Алгоритмы и Алгоритмические Языки

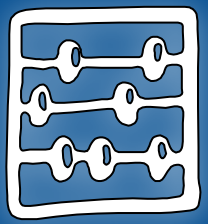
Семинар #6:

- Результаты работы по выражениям;
- Функции и рекурсия;
- Указатели на функцию;
- Хвостовая рекурсия;

03.10.2023

HSE University

Результаты работы по выражениям



Работа над ошибками

Задание №2:

```
b <<= a >>= b      (x = 0) || x / 2      x = y | y = z
```

Задание №3:

```
unsigned short c; signed char h; typeof(c + h) = ???
```

Задание №4:

Выпишите **выражение** типа **`unsigned`**,
во втором байте значения которого все биты **равны 1**,
а **остальные байты совпадают** с соответствующими байтами значения переменной **`unsigned t`**.
Нумерация байтов начинается **с нуля, с младшего байта**.

Функции и рекурсия

Общие понятия про функциям

Характеристики функций:

- Уникальное имя;
- Возвращаемое значение;
- Параметры и аргументы;
- Объявление (declaration), определение (definition), вызов (call);

Особенности вызова функций:

```
int max_value = max(expr1, expr2);
```

Выражения `expr1` и `expr2` упорядочены только с вызовом функции, но не друг с другом!

Расчёт N-го числа Фибоначчи

См. пример **06_recursion**, считающий N-ое число Фибоначчи:

```c
typedef unsigned long long ull_t;

// Function declaration:
ull_t fibs(unsigned n);

int main(void)
{
    // Print the Fibonacci sequence:
    for (unsigned i = 0U; i < 50U; ++i)
    {
        // Function call:
        printf("fibs[%03u] = %llu\n", i, fibs(i));
    }

    return EXIT_SUCCESS;
}
```

Расчёт N-го числа Фибоначчи

```
ull_t fibs(unsigned n)
{
    if (n == 0U) { return 0ULL; }
    if (n == 1U) { return 1ULL; }

    ull_t prev = 0ULL;
    ull_t cur  = 1ULL;
    for (unsigned i = 1U; i < n; ++i)
    {
        ull_t tmp = prev;
        prev = cur;
        cur = tmp + cur;
    }

    return cur;
}
```

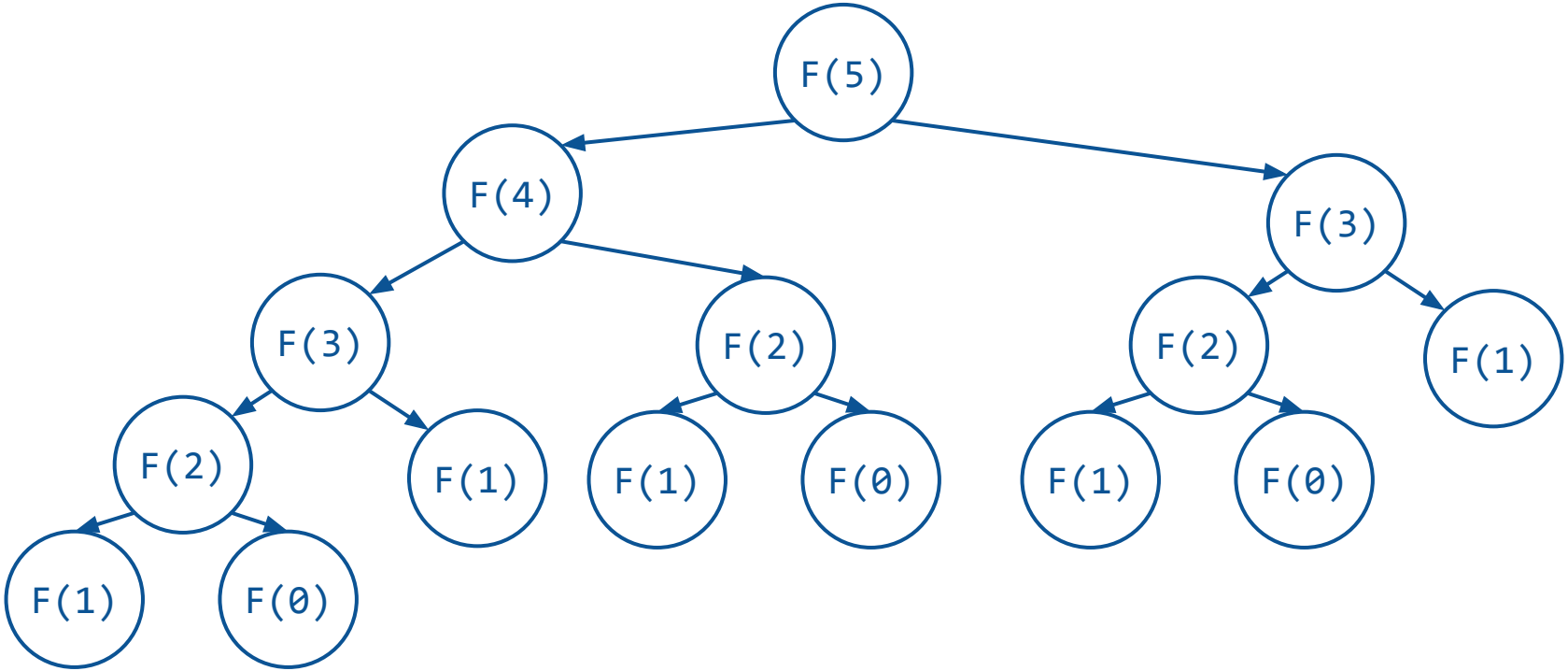
Рекурсия

Рекурсия – вызов функции внутри себя самой.

```
ull_t fibs(unsigned n)
{
    if (n == 0U)
    {
        return 0ULL;
    }
    if (n == 1U)
    {
        return 1ULL;
    }

    return fibs(n - 1ULL) + fibs(n - 2ULL);
}
```

