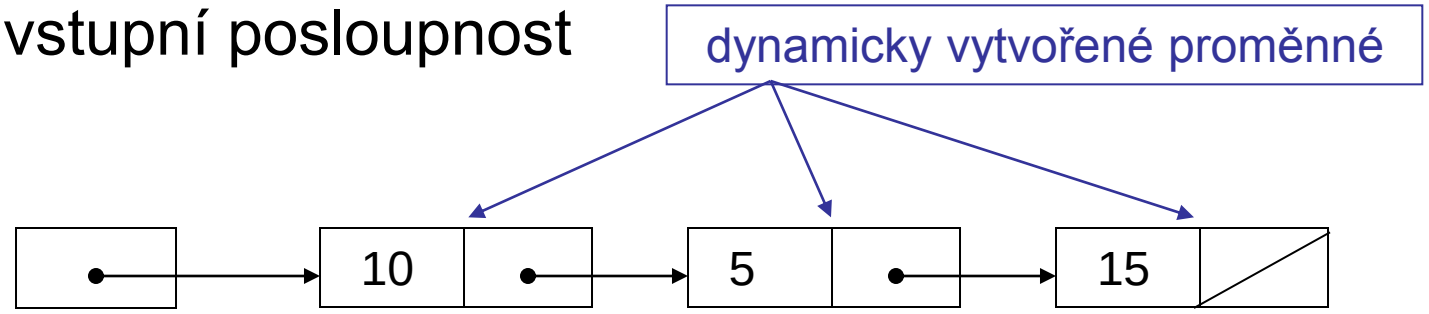
```
struct { int a; char b; } x, *px = &x;
x.a = 1;
px -> b = ‘a’;
(*px).b = ‘a’; /* totéž */
```

Úvodní příklad na spojové seznamy

- Program, který přečte řadu celých čísel zakončených nulou a vypíše je v opačném pořadí (bez závěrečné nuly)
- Úlohu jsme řešili:
  - pomocí rekurzivní procedury (nebylo třeba omezit počet čísel)
  - pomocí pole (bylo třeba omezit počet čísel)
- Vyřešme ji pomocí dynamických proměnných, z nichž vytvoříme *spojový seznam* (nebude potřeba omezit počet čísel). Princip:
  - pro každé přečtené číslo dynamicky vytvoříme proměnnou typu struktura, v jejíž jedné položce bude číslo a ve druhé ukazatel na další proměnnou obsahující dříve přečtené číslo
- Příklad: pro vstupní posloupnost

  15 5 10 0

  vytvoříme:



Úvodní příklad na spojové seznamy

Dynamicky vytvářené proměnné budou typu *Prvek*:

```
typedef struct prvek {
    int hodn;
    struct prvek *dalsi;
} Prvek;
```

Pro vytvoření dynamické proměnné typu *Prvek* zavedeme funkci:

```
Prvek *vytvorPrvek(int hodn, Prvek *dalsi) {
    Prvek *pom = (Prvek*)malloc(sizeof(Prvek));
    pom->hodn = hodn;
    pom->dalsi = dalsi;
    return pom;
}
```

lépe **sizeof(*pom)**,
velikost vypočítá překladač (od C99 až za běhu)
podle proměnné, odolnější proti chybě programátora

Ukazatelem na první prvek spojového seznamu bude proměnná

```
Prvek *zacatek = NULL;
```

Úvodní příklad na spojové seznamy

- Postup při vytvoření spojového seznamu:
  - pokud přečtené číslo není nula, pak
    - uložíme ho do dynamicky vytvořené proměnné typu *Prvek,* která v položce *další* bude obsahovat ukazatel na začátek doposud vytvořeného spojového seznamu
    - ukazatel na novou proměnnou typu *Prvek* se stane ukazatelem na rozšířený spojový seznam
  - skončíme, když je přečtené číslo nula
  - pak spojový seznam projdeme od začátku do konce a čísla vypíšeme

