Введение. Основные понятия.

Алгоритмы и структуры данных

Мулюгин Николай

03.09.2022

Информационные источники

1 А. В. Столяров. Введение в профессию.
http://stolyarov.info

2 К. Владимиров.
youtube.com/channel/UCvmBEbr9NZt7UEh9doI7_A

3 Ю. Окуловский.
https://ulearn.me/

Информационные источники

4 Томас Х. Кормен. Алгоритмы: построение и анализ.

5 Томас Х. Кормен. Алгоритмы. Вводный курс.

6 Род Стивенс. Алгоритмы. Теория и практическое применение.

7 Никлаус Вирт. Алгоритмы и структуры данных.

8 Д. Кнут. Искусство программирования.

Мотивация

- Применение в науке.
- Трудоустройство. Вопросы на собеседовании.
- Общение с коллегами.
- Повседневное применение.

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Свойства алгоритма:

обязательные

необязательные

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Свойства алгоритма:

обязательные

1. Конечность.

необязательные

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Свойства алгоритма:

обязательные

1. Конечность.
2. Дискретность.

необязательные

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Свойства алгоритма:

обязательные

1. Конечность.
2. Дискретность.
3. Определенность.

необязательные

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Свойства алгоритма:

обязательные

1. Конечность.
2. Дискретность.
3. Определенность.

необязательные

4. Правильность.

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Свойства алгоритма:

обязательные

1. Конечность.
2. Дискретность.
3. Определенность.

необязательные

4. Правильность.
5. Завершаемость.

Алгоритм

Алгоритм представляет собой набор правил преобразования исходных данных(input) в выходные(output).

Алгоритмы строятся для решения **вычислительных задач**.

Свойства алгоритма:

обязательные

1. Конечность.
2. Дискретность.
3. Определенность.

необязательные

4. Правильность.
5. Завершаемость.
6. Массовость.

Сложность алгоритма мотивация

Есть алгоритм. Что дальше?

Сложность алгоритма мотивация

Есть алгоритм. Что дальше?

- Анализировать алгоритм.

Сложность алгоритма мотивация

Есть алгоритм. Что дальше?

- Анализировать алгоритм.
- Сравнивать различные алгоритмы.

Сложность алгоритма мотивация

Есть алгоритм. Что дальше?

- Анализировать алгоритм.
- Сравнивать различные алгоритмы.
- Понять как поведет себя алгоритм в будущем.

Сложность алгоритма

Время работы алгоритма - число элементарных операций, которые он выполняет.

Сложность алгоритма

Время работы алгоритма - число элементарных операций, которые он выполняет.

Время работы алгоритма в худшем случае (временная сложность) - максимальное время работы для входов данного размера.

Сложность алгоритма

Время работы алгоритма - число элементарных операций, которые он выполняет.

Время работы алгоритма в худшем случае (временная сложность) - максимальное время работы для входов данного размера.

Емкостная сложность - максимально используемый объем памяти для входов данного размера.

Сложность алгоритма

Время работы алгоритма - число элементарных операций, которые он выполняет.

Время работы алгоритма в худшем случае (временная сложность) - максимальное время работы для входов данного размера.

Емкостная сложность - максимально используемый объем памяти для входов данного размера.

Асимптотическая сложность алгоритма - оценка, характеризующая вид зависимости времени работы алгоритма от длины входа.

Сложность алгоритма

```
#include<iostream> //headers
#include<string>
//g++ -Wall -Werror -g --std=c++20 complex_1.cpp
//g++ - compiler; -Wall - show all the errors;
//-Werror - treat warnings as errors
int main()//entry point
{
    std::string input;//variable with type string
    std::cin >> input;//standard input
    int n = input.size();
    int sum = 0;
    for( int i = 0; i < n; i++ )//n
        for( int j = 0; j < 2 * i; j++ )
        //01 0123 012345 0123456 012..2i-1
            sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
    std::cout << sum << std::endl;
    return 0;//code of return
}
```

Сложность алгоритма

Посчитаем сложность алгоритма - функцию $f : N \to N$ которая на вход берет n - длину входа и возвращает число элементарных операций для данного n.

```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )
    for( int j = 0; j < 2 * i; j++ )
        sum++;
```

Сложность алгоритма

```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )//n
    for( int j = 0; j < 2 * i; j++ )
    //01 0123 012345 0123456 012..2i-1
        sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
```

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out}$$

f(n) - характеристика нашего алгоритма.
Какие выводы мы можем сделать по ней?

