ВЪВЕДЕНИЕ В РЕКУРСИВНОТО ПРОГРАМИРАНЕ

В компютърната наука рекурсивна програма е програма, включваща функция, която в дефиницията си съдържа обръщение към самата себе си. Рекурсията може да е директна – в дефиницията на функцията се извиква непосредствено самата функция или индиректна – извиква се последователност от други функции, в някоя от които се съдържа извикване на рекурсивната функция.

```
int funcR(int p){
   ...
   a= b+funcR(c);
   ...
}
```

Директна рекурсия – в дефиницията на функцията се съдържа непосредствено обръщение към самата функция.

```
int funcR(int p){
   ...
   a= b+funcD(c);
   ...
}
int funcD(int p){
   ...
   if(a>2)d= b+funcR(m);
   ...
}
```

Индиректна рекурсия – функцията `funcR` извиква функцията `funcD`, която от своя страна вика отново функцията `funcR`.

Директна и индиректна рекурсия

За да може да се осъществи рекурсивно извикване, езикът трябва да бъде стеково ориентиран. Това означава, че при всяко извикване на функция за нея се заделя ново място в системния стек, в което се съхраняват локални променливи, параметри и всички необходими данни за изпълнението на функцията. Всички съвременни езици за програмиране спазват това условие.

Рекурсивен механизъм

Следващата програма включва извикване на рекурсивна функция , която въвежда символ от клавиатурата до срещане на точка или нов ред и отпечатва прочетените символи в обратен ред.

```
#include <stdio.h>
void prtBk();
int main(){
    printf("Please input a line: ");
    prtBk();
    return 0;
}
void prtBk(){
   int c;
   if((c= getchar())!=EOF){
      if(!((c=='.') || (c=='\n'))){
         prtBk();
      }
      printf("%c",c);
   }
   else{
      printf("Error");
   }
}
```

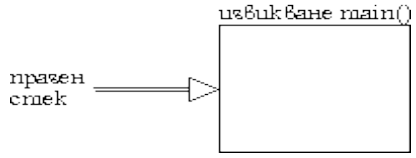
Типичен изход от изпълнението на програмата:

```
Please input a line: ex1. ex2. ex3
.1xe
```

Обърнете внимание, че рекурсивното извикване на функцията е включено в условен оператор. За да може рекурсивната програма да завърши успешно, задължително трябва да има просто нерекурсивно решение при някакво условие. Нещо повече – трябва да има гаранция, че до това нерекурсивно решение задължително ще се достигне в процеса на изпълнението на програмата, за да не се получи безкрайна поредица от рекурсивни извиквания.

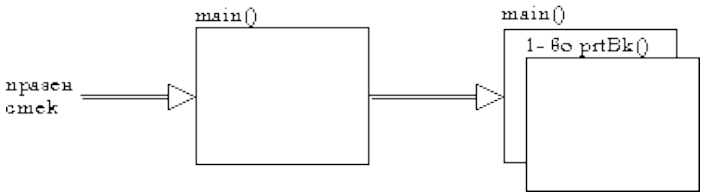
Изпълнението на програмата протича по следния начин:

- извиква се за изпълнение функцията `main()`. В стека се заделя памет, необходима за изпълнението й, след което се извежда подканващото съобщение.



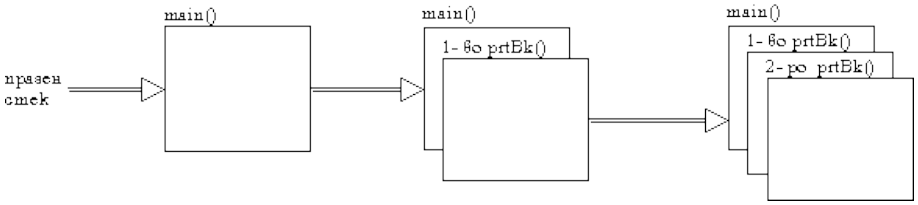
Фиг. Извикана функция `main()`

- Функцията `main()` извиква рекурсивната функция `prtBk()`.



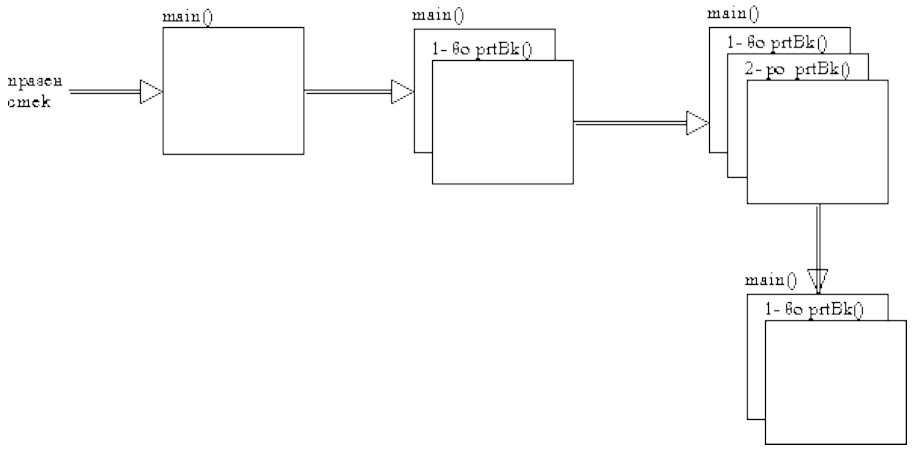
Фиг. Извикана функция `prtBk()` за първи път

- Функцията `prtBk()` прочита следващия символ; установява, че не е точка или символ за нов ред и прави рекурсивно извикване, като в стека се заделя необходимата за това памет.