Úvodní příklad - prog12-seznam1.c

```
int main(void) {
    Prvek *zacatek = NULL, *p;
    int cislo;
    printf("zadej posloupnost cisel zakoncenou nulou\n");
    scanf("%d", &cislo);
    while (cislo) {
        zacatek = vytvorPrvek(cislo, zacatek);
        scanf("%d", &cislo);
    }
    printf("vypis cisel v opacnem poradi\n");
    p = zacatek;
    while (p) {
        printf("%d ", p->hodn); p = p->dalsi;
    }
    printf("\n"); system("PAUSE"); return 0;
}
```

Druhý příklad

- Úvodní příklad ilustroval:
  - vytváření spojového seznamu vkládáním prvku na začátek
  - průchod spojovým seznamem a provedení operace pro všechny prvky
- Další příklad na průchod spojovým seznamem (prog12-seznam2.c):
  - přečteme čísla zakončená nulou
  - vytvoříme z nich spojový seznam (vkládáním prvku na začátek seznamu)
  - pak čteme další posloupnost čísel zakončenou nulou
  - pro každé číslo vypíšeme, zda je či není prvkem spojového seznamu

Druhý příklad

- Funkce pro výpis seznamu

```
void vypisSeznamu(Prvek *p) {
        while (p) {
                printf("%d ", p->hodn);
                p = p->dalsi;
        }
        printf("\n");
}
```

- Funkce pro hledání hodnoty v seznamu

```
int jePrvkem(Prvek *p, int x) {
        while (p && p->hodn!=x) p = p->dalsi;
        if (p) return 1;
        else return 0;
}
```

Druhý příklad – funkce main

```
int main(void) {
    Prvek *zacatek = NULL, *p;
    int cislo;
    printf("zadej posloupnost cisel zakoncenou nulou\n");
    scanf("%d", &cislo);
    while (cislo) {
        zacatek = vytvorPrvek(cislo, zacatek);
        scanf("%d", &cislo);
    }
    printf("vypis cisel v opacnem poradi\n");
    vypisSeznamu(zacatek);
    printf("zadej dalsi posloupnost cisel zakoncenou nulou\n");
    ...
```

Druhý příklad – funkce main, 2. část

```
...
    printf("zadej dalsi posloupnost cisel zakoncenou nulou\n");
    scanf("%d", &cislo);
    while (cislo) {
            printf("cislo %d ", cislo);
            if (jePrvkem(zacatek, cislo)) printf("je ");
            else printf("neni ");
            printf("prvkem seznamu\n");
            scanf("%d", &cislo);
    }
    printf("\n");
    system("PAUSE");
    return 0;
}
```

Vložení prvku na konec spojového seznamu I

1. přečteme posloupnost čísel zakončenou nulou
2. vytvoříme spojový seznam, ve kterém budou čísla v pořadí, v jakém jsou čtena
3. seznam vypíšeme

- Seznam budeme vytvářet vkládáním nového prvku na konec
- K vložení na konec musíme najít poslední prvek seznamu
  - poslední prvek má v položce *dalsi* hodnotu **NULL**
  - na počátku je seznam prázdný, tj. *zacatek* je **NULL**
- Zavedeme funkci:

```
Prvek *najdiPosledni(Prvek *p) {
    if (!p) return NULL;
    while (p->dalsi) p = p->dalsi;
    return p;
}
```

Vložení prvku na konec spojového seznamu I

```
/*prog12-seznam3a.c*/
int main(void) {
        Prvek *zacatek = NULL, *posledni, *p;  int cislo;
        printf("zadej posloupnost cisel zakoncenou nulou\n");
        scanf("%d", &cislo);
        while (cislo) {
                p = vytvorPrvek(cislo, NULL);
                posledni = najdiPosledni(zacatek);
                if (posledni) posledni->dalsi = p;
                else zacatek = p;
                scanf("%d", &cislo);
        }
        printf("vypis cisel\n");
        vypisSeznamu(zacatek);
        system("PAUSE");
        return 0;
}
```

Vložení prvku na konec spojového seznamu II

- Pokud při vkládání prvku na konec spojového seznamu hledáme poslední prvek, má operace vložení lineární časovou složitost O(n)
- Poznámka: operace vložení prvku na začátek spojového seznamu má konstantní časovou složitost O(1)

```
int main(void) {
        Prvek *zacatek = NULL, *posledni = NULL, *p;
        int cislo;
        scanf("%d", &cislo);
        while (cislo) {
            p = vytvorPrvek(cislo, NULL);
            if (posledni) posledni->dalsi = p;
            else zacatek = p;
            posledni = p;
            scanf("%d", &cislo);
        }
... }
```

