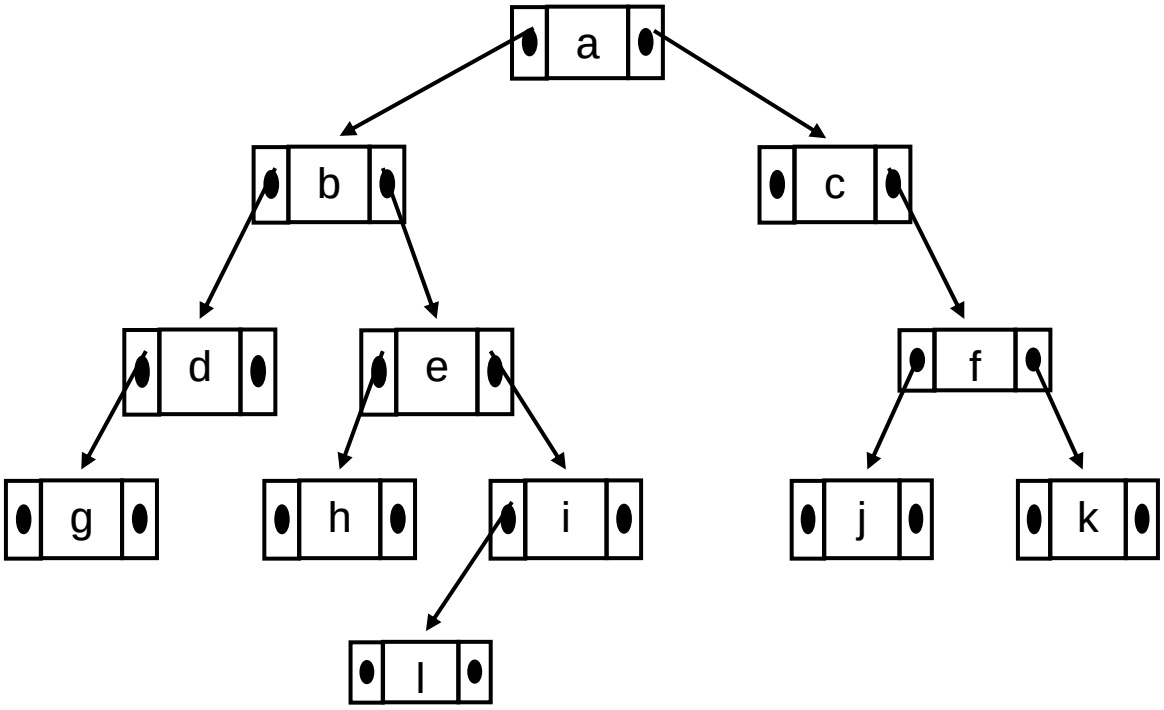
CVIČENIE 10/13 (S7)

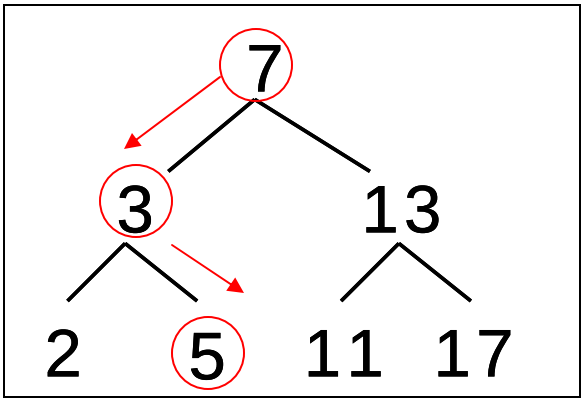
Programovanie v jazyku C – stromy

Binarne stromy:

- zlozene z nula alebo viac prvkov
- kazdy prvok obsahuje
  - nejaku hodnotu (data)
  - smernik na laveho potomka
  - smernik na praveho potomka
- moze byt *empty*
- ak nie je *empty* ma *root node*
  - kazdy prvok binarneho stromu je dosazitelny z korenoveho prvku po unikatnej ceste
- prvok bez potomkov je nazyvany list (*leaf*)
  - existuju binarne stromy kde len listy obsahuju data



