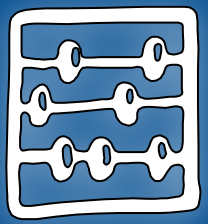
Алгоритмы и Алгоритмические Языки

Семинар #8:

1. Результаты работы по выражениям.
2. Функции и рекурсия.
3. Указатели на функции.

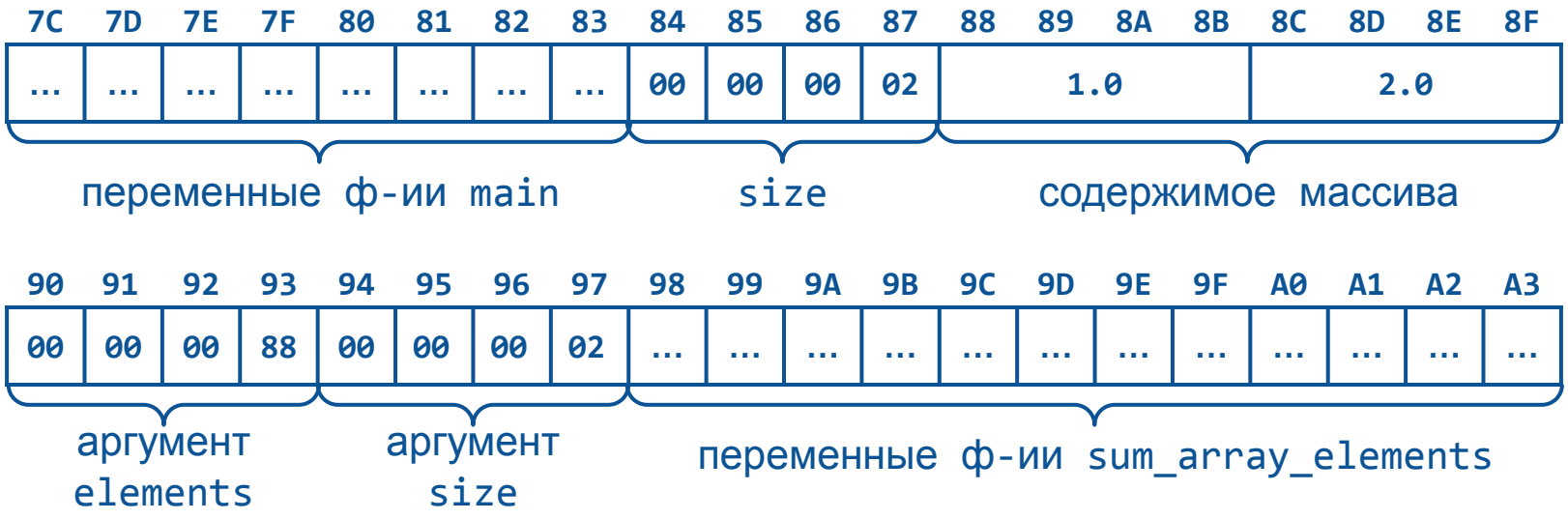
08.10.2024

Результаты работы по выражениям



Функции и рекурсия

Передача аргументов в функцию




```
size_t size = ...;
double elements[size] = ...;
double sum = sum_array_elements(elements, size);
```

Стек вызовов

Расчёт N-го числа Фибоначчи

```c
// Конструкция typedef позволяет создавать псевдонимы типов
typedef unsigned long long ull_t;

// Объявление функции
ull_t fibs(unsigned n);

int main(void)
{
    // Печать 50 чисел Фибоначчи:
    for (unsigned i = 0U; i < 50U; ++i)
    {
        // Вызов функции:
        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;
}
```