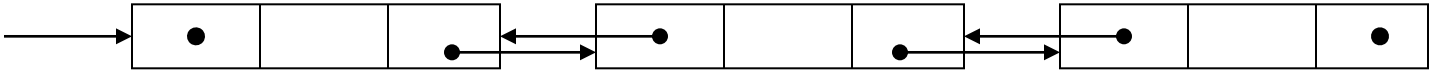
Vložení prvku na konec spojového seznamu bude mít konstantní složitost, budeme-li si pamatovat ukazatel na poslední prvek

(prog12-seznam3b.c)

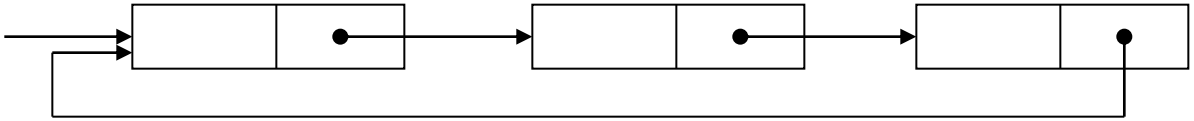
Spojové struktury

- Spojový seznam z předchozích příkladů patří mezi *spojové struktury*
- Spojová struktura (linked structure):
  - množina objektů propojených pomocí spojů (ukazatelů)
- Spoj často vyjadřuje vztah předchůdce – následník
- Lineární spojové struktury (spojové seznamy): každý prvek struktury má nanejvýš jednoho následníka
- Příklady spojových seznamů:
  - jednosměrný

