

## Funkce bez parametrů

Má smysl definovat funkci bez parametrů (kupř. chceme-li načítat).  
V tom případě můžeme vynechat kulaté závorky jak při definici,  
tak při volání:

```
function x:integer;
```

```
begin
```

```
    x:=10;
```

```
end;
```

```
...
```

```
a:=x;
```

```
...
```

Pozor na chybu ve Free Pascalu!

# Procedury

"Procedury jsou funkce, které nevracejí hodnotu."

```
procedure jmeno(argumenty);
```

```
... jmeno(argumenty);...
```

Příklad:

```
procedure vyp(a:integer;b:integer);
```

```
begin
```

```
    writeln(a); writeln(b);
```

```
    {Vypsali jsme argumenty}
```

```
end;
```

```
... vyp(5,10);...
```

## Vnořené funkce a procedury

Proceduru či funkci lze definovat i uvnitř jiné procedury nebo funkce:

```
procedure f(a:integer);  
    procedure g(b:integer);  
    begin  
        writeln('Proc. g v procedure f s par-em ',b);  
    end;  
begin  
    writeln('Procedura f s parametrem ',a);  
    g(2);{Volame vnorenou proc. g}  
end;
```

## Scope resolution

- Procedura/funkce vidí jen funkce (procedury) vnořené přímo do sebe (ne do svých vnořených funkcí/procedur)!
- Vnořené funkce vnášejí další zmatek do významu proměnných (a dokonce procedur a funkcí).
- Identifikátor má takový význam, jaký je v danou chvíli "nejlokálnější".

## Příklad

```
procedure f(h:integer);
  procedure g(b:integer);
    procedure h(c:integer);
      begin
        writeln('Procedura h s par. ',c);
      end;
    begin
      writeln('Procedura g s parametrem ',b);
      h(5);
    end;
  begin
    writeln('Funkce f s parametrem ',h);
    g(3); f(5); {zatim OK, ale h(4) udela chybu!}
  end;
```

# Rekurze

- Může mít dobrý smysl volat z funkce sebe sama.
- Této metodě říkáme **rekurze**.
- **Rekurze není nic jiného, než vtipně pojmenovaná indukce!**
- Příklady: Úředníci na úřadech, Faktoriál, Caesarův kód...

## Úředníci na úřadech

- Člověk chce nechat provést úřední výkon.
- Úředník požaduje vyplnění papírů vyžadující návštěvy dalších úřadů.
- Řešení:

```
procedure vypln(musime_vyplnit:seznam_papiru);  
var x:seznam_papiru;  
for papir in musime_vyplnit do  
begin  
    x:=zeptej_se_urednika(papir);  
    if nonempty(x) then  
        vypln(x);  
end;
```

# Faktoriál

- $n! = 1 \cdot 2 \cdot \dots \cdot n$
- Jak tuto funkci naprogramovat?
- Cyklem:  
fakt:=1;  
for i:=1 to n do  
    fakt:=fakt\*i;
- nebo rekurzí.



## Faktoriál rekurzí:

```
function faktorial(a:integer):integer;  
begin  
    if a<2 then  
        faktorial:=1  
    else faktorial:=a*faktorial(a-1);  
end;
```

Jaká je složitost výpočtu funkce faktoriál?

## Přednášející jde do posluchárny

- Přednášející při cestě po posluchárny na schodech vždycky buď to šlápne na následující schod, nebo jeden schod překročí.
- Kolika způsoby může vylézt do posluchárny F1?  
(schody nepočítejte, jejich počet odhadněte)
- Nápady?

## Přednášející do posluchárny – řešení

- Vzniká rekurence  $f_n = f_{n-1} + f_{n-2}$ .
- Rekurence není nic jiného, než matematický zápis rekurze.
- Řešení:

```
function schody(a:integer):integer;
begin
    if a=1 then schody=1
    else if a=2 then schody=2
    else
        schody:=schody(a-1)+schody(a-2);
    end;
```

- Jaký je problém?
- Ano, příšerná složitost.

## Maximální jedničková podmatice

Problém: V matici tvaru  $m \times n$  vyplněné nulami a jedničkami máme najít největší podmatici (souvislou) obsahující pouze jedničky.

# Naivní algoritmus

- Pro každý možný levý horní a pravý dolní roh prohlédni vnitřek matice.
- Algoritmus funguje, ale s jakou složitostí?
- $\Theta(mn)$  levých horních rohů,  $\Theta(mn)$  pravých dolních rohů,  $\Theta(mn)$  prvků uvnitř (proč?), celkem tedy  $\Theta(m^3n^3)$ .
- Nápady na zlepšení?