01 – Rekurze

Miroslav Jílek

01 – Rekurze

Rekurzivní spouštění funkce znamená, že funkce spouští (startuje) sama sebe.

Při každém novém spuštění se vytvoří v paměti nová a nezávislá instance proměnných, ve které jsou nové hodnoty. Změna hodnoty proměnné v jedné instanci neovlivní hodnotu stejné proměnné v ostatních instancích.

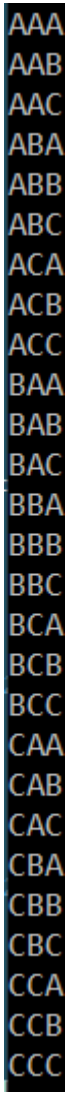
Tyto nové proměnné mají stejný název, ale jinou adresu v paměti.
Tyto nové proměnné a jejich hodnoty jsou přístupné pouze z toho spuštění funkce, které je v paměti vytvořilo.

Musíme si dát pozor na změny hodnot parametru nového spuštění funkce. Jestli máme jako parametr hodnotu proměnné a změníme ji, mění se tato hodnota v celé instanci funkce, ze které děláme nové spuštění. Změnit hodnotu parametru předávaného proměnnou je možné i tak, že do parametru dáme výraz proměnná Prom+1. Tímto způsobem se hodnota proměnné v dané instanci nezmění, ale do nové instance přijde vstupní hodnota o jedničku větší, než je hodnota proměnné Prom.

Příklad:

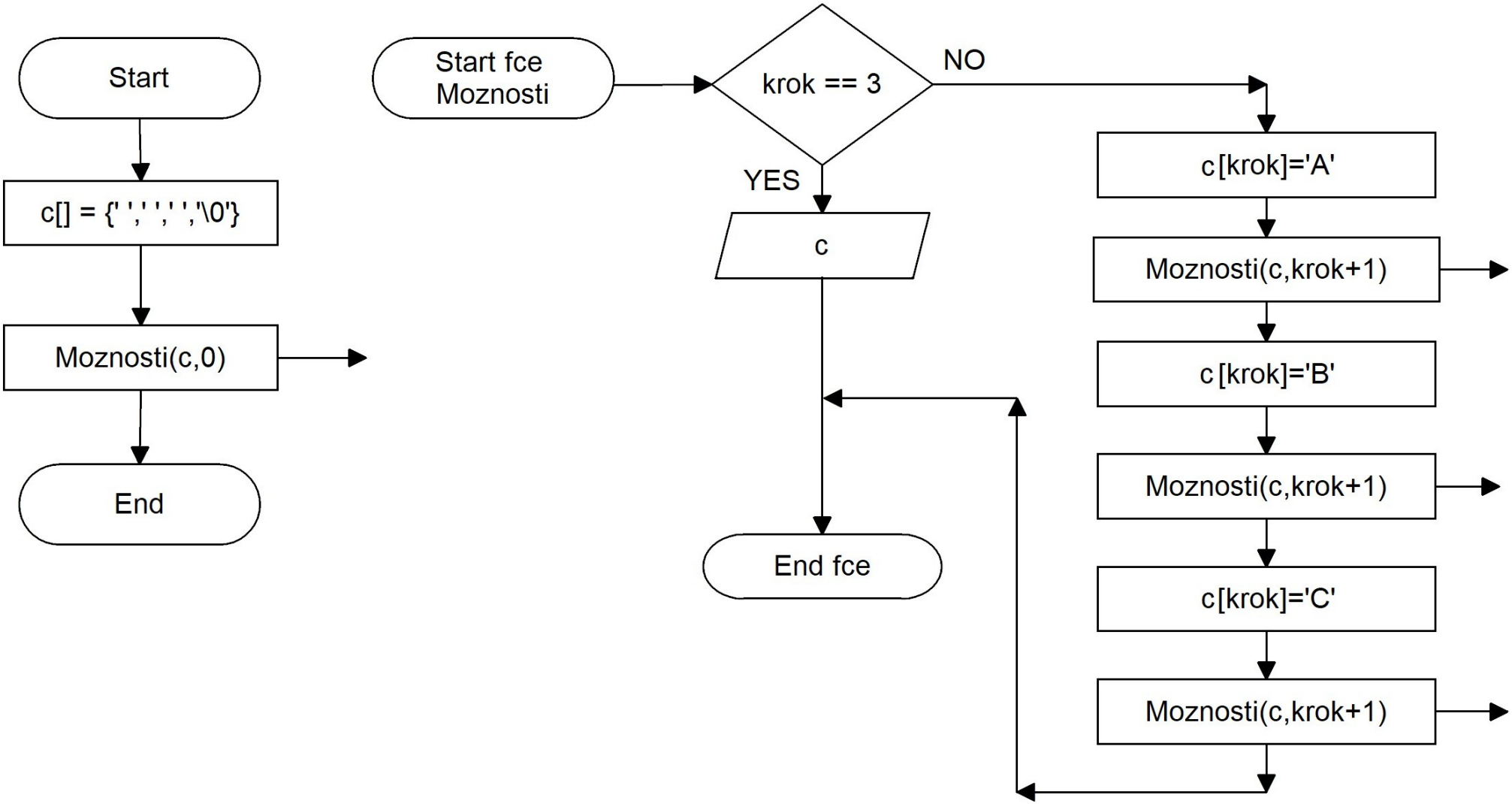
Navrhněte program, který vytvoří všechny možné variace s opakováním tří písmen A, B a C. Použijte rekurzi.