Сложность алгоритма мотивация

- Анализировать алгоритм.
- Сравнивать различные алгоритмы.
- Понять как поведет себя алгоритм в будущем.

Переменные

$\boxed{0100}$

Переменные

a 4 $[-7 \div 8]$

$\boxed{0100}$

Переменные

a 4 $[-7 \div 8]$

$\boxed{0100}$

Мы знаем тип a?

Мы знаем диапозон значений!

Переменные

a 4 [-7÷8] 4 b

0100 0011

Мы знаем тип a?

Мы знаем диапозон значений!

Мы не знаем операции!

Типы

- Что такое тип?
- value type: диапозон значений объекта
- object type: совокупность операций над объектом
  - 5/2 даст 2 для int но 2.5 для double
- Назовем целочисленный арифметический тип i4

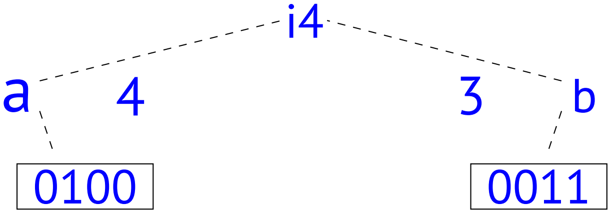
a 4 i4 3 b

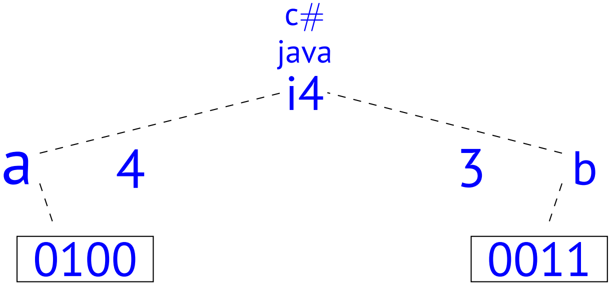
$\boxed{0100}$ $\boxed{0011}$

ruby
python

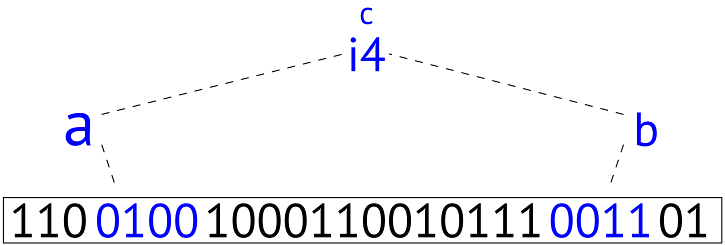
a 4 i4 3 b

0100 0011

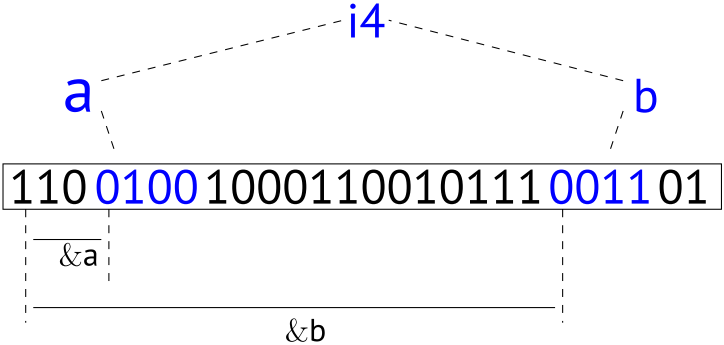

i4
a
4
3
b
0100
0011



