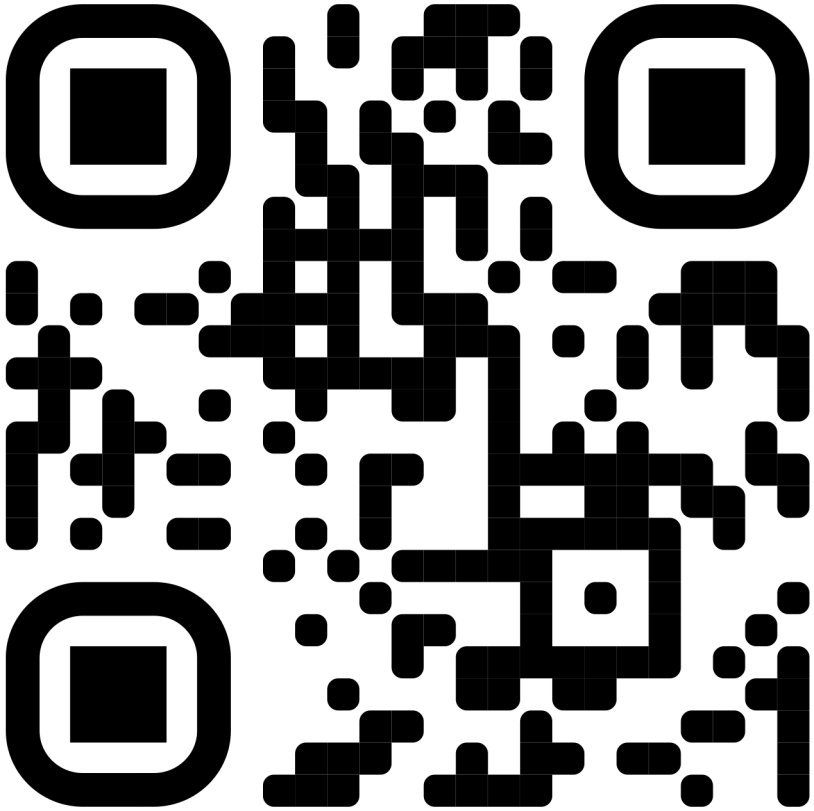
Введение

Алгоритмы
и алгоритмические языки

u.to/72D_Gg



Лекция 1, 05 февраля, 2021

Лектор:

Дмитрий Северов, кафедра информатики 608 КПМ

dseverov@mail.mipt.ru

Обратная связь: **u.to/7Wn7Gg**

Алгоритмы и структуры данных (1/2)

- Часть I Структуры данных
  - О диалектах Си
  - Составные типы данных
  - Абстрактные типы данных

- Часть II Основы теории алгоритмов
  - Анализ эффективности алгоритмов
  - Рекурсия

Алгоритмы и структуры данных (2/2)

- Часть III Основные (базовые) алгоритмы
  - Задачи сортировки
  - Задачи поиска
  - Поиск на графах
- Часть IV Теоретические основы информатики
  - Формализация понятия алгоритм
  - Машина Тьюринга
  - Алгорифмы Маркова

Литература ?

	ISBN: 978-5-8459-	Тысяч страниц	Тысяч рублей
Язык программирования C. Лекции и упражнения	0986-2	1,0	3,0
Язык программирования C++. Лекции и упражнения	2048-5	1,2	3,0
Программирование. Принципы и практика с использованием C++	1949-6	1,3	2,9
Язык программирования C++. Базовый курс	1839-0	1,1	2,2
Алгоритмы. Построение и анализ	2016-4	1,3	3,8
Алгоритмы на C++	2070-6	1,1	2,9

Предусловие – умения…

алгоритмически мыслить

Можно проверить с помощью **http://lightbot.com**

формально излагать решение задачи

Можно проверить с помощью **https://drakon-editor.com**

самостоятельно изучать ЯП

Можно проверить с помощью **http://cs.mipt.ru/c_intro**

Из википедии (1/2)

- **Программи́рование**
  - процесс создания компьютерных программ
- **Компью́терная програ́мма**
  1. как исполняемый код: комбинация компьютерных инструкций и данных, позволяющая аппаратному обеспечению вычислительной системы выполнять вычисления или функции управления (стандарт ISO/IEC/IEEE 24765:2010)[1];
  2. как исходный текст: синтаксическая единица, которая соответствует правилам определённого языка программирования, состоящая из определений и операторов или инструкций, необходимых для определённой функции, задачи или решения проблемы (стандарт ISO/IEC 2382-1:1993)[2].