  - dvousměrný

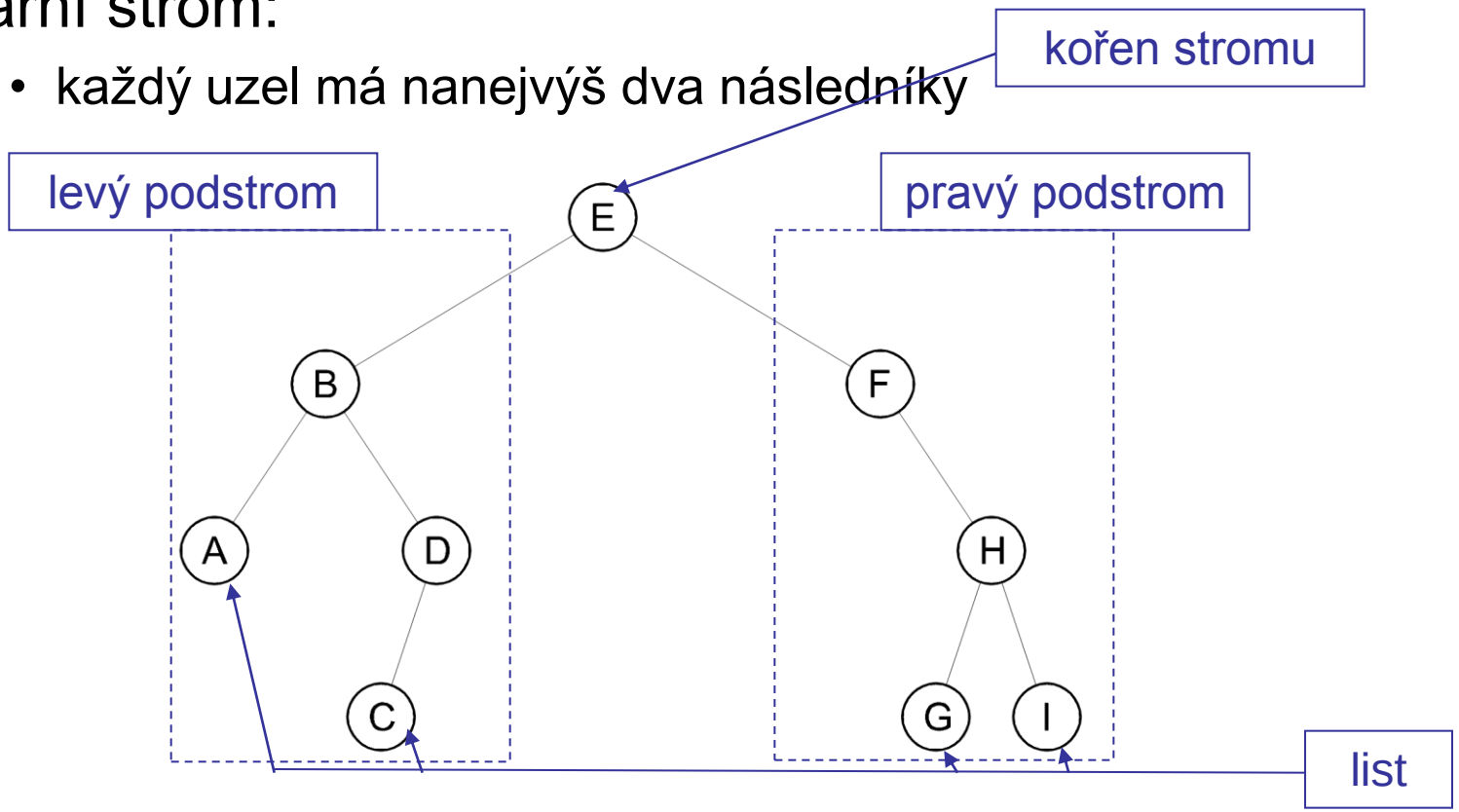


  - cyklický jednosměrný



Stromy

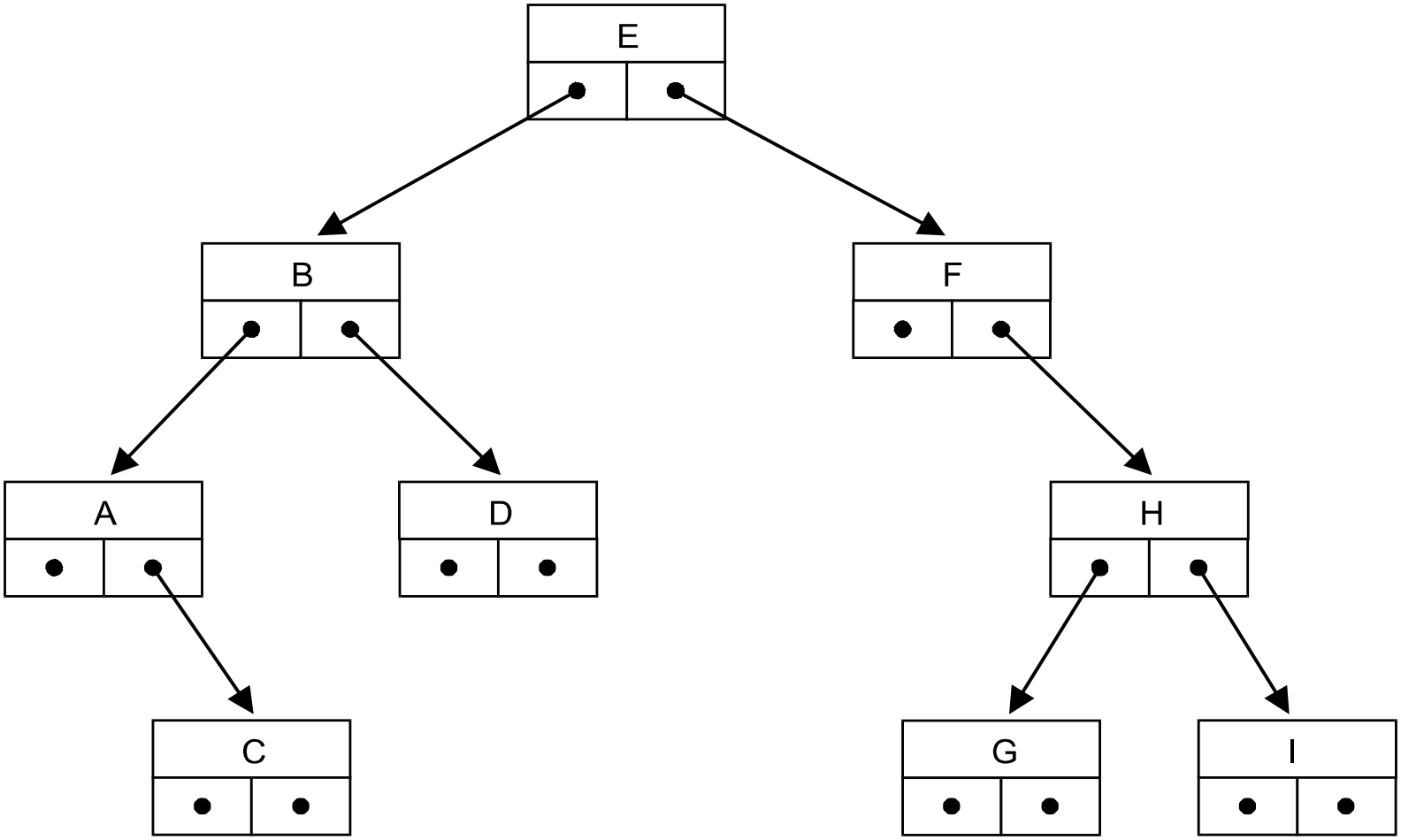
- Nelineární spojová struktura:
  - každý prvek může mít více následníků
- Příkladem nelineární spojové struktury je strom (prvky nazýváme *uzly*)
- Binární strom:
  - každý uzel má nanejvýš dva následníky



