Программирование на языке C++

Зариковская Наталья Вячеславовна

Лекция 9

Рекурсия в языке C++

- В программировании рекурсия — вызов функции (или процедуры) из неё же самой, непосредственно (простая рекурсия) или через другие функции (сложная или косвенная рекурсия). Количество вложенных вызовов функции или процедуры называется глубиной рекурсии. Рекурсивная программа позволяет описать повторяющееся или даже потенциально бесконечное вычисление, причём без явных повторений частей программы и использования циклов.

- Язык C++, как и большинство языков программирования высокого уровня, поддерживает рекурсивные вызовы. В C++ любая функция, кроме main() может быть рекурсивной.

Общий вид

- Прямая рекурсия

```
int function() {
    …
    function()
    …
}
```

- Косвенная рекурсия

```
int firstFunction() {
    …
    secondFunction()
    …
}

int secondFunction() {
    …
    firstFunction()
    …
}
```

Факториал

Факториал числа n (n!) – произведение всех натуральных чисел от 1 до n включительно (принято, что 0! = 1).

```
int factorial(int n) {
 if (n < 2)
      return 1;
 return n*factorial(n - 1);
}
```

Число Фибоначчи

Числа Фибоначчи – элементы последовательности, в которой первые два числа равны 0 и 1, а каждое последующее число равно сумме двух предыдущих чисел. (0, 1, 1, 2, 3, 5, 8…)

```
int fibonacci(int n) {
 if (n < 1)
      return 0;
 if (n == 1)
      return 1;
 return fibonacci(n - 1)*fibonacci(n – 2);
}
```

QuickSort

- Другим интересным примером использования рекурсивных функций может служить алгоритм быстрой сортировки (quicksort). Этот алгоритм получил широко распространение во многих прикладных программах. Однако эффективность его использования существенным образом зависит от числа элементов сортируемого массива, а также характера распределения этих данных.
- 1. Выбирается какой-либо элемент (x) (обычно находящийся в середине последовательности). Будем просматривать массив слева до тех пор, пока не обнаружим элемент a_i>x, затем будем просматривать массив справа, пока не встретится a_j<x. Теперь поменяем местами эти два элемента и продолжим наш процесс просмотра и обмена, пока оба просмотра не встретятся где-то в середине массива (значение i и j равны). В результате массив окажется, разбит на две половины с ключами <=x и >=x, а сам элемент “x” будет находиться в необходимом месте
- 2. Итерационно повторяем процесс 1 для каждой из полученных половин (в первый раз получим четыре поддиапазона исходной последовательности) до тех пор, пока значение индексов для каждого вложенного диапазона будут или равны, или изменят отношение. В результате мы получим упорядоченную последовательность
-

Рекурсия в программе – это плохо

По крайней мере в большинстве случаев.

Достоинства рекурсии:

- высокая читаемость кода;
- малое количество кода.

Недостатки рекурсии:

- низкая скорость работы;
- высокое использование памяти.

Итеративный факториал

```
int factorial(int n) {
 if (n < 2)
        return 1;
 int result = 1;
        while(n > 1) {
                result *= n;
                n--;
        }
        return result;
}
```