Из википедии (2/2)

- **Програ́мма** (от греч. προ — пред, греч. γράμμα — запись)

  термин, в переводе означающий «предписание», то есть предварительное описание предстоящих событий или действий. Данное понятие непосредственно связано с понятием алгоритм. …

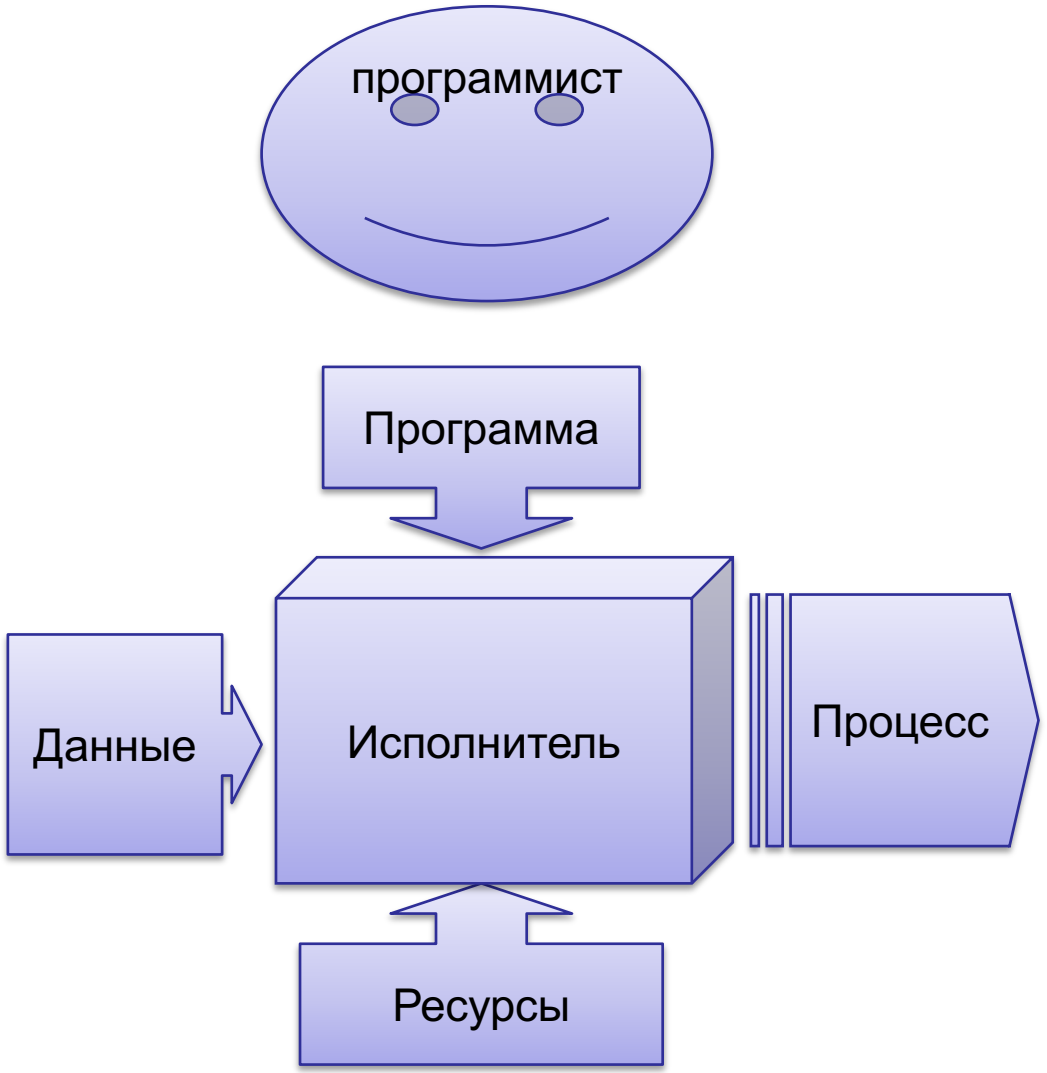
- **Алгори́тм**

  набор инструкций, описывающих порядок действий исполнителя для достижения некоторого результата. <…> Независимые инструкции могут выполняться в произвольном порядке, параллельно, если это позволяют используемые исполнители

- **Инстру́кция** или **опера́тор** (англ. *statement*)

  наименьшая автономная часть языка программирования; команда или набор команд. Программа обычно представляет собой последовательность инструкций.