Ukázka výstupu:



Variace s opakováním: $V_{opak.}(n,k) = n^k$ v našem případě to je 3^3 = 27 možností

Vývojový diagram:



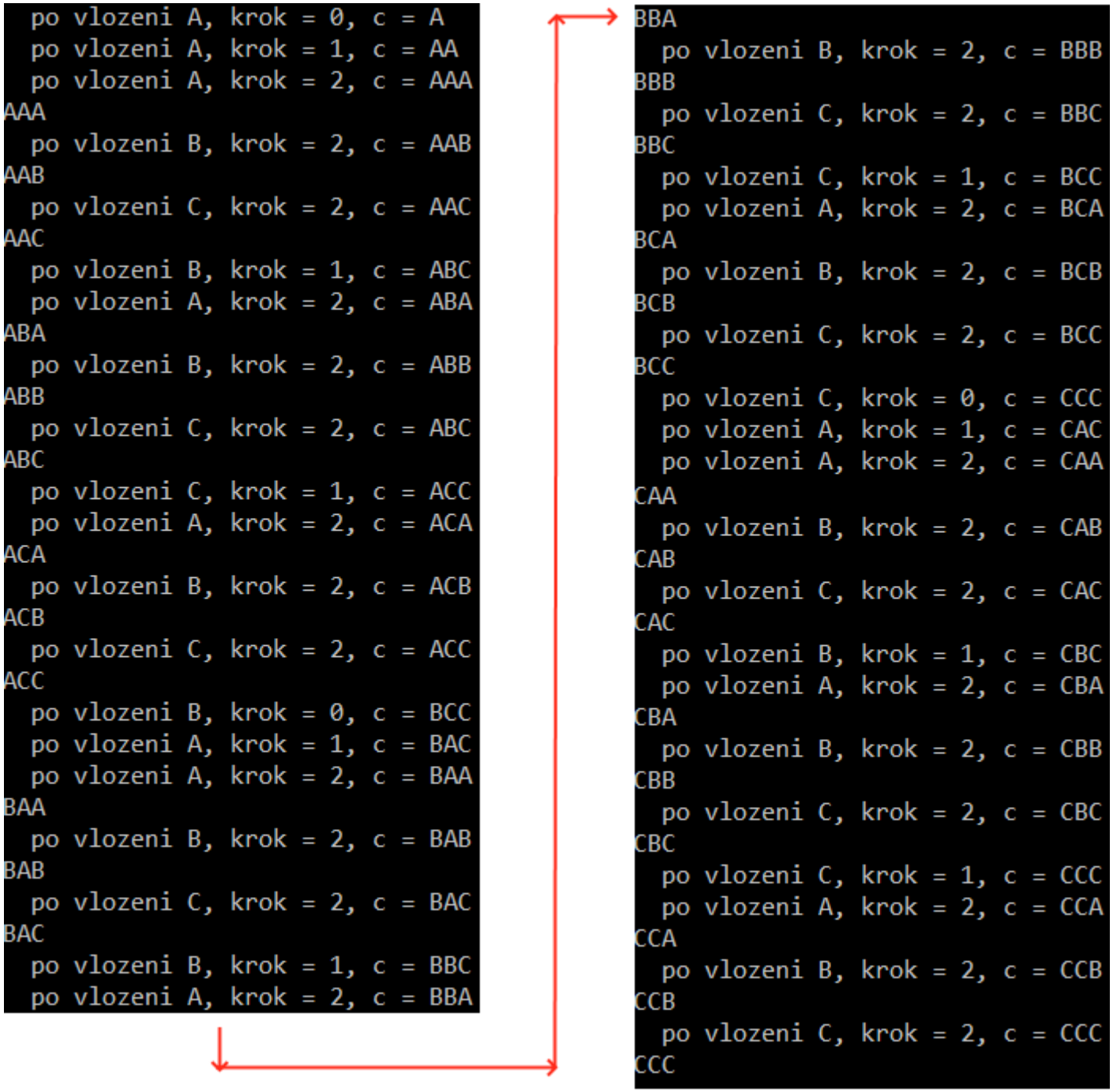
Kód programu:

```
#include <stdio.h>

void Moznosti(char * c, int krok)
{
	if (krok == 3)
	{
		printf("%s\n",c);
		return;
	}
	c[krok] = 'A';
	Moznosti(c,krok+1);
	c[krok] = 'B';
	Moznosti(c,krok+1);
	c[krok] = 'C';
	Moznosti(c,krok+1);
}

int main (void)
{
	char c[] = {' ',' ',' ','\0'};
	Moznosti(c,0);
	return 0;
}
```