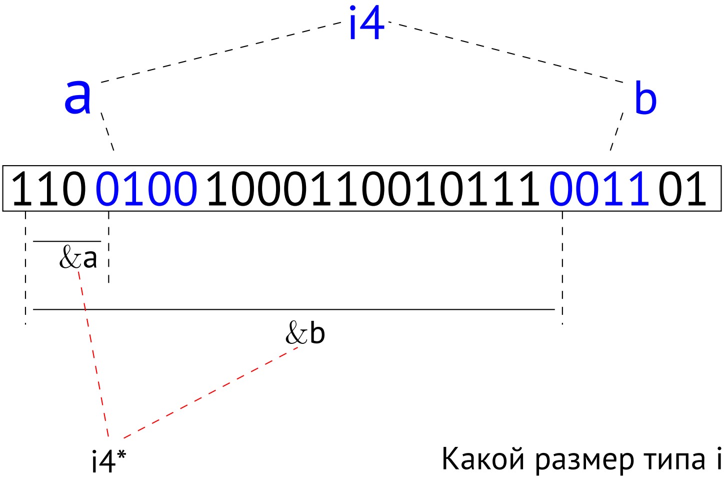
Имя навсегда связано с типом.



RAM - random access memory

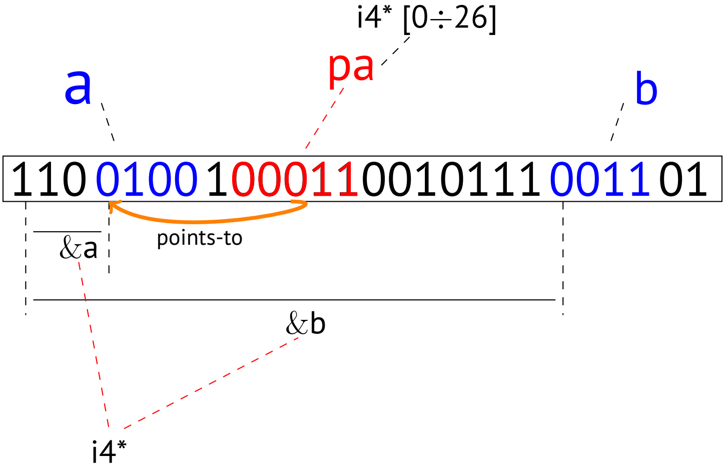


i4
a
b
110 0100 10001100101110011 01
&a
&b



Какой размер типа i4* ?

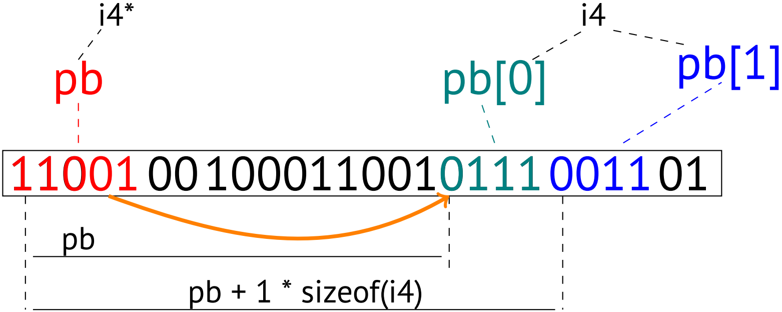
Какое значение CHAR_BIT ?



i4* [0÷26]
pa
a
b
110 0100 1 00011 00010111 0011 01
points-to
&a
&b
i4*

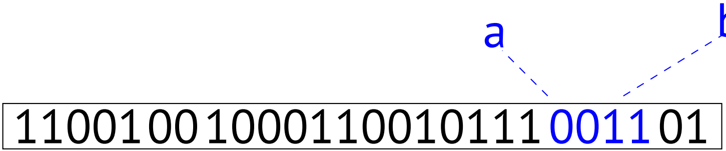
Указатели

- Если указатель это просто расстояние, то может быть нулевое расстояние?
  - nullptr
- p[2] == *(p+2)


i4*
i4
pb
pb[0]
pb[1]
11001 00 1000011001 0111 0011 01
pb
pb + 1 * sizeof(i4)

Ссылки

- Два имени у одного объекта в C++.



Ссылки

Базовый синтаксис lvalue ссылок это одинарный амперсанд

```
int x;
int &y = x; // y - another name for x
```

Ссылки

Базовый синтаксис lvalue ссылок это одинарный амперсанд

```
int x;
int &y = x; // y - another name for x
```

Не путать со взятием адреса!

