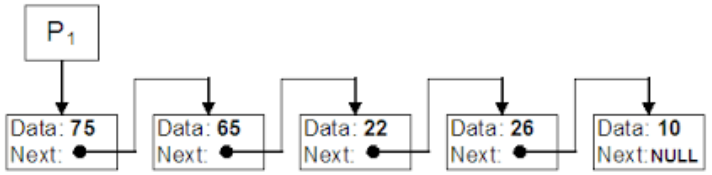
TIQXMMI
Toshkent Irrigatsiya va Qishloq Xo'jaligini Mexanizatsiyalash Muhandislari Instituti

Ma'lumotlar tuzilmasi

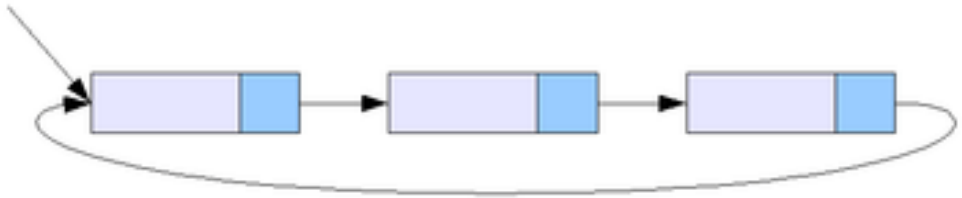
Data structures



3-ma'ruza: Rekursiya va rekusiv triada tushunchasi

Lecture #6. Concepts of recursion and recursive triad

PhD. Sh.A.Toirov



Ma’ruza maqsadi va rejasi
Purpose and plan of the lecture

- **Rekursiya haqida tushuncha**
- **Rekursiv triada**
- **Rekursiv algoritmlar**
- **Rekursiyaga doir misollar**

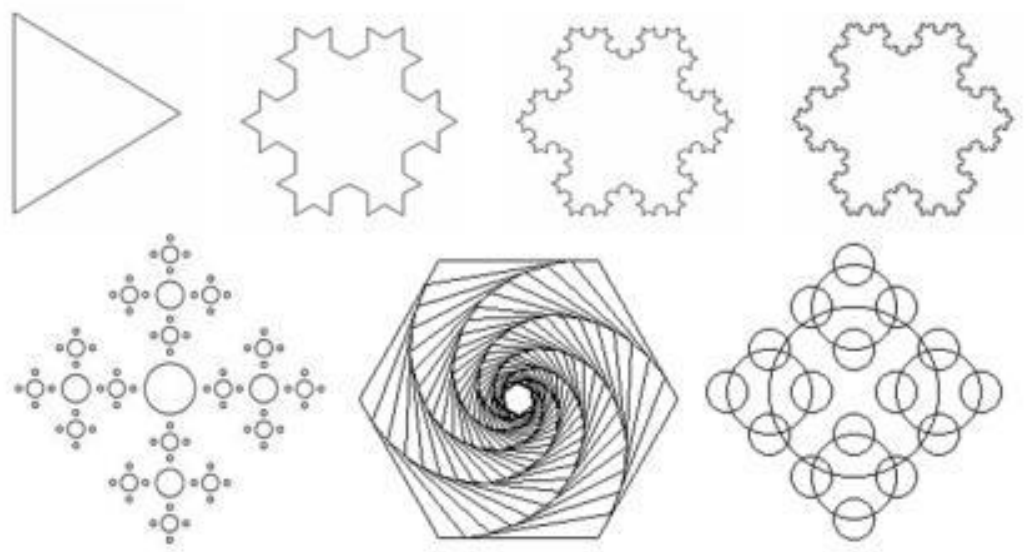
Ma’ruza maqsadi: rekursiya tushunchasi, dasturlashda rekursiv funktsiyalar, masalalarni yechishda rekursiv triadani qurish usullari, C++ va Python dasturlash tilida masalalarni yechishda rekursiv usullarni qo’llashni o’rganish.