Vysvětlení algoritmu:

Pro všechny instance je společná proměnná c. Ta má hodnotu adresy v operační paměti, kde je uloženo pole charů – 4 prvky, poslední obsahuje '\0', tedy symbol pro označení konce řetězce. Každá instance má jen jednu lokální proměnnou – krok. Hodnota této proměnné určuje adresu buňky – index v poli charů c.

```
void Moznosti(char * c, int krok)
{
        if (krok == 3) {printf("%s\n",c);return;}
        c[krok] = 'A';
        printf("  po vlozeni A, krok = %d, c = %s\n",krok,c);
        Moznosti(c,krok+1);
        c[krok] = 'B';
        printf("  po vlozeni B, krok = %d, c = %s\n",krok,c);
        Moznosti(c,krok+1);
        c[krok] = 'C';
        printf("  po vlozeni C, krok = %d, c = %s\n",krok,c);
        Moznosti(c,krok+1);
}
```


po vlozeni A, krok = 0, c = A
po vlozeni A, krok = 1, c = AA
po vlozeni A, krok = 2, c = AAA
AAA
po vlozeni B, krok = 2, c = AAB
AAB
po vlozeni C, krok = 2, c = AAC
AAC
po vlozeni B, krok = 1, c = ABC
po vlozeni A, krok = 2, c = ABA
ABA
po vlozeni B, krok = 2, c = ABB
ABB
po vlozeni C, krok = 2, c = ABC
ABC
po vlozeni C, krok = 1, c = ACC
po vlozeni A, krok = 2, c = ACA
ACA
po vlozeni B, krok = 2, c = ACB
ACB
po vlozeni C, krok = 2, c = ACC
ACC
po vlozeni B, krok = 0, c = BCC
po vlozeni A, krok = 1, c = BAC
po vlozeni A, krok = 2, c = BAA
BAA
po vlozeni B, krok = 2, c = BAB
BAB
po vlozeni C, krok = 2, c = BAC
BAC
po vlozeni B, krok = 1, c = BBC
po vlozeni A, krok = 2, c = BBA
BBA
po vlozeni B, krok = 2, c = BBB
BBB
po vlozeni C, krok = 2, c = BBC
BBC
po vlozeni C, krok = 1, c = BCC
po vlozeni A, krok = 2, c = BCA
BCA
po vlozeni B, krok = 2, c = BCB
BCB
po vlozeni C, krok = 2, c = BCC
BCC
po vlozeni C, krok = 0, c = CCC
po vlozeni A, krok = 1, c = CAC
po vlozeni A, krok = 2, c = CAA
CAA
po vlozeni B, krok = 2, c = CAB
CAB
po vlozeni C, krok = 2, c = CAC
CAC
po vlozeni B, krok = 1, c = CBC
po vlozeni A, krok = 2, c = CBA
CBA
po vlozeni B, krok = 2, c = CBB
CBB
po vlozeni C, krok = 2, c = CBC
CBC
po vlozeni C, krok = 1, c = CCC
po vlozeni A, krok = 2, c = CCA
CCA
po vlozeni B, krok = 2, c = CCB
CCB
po vlozeni C, krok = 2, c = CCC
CCC

Hlavní program spustí funkci Možnosti. Parametry jsou adresa pole c a 0 (číslo – hodnota nula). Vytvoří se první instance rekurze:

false
krok = 0
c[0]='A'
nové spuštění funkce Možnosti. Parametry jsou adresa pole c a krok +1 (číslo – hodnota 1). Vytvoří se druhá instance rekurze:

false
krok = 1
c[1]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:

false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:

true
výstup **AAA**
return – návrat o instanci zpět

false
krok = 2
c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:

true
výstup **AAB**
return – návrat o instanci zpět

false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:

true
výstup **AAC**
return – návrat o instanci zpět

funkce je u konce, proto návrat o instanci zpět

false
krok = 1
c[1]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:

false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:

true
výstup **ABA**
return – návrat o instanci zpět
false
krok = 2
c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **ABB**
