Рекурсия

гл. ас. д-р. Нора Ангелова

Рекурсия

Ако в дефиницията на някаква функция се използва самата функция, дефиницията на функцията се нарича **рекурсивна**.

$$n! = n * (n-1) * (n-2) * \cdots * 1$$

$$n! = \begin{cases} 1, & n = 0 \\ n * (n-1)!, & n > 0 \end{cases}$$

Условието при $n = 0$ не съдържа обръщение към функцията факториел и се нарича **гранично**.

Рекурсия

Пример:

$$x^n = \begin{cases} 1, & n = 0 \\ x * x^{n-1}, & n > 0 \\ \frac{1}{x^{-n}}, & n < 0 \end{cases}$$

Гранично условието при $n = 0$.

Рекурсия

Пример:

$$1\ 1\ 2\ 3\ 5\ 8\ \ldots$$

$$fib(n) = \begin{cases} 1, & n = 1 \\ 1, & n = 2 \\ fib(n-1) + fib(n-2), & n > 1 \end{cases}$$

Гранично условието при $n = 1,\ n = 2.$

Рекурсия

Рекурсия в C++

В C++ е разрешено функция да вика в тялото си сама себе си.

- Функция, която се обръща **пряко или косвено** към себе си, се нарича **рекурсивна**.

- **Пряко рекурсивна функция -** в тялото на функция се извършва извикване на същата функция.

- **Непряко (косвено) рекурсивна -** функция A вика функция B, B вика C, а C отново вика A.

Рекурсия

Дъно на рекурсията (гранично условие) -
Един или повече случаи, които не изискват рекурсивно извикване за намиране на решение.

Пример:
В редицата на Фибоначи – n = 1 && n = 2;

** Възможно е броенето да започва от n = 0.*

Рекурсия

```
void func () {
  cout << "1" << endl;
}

int main() {
  cout << "0" << endl;
  func();
  cout << "2" << endl;

  return 0;
}
```

Рекурсия

```
void func () {
  cout << "1" << endl;
  func();                    // Зацикляне
}

int main() {

  cout << "0" << endl;
  func();
  cout << "2" << endl;


  system("pause");
  return 0;
}
```

Рекурсия

```
void func (int n) {
  if (n>10) {
    cout << "11!!!" << endl;
    return;
  }
  cout << n << endl;
  func(++n);
}

int main() {
  cout << "0" << endl;
  func(1);
  cout << "2" << endl;

  return 0;
}
```

```
0
1
2
3
4
5
6
7
8
9
10
11!!!
2
```

Рекурсия

Да се напише рекурсивна функция, която пресмята n-ти член на редицата на Фибоначи.

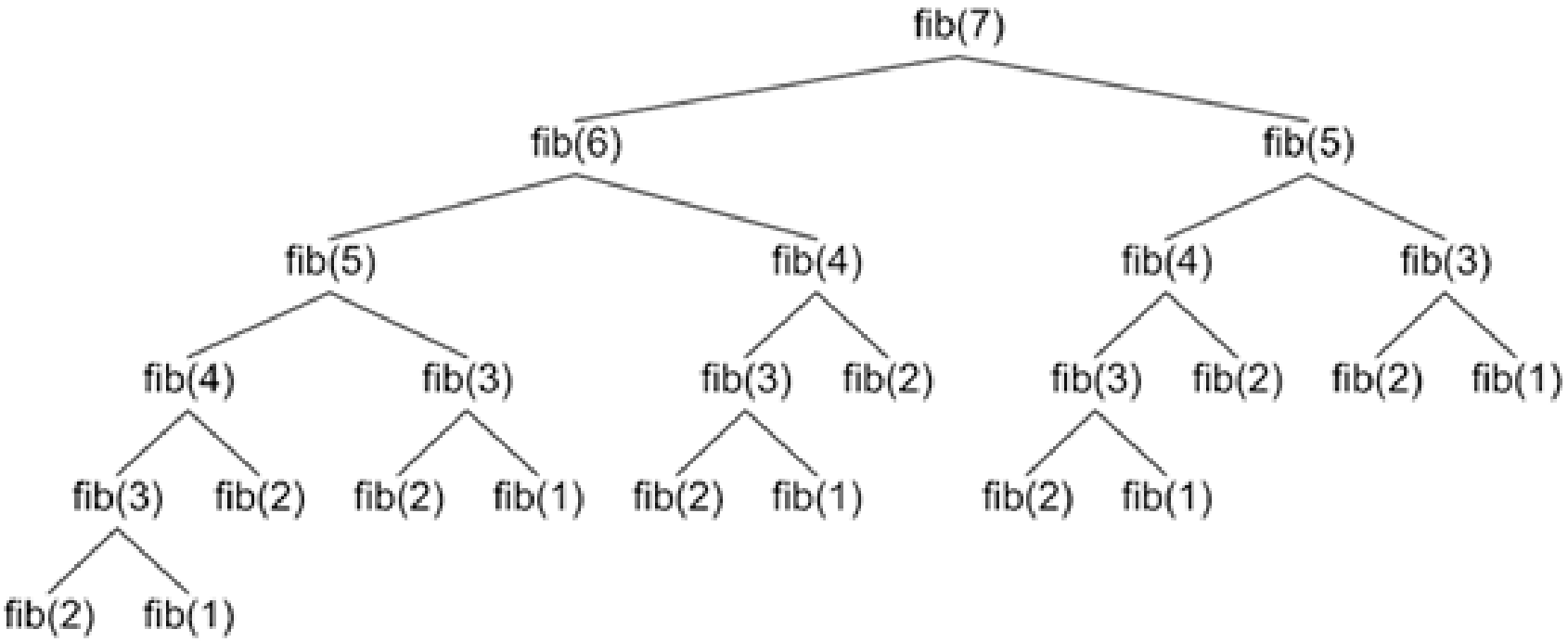
```
int fib (int n) {
  if (n >= 1 && n <= 2) {
    return 1;
  }
  return fib(n-1) + fib(n-2);
}

int main() {
  cout << fib(5);
  return 0;
}
```

Рекурсия

Намиране на n-ти член на редицата на Фибоначи:

- Броят на стъпките за рекурсивно изчисление на fib(100) ~ 1.6 на степен 100.
- Броят на стъпките за линейно решение - 100.



Рекурсия

Намиране на n-ти член на редицата на Фибоначи

- Оптимизация – записване на пресметнатите числа в масив. Извършва се рекурсивно извикване за членовете на редицата, които не са били пресметнати до момента.

Рекурсия

Да се напише рекурсивна функция, която намира минималния елемент на редица от реални числа.

```
double min(double* arr, int n) {
  double subArrayMin;
  if (n == 1) {
    return arr[0];
  }

  subArrayMin = min(arr, n-1);

  if (subArrayMin < arr[n-1]) {
    return subArrayMin;
  }
  return arr[n-1];
}
```

Рекурсия

Да се напише рекурсивна функция, която решава задачата за ханойските кули.

- Ако броят на дисковете е 0 – не се прави нищо.
- Да се преместят n-1 диска от стълб A на стълб B (същата задача с размерност n-1).
  Да се премести последният останал диск от стълб A на стълб C (нерекурсивна задача).
- Да се преместят поставените на стълб B n-1 диска на стълб C (същата задача с размерност n-1).

Рекурсия

Да се напише рекурсивна функция, която решава задачата за ханойските кули. Програмата да се реализира с извеждане на преместванията.

```
void hanoi(int n, char X, char Y, char Z) {
  if (n <= 0) {
    return;
  }

  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 successfully moved\n", n);

  return 0;
}
```

Следва продължение...