Realizace binárního stromu

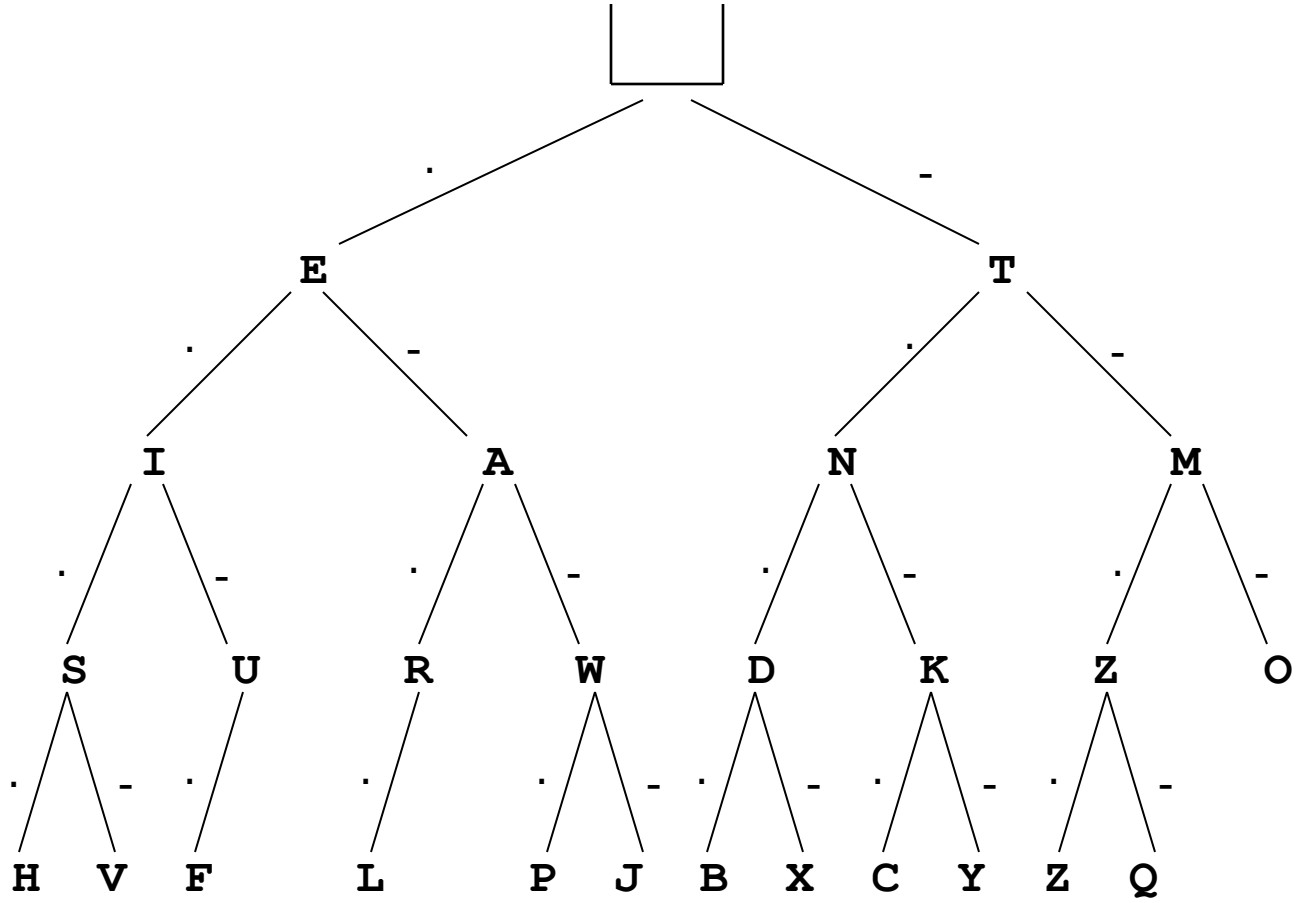
Struktura uzlů:

Příklad binárního stromu:



Příklad – dekódování morseovky

Pro dekódování textu zapsaného v Morseově abecedě lze použít následující binární strom



Příklad – dekódování morseovky

- Strom vytvoříme ze struktur typu MUzel

```
typedef struct MUzel {
    char znak;
    struct MUzel *tecka, *carka;
} MUzel;
```

…… a pomocí funkcí:

```
MUzel *novyMUzel(char z, MUzel *t, MUzel *c) {
    MUzel *p = (MUzel*)malloc(sizeof(MUzel));
    p->znak = z; p->tecka = t; p->carka = c;
    return p;
}
MUzel *novyMUzel1(char z) {
    MUzel *p = (MUzel*)malloc(sizeof(MUzel));
    p->znak = z; p->tecka = NULL; p->carka = NULL;
    return p;
}
```

Dekódování morseovky, vytvoření stromu

```
MUzel *strom()                                                    {
    return novyMUzel('  ',
                novyMUzel('E',                  /* . */
                   novyMUzel('I',               /* .. */
                      novyMUzel('S',            /* ... */
                         novyMUzel1('H'),       /* .... */
                         novyMUzel1('V')        /* ...- */
                                  ),
                      novyMUzel('U',            /* ..- */
                         novyMUzel1('F'),       /* ..-. */
                         NULL     )             /* ..-- */
                               ),
                   novyMUzel('A',               /* .- */    ...
                               )
                            ),
                novyMUzel('T',                  /* - */     ...)   );   }
```

Příklad – dekódování morseovky

- Prvním parametrem je řetěz obsahující Morseův kód a druhým parametrem je pole, kam se uloží dekódovaný text

```
void dekoduj(char odkud[], char kam[]) {
    MUzel *aktualni = koren;
    int i = 0, j = 0, delka = strlen(odkud);
    while (i<delka) {
        char z = odkud[i];
        if (aktualni!=NULL) {
            if (z=='.') aktualni = aktualni->tecka;
            else if (z=='-') aktualni = aktualni->carka;
            else {    kam[j++] = aktualni->znak;  aktualni = koren; }
            i++;
        } else {
            kam[j++] = '?';
            while (odkud[i]=='.' || odkud[i]=='-') i++;
            aktualni = koren;
        }
    }
    kam[j] = 0;
}
```

Dekódování morseovky - main

```
/* prog12-morseovka.c*/
MUzel *koren;
#define MAXDELKA 100
int main() {
    koren = strom();
    char morse[MAXDELKA], text[MAXDELKA];
    fgets(morse, MAXDELKA, stdin);
    dekoduj(morse, text);
    fprintf(stdout, "%s\n", text);
    system("PAUSE");
    return 0;
}
```

Příklad - hra „Jaké zvíře si myslíš“

- Příklad dialogu počítače a člověka :

  Myslite si nejake zvire?

  ano

  leta?

  ne

  Je to ryba?

  ne

  Neuhadl jsem. Prosim o doplneni znalosti.