Рекурсия в языке C++

- Примером задачи, где оправдано использование рекурсии, может служить решение задачи о Ханойских башнях.
- Эта задача формулируется следующим образом: имеется три стержня. На одном из них находятся диски, разных размеров, сужающиеся к верху (пирамида). Требуется перенести эти диски на другой стержень, используя только один, вспомогательный стержень. При этом за один раз можно перенести только один диск, а больший по размеру диск не допустимо ставить на диск меньшего размера
- Доказательство решения этой задачи основано на использовании индукционного метода, из которого непосредственно следует алгоритм, основанный на использовании рекурсивных функций. Так, для случая одного диска решение, очевидно, он просто перекладывается с исходного стержня на заданный. Для случая из двух дисков сначала верхний диск перекладывается на вспомогательный стержень, а затем нижний диск с исходного стержня и верхний со вспомогательного стержня перекладываются на заданный. Аналогично, для случая N дисков, сначала N-1 дисков при соблюдении правил перестановки перемещаются на вспомогательный стержень, а затем на заданный стержень переносится нижний диск (он нашёл своё место). Далее, учитывая, что последний диск, перемещённый на заданный стержень, никак не мешает, поскольку он больше всех остальных, задача решается аналогично. Исключение состоит в том, что N-1 диск, из оставшихся, перемещаются на вспомогательный, который ранее был исходным, используя в качестве промежуточного заданный, а последний из оставшихся снова перемещается на заданный стержень. Таким образом, после двух циклов перестановок, мы приходим к исходной ситуации использования стержней, но количество дисков, которых необходимо переставить меньше на два диска и т.д.. Процесс рекурсии останавливается, когда остаётся только один диск, который перемещается на заданный. Доказано, что для n дисков минимальное число необходимых перемещений равно 2^n-1.

Рекурсия в языке C++

- Рекурсивная функция должна всегда определять условие окончания, иначе рекурсия станет бесконечной.
- #include<iostream.h>
- int k;
- void Hanoy1 (char a,char b,char c,int n);
- void main()
- {cout<<”введите колличество дисков”<< “\n”;
- cin>>k;
- Hanoy1 (‘A’,‘C’,‘B’,k);//перенесено k дисков с A на C промежуточный B
- cin>>k;
- }
- void Hanoy1 (char a,char b,char c,int n)
- {int v;
- v=n;
- if (n==1)
- cout<<”переместить “<< v<<” с “<< a<<” на “<<b<<”\n”;
- else
- {Hanoy1(a,c,b,n-1);
- //перенесено n-1 дисков с A на C промежуточный B
- cout<<”переместить “<<v<<” с “<< a<<” на “<< b<<”\n”;
- Hanoy1 (c, b, a, n - 1);
- //перенесено n - 1 дисков с C на B промежуточный A
- } }

Рекурсия в языке C++

```
#include<iostream.h>
#include<stdio.h>
#define b=16
int n=6;
int a[6];
int i,j,l,r=6;
void swap(int*,int*);
void quicksort(int,int);
void part(int,int,int&,int&);
void main()
{for(int k=1;k<=n;k++){
cout<<"введите a["<<k<<"] исходного массива \n";
cin>>a[k];
                    }
quicksort(1,n);
for(k=1;k<=n;k++)
{printf("a[ %d ]= %d \n",k,a[k]);}
}
cin>>k;
}
void swap(int* p,int* q)
{int prom;
prom=*p;*p=*q;*q=prom;
}
```

```
void quicksort(int l,int r)
{int i,j;
i=l;j=r;
{part(l,r, i, j);
if(i<r)quicksort(i,r);// переход к сортировке левой
 части
if(j>l)quicksort(l,j); //переход к сортировке правой
 части
  }
 }
   void part(int l,int r,int &i,int &j)
 {int x ;
i=l;j=r;x=(l+r)/2;
do{
        while(a[i]<a[x])i++;//просмотр ->: найдём a[j]>
           a[x]
while(a[j]>a[x])j--;
//просмотр<->: найдём a[j]< a[x]: меняем местами
if(i<=j)
{ swap(&a[i],&a[j]);   //  обмен этих элементов
i++;j--;}
 } while(i<j);
  }
```

Рекурсия в языке C++

- Технология использования рекурсивных функций весьма проста. Однако при использовании рекурсивных функций необходимо особое внимание уделять анализу целесообразности их применения. Например, рассмотрим пример решения задачи расчёта долга клиента банка, который взял некоторую сумму под ежемесячный процент от текущей суммы долга. Эта задача может быть решена как с использованием рекурсивных функций, так и с помощью итерационных алгоритмов. При использовании рекурсивных функций для решения этой задачи необходимо: определить условие окончания - начальная сумма кредита (if (n==0) kuz=b;); определить в выражение расчёта долга относительно текущего (kuz=kuzy(n-1)*(1+kk);). После чего алгоритм и программа решения задачи, приведенная ниже, становится весьма прозрачной
-