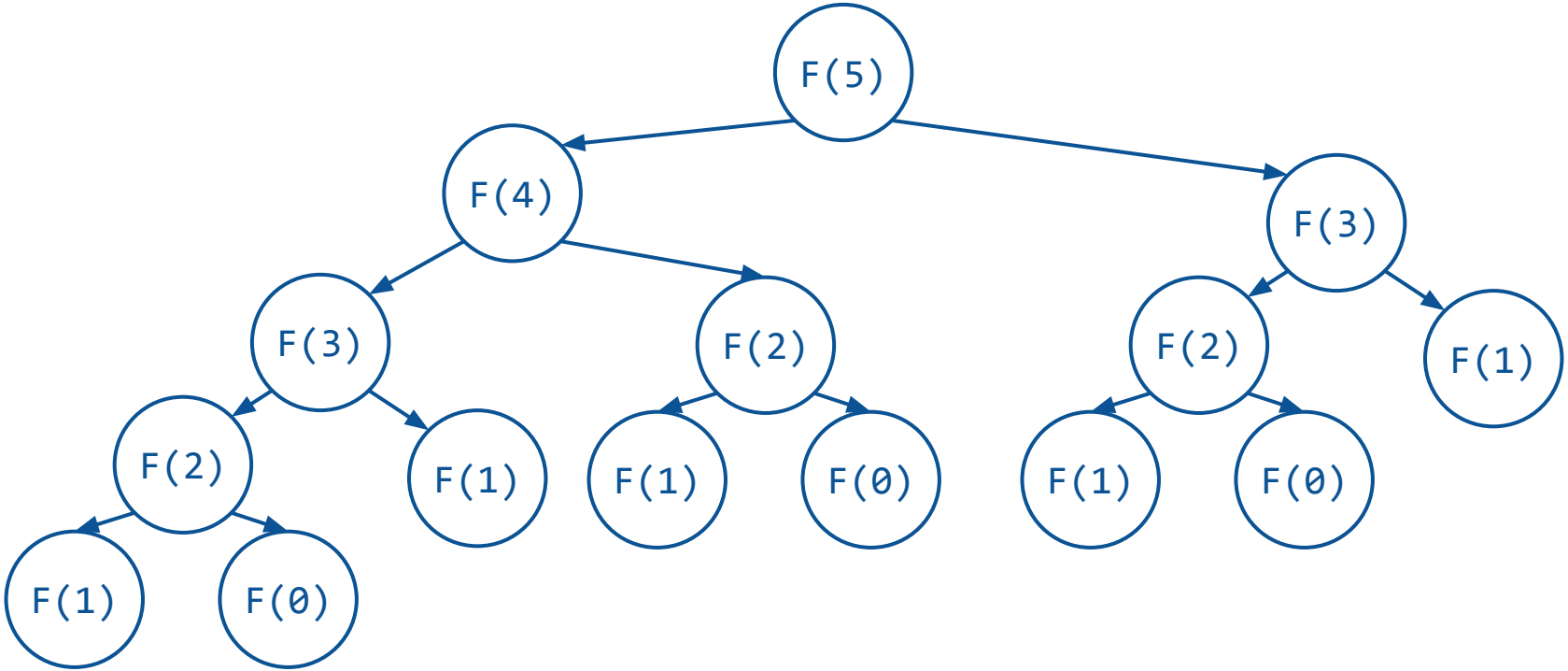
Рекурсивное решение

```
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

Количество вызовов функции F = F(N) = $O\left(\left(\frac{1+\sqrt{5}}{2}\right)^N\right)$



Более грамотная рекурсия

```
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);
}
```

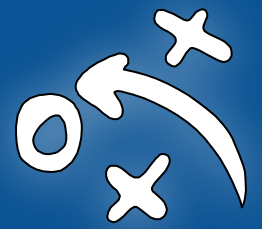
Более грамотная рекурсия

Оптимизация хвостовой рекурсии с флагом -O2 (decompiled)

```
000011e0  int64_t fibs_recursion_helper(int64_t arg1, int64_t arg2, int32_t arg3) __pure

000011e0  {
000011e7      if (arg3 == 1)
000011e4      {
0000120b          return arg2;
0000120b      }
000011f3      int64_t rax_1;
000011f3      while (true)
000011f3      {
000011f3          arg3 = (arg3 - 1);
000011f6          rax_1 = (arg1 + arg2);
000011fa          arg1 = arg2;
00001200          if (arg3 == 1)
000011fd          {
00001200              break;
00001200          }
000011f0          arg2 = rax_1;
000011f0      }
00001202      return rax_1;
00001202  }
```

Указатели на функцию



Сумма элементов ряда

```
// Ввод количества элементов ряда:
printf("Enter number of sequence elements:\n");

unsigned sequence_size;
if (scanf("%u", &sequence_size) != 1)
{
    printf("Expected one unsigned integer\n");
    return EXIT_FAILURE;
}

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));
```

Сумма элементов ряда

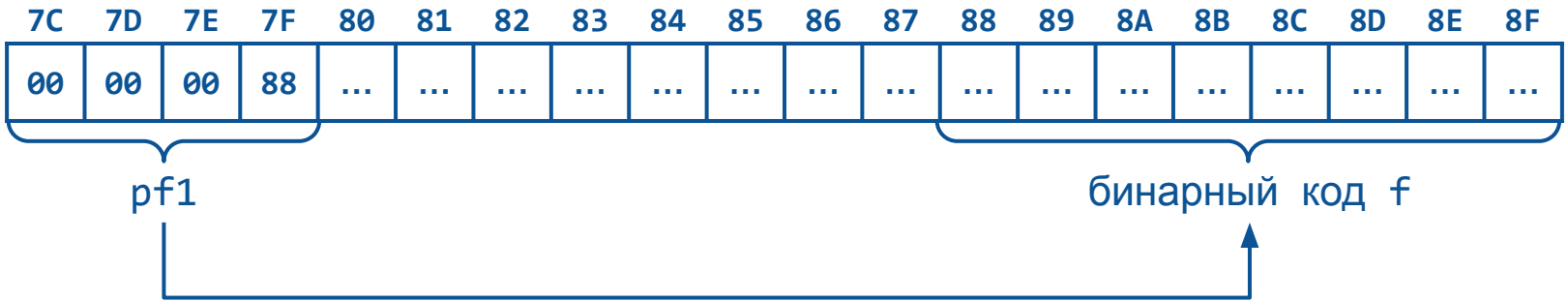
См. пример 08_generators

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

    return acc;
}
```