return – návrat o instanci zpět
false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **ABC**
return – návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět, tedy na pozici 0 (krok = 0)
krok=0
c[0]='B'
nové spuštění funkce Možnosti. Parametry jsou adresa pole c a krok +1 (číslo – hodnota 1). Vytvoří se druhá instance rekurze:
false
krok = 1
c[1]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:
false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BAA**
return – návrat o instanci zpět
false
krok = 2
c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BAB**

return – návrat o instanci zpět
false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BAC**
return – návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
false
krok = 1
c[1]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:
false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BBA**
return – návrat o instanci zpět
false
krok = 2
c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BBB**
return – návrat o instanci zpět
false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BBC**
return – návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
false
krok = 1
c[1]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:

false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BCA**
return – návrat o instanci zpět
false
krok = 2
c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BCB**
return – návrat o instanci zpět
false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **BCC**
return – návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
krok=0
c[0]='C'
nové spuštění funkce Možnosti. Parametry jsou adresa pole c a krok +1 (číslo – hodnota 1). Vytvoří se druhá instance rekurze:
false
krok = 1
c[1]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:
false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CAA**
return – návrat o instanci zpět
false
krok = 2

c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CAB**
return – návrat o instanci zpět
false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CAC**
return – návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
false
krok = 1
c[1]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:
false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CBA**
return – návrat o instanci zpět
false
krok = 2
c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CBB**
return – návrat o instanci zpět
false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CBC**
return – návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět

false
krok = 1
c[1]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 2. Vytvoří se třetí instance rekurze:
false
krok = 2
c[2]='A'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CCA**
return – návrat o instanci zpět
false
krok = 2
c[2]='B'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CCB**
return – návrat o instanci zpět
false
krok = 2
c[2]='C'
nové spuštění funkce Možnosti(c, krok+1), druhý parametr je hodnota 3. Vytvoří se třetí instance rekurze:
true
výstup **CCC**
return – návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
funkce je u konce, proto návrat o instanci zpět
návrat do mainu
konec programu