Рекурсия в языке C++

```
#include<iostream.h>
float kk,b;
float kuzy (int);
void main()
 { int  n ;
cout<<" введите кол месяцев"<< "\n";
cin>>n;
cout<<"введите начальную сумму долга"<< "\n";
cin>>b;
cout<<"введите ежемесячный процент - вещественное число" << "\n";
cin>>kk;
cout<<"долг="<<kuzy(n)<<'\n';
cin>>kk;
   }
float kuzy(int n )
{float kuz;
if (n<0)
cout<<"ошибка в задании количества месяцев"<< "\n";
else
if (n==0)  kuz=b;
else  kuz=kuzy(n-1)*(1+kk);
return kuz;
        }
```

Рекурсия в языке C++

- Эта же программа при использовании рекурсивного алгоритма не менее красива, но значительно эффективнее.

```
#include<iostream.h>
float kk,b;
void main()
        {  int  n ;
     float kuz;
cout<<" введите кол месяцев"<< '\n';
cin>>n;
cout<<"введите начальную сумму долга"<< '\n';
cin>>b;
cout<<"введите ежемесячный процент"<< "\n";
cin>>kk;
kuz=b;
while (n>0)
{kuz=kuz*(1+kk);
n--;
                }
cout<<"долг="<<kuz<<'\n';
          cin>>kk;   }
```

Рекурсия в языке C++

- Незаменимы рекурсивные процедуры при решении интеллектуальных задач, где используются алгоритмы с возвратом. Целесообразно проанализировать приведенный ниже листинг программы решения задачи обхода шахматным конём шахматной доски.