  Jake zvire jste myslel?

  pes

  Napiste otazku vystihujici rozdil mezi pes a ryba

  steka

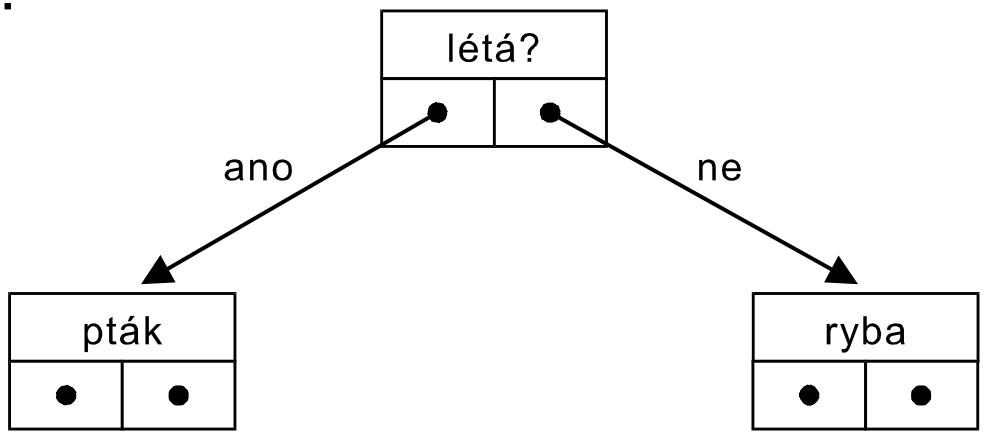
  Pro pes je odpoved ano ci ne

  ano

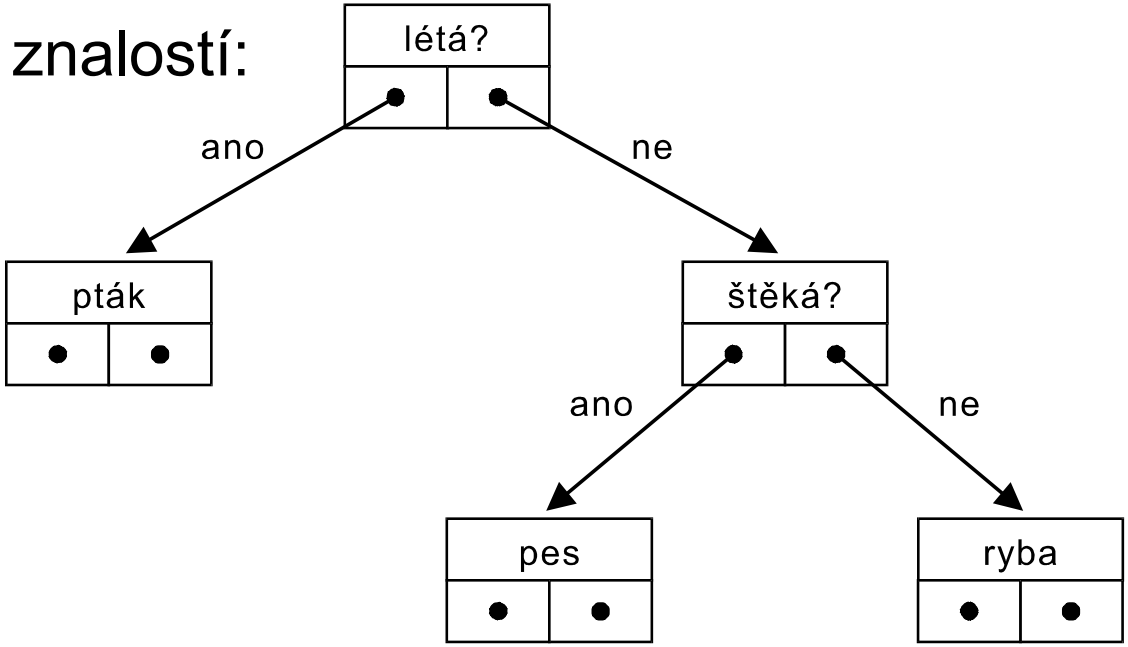
  Dekuji, chcete pokracovat?...

