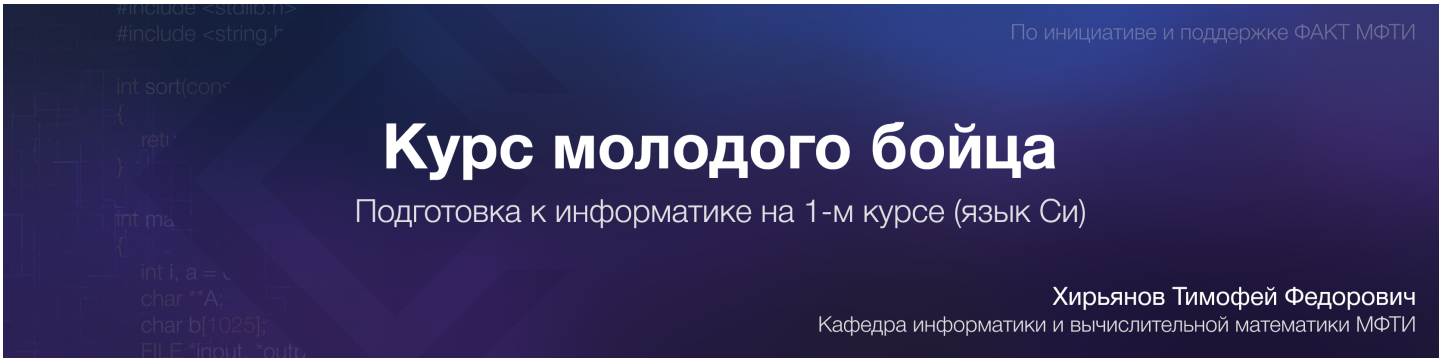


Главная

Рекурсия и динамическое программирование

Функции в языке Си

Что можно сделать с функцией. Синтаксис описания функции в Си. Синхронные и асинхронные вызовы. Адрес возврата и стек вызовов. Пример отладки программы с наблюдением за Call stack. Локальные переменные и параметры хранятся на стеке.

functions.c

```c
#include <stdio.h>

void A();
void B();
void C();

int main(int argc, char* argv[])
{
    printf("main() called.\n");
    A();
    printf("main() returns.\n");
```

```
    return 0;
}

void A()
{
    printf("  A() called.\n");
    B();
    printf("  A() returns.\n");
}

void B()
{
    printf("    B() called.\n");
    C();
    printf("    B() returns.\n");
}

void C()
{
    printf("      C() called.\n");
    printf("      C() returns.\n");
}
```

Рекурсия. Репка и матрёшка

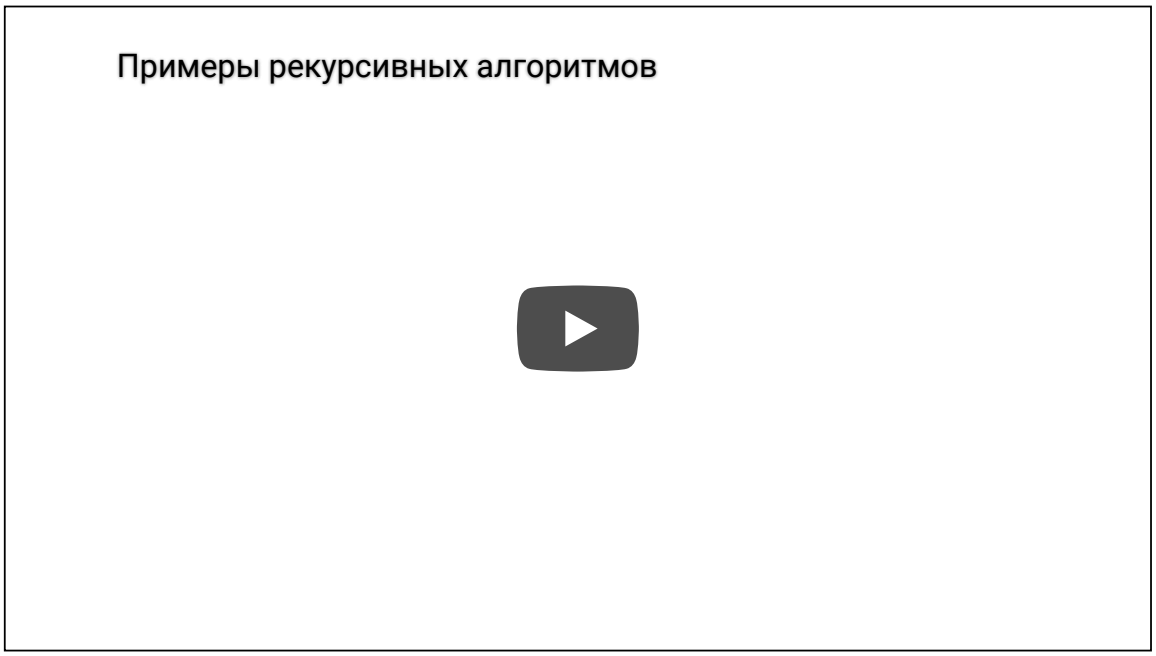
Сказка "Репка". Крайний случай. Прямой и обратный ход рекурсии. Алгоритм изготовления матрёшки. Программа, печатающая матрёшку.