Указатель на функцию

```
void f(int);                    // Объявление функции
void (*pf1)(int) = &f; // Указатель на функцию
void (*pf2)(int) =  f; // Указатель на функцию
(*pf1)(10);                     // Вызов по указателю
pf2(10);                        // Вызов по указателю
```



Указатель на функцию

```
ull_t sum(unsigned (*get_element)(unsigned), unsigned nmemb)
{
    ull_t acc = 0ULL;

    for (unsigned i = 0U; i <= nmemb; ++i)
    {
        acc += get_element(i);
    }

    return acc;
}
```

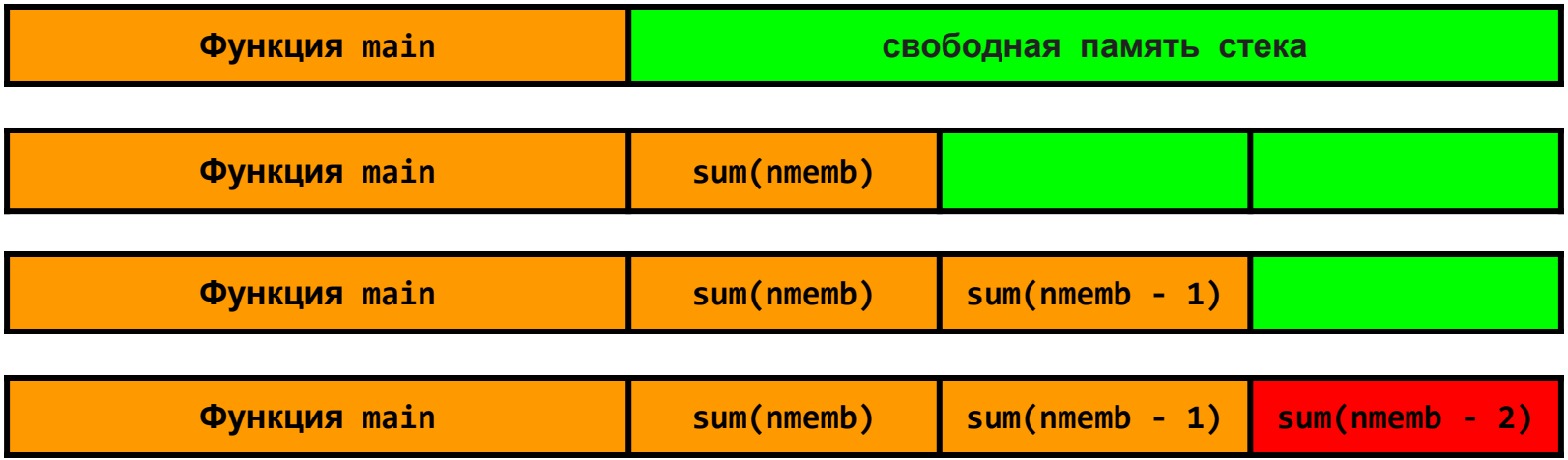
Рекурсивная реализация

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

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

Переполнение стека

```
return sum(get_element, nmemb - 1) + get_element(nmemb);
```



Снова хвостовая рекурсия

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

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

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

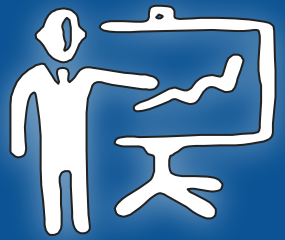
Снова хвостовая рекурсия

Оптимизация хвостовой рекурсии с флагом -O2 (decompiled)

```
00001290  int64_t sum(int64_t arg1, int32_t arg2)

00001290  {
00001297      int64_t rbp = 0;
0000129c      if (arg2 != 0)
0000129a      {
000012a1          int32_t i_1 = arg2;
000012b5          int32_t i;
000012b5          do
000012b5          {
000012af              rbp = (rbp + ((uint64_t)arg1(((uint64_t)i_1))));
000012b2              i = i_1;
000012b2              i_1 = (i_1 - 1);
000012b2          } while (i != 1);
000012b2      }
000012be      return rbp;
000012be  }
```

Вопросы?



Красивые иконки взяты с сайта handdrawngoods.com