Příklad - hra „Jaké zvíře si myslíš“

- Počáteční strom :



- Strom po doplnění znalostí:



„Jaké zvíře si myslíš“ -hrubé řešení

```
"úvod dialogu";
"aktuálním uzlem je kořen stromu";
do {
        "polož otázku uvedenou v aktuálním uzlu";
        if ("odpověď je ano") "aktuálním uzlem je levý následník“;
        else                          "aktuálním uzlem je pravý následník“;
} while ("aktuální uzel není list");
"polož závěrečnou otázku, název zvířete vyber z aktuálního uzlu";
if ("odpověď je ano")   "hádání bylo úspěšné“;
else {
                              "hádání bylo neúspěšné";
                               "doplň znalosti"
}
```

„Jaké zvíře si myslíš“ – uzel a vytvoření

```
typedef struct Uzel {
    char text[MAXDELKA];
    struct Uzel *ano, *ne;
 } Uzel;
Uzel *vytvorUzel1(char t[]) {
        Uzel *u = (Uzel*)malloc(sizeof(Uzel));
        strcpy(u->text, t);
        u->ano = NULL;
        u->ne = NULL;
        return u;
}
Uzel *vytvorUzel(char t[], Uzel *a, Uzel *n) {
    Uzel *u = (Uzel*)malloc(sizeof(Uzel));
    strcpy(u->text, t);
    u->ano = a; u->ne = n;
    return u;
}
```

„Jaké zvíře si myslíš“

- Test, zda uzel je listem, inicializaci stromu a čtení odpovědi:

```
int jeList(Uzel *u) {
    return u->ano==NULL && u->ne==NULL;
}

Uzel *inicializaceStromu() {
    return vytvorUzel("leta?", vytvorUzel1("ptak"), vytvorUzel1("ryba") ); }

int odpovedAno() {
    char odpoved[MAXDELKA];
    scanf("%s", odpoved);
    if (odpoved[0]=='a' || odpoved[0]=='A') return 1;
    else return 0;
}
```

„Jaké zvíře si myslíš“ – main /* prog12-hra.c */

```
int main() { Uzel *koren = inicializaceStromu();
             Uzel *aktualni = koren;
  for (;;) { printf("Myslite si nejake zvire?\n");
             if (!odpovedAno()) break;
             do { printf("%s\n", aktualni->text);
                  if (odpovedAno())  aktualni = aktualni->ano;
                  else               aktualni = aktualni->ne;
                } while (!jeList(aktualni));
             printf("Je to %s?\n", aktualni->text);
             if (odpovedAno()) printf("Uhadl jsem\n");
             else { printf("Neuhadl jsem. Prosim o doplneni znalosti\n");
                    dopInPodstrom(aktualni);
                  }
             printf("Dekuji. Chcete pokracovat?\n");
             if (!odpovedAno()) break;
           }  return 0;  }
```

„Jaké zvíře si myslíš“- doplnění znalostí:

```
void doplnPodstrom(Uzel *p) {
    char noveZvire[MAXDELKA], novaOtazka[MAXDELKA];
    Uzel *novyAno, *novyNe;
    printf("Jake zvire jste myslel?\n");
    scanf("%s", noveZvire);
    printf("Napiste otazku vystihujici rozdil mezi %s a %s\n",
            noveZvire, p->text);  scanf("%s", novaOtazka);
    printf("Pro %s je odpoved ano ci ne?\n", noveZvire);
    if (odpovedAno()) { novyAno = vytvorUzel1(noveZvire);
                        novyNe = vytvorUzel1(p->text);
    } else {            novyAno = vytvorUzel1(p->text);
                        novyNe = vytvorUzel1(noveZvire);
    }
    strcpy(p->text, novaOtazka);  p->ano = novyAno;  p->ne = novyNe;
}
```