Сложность рекурсивного решения

Сложность решения – экспоненциальная ($O\left(\left(\frac{1+\sqrt{5}}{2}\right)^N\right)$)

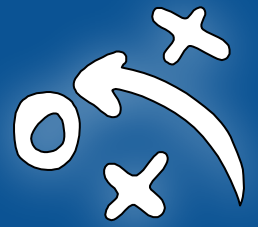


Более грамотное решение

```
ull_t fibs_helper(ull_t acc_prev, ull_t acc_cur, unsigned n)
{
    if (n == 1U) { return acc_cur; }

    return fibs_helper(acc_cur, acc_prev + acc_cur, n - 1ULL);
}

ull_t fibs(unsigned n)
{
    if (n == 0U) { return 0U; }

    return fibs_helper(0ULL, 1ULL, n);
}
```

HSE University

Указатели на функцию, хвостовая рекурсия



Указатели на функции

Объявление указателя на функцию:

```
void f(int);
void (*pf1)(int) = &f;
void (*pf2)(int) = f; // same as &f
```

Вызов функции по указателю:

```
int f();          // declaration
int (*p)() = f; // pointer assignment
(*p)();           // call by pointer
p();              // another call by pointer
```

Сумма арифметической прогрессии

См. пример 06_generators из репозитория:

```
int main(void)
{
    <input sequence_size>

    unsigned array[sequence_size];
    for (unsigned i = 0U; i < sequence_size; ++i)
    {
        array[i] = i;
    }

    printf("Progression sum of %u elements is: %llu\n",
        sequence_size, sum(array, sequence_size));

    return EXIT_SUCCESS;
}
```

Сумма арифметической прогрессии

```
ull_t sum(const unsigned* array, unsigned size)
{
    ull_t acc = 0ULL;
    for (unsigned i = 0U; i < size; ++i)
    {
        acc += array[i];
    }

    return acc;
}
```

Возможные проблемы?

Решение через генераторы

Решение – функция, задающая последовательность:

```
ull_t sum(unsigned (*get_element)(unsigned), unsigned size)
{
    if (size == 0U) { return 0ULL; }

    return sum(get_element, size - 1) + get_element(size);
}
```

```
unsigned generate_progression(unsigned index)
{
    return index;
}
```

Решение через генераторы

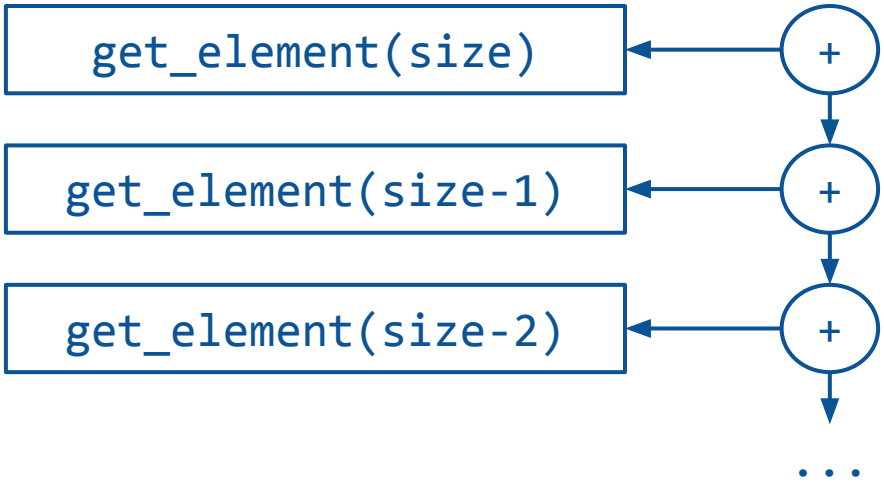
Вызов функции по указателю:

```
printf("Progression sum of %u elements is: %llu\n",
    sequence_size, sum(&generate_progression, sequence_size));
```

Возможные проблемы?

Схема рекурсивных вызовов

```
ull_t sum(unsigned (*get_element)(unsigned), unsigned size)
{
    return sum(get_element, size - 1) + get_element(size);
}
```

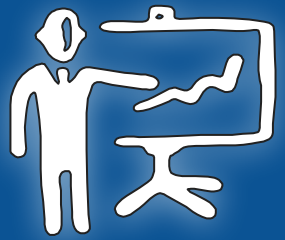


Решение через хвостовую рекурсию

```
ull_t sum_helper(
    unsigned (*get_element)(unsigned), unsigned size, ull_t acc)
{
    if (size == 0U) { return acc; }

    return sum_helper(
        get_element, size - 1U, acc + get_element(size));
}

ull_t sum(
    unsigned (*get_element)(unsigned), unsigned size)
{
    return sum_helper(get_element, size, 0U);
}
```

С уровнем оптимизации -O2 компилятор сгенерирует цикл!

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com