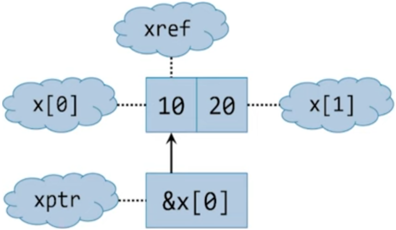
Ссылки

Базовый синтаксис lvalue ссылок это одинарный амперсанд

```
int x;
int &y = x; // y - another name for x
```

Не путать со взятием адреса!

```
int x[2] = {10, 20};
int &xref = x[0];
int *xref = &x[0];
xref += 1;
xptr += 1;
assert(xref == 11);
assert(xptr == 20);
```



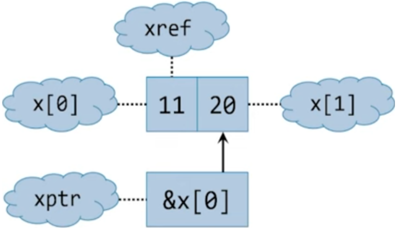
Ссылки

Базовый синтаксис lvalue ссылок это одинарный амперсанд

```
int x;
int &y = x; // y - another name for x
```

Не путать со взятием адреса!

```
int x[2] = {10, 20};
int &xref = x[0];
int *xref = &x[0];
xref += 1;
xptr += 1;
assert(xref == 11);
assert(xptr == 20);
```



Оценка сложности алгоритма

Чем мы тут занимаемся???

```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )//n
  for( int j = 0; j < 2 * i; j++ )
  //01 0123 012345 0123456 012..2i-1
      sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
```

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out}$$

- f(n) - характеристика нашего алгоритма.
- Какие выводы мы можем сделать по ней?

Мотивация!!!

Есть алгоритм. Что дальше?

- Анализировать алгоритм.
- Понять как поведет себя алгоритм в будущем.
- Сравнивать различные алгоритмы.

Анализировать алгоритм

```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )//n
  for( int j = 0; j < 2 * i; j++ )
  //01 0123 012345 0123456 012..2i-1
      sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
```

$$f(n) = n \cdot (C_{=} + C_{<} + C_{*} + C_{+}) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_{=} + C_{<} + C_{*} + C_{+}) + C_{in} + C_{out}$$

Анализировать алгоритм

```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )//n
  for( int j = 0; j < 2 * i; j++ )
  //01 0123 012345 0123456 012..2i-1
      sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
```

$$f(n) = n \cdot (C_{=} + C_{<} + C_{*} + C_{+}) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_{=} + C_{<} + C_{*} + C_{+}) + C_{in} + C_{out}$$

f(n)- показывает зависимость алгоритма от длины входа.

Понять как поведет себя алгоритм в будущем

```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )//n
  for( int j = 0; j < 2 * i; j++ )
  //01 0123 012345 0123456 012..2i-1
      sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
```

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out}$$

Понять как поведет себя алгоритм в будущем

```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )//n
  for( int j = 0; j < 2 * i; j++ )
  //01 0123 012345 0123456 012..2i-1
      sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
```

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out}$$

$$f(n) : \; n \to \infty$$

Понять как поведет себя алгоритм в будущем

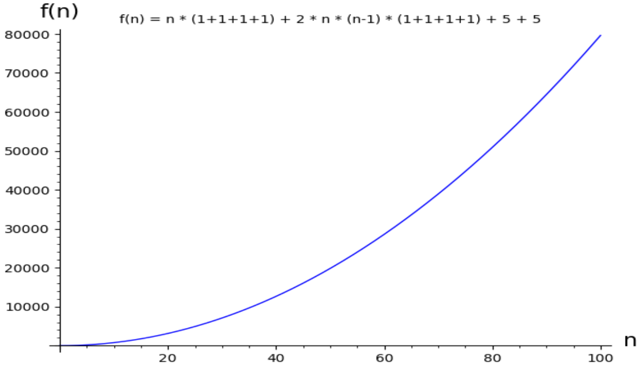
```
int n = input.size();
int sum = 0;
for( int i = 0; i < n; i++ )//n
  for( int j = 0; j < 2 * i; j++ )
  //01 0123 012345 0123456 012..2i-1
      sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
```