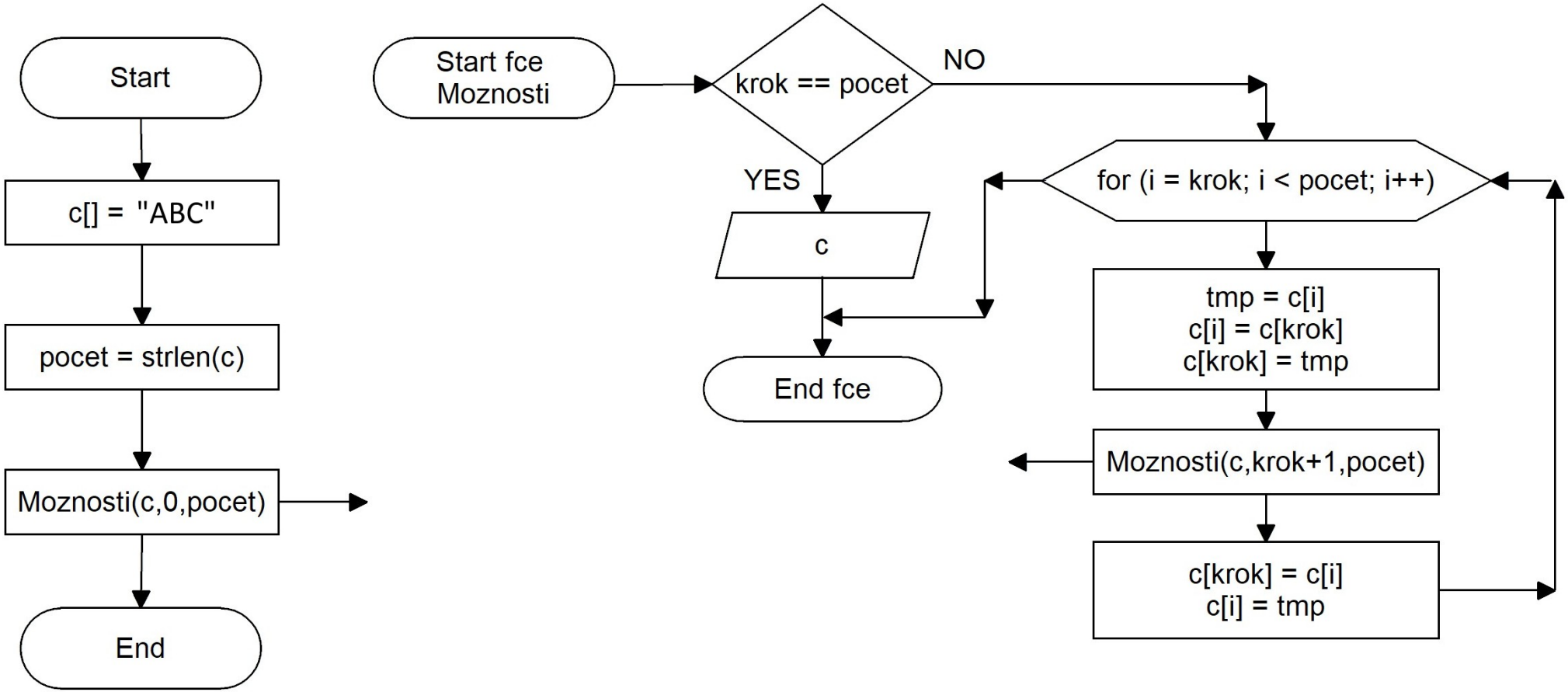
Příklad:

Navrhněte program, který vytvoří všechny možné permutace tří písmen A, B a C. Použijte rekurzi.