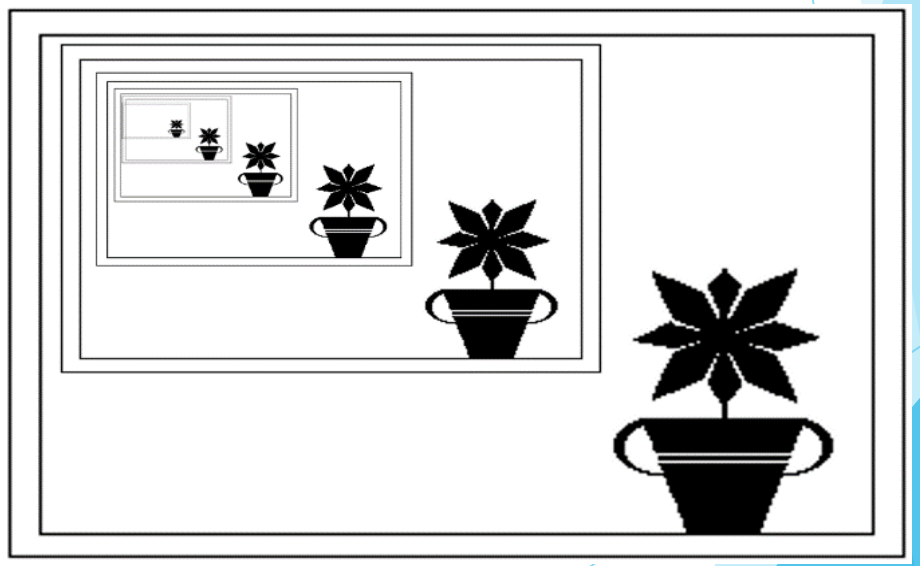
Rekursiya haqida tushuncha

- Bizni o’rab turgan dunyoda o’z-o’ziga o’xshash ob’ektlardan tashkil topgan buyumlarni uchratamiz. Ya’ni katta ob’ektining qismlari aynan ushbu ob’ektning o’zidan iborat bo’ladi. Bunday ob’ektlar reskursiv deyiladi
- Masalan, daraxtning bargi aynan daraxtning shoxlanishiga o’xshash tasvirni taqdim etadi.
- Tashqaridan qaragandan rekursiya yetarli darajada juda oddiy va maxsus bilimlarni talab qilmaydigandek ko’rinadi.

Rekursiv ob’ektlarga misollar:

- Rekursiv ob’ektlarga misol sifatida quyidagi grafik tasvirlarni olish mumkin. Bunda tasvirlar o’z-o’zini takrorlovchi, bitta ob’ekt sifatida qaraladi.





Rekursiya haqida tushuncha

- ***Rekursiya*** – o’z-o’zi orqali aniqlanuvchi ob’ekt hisoblanadi. Matematikada rekursiya yordamida bir qancha cheksiz to’plamlarni aniqlash mumkin, masalan, natural sonlar to’plami.
- Haqiqatan ham, natural sonlarni quyidagicha ifodalash mumkin:
  - **Natural son:**
  - **1 - natural son.**
  - **Natural sondan keyin keluvchi son - natural son.**
- Xuddi shunday faktorial tushunchasi **n!=1·2·3·…·(n-1)·n** ni ham rekursiya yordamida tushuntirish mumkin:
  - **Faktorial:**
  - **0!=1** (shartli qabul qilingan).
  - **n>0 uchun n!=n*(n-1)!**

Rekursiv triada

Masalalarni rekursiv usulda yechish uchun ***rekursiv triada*** deb ataluvchi quyidagi bosqichlar ishlab chiqiladi::

- ***Parametrlarni aniqlash*** – masalaning shartlarini tavsiflash uchun va yechimni olishda qo'llaniladigan parametrlarni tanlash;
- **Rekursiya tayanchi (bazisi)** - yechimni olish vaqtida funktsiyaning o'ziga murojaatni talab etmaydigan arzimas holatlarni aniqlash;
- ***dekompozitsiya*** – umumiy masalani parametrlarni o'zgartirish orqali ancha sodda qism masalalarga ajratgan holda ifodalash.

1-Misol. Manfiy bo’lmagan butun n sonining faktorialini aniqlang.

Qo’yilgan masala uchun rekursiv triadani ishlab chiqamiz:

- *Parametrlarni aniqlash*: n – manfiy bo’lmagan butun son.
- Rekursiya tayanchi: n =0 uchun *faktorial* 1 ga teng.
- *Dekompozitsiya qilish (qismlarga ajratish)*: n!=(n-1)!*n.

```
long factor (int n) {
  if (n<0) return 0;          // manfiy sonlar uchun
  if (n==0) return 1;         // tayanch holat: n=0
  return factor(n-1)*n;       // umumiy holat (dekompozitsiya)
}
```

Rekursiv algoritmlar

- Agar protsedura yoki funktsiyaning o’zida o’ziga murojaat bo’lsa ***rekursiv*** deb ataladi.
  - Misol uchun, faktorialni hisoblash funktsiyasini quyidagicha yozish mumkin:

```
int Factorial (int n)
{
if (n<=0) return 1; //1 ni qaytarish
   else
return n*Factorial(n-1); //rekursiv chaqirish
 }
```

Rekursiyaga doir Pythonda misollar

```
def faktorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * faktorial(n - 1)
# Test qilish
print(faktorial(5))  # Natija: 120
```

- Agar **n > 0** bo’lsa, **Factorial** funktsiyasi o’zini o’zi chaqiradi. Bu masalani yechish uchun rekursiv protsedurani (funktsiyani emas) qo’llash mumkin.