Понятия программирования



Программа

определённый набор предписаний

Данные

неопределённый набор информации

Процесс

ожидаемый набор **фактических** событий

Ресурсы

- Время работы
  - Программиста
  - Исполнителя
- Энергия
- Пространство

Из определений ГОСТ 33707 (1/3)

- Программа
  - Данные, предназначенные для управления конкретными компонентами системы обработки информации в целях реализации определенного алгоритма.
- Язык программирования
  - Искусственный язык для изложения текстов программ

Из определений ГОСТ 33707 (2/3)

- Алгоритм
  - 2015: Конечное упорядоченное множество точно определенных правил для решения конкретной задачи
  - 1984: Конечный набор предписаний, определяющий решение задачи посредством конечного количества операций
- Алгоритмический язык
  - Искусственный язык, предназначенный для выражения алгоритмов

Из определений ГОСТ 33707 (3/3)

- Язык спецификаций
  - Язык прикладного характера, который,
    1. часто являясь ориентированным на компьютеры синтезом естественного языка и искусственного языка,
    2. используются для выражения требований к системе или компоненте,
    3. для описания их конструкции,
    4. а иногда и протоколов проверки,
    5. для проектирования, исследования и документирования указанных характеристик.

[Около]компьютерные явления

- Модель – упрощённое представление законов поведения частей и отношений системы
- Язык модели - знаковая подсистема фиксации, переработки и передачи информации
- Машина – модельный исполнитель, которому направлены предписания на языке модели

- Рынки
- Потребители
- Задачи

- Алгоритмы
- Программы
- Система/Аппаратура
- Цифровые схемы
- Аналоговые схемы

- Физические явления в [не]линейных структурах

Некоторые [около]компьютерные языки

■ Задачи	■ Языки спецификаций ■ Языки проектирования
■ Алгоритмы	■ Алгоритмические языки
■ Программы	■ Языки программирования
■ Архитектура «системы» ■ Архитектура аппаратуры	■ Языки обращений к «системе» ■ Языки команд аппаратуры
■ Блоки архитектуры ■ Цифровые схемы ■ Аналоговые схемы	■ Языки микрокоманд ■ Языки цифровых схем ■ Языки аналоговых схем
■ Физические явления в [не]линейных структурах	■ Языки физических моделей

Алгоритм интуитивно

Поиск НОД(a,b), где a,b > 0, a < b

- Перебор: 1 – подходит, 2 - ?, 3 - , … a -?
- Алгоритм Эвклида
    1. Разделить первое на второе и получить остаток
    2. Если остаток равен нулю, то второе – результат
    3. Иначе: заменить первое на второе, второе – на остаток и перейти к шагу 1

Свойства алгоритма

Дискретность данных и действий над ними

Понятность: доступность и однозначность правил

Конечность: решение задачи за конечное число шагов

Определённость: воспроизводимость результата

Массовость: применимость к различным данным

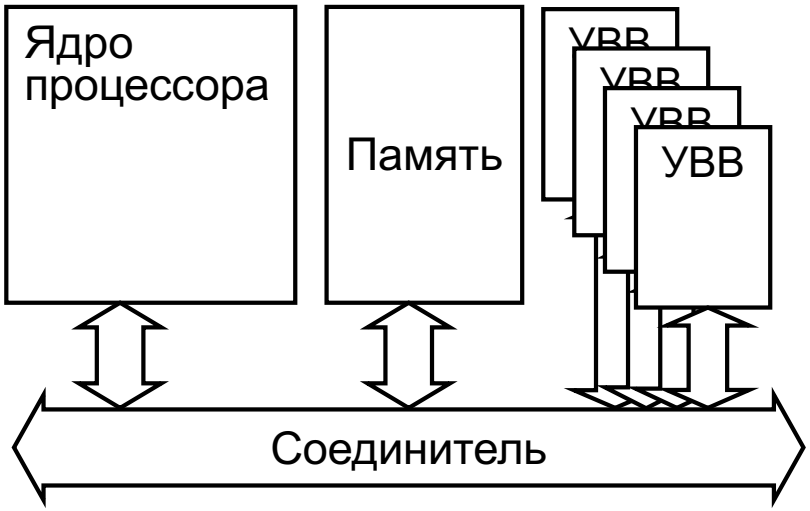
Программа

Реализация алгоритма на языке программирования, определяющем…

1. Действия - операции и операторы
2. Данные - типы и экземпляры
3. Конструкцию сложных данных и действий
4. Взаимодействие со средой

