Информатика

Лекция 1

Осенний семестр 2024

Организационные вопросы

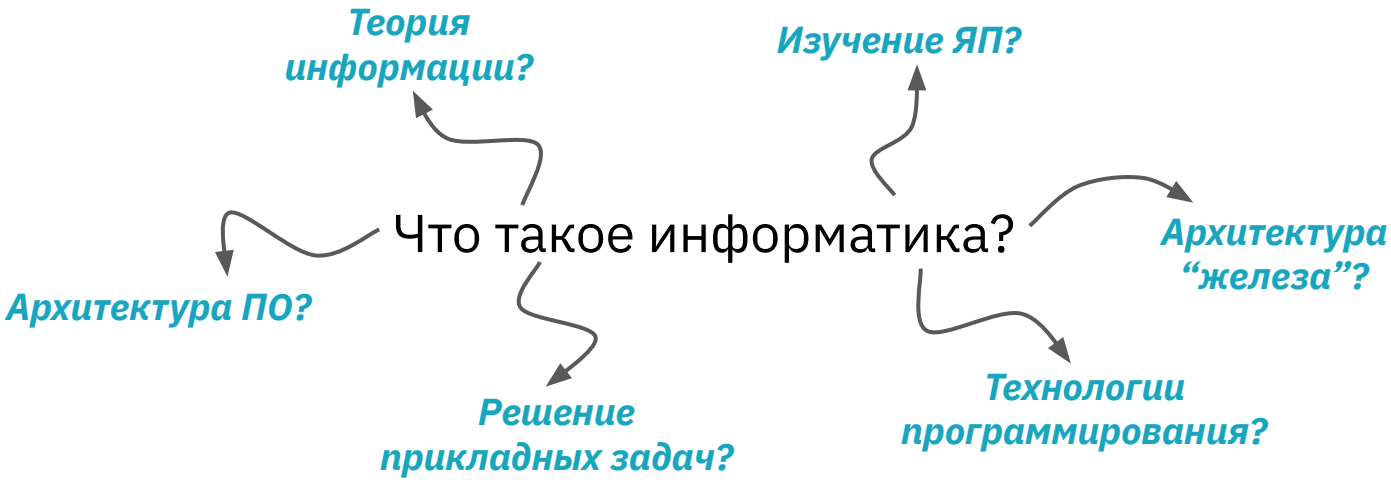
Знакомимся

Гольдштейн Клим Дмитриевич (Клим, ты/вы – как удобно)
ФЭФМ’22, аспирант
н.с. лаборатории компьютерного дизайна материалов
Senior ML-инженер в Huawei
ст. преподаватель ФЭФМ и ФПМИ

По каким вопросам обращаться:

- Вопросы по материалу лекций
- Вопросы по наполнению курса
- Консультация по стажировкам в лабораториях/компаниях
- Проблемы с контестами
- Спорные ситуации по оцениванию

Цель и особенности курса



Наши цели:

1) Обрести практические навыки программирования, необходимые для решения научных и повседневных задач
2) Получить универсальные теоретические знания, достаточные для профессионального общения и дальнейшей специализации в любой из областей IT
3) Приобрести набор умений, достаточный для трудоустройства в топовую IT-компанию или лабораторию, связанную с IT

Почему Python?

Aug 2023	Aug 2022	Change	Programming Language	Ratings
1	1		Python	13.33%
2	2		C	11.41%
3	4	▲	C++	10.63%
4	3	▼	Java	10.33%
5	5		C#	7.04%

1) Простота освоения
2) Актуальность/популярность
3) Наличие большого количества прикладных библиотек

Почему не только Python?

Дает мало контроля пользователю – не все концепции программирования можно изучить