Rekursiv algoritmlar

- Bunda ssilka bilan berilgan parametr orqali (protsedurani e’lon qilishda uning nomi oldiga ssilka & belgisi qo’yilgan) qo’llaniladi. Protsedurani rekursiv chaqirishda bu qiymat o’zgaradi.

```
void Factorial (int n, int &fact)
{
 if (n==0) fact=1; //rekursiya yakunlanadi
   else {
Factorial(n-1, fact); //(n-1)! ni rekursiv hisoblash
  fact*=n; //n!=n*(n-1)!
   }
}
```

- Funktsiyadan farqli ravishda protsedura ssilka orqali berilgan parametr yordamida bir nechta qiymatlarni qaytarish mumkin.

Rekursiv algoritmlar

- Yangi rekursiv chaqiruvni amalga oshirishda kompyuter quyidagilarni bajaradi:
  - ushbu bosqichdagi hisoblashlar holatini eslab qoladi.
  - stek (asosan xotira sohasi)da lokal o’zgaruvchilarning yangi to’plamini hosil qiladi (chunki, joriy chaqiruvda o’zgaruvchini yo’qotib qo’ymaslik kerak).
  - har bir chaqiruvda yangi xotira sarflanadi va protsedura chaqiruvi hamda protseduraga qaytish uchun vaqt yo’qotiladi.

Rekursiya chuqurligi

- Shuning uchun ham rekursiyani qo’llashda quyidagiga alohida e’tibor berish kerak:
- **Rekursiya chuqurligi** (tarkibidagi chaqiruvlar soni) – yetarli darajada kichik bo’lishi shart.
- Katta chuqurlikdagi rekursiyadan foydalanishda dasturda uzoq vaqt ishlash va stekning to’lib toshib ketishi (stekli xotiraning yetishmovchiligi) holati yuz berishi mumkin. Shuning uchun ham, agar masalani rekursiyasiz ham yechish mumkin bo’lsa, u holda rekursiyani qo’llash tavsiya etilmaydi.

Rekursiya chuqurligi

- Masalan, faktorialni hisoblash uchun oddiygina **for** tsiklidan foydalanish mumkin (bunday tsikl yordamida olinadigan yechim iterativ (qadamma-qadam) deb ataladi):
- **int Factorial (int n)**
- **{**
- **int i, fact=1;**
- **for (i=2; i<=n; i++)**
- **fact*=i;**
- **return fact;**
- **}**
- Bu dastur rekursiv dasturga nisbatan tezroq ishlaydi. Ixtiyoriy rekursiv dasturlarni juda murakkab bo’lsa ham rekursiyasiz yozish mumkinligi isbotlangan.

Rekursiyaga doir misollar

- **Masala.** Fibonachchi sonini hisoblash uchun **Fib** funktsiyasini tuzing. Bunda quyidagilar berilgan:
  - **f0=0, f1=1.**
  - **i>1** uchun **fi=fi-1+fi-2.**
- Rekursiyani qo’llanilgan funktsiya quyidagicha bo’ladi:

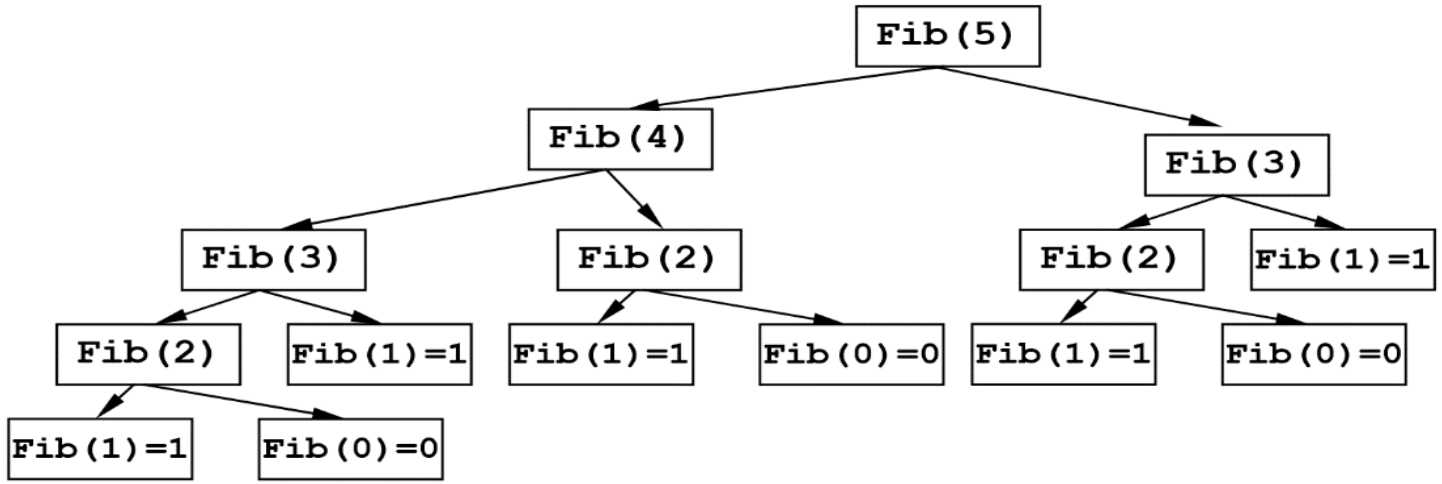
```
int Fib (int n)
{
 if (n==0) return 0;
 if (n==1) return 1;
 return Fib(n-1)+Fib(n-2);
}
```

Rekursiyaga doir Pythonda misollar

```
def fibonachchi(n):
    if n <= 1:
        return n
    else:
        return fibonachchi(n - 1) + fibonachchi(n - 2)
# Test qilish
for i in range(6):
    print(fibonachchi(i))  # Natija: 0, 1, 1, 2, 3, 5
```

Rekursiyaga doir misollar

- E’tibor berish kerakki, har bir **n>1** uchun rekursiv chaqiruvda 2 ta funktsiyani chaqirish kerak, ya’ni kichik **n** uchun Fibanachchi sonini ko’p ifodalar bir necha marta hisoblanadi. Shuning uchun ham amaliyotda asosan katta **n** lar uchun bu algoritm qulay emas. **Fib(5)** ni hisoblash sxemasi quyidagi daraxat shaklida tasvirlangan:



Rekursiyaga doir misollar

- Navbatdagi Fibanachchi sonini topish undan oldingi ikkita **f1** va **f2** o’zgaruvchilarda saqlanayotgan sonlarga bog’liq ekanligi ma’lum. Oldin **f1=1** va **f2=0** larni qabul qilib, keyingi fibanachchi sonini hisoblaymiz va uni **x** o’zgaruvchiga saqlab qo’yyamiz. Endi **f2** qiymat bizga kerak emas, shuning uchun **f1** ni **f2** ga va **x** ni **f1** ga nusxalaymiz.
- **int Fib2(int n)**
- **{**
- **int i, f1=1, f2=0, x;**
- **for (i=2; i<=n; i++) {**
- **x=f1+f2;** //keyingi son
- **f2=f1; f1=x;** //qiymatlarni siljitish
- **}**
- **return x;**
- **}**

Xulosa

- Bunday protsedura rekursiv funktsiyaga nisbatan katta **n (>20)** uchun bir necha yuz ming marta (!!!) tez ishlaydi. Bundan kelib chiqadiki, qaerda rekursiyasiz masala yechimini olish mumkin bo’lsa, rekursiyani qo’llash tavsiya etilmaydi.

Adabiyotlar

- Алфред В. Ахо., Джон Э. Хопкрофт, Джефри Д. Ульман. Структура данных и алгоритмы. //Учеб.пос., М.: Изд.дом: "Вильямс", 2000, — 384 с.
- Adam Drozdek. Data structures and algorithms in C++. Fourth edition.
  Cengage Learning, 2013.
- Бакнелл Джулиан М. Фундаментальные алгоритмы и структуры данных в Delphi//СПб: ООО «ДиаСофтЮП», 2003. 560с.
- Narzullaev U.X., Qarshiev A.B., Boynazarov I.M. Ma’lumotlar tuzilmasi va algoritmlar. //O’quv qo’llanma. Toshkent: Tafakkur nashriyoti, 2013 y. – 192 b.
- Лойко В.И. Структуры и алгоритмы обработки данных. Учебное пособие для вузов. - Краснодар: КубГАУ. 2000. - 261 с., ил.

Mustaqil ishlash uchun topshiriqlar:

- Bog’lamli ro’yxatlar ustida bajariladigan amallarga doir misollar yechish
- *Izoh: dars mashg’ulotida berilgan bilimlarga qo’shimcha ma’lumotlarni to’plash-konspekt qilish*