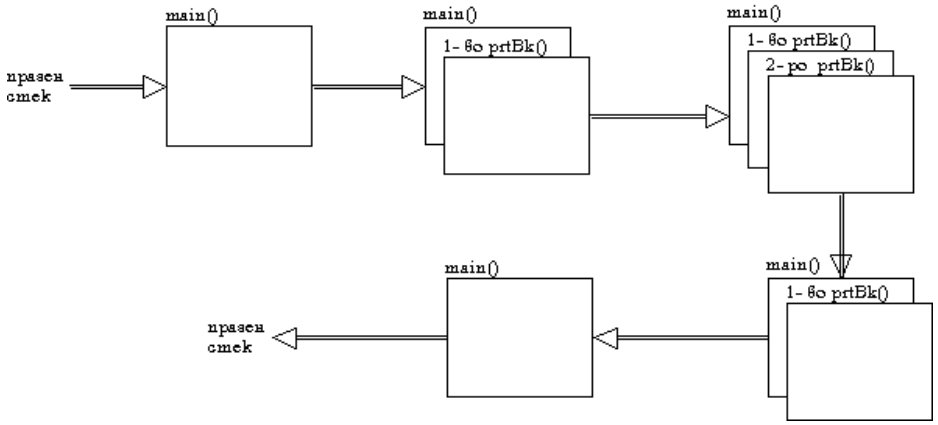
Фиг. Извикана функция `prtBk()` за втори път

- Нека този път въведеният символ да е точка. Функцията отпечатва символа и завършва изпълнението си, като освобождава паметта от стека и връща управлението в извикващата функция – извиканата за първи път функция `prtBk()`.



Фиг. Извиканата повторно функция `prtBk()` завършва

- Извиканата за първи път функцията `prtBk()` завършва изпълнението си, като отпечатва прочетения символ, освобождава паметта от стека и връща управлението на функцията `main()`. Тя също завършва и освобождава заеманата в стека памет



Фиг.Програмата завършва изпълнението си

Разбира се тази програма има и нерекурсивно итеративно решение. В този случай символите трябва просто итеративно да бъдат въведени в буфер от паметта, след което да бъдат отпечатани в обратен ред.

Условия за избор на рекурсивно решение

Рекурсивното програмиране е мощен и елегантен механизъм за решаване на определен кръг сложни задачи. То е стандартен механизъм при обработката на редица класове структурни данни – дървета, списъци и др. То обаче има и своите ограничения и недостатъци.

Броят на последователните рекурсивни извиквания е от изключително значение при изпълнението на програмата и се нарича дълбочина на рекурсията. Възможната дълбочина е ограничена от големината на стека и при писането на рекурсивни програми на нея се отделя особено внимание. Ако дълбочината надхвърли някакъв праг, е необходимо да се търси друго, нерекурсивно решение.

Многократното рекурсивно извикване на една функция е по-неефективно от гледна точка на необходимата памет. Губи се и време за многократното извикване и обмен на данни между функциите.

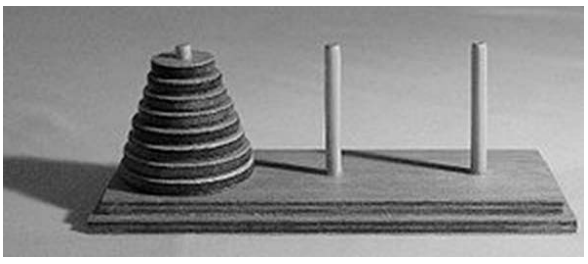
По принцип за всяка рекурсивно поставена задача може да се намери нерекурсивно решение. Важен критерии е колко по-сложно е това решение. Има широк кръг от задачи, при които в тялото на рекурсивната функция се прави едно единствено рекурсивно извикване. Такъв е разгледаният по-горе пример. За този клас задачи не представлява трудност намирането на итеративно решение. Обикновено итеративното решение е по-малко елегантно, изисква въвеждането на допълнителни структури от данни, но е по-ефективно от гледна на компютърните ресурси. В някой случай обаче намирането на нерекурсивно решение е нетривиална и сложна задача.