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out}$$

$$f(n) : \ n \to \infty$$

```
f(n) = n*(1+1+1+1) + 2*n*(n-1)*(1+1+1+1) + 5 + 5
plot(f, (n,0,100), axes_labels=['n','f(n)'])
```

Понять как поведет себя алгоритм в будущем



Масштаб роста фунций

$$f(n) = n \cdot (C_{=} + C_{<} + C_{*} + C_{+}) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_{=} + C_{<} + C_{*} + C_{+}) + C_{in} + C_{out} =$$

Масштаб роста фунций

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out} =$$
$$= n \cdot C_1 + 2 \cdot n \cdot (n-1) \cdot C_2 + C_3 =$$

Масштаб роста фунций

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+ 2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out} =$$
$$= n \cdot C_1 + 2 \cdot n \cdot (n-1) \cdot C_2 + C_3 =$$
$$= 2 \cdot n^2 \cdot C_2 + n \cdot (C_1 - 2 \cdot C_2) + C_3 =$$

Масштаб роста фунций

$$
\begin{aligned}
&f(n) = n \cdot (C_= + C_< + C_* + C_+) + \\
&+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out} = \\
&= n \cdot C_1 + 2 \cdot n \cdot (n-1) \cdot C_2 + C_3 = \\
&= 2 \cdot n^2 \cdot C_2 + n \cdot (C_1 - 2 \cdot C_2) + C_3 = \\
&= 2 \cdot n^2 \cdot C_2 + n \cdot C_4 + C_3.
\end{aligned}
$$

Масштаб роста фунций

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out} =$$
$$= n \cdot C_1 + 2 \cdot n \cdot (n-1) \cdot C_2 + C_3 =$$
$$= 2 \cdot n^2 \cdot C_2 + n \cdot (C_1 - 2 \cdot C_2) + C_3 =$$

$$= 2 \cdot n^2 \cdot C_2 + n \cdot C_4 + C_3.$$

Может быть нам более интеерсен масштаб роста функции?

Масштаб роста фунций

$$f(n) = n \cdot (C_= + C_< + C_* + C_+) +$$
$$+2 \cdot n \cdot (n-1) \cdot (C_= + C_< + C_* + C_+) + C_{in} + C_{out} =$$
$$= n \cdot C_1 + 2 \cdot n \cdot (n-1) \cdot C_2 + C_3 =$$
$$= 2 \cdot n^2 \cdot C_2 + n \cdot (C_1 - 2 \cdot C_2) + C_3 =$$
$$= 2 \cdot n^2 \cdot C_2 + n \cdot C_4 + C_3.$$

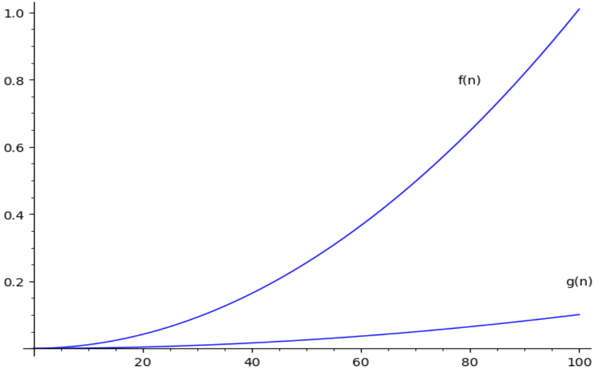
Может быть нам более интеерсен масштаб роста функции?

Не все ли равно какие коэффициенты C_1, C_2, C_3, C_4 при $n \to \infty$?