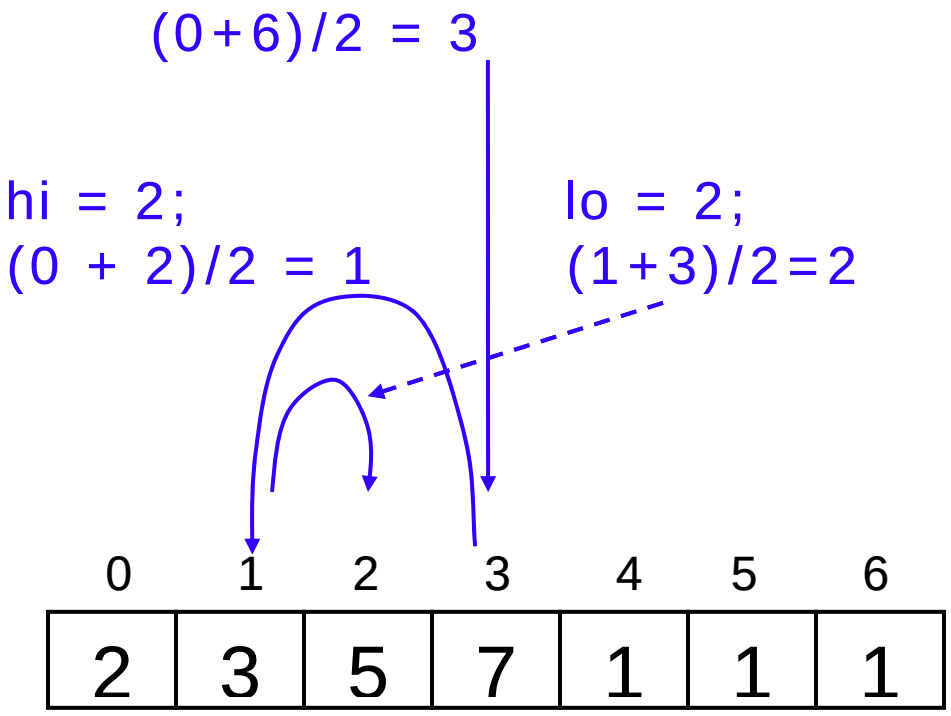
- velkost binarneho stromu je pocet prvkov v nom (12)
- hlbka prvku je vzdialenost od korena (e – 2)
- hlbka stromu je maximalna vzdialenost od korena (4)
- vyvazeny binarny strom – vsetky prvky okrem tych v najspodnejsie vrstve maju dvoch potomkov



Prehladavanie v zatriedenom poli:

Zacni: (lo + hi)/2

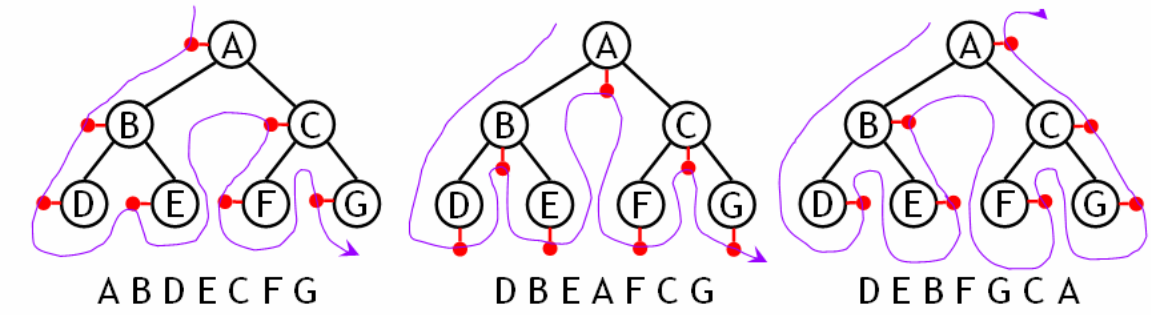
Hladame 5:



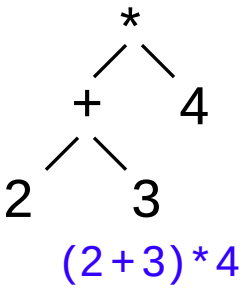
Prechadzanie stromu:

- kazdy binarny strom je definovany rekurzivne, pomocou korenoveho nodu a laveho podstromu a praveho podstromu
- prechadzanie takeho to stromu tak aby sme kazdy prvok navstivili prave raz je teda rekurzivne

Preorder : root, left, right
Inorder : left, root, right
Postorder: left, right, root

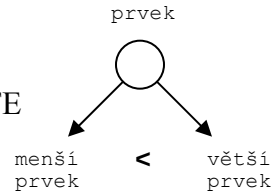


Binarne stromy su vhodne napriklad na vyjadrenie vyrazov:



BINÁRNÍ STROMY
- vyhledávací stromy

1. Navrhnout datové struktury
2. Operace INSERT, SHOW, DELETE



Tato podmínka platí pro každý prvek!!

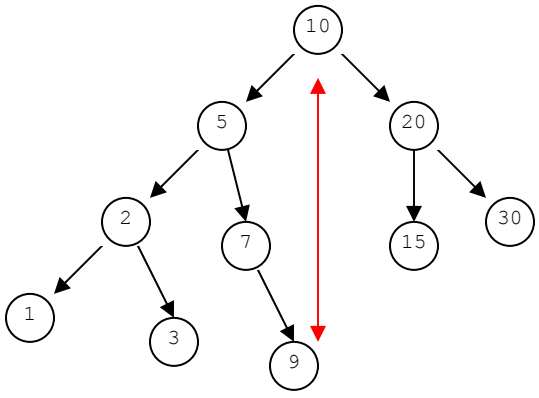
udělat INSERT bez podmínky if (vyjma při testování, zda se naalokovala paměť)
→ hint: každá podmínka vrací 0 nebo 1 → tedy int

Pozn.
- pro tisk stromu se hodí tisk několika mezer
  printf("%5c",'a') ... vytiskne se znak 'a' na šířku 5
  printf("%*c",k,'a') ... vytiskne se znak 'a' na šířku k

→ pokud je šířka udána * → určuje počet znaků argument typu int, který musí v seznamu argumentů bezprostředně předcházet argumentu, který se má tisknout

INSERT
- najdu místo ve stromě, kam uzel patří a tak vložím nový list

DELETE
- list ... smažu
- uzel s 1 synem ... přepojím a smažu
- uzel se 2 syny ... najdu nejpravější v levém podstromu (nebo nejlevější v pravém podstromu) a ten uzel prohodím s mazaným uzlem



Prolézání stromu

1. pomocí if

```
if (x<p->x)
  p = p->l;
else
  p = p->r;
```

2. bez if

```
enum {left, right};
struct prvek {
  int x;
  prvek* dalsi[2];
}
p = p->dalsi[x > p->x];
```