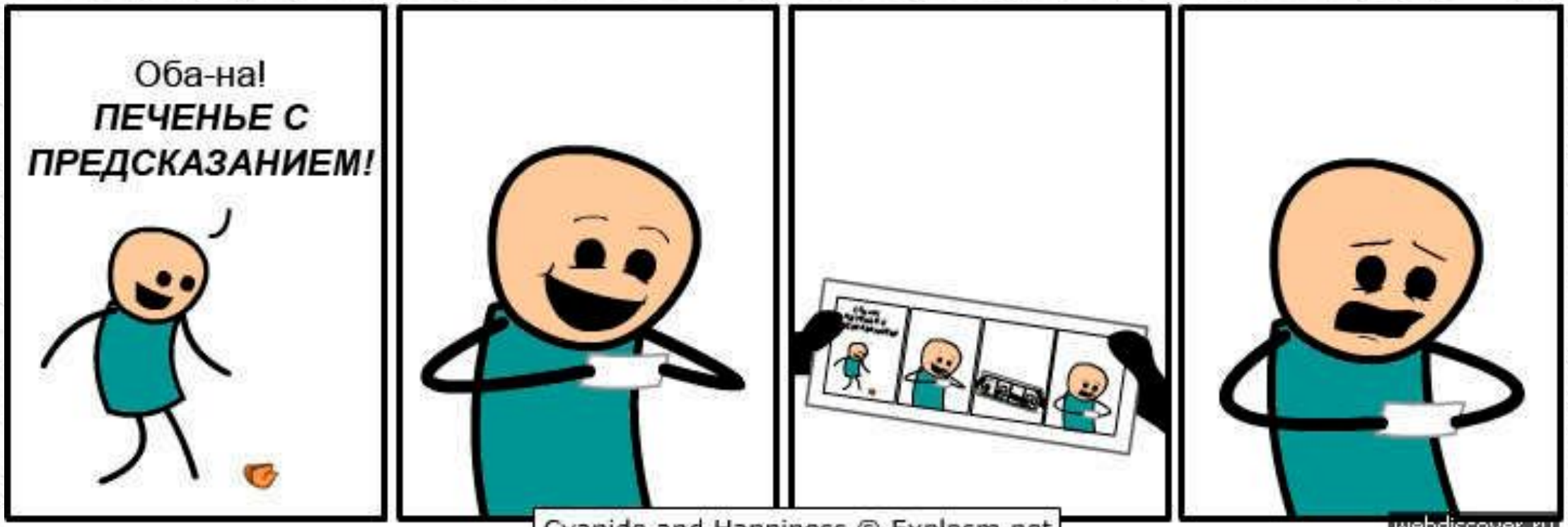
```
//программа поиска клеток на шахматной доске, из которой //конь обойдёт всю доску не побывав дважды
 //ни в одной из клеток
#include<iostream.h>
#include<stdlib.h>
#include <iomanip.h>
const int dx[8]={2,1,-1,-2,-2,-1, 1, 2};
const int dy[8]={1,2,2,  1,-1,-2,-2,-1};
const   int nx=15;
int h[15][15];
void move (int i,int x, int y,bool &q);
void ShowDesk(int dk[ ][nx],int nx);
 int n;
  void main(void)
{  bool nq;
  char ch;
cout<<" Введите размер доски: "<<'\n';
cin>>n;
 for (int i=0;i<n;i++)
 for (int j=0;j<n;j++)
    {    // очистка массива
    for (int i2=0;i2<n;i2++)
    for (int j2=0;j2<n;j2++)
      h[i2][j2]=0;
    h[i][j]=1; //инициализация первого хода
    nq=false;
    move(2,i,j,nq);
    if (nq==true) {cout<<" Решение:"<<i<<","<<j<<'\n';
                  cout<<" Показать доску(Y/N)?";
                  cin>>ch;
                  if (ch=='y'||ch=='Y')
                      { ShowDesk(h,n);
                       cout<<" Продолжить(Y/N)?";
                       cin>>ch;
                          if (ch=='n' ||ch=='N') return;
              }     }
   }
cout<<" Все варианты пройдены"<<'\n';
cin>>n;
}
void move (int ii,int x, int y,bool &q)
{int k,u,v;
bool q1;
 k=0;
if(q==true) return;
do
{if (k<8) k++;
else k=0;
q1=false;
u=x+dx[k];
v=y+dy[k];
if(u>=0&&u<n&&v>=0&&v<n&&h[u][v]==0)
            {h[u][v]=ii;
          if(ii<n*n){ move(ii+1,u,v,q1);}
              else{q=q1=true;return;}
       if(q1==true)  {q=q1;return;}
        h[u][v]=0;
    }       }
 while(k<8);
 q=q1;
 }//move

void ShowDesk(int dk[][nx],int nx)
{for(int i=0;i<nx;i++)
 {for(int j=0;j<nx;j++)
cout<< setw(4)<<dk[i][j]<<" ";
cout<<'\n';}
}
```

Рекурсия в языке C++

```
Введите размер доски:
5
Решение:0,0
Показать доску(Y/N)?y
  1  20  15   8   3
 14   9   2  21  16
 19  22  13   4   7
 10   5  24  17  12
 23  18  11   6  25
Продолжить(Y/N)?y
Решение:3,1
Показать доску(Y/N)?y
 25  16  11   4  23
 10   5  24  17  12
 15  18   9  22   3
  6   1  20  13   8
 19  14   7   2  21
Продолжить(Y/N)?y
```

```
Решение:4,0
Показать доску(Y/N)?y
 25  22  17  10   5
 16  11   6  23  18
 21  24  15   4   9
 12   7   2  19  14
  1  20  13   8   3
Продолжить(Y/N)?y
Решение:4,4
Показать доску(Y/N)?y
 25  20   9  14   3
  8  15   4  19  10
 21  24  13   2   5
 16   7  22  11  18
 23  12  17   6   1
Продолжить(Y/N)?y
Все варианты пройдены
```


Оба-на!
ПЕЧЕНЬЕ С ПРЕДСКАЗАНИЕМ!
Cyanide and Happiness © Explosm.net
webdiscover.ru