Ukázka výstupu:

Permutace = Variace bez opakování: $V_{\text{bez opak.}}(n,k)$, *(kde n=k)* = $\frac{n!}{(n-k)!} = n!$ v našem případě to je 3! / 0! = 6 možností

Vývojový diagram:



Kód programu:

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

void Moznosti(char * c, int krok, int pocet)
{
	if (krok == pocet)
	{
		printf("%s\n", c);
		return;
	}
	int i;
	char tmp;
	for (i = krok; i < pocet; i++)
	{
		tmp = c[i];
		c[i] = c[krok];
		c[krok] = tmp;
		Moznosti(c, krok + 1, pocet);
		c[krok] = c[i];
		c[i] = tmp;
	}
}

int main (void)
{
	char c[] = "ABC";
	int pocet = strlen(c);
	Moznosti(c, 0, pocet);
	return 0;
}
```

Vysvětlení algoritmu:

Pro všechny instance je společná proměnná c. Ta má hodnotu adresy v operační paměti, kde je uloženo pole charů – 4 prvky, poslední obsahuje '\0', tedy symbol pro označení konce řetězce. Každá instance má lokální proměnnou – krok, která se mění, druhá proměnná počet zůstává konstantní. Hodnota proměnné krok určuje adresu buňky – index v poli charů c.

Hodnoty proměnných i, krok a pole znaků c v průběhu chodu programu:

```
void Moznosti(char * c, int krok, int pocet)
{
	if (krok == pocet)
	{
		printf("   %s\n", c);
		return;
	}
	int i;
	char tmp;
	for (i = krok; i < pocet; i++)
	{
		tmp = c[i];
		c[i] = c[krok];
		c[krok] = tmp;
		printf("   pred rekurzivnim startem funkce: i= %d,krok = %d, %s\n",i,krok,c);
		Moznosti(c, krok + 1, pocet);
		c[krok] = c[i];
		c[i] = tmp;
		printf("   po navratu z funkce: i= %d,krok = %d, %s\n",i,krok,c);
	}
}
```

```
pred rekurzivnim startem funkce: i= 0,krok = 0, ABC
pred rekurzivnim startem funkce: i= 1,krok = 1, ABC
pred rekurzivnim startem funkce: i= 2,krok = 2, ABC
ABC
po navratu z funkce: i= 2,krok = 2, ABC
po navratu z funkce: i= 1,krok = 1, ABC
pred rekurzivnim startem funkce: i= 2,krok = 1, ACB
pred rekurzivnim startem funkce: i= 2,krok = 2, ACB
ACB
po navratu z funkce: i= 2,krok = 2, ACB
po navratu z funkce: i= 2,krok = 1, ABC
po navratu z funkce: i= 0,krok = 0, ABC
pred rekurzivnim startem funkce: i= 1,krok = 0, BAC
pred rekurzivnim startem funkce: i= 1,krok = 1, BAC
pred rekurzivnim startem funkce: i= 2,krok = 2, BAC
BAC
po navratu z funkce: i= 2,krok = 2, BAC
po navratu z funkce: i= 1,krok = 1, BAC
pred rekurzivnim startem funkce: i= 2,krok = 1, BCA
pred rekurzivnim startem funkce: i= 2,krok = 2, BCA
BCA
po navratu z funkce: i= 2,krok = 2, BCA
po navratu z funkce: i= 2,krok = 1, BAC
po navratu z funkce: i= 1,krok = 0, ABC
pred rekurzivnim startem funkce: i= 2,krok = 0, CBA
pred rekurzivnim startem funkce: i= 1,krok = 1, CBA
pred rekurzivnim startem funkce: i= 2,krok = 2, CBA
CBA
po navratu z funkce: i= 2,krok = 2, CBA
po navratu z funkce: i= 1,krok = 1, CBA
pred rekurzivnim startem funkce: i= 2,krok = 1, CAB
pred rekurzivnim startem funkce: i= 2,krok = 2, CAB
CAB
po navratu z funkce: i= 2,krok = 2, CAB
po navratu z funkce: i= 2,krok = 1, CBA
po navratu z funkce: i= 2,krok = 0, ABC
```