Има смисъл да бъде търсено рекурсивно решение при изпълнение на следните условия:

1. **Проблемът е достатъчно сложен и не дава възможност за намиране на относително просто итеративно решение**
2. **Може ясно да бъде определена размерност на проблема – целочислена функция, свързана със сложността на задачата. Тази размерност в последствие се свързва с дълбочината на рекурсията.**
3. **Проблемът може да бъде сведен към един или няколко проблема от същия вид, но с по-малка размерност.**
4. **Проблемът има просто, нерекурсивно решение при нулева или единична размерност.**

Типичен пример – ханойски кули

Задачата за ханойските кули представлява логическа игра, чийто автор е френският математик Едуард Лука. Играта представлява три стълба, на левия от които са нанизани дискове, подредени по размер, като най-големият е най-отдолу, а най-малкият – отгоре. Целта е дисковете да бъдат преместени на десния стълб. Може да се мести само по един диск на ход и той може да бъде поставен върху по-голям или на празен стълб. Всеки ход е съставен от взимането на горния диск от даден стълб и поставянето му върху дисковете на друг стълб.



Нека да анализираме задачата за преместване на **n** на брой диска от стълб **A** (левия) на стълб **C** (десния), като може да бъде използван стълб **B** (средният) от гледна точка на изложените по-горе условия.

- Сложност. Задачата разбира се има нерекурсивно решение, но то далеч не е тривиално. Следователно това условие се удовлетворява.
- Размерност. Сложността на задачата зависи от броя на дисковете, което очевидно свързва търсената размерност с броя на дисковете, които трябва да се преместят.
- Задачата, която е с размерност **n**, лесно може да се сведе до две задачи от същия тип, но с по-малка размерност: – 1) Да се преместят **n-1** диска от стълб **A** на стълб **B** (същата задача с размерност **n-1**). 2) Да се премести

последният останал диск от стълб **A** на стълб **C** (нерекурсивна задача). 3) Да се преместят поставените на стълб **B n-1** диска на стълб **C** (същата задача с размерност **n-1**).

- Задачата има просто нерекурсивно решение при нулева размерност – просто не трябва да се прави нищо.

Очевидно е, че задачата е особено подходяща за рекурсивно решение. При реализацията на задачата местенето на диск ще бъде извършено с извеждане на съответно съобщение на екрана. Рекурсивната функция ще има 4 параметъра – едно цяло число – брой на дисковете за местене, и 3 символни променливи – стълбовете, които ще се използват при местенето. Нека задачата да бъде пусната за 3 диска:

```
#include <stdio.h>
void hanoi(int n, char X, char Y, char Z){
     if(n>0){
        hanoi(n-1,X,Z,Y);
        printf("move a disk from %c to %c\n",X,Z);
        hanoi(n-1,Y,X,Z);
     }
}
int main()
{
  int n=3;
  hanoi(n,'A','B','C');
  printf("the %d disks are sucessfuly moved\n",n);
  return 0;
}
```

В резултат на пускането на програмата:

```
move a disk from A to C
move a disk from A to B
move a disk from C to B
move a disk from A to C
move a disk from B to A
move a disk from B to C
move a disk from A to C
the 3 disks are sucessfuly moved
```

Очевидно дълбочината на рекурсията е равна на броя на дисковете. С увеличаването им броят на необходимите премествания нараства много бързо. Рекурсивният подход позволява много просто изчисляването на необходимия брой премествания на дискове за решението на задачата. Тъй като за n диска се извършва два пъти преместване на n-1 диска и още един диск, а за един диск е необходимо само едно преместване, то общият брой необходими премествания се изчислява на 2^n-1.

Съществува легенда, за която не се знае дали е инспирирана от Едуард Лука или той е повлиян от нея. В индийски храм с името Брахма съществува стая, в която монаси решават задачата, като местят 64 златни диска. Когато те преместят последния диск, ще настъпи краят на света. Легендата е известна под името кулите на Брахма.

Нека дисковете да бъдат местени със скорост един диск в секунда. Ако дисковете са 3, са необходими 7 премествания или 7 секунди. Ако те са 8, са необходими 255 премествания или 4 минути и 15 секунди. За кулите на Брахма са необходими 2^{64}-1 хода, което е приблизително $10^{19,3}$ хода или около 585 милиарда години, което в момента изглежда напълно приемливо за край на света.

Задачи за самостоятелна работа

Задача 1: Функцията факториал от положително цяло число **n!** се задава със следната дефиниция: **1! = 1; n!=(n-1)!*n**. Напишете рекурсивна и итеративна програма за пресмятане на функцията.

Задача 2: Напишете рекурсивна и итеративна функция с два параметъра, която връща като резултат стойността на първия аргумент (реално число), повдигната на степен втория си аргумент (цяло неотрицателно число). Вземете необходимите мерки при подаване на неподходящи стойности на аргументите.

Задача 3: Напишете рекурсивна функция с два параметъра. Първият е указател към масив от цели числа. Функцията връща сумата от първите n елемента на масива. Стойността на n е вторият параметър на функцията (цяло неотрицателно число).