Масштаб роста фунций

```
f(n) = 10*n^2 + n *10 + 10
g(n) = 100*n^2 + n *100 + 100

f_p = plot(f, (n,0,100))
g_p = plot(g, (n,0,100))
ans = f_p + g_p

ans += text("g(n)", (100, +200000))
ans += text("f(n)", (80, +800000))
ans
```

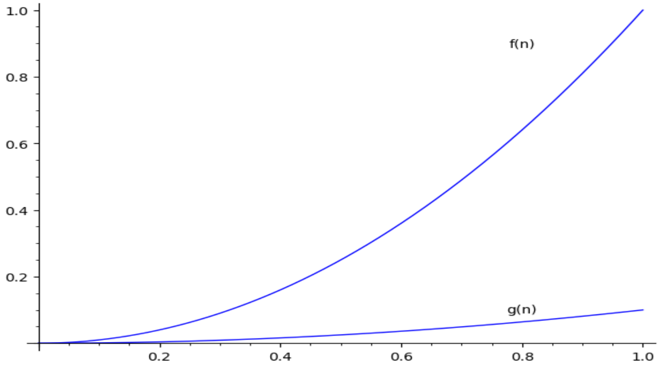
Масштаб роста фунций



Масштаб роста фунций

```
f(n) = 10*n^2 + n *10 + 10
g(n) = 100*n^2 + n *100 + 100

f_p = plot(f, (n,0,100000000))
g_p = plot(g, (n,0,100000000))
ans = f_p + g_p
ans+= text("g(n)", (80000000, +1000000000000000000))
ans+= text("f(n)", (80000000, +9000000000000000000))
ans
```

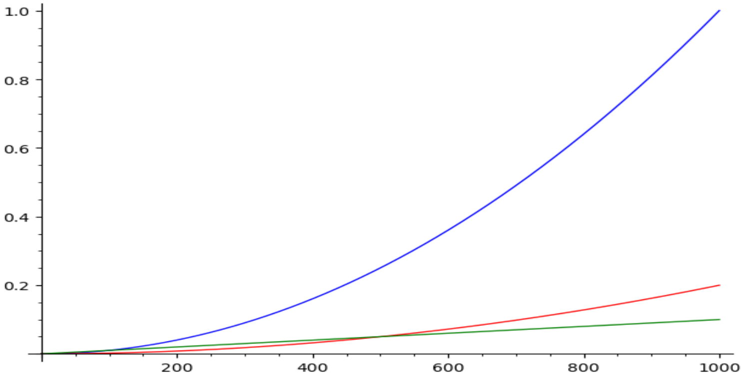
Масштаб роста фунций



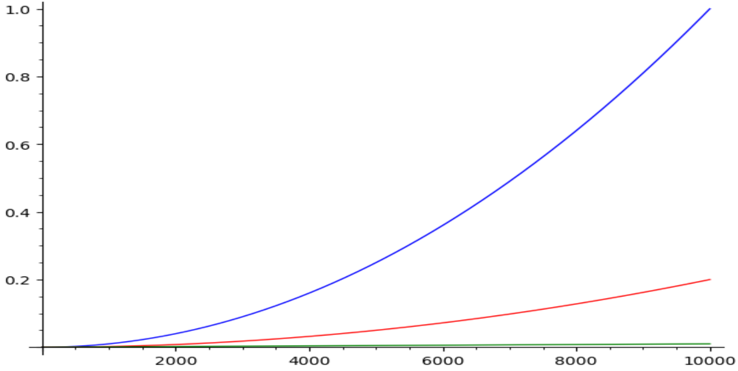
Масштаб роста фунций

```
f(n) = 10*n^2 + n *10 + 10
quad(n) = 2*n^2
lin(n) = 1000*n

f_p = plot(f, (n,0,1000))
quad_p = plot(quad, (n,0,1000), color='red')
lin_p = plot(lin, (n,0,1000),color = 'green')
ex_p = plot(ex, (n,0,1000),color = 'brown')

ans = f_p + quad_p + lin_p
ans
```