Масштабы событий вашей программы

- **Ваш исполнитель**



- **Ваш процесс**
  - **Выполнение операций**
  - **Изменение состояния**
- Другие процессы

Крупнее:

- Среда разработки
  - Редактирование
  - Трансляция
  - Связывание
  - …
- Среда исполнения
  - Загрузка
  - Выделение ресурсов
  - Нормирование работы и реагирование на запросы
  - Освобождение ресурсов
  - …

Мельче – за рамками семестра

Создание программы

Редактирование

⇓ Текст программы

Трансляция

⇓ Объектный код

Компоновка

⇓ Загрузочный код

Загрузка

⇓ Исполняемый код

Выполнение, отладка

⇓ Результат

Создание программы вместе

Редактирование

⇓ Текст программы ⇐ Ваши изменения вручную

Трансляция

⇓ Объектный код ⇐ Библиотечный исходный текст

Компоновка

⇓ Загрузочный код ⇐ Библиотечный машинный код

Загрузка

⇓ Исполняемый код ⇐ Решения среды исполнения

Выполнение, отладка

⇓ Результат ⇐ Внешние данные, коды, события

Создание программы в окружении

Редактирование	☜ **«Предписания» редактору**
⇓ Текст программы	⇐ Ваши изменения вручную
Трансляция	☜ Предписания трансляции
⇓ Объектный код	⇐ Библиотечный исходный текст
Компоновка	☜ Предписания компоновки
⇓ Загрузочный код	⇐ Библиотечный машинный код
Загрузка	☜ Предписания загрузки
⇓ Исполняемый код	⇐ Решения среды исполнения
Выполнение, отладка	☜ Предписания исполнения
⇓ Результат	⇐ Внешние данные, коды, события

(Само)обман

- Что есть (само)обман ?
  - Брать чужой код: копируя, перенабирая, **подглядывая**, принимая файлы
  - Пересказывать: устное описание кода одним человеком другому
  - Натаскивание: помощь другу в построчном написании заданий
  - Поиск решения в сети
  - Копирование кода предыдущих кусов и экзаменов

- Что НЕ есть (само)обман?
  - Объяснять как использовать инструменты или системы
  - Помогать другим в вопросах конструкции верхнего уровня

Переменная

Именованная область памяти типизованных значений.

1. Имя (адрес начала),
2. Тип (размер и правила операций),
3. Значение (содержимое памяти)

Эволюция базовых типов данных

- Базовые типы данных
  - Целые (char, int)
  - Вещественные (float, double)
  - Никакой (void)
- Модификаторы типа
  - Без/Знаковые (un/signed)
  - Короткие/Длинные (short, long)
- Спецификаторы класса памяти:
  - Статические/Автоматические (static, auto)
  - Регистровые, Внешние (register, extern)
- Квалификаторы изменчивости:
  - Неизменяемые (const),
  - Изменчивые (volatile), исключаемые из оптимизации

Приоритет операций

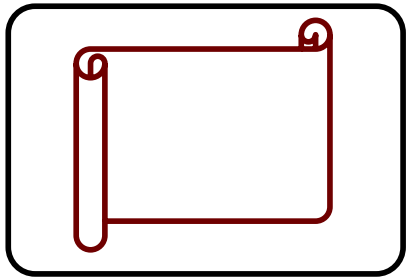
Приоритет и порядок	Обозначение	Назначение
1⇨	::	Разрешения контекста
2⇦	()	Вызов или описание функции
	()	Конструкция значения
	[]	Индекс массива
	->	Косвенная принадлежность
	.	Прямая принадлежность
	++	Инкремент (постфиксная)
	--	Декремент (постфиксная)
3⇨	!	Логическое отрицание
	~	Битовое отрицание (сумма по модулю 2)
	+	Унарный плюс
	-	Унарный минус
	++	Инкремент (префиксная)

Приоритет и порядок	Операция	Назначение
3⇨	--	Декремент (префиксная)
	&	Адрес
	*	Разыменование
	()	Преобразование типа
	sizeof	Размер в байтах
	new	Динамическое выделение памяти
	delete	Динамическое освобождение памяти
4⇨	.*	Разыменование члена
	->*	Косвенное разыменование члена
5⇨	*	Умножение
	/	Деление
	%	Остаток от деления
6⇨	+	Сложение
	-	Вычитание
7⇨	<<	Сдвиг влево
	>>	Сдвиг вправо