Плох для высоконагруженных вычислений

Диверсификация: разные языки для разного времени и задач

Total					
	Energy		Time		Mb
(c) C	1.00	(c) C	1.00	(c) Pascal	1.00
(c) Rust	1.03	(c) Rust	1.04	(c) Go	1.05
(c) C++	1.34	(c) C++	1.56	(c) C	1.17
(c) Ada	1.70	(c) Ada	1.85	(c) Fortran	1.24
(v) Java	1.98	(v) Java	1.89	(c) C++	1.34
(c) Pascal	2.14	(c) Chapel	2.14	(c) Ada	1.47
(c) Chapel	2.18	(c) Go	2.83	(c) Rust	1.54
(v) Lisp	2.27	(c) Pascal	3.02	(v) Lisp	1.92
(c) Ocaml	2.40	(c) Ocaml	3.09	(c) Haskell	2.45
(c) Fortran	2.52	(v) C#	3.14	(i) PHP	2.57
(c) Swift	2.79	(v) Lisp	3.40	(c) Swift	2.71
(c) Haskell	3.10	(c) Haskell	3.55	(i) Python	2.80
(v) C#	3.14	(c) Swift	4.20	(c) Ocaml	2.82
(c) Go	3.23	(c) Fortran	4.20	(v) C#	2.85
(i) Dart	3.83	(v) F#	6.30	(i) Hack	3.34
(v) F#	4.13	(i) JavaScript	6.52	(v) Racket	3.52
(i) JavaScript	4.45	(i) Dart	6.67	(i) Ruby	3.97
(v) Racket	7.91	(v) Racket	11.27	(c) Chapel	4.00
(i) TypeScript	21.50	(i) Hack	26.99	(v) F#	4.25
(i) Hack	24.02	(i) PHP	27.64	(i) JavaScript	4.59
(i) PHP	29.30	(v) Erlang	36.71	(i) TypeScript	4.69
(v) Erlang	42.23	(i) Jruby	43.44	(v) Java	6.01
(i) Lua	45.98	(i) TypeScript	46.20	(i) Perl	6.62
(i) Jruby	46.54	(i) Ruby	59.34	(i) Lua	6.72
(i) Ruby	69.91	(i) Perl	65.79	(v) Erlang	7.20
(i) Python	75.88	(i) Python	71.90	(i) Dart	8.64
(i) Perl	79.58	(i) Lua	82.91	(i) Jruby	19.84

План на семестр

Первая половина – практика программирования

На лекциях – особенности языка, теория программирования
Примеры заданий на семинарах:

1) Обработка лабораторной работы
2) Построение графиков, работа с БД
3) Визуализация модели Солнечной системы
4) Написание простой игры

Вторая половина – введение в алгоритмы и структуры данных

На лекциях – теория по алгоритмам
На семинарах – решение алгоритмических задач (как теоретических, так и с реализацией в коде), набор задач зависит от уровня группы

Возможные сложности

1) Где искать информацию, если упустил что-то из теории ʕ ͡° ͜ʖ ͡°ʔ
   a) **Just google it.** Лучше на английском. На 95% вопросов есть ответы на stackoverflow, github, habr, в документации языка/библиотек и других подобных ресурсах. Лучше сразу смотреть ответ на вопрос в нескольких источниках.
   b) Основная книга по алгоритмам: Т.Кормен “Алгоритмы: построение и анализ”. Книги по синтаксису языка лучше не читать, а искать статьи по конкретным темам.
2) У меня ошибка в коде, я не понимаю, что не так **(ノಠ益ಠ)ノ彡┻━┻**
   a) **Загуглите** текст ошибки. Переберите все возможные крайние случаи, которые могли возникнуть в вашем коде.
   b) Спросите соседа/одногруппника/однокурсника, чтобы он посмотрел на ваш код свежим взглядом.
   c) Спросите семинариста. Спросите в чате курса.
3) Я ничего не понимаю и не могу написать ни строчки кода **(ಥ_ಥ)**
   a) **Все с этого начинали.** Попробуйте представить, как бы вы решали задачу “вручную” и написать код, соответствующий этим действиям.
   b) Попросите знакомого объяснить вам тему и **фиксируйте логику рассуждений.** Постарайтесь понять, как разбивать задачи на подзадачи и как подходить к решению подзадач и “сборке” итогового решения. **Не пишите то, чего не понимаете.**

Система оценивания

$$G = 0.6*S + 0.4*C + E$$

G – итоговая оценка за семестр

S – накопленная оценка за семестр (до 10)
5 баллов – выполнение лабораторных работ на семинарах
5 баллов – выполнение контестов (домашних заданий)
1 балл – лекционные “разогревочные тесты”

C – оценка за зачет (до 10)
Зачет сдается другому семинарскому преподавателю на зачетной неделе в формате теоретического опроса по программе курса.

E – бонус от семинариста (до 1)

Блокирующие оценки
Для допуска к зачету необходимо набрать минимум 2 балла за лабораторные и 2 балла за контесты.

Списывание жестко наказывается
Контесты после дедлайна будут анализироваться в системе поиска плагиата. При поимке на
1 задаче – аннулируется задача (у обоих).
2 задачах – аннулируется контест.
3 задачах – недопуск к зачету.

Доска позора (тест)

File 1	File 2
./submits_2/dudkina.ea@phystech.edu-119103724/ (98%)	./submits_2/saveleva.ea@phystech.edu-119103691/ (98%)
./submits_2/merkulov.kv@phystech.edu-119091354/ (33%)	./submits_2/shokorov.f@phystech.edu-119091350/ (86%)
./submits_2/milogradov.ma@phystech.edu-119091305/ (34%)	./submits_2/parfentev.ea@phystech.edu-119104346/ (53%)
./submits_2/mikheev-zhdanovich.kiu@phystech.edu-119091334/ (40%)	./submits_2/parfentev.ea@phystech.edu-119104346/ (53%)
./submits_2/mikheev-zhdanovich.kiu@phystech