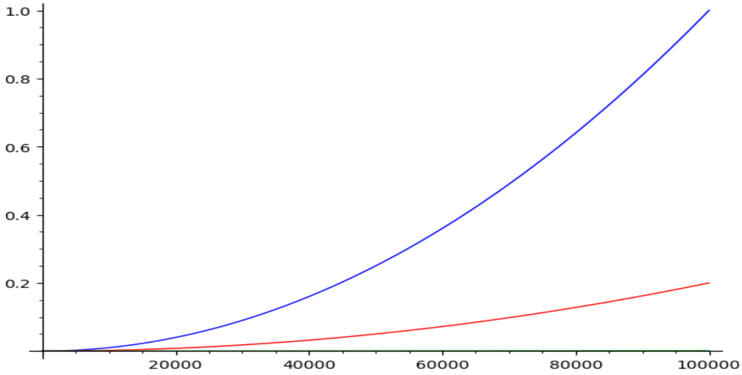
Масштаб роста фунций



Масштаб роста фунций



Масштаб роста фунций



О-символика

$f(n) = o(g(n))$ $\forall k > 0\ \exists n_0\ \forall n > n_0$ $f(n) < k \cdot g(n)$ $\Leftrightarrow \lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = 0$ $f(n) \prec g(n)$

$f(n) = O(g(n))$ $\exists k > 0\ \exists n_0\ \forall n > n_0$ $f(n) < k \cdot g(n)$ $\Leftrightarrow \lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} < \infty$ $f(n) \preceq g(n)$

$f(n) = \Theta(g(n))$ $\exists k_1 k_2 > 0\ \exists n_0$ $\forall n > n_0$ $k_1 \cdot g(n) < f(n)$ $f(n) < k_2 \cdot g(n)$ $\Leftarrow \lim\limits_{n\to\infty} \dfrac{f(n)}{g(n)} = c$ $f(n) \approx g(n)$

```cpp
#include<iostream> //headers
#include<string>
//g++ -Wall -Werror -g --std=c++20 complex_1.cpp
//g++ - compiler; -Wall - show all the errors;
//-Werror - treat warnings as errors
int main()//entry point
{
    std::string input;//variable with type string
    std::cin >> input;//standard input
    int n = input.size();
    int sum = 0;
    for( int i = 0; i < n; i++ )//n
        for( int j = 0; j < 2 * i; j++ )
        //01 0123 012345 0123456 012..2i-1
            sum++;//0 + 2 + 4 + 6 + ... 2(n-1)
    std::cout << sum << std::endl;
    return 0;//code of return
}
```

Классы сложности

Алгоритм со сложностью f(n) называется:

- Если f $= \Theta(\log^k n)$: логарифмическим при k=1, полилогарифмическим при k>1.
- Если f $= \Theta(n)$: линейным.
- Если f $= \Theta(n \log^k n)$: linearithmic при k=1, квазилинейным при k>1.
- Если f $= \Theta(n^k)$: полиномиальным, при k=2 квадратическим.
- Если f $= \Theta(2^{n^k})$: экспоненциальным.

```cpp
#include<iostream>
#include<string>
#include<cmath>
int main()
{
    int n, root;
    std::cin >> n;
    root = std::sqrt( n );
    for( int i=2;i<=root;i++ ){
        if(n%i == 0){
            std::cout << i << std::endl;
            return 0;
        }
    }
    std::cout << n << " is prime" << std::endl;
    return 0;
}
```

```
#include<iostream>
#include<string>
#include<cmath>
int main()
{
    int n, root;
    std::cin >> n;
    root = std::sqrt( n );
    for( int i=2;i<=root;i++ ){
        if(n%i == 0){//2 3 4 5 ... sqrt(n)
            std::cout << i << std::endl;
            return 0;
        }
    }
    std::cout << n << " is prime" << std::endl;
    return 0;
}
```

```cpp
int n, root;
std::cin >> n;
root = std::sqrt( n );
for(int i=2;i<=root;i++)
```

$$f(n) = \Theta(\sqrt{n})$$

```cpp
int n, root;
std::cin >> n;
root = std::sqrt( n );
for(int i=2;i<=root;i++)
```

$$f(n) = \Theta(\sqrt{n})$$

$$n = \Theta(10^{|x|})$$

```
int n, root;
std::cin >> n;
root = std::sqrt( n );
for(int i=2;i<=root;i++)
```

$$f(n) = \Theta(\sqrt{n})$$

$$n = \Theta(10^{|x|})$$

$$f(|x|) = \Theta\left(\sqrt{10^{|x|}}\right)$$

Домашнее задание

- Написать программу реализующую алгоритм Эратосфена.
- Оценить сложность - написать в программе комментариями мысли и результат.
- Сделать pull request в папку lecture_01/homework/ с файлом решения.