Př. naprogramujte binární vyhledávací strom – použít pointer na pointer

- **insert**
- **tisk stromu**
- **(delete)**

```
#include <malloc.h>
#include <stdio.h>

struct prvek {
  int val;
  prvek* kam[2];    // v kam[0] bude ukazatel doleva, v kam[1] bude ukazatel doprava
};

struct strom {
  prvek* koren;
};

// pridani prvku do stromu
int add_strom(strom* st, int x){
  printf("Add %d \n",x);
  prvek* n;
  n = (prvek*)malloc(sizeof(prvek));
  if (!n) {
          printf("neni pamet");
          return -1;
  }
  n->val = x;
  n->kam[0] = NULL;
  n->kam[1] = NULL;

  prvek** p;
  p = &(st->koren);
```

```
  while (*p)
    p = &((*p)->kam[x>(*p)->val]);

  *p = n;
  return 0;
}


  // smazani prvku ve stromu
int del_strom(strom* st, int x){
  printf("Del %d\n",x);
  prvek** p;
  prvek* tm;
  prvek** tmp;

  p = &(st->koren);

  // prazdny strom
  if (!(*p)) {
    printf("prazdny strom - nelze mazat");
          return -1;
  }

  // hledam uzel, ktery se ma smazat
  while ( (*p) && ((*p)->val!=x) ) {
    p = &((*p)->kam[x>(*p)->val]);
  }

  // uzel ve stromu neni
  if (!(*p)) {
    printf("prvek ve strome neni - nelze mazat \n");
    return 1;
  }

  // uzel ve stromu JE
  // - uzel “p” ma 2 syny
  if( (*p)->kam[0] && (*p)->kam[1] ){
    // najdu nejmensi (nejlevejsi) prvek "tmp" v pravem podstrome
    tmp = &((*p)->kam[1]);
    while ( (*tmp) && ((*tmp)->kam[0]) )
          tmp = &((*tmp)->kam[0]);

    // zkopiruju hodnotu do uzlu, ktery chci mazat
    (*p)->val = (*tmp)->val;
    //smazu nejlevejsi uzel v prazem podstromu (*tmp)
```

```
    // - uzel (*tmp) nema leveho syna
    tm = *tmp;
    (*tmp) = (*tmp)->kam[1]; // prepojime praveho syna
    free(tm);
   }
   else { // jeden ze synu uzlu “p” chybi
    tm = *p;
    // kdyz neni levy syn, prepojime o patro vys praveho syna
    *p = (!((*p)->kam[0])) ? (*p)->kam[1] : (*p)->kam[0];
    free(tm);
   }

   return 0;
}


// tisk jednoho prvku
int print_prvek(prvek* p){
   if (p) {
    if ((p->kam[0]) &&(p->kam[1]))
      printf("uzel %d s L:%d a P:%d\n",p->val,p->kam[0]->val,p->kam[1]->val);
    if ((!p->kam[0]) &&(p->kam[1]))
      printf("uzel %d s P:%d\n",p->val,p->kam[1]->val);
    if ((p->kam[0]) &&(!p->kam[1]))
      printf("uzel %d s L:%d\n",p->val,p->kam[0]->val);
    if ((!p->kam[0]) &&(!p->kam[1]))
      printf("uzel %d \n",p->val);

    print_prvek(p->kam[0]);
    print_prvek(p->kam[1]);
   }
   return 0;
}


// tisk celeho stromu
int print_strom(strom* st){
          if (!(st->koren)) {
            printf("Prazdny strom \n");
            return 1;
   }
   print_prvek(st->koren);
   printf("\n\n");
   return 0;
}
```

```
// inicializace stromu
strom* init_strom(){
  strom* st = (strom*) malloc(sizeof(strom));
  st->koren=NULL;
  return st;
}

// main
int main (int argc, char** argv){
        strom* tr;
        tr = init_strom();
        print_strom(tr);
        add_strom(tr,3);
        print_strom(tr);
        add_strom(tr,5);
        print_strom(tr);
        del_strom(tr,5);
        print_strom(tr);
        del_strom(tr,10);
        print_strom(tr);

return 0;
}
```

BINÁRNÍ VYVÁŽENÝ VYHLEDÁVACÍ STROM
- binární vyhledávací strom
- pro každý uzel platí navíc podmínka, že výška v pravém a levém podstromu se liší nejvýše o 1

3. DOMÁCÍ ÚKOL

Př. program, který načte jako parametry příkazy pro práci s binárním stromem
ix ... vloží prvek s hodnotou "x" do binárního stromu
dx ... vybere prvek s hodnotou "x" z binárního stromu

např. myprog i4 d4 i8 i9 d7

Možná struktura main:

```
int main (int argc, char** argv){
        char* param;

        .....
```

```
        argv++; // posuneme se na prvni skutečný parametr;

        // dokud mame nejaky parametr
        while (*argv){
                switch (**argv){ // switch podle prvního znaku v parametru
                        case 'i':
                                .....
                                break;
                        case 'd':
                                ......
                                break;
                }
                argv++;
        }

        return 0;
}
```

Řešení main:

```
int main (int argc, char** argv){
        strom* tr;
        tr = init_strom();

        argv++; // posuneme se na prvni parametr;

        // dokud mame nejaky parametr
        while (*argv){
                switch (**argv){
                        case 'i':
                                add_strom(tr,atoi(*argv+1));
                                print_strom(tr);
                                break;
                        case 'd':
                                del_strom(tr,atoi(*argv+1));
                                print_strom(tr);
                                break;
                }
                argv++;
        }

        return 0;
}
```