Приоритет и порядок	Операция	Назначение
8⇨	<	Меньше
	<=	Меньше или равно
	>=	Больше или равно
	>	Больше
9⇨	==	Равно
	!=	Не равно
10⇨	&	Побитовая конъюнкция
11⇨	^	Побитовое сложение по модулю два
12⇨	|	Побитовая дизъюнкция
13⇨	&&	Конъюнкция
14⇨	||	Дизъюнкция
15⇨	?:	Условное выражение

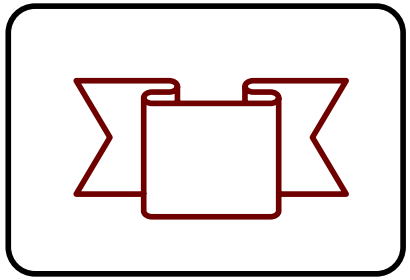
Приоритет и порядок	Операция	Назначение
16⇨	=	Присваивание
	*=	Умножение и присваивание
	/=	Деление и присваивание
	%=	Нахождение остатка и присваивание
	+=	Сложение и присваивание
	-=	Вычитание и присваивание
17⇨	&=	Побитовая конъюнкция и присваивание
	^=	Побитовое сложение по модулю два и присваивание
	|=	Побитовая дизъюнкция и присваивание
	<<=	Сдвиг влево и присваивание
	>>=	Сдвиг вправо и присваивание
18⇨	throw	Генерация исключения
19⇨	,	Объединение двух выражений в одно

Составные типы данных



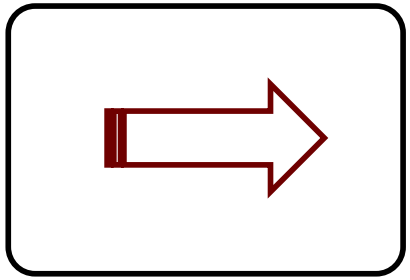
Массивы – наборы нумерованных однородных данных

- Строки



Структуры – наборы именованных разнородных данных

- Объединения
- Перечисления



Указатели – переменные содержащие адрес

- Ссылки

Массивы

- Описание

```
int a[3],b[2][3];
```

- Описание с инициализацией

```
char c[] = {'0','2','4'};
int d[][3] =
 {{1,2,3},{4,5},{6}};
```

- Доступ к элементу

```
a[0] = d[0][1] + 1;
```

Строки

- Одномерные массивы однобайтных целых элементов (символов), содержащие последним элемент со значением ноль.

- ```
  char Str1[] = {'H','e','l','l','o','\0'};
  char Str2[5] = {',',' '};
  char Str3[] = "world";
  ```

- Использование

  ```
  cout << Str1 << Str2 << Str3 << endl;
  ```
  ```

Структуры

- Определение конструкции набора данных:

```
struct My_type {
        type_a Member_1,Member_2;
        type_b Member_3;… } My_object;
```

- Частные случаи:

```
struct Complex { double Re,Im; };
struct Complex A;
struct  { double Re,Im; } B;
```

- Доступ к полям: `B.Re` или `A.Im`

Объединение

совмещение полей — «точек зрения»

`union` *тег* `{` *список_полей* `}` *имя_объекта*`;`

Перечисление

Набор именованных целых констант

`enum` *тег* `{` *список_имен* `}` *имя_объекта*`;`

Битовые поля

Набор целых полей определённой длины

`struct` *тег* `{` *поле*`:`*число_бит*`;` `}` *имя*`;`

`union` *тег* `{` *поле*`:`*число_бит*`;` `}` *имя*`;`

```
#include <iostream>
using namespace std;

enum {Z=0,X,Y=4};
union {
    int A;
    char B[4];
    struct { int a:4,b:4,c:4,d:4,e:4,f:4,g:4,h:4;} X;
    struct { short int L_word, H_word;} W;
} N;

int main() {
    N.B[0]=Y; N.B[1]=Z; N.B[2]=X; N.B[3]=Z;
    cout << N.W.H_word << " " << N.W.L_word << endl;
    cout << N.A << endl;
    cout << N.X.h << N.X.g << N.X.f << N.X.e
         << N.X.d << N.X.c << N.X.b << N.X.a << endl;
    return 0;
}
```

A								
B	B[0]=Y		B[1]=Z		B[2]=X		B[3]=Z	
X	a	b	c	d	e	f	g	h
W	L_word				H_word			

1 4
65540
00010004