matryoshka.c

```
#include <stdio.h>

void matryoshka(int n);

int main(int argc, char* argv[])
{
    matryoshka(7);

    return 0;
}

void matryoshka(int n)
{
    if (n == 1)
        printf(" Last matryoshka!!! %d\n", n);
    else
```

```
    {
        printf(" Top side of matryoshka %d\n", n);
        matryoshka(n-1);
        printf(" Bottom side of matryoshka %d\n", n);
    }
}
```

Примеры рекурсивных алгоритмов



Факториал числа. Алгоритм Евклида. Быстрое возведение в степень. Числа Фибоначчи.

recursion_examples.c

```
#include <stdio.h>

int factorial(int n)
{
    if (0 == n)
        return 1;
    return factorial(n-1)*n;
}

int gcd(int a, int b)
{
    if (0 == b) return a;
    return gcd(b, a%b);
}

double fast_power(double a, int n)
{
    if (0 == n) return 1;
    if (n%2 == 1)
        return a*fast_power(a, n-1);
    else
        return fast_power(a*a, n/2);
}

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

int main(int argc, char* argv[])
{
    int n, m;
```

```
    scanf("%d%d", &n, &m);
    printf("factorial(%d) = %d\n", n, factorial(n));
    printf("gcd(%d, %d) = %d\n", n, m, gcd(n, m));
    printf("fast_power(%d, %d) = %lf\n", n, m, fast_power(n, m));
    printf("fib(%d) = %d\n", n, fib(n));

    return 0;
}
```

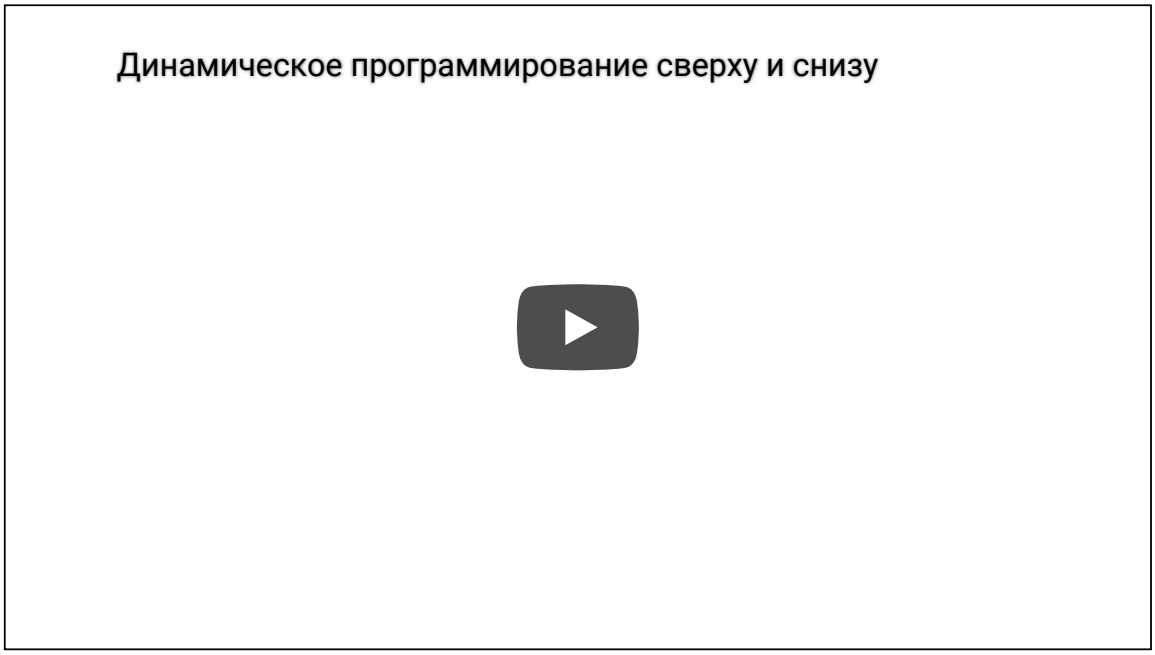
Ханойские башни

Постановка задачи. Рекуррентный алгоритм и его реализация.

hanoi.c

```
#include <stdio.h>

void hanoi(int n, int i, int k);

int main(int argc, char* argv[])
{
    hanoi(3, 1, 2);

    return 0;
}

void hanoi(int n, int i, int k)
{
    if (n == 1)
        printf("Move disk 1 from pin %d to %d.\n", i, k);
    else
    {
        int tmp = 6 - i - k;
        hanoi(n-1, i, tmp);
        printf("Move disk %d from pin %d to %d.\n", n, i, k);
        hanoi(n-1, tmp, k);
    }
}
```

Динамическое программирование сверху и снизу

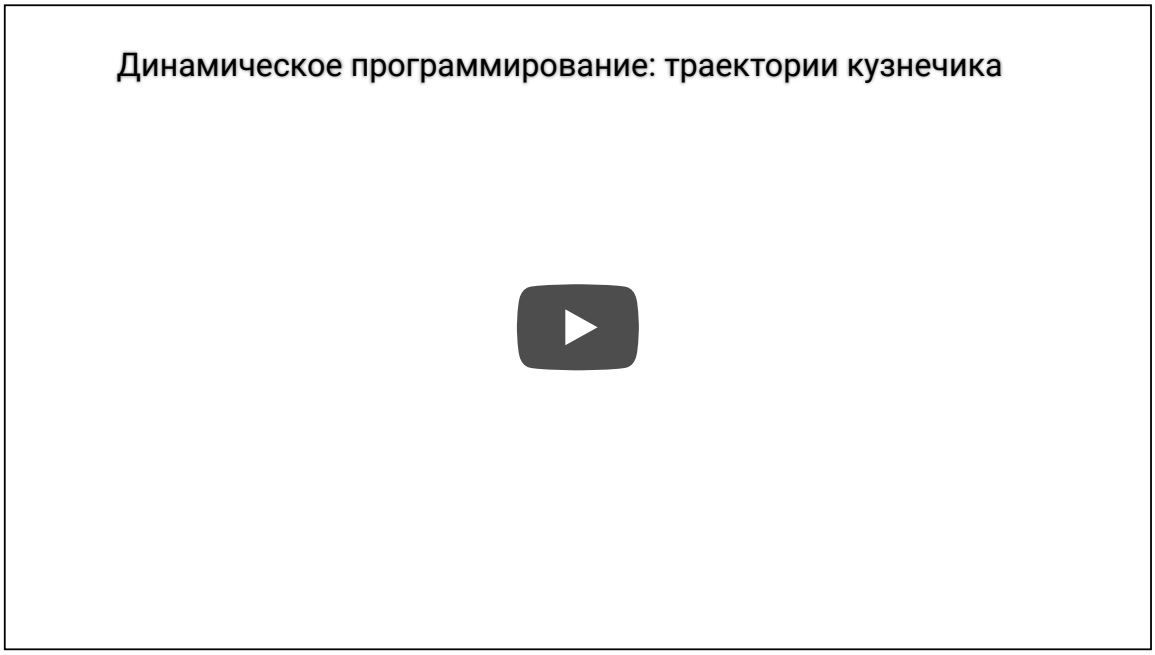


Скорость рекуррентного вычисления чисел Фибоначчи. Проблема повторных вычислений. Подход динамического программирования состоит в том, чтобы решить каждую подзадачу только один раз, сократив тем самым количество вычислений Динамическое программирование сверху и снизу.

fibonacci_time.c

```c
#include <stdio.h>
#include <time.h>

static int cache[100] = {0};

int fib(int n)
{
    if (n <= 1) return n;
    if (cache[n] == 0)
        cache[n] = fib(n-1) + fib(n-2);
    return cache[n];
}

int fib_dynamic(int n)
{
    int Fib[n+1];
    Fib[0] = 0;
    Fib[1] = 1;
    for (int i = 2; i <= n; ++i)
        Fib[i] = Fib[i-1] + Fib[i-2];
    return Fib[n];
}

int main(int argc, char* argv[])
{
    for (int n = 1; n < 50; n += 1)
    {
        clock_t time1 = clock();
        int result = fib_dynamic(n);
        clock_t time2 = clock();
        int delta_ms = (time2 - time1)*1000/CLOCKS_PER_SEC;
        printf("fib(%d) = %d,\t time = %d ms\n",
               n, result, delta_ms);
    }

    return 0;
}
```

Динамическое программирование: траектории кузнечика



Задача из ЕГЭ про граф дорог. Количество различных траекторий кузнечика из 1 в N. Реализация динамическим программированием.

grasshopper.c

```c
#include <stdio.h>

int number_of_trajectories(int n)
{
    int K[n+1];
    K[0] = 0;
    K[1] = 1;
    for (int i = 2; i <= n; ++i)
        K[i] = K[i-1] + K[i-2];
    return K[n];
}

int main(int argc, char* argv[])
{
    int finish;
    scanf("%d", &finish);
    printf("Grasshopper has %d trajectories from 1 to %d\n",
           number_of_trajectories(finish), finish);
    return 0;
}
```

Самостоятельная работа

Уважаемые студенты!

К 4-му уроку есть домашняя работа в форме контеста: **ссылка на ДЗ №4**. Ссылка на неё также находится на главной странице сайта.

Если у вас нет логина и пароля, **зарегистрируйтесь на 1-й контест**, и доступ к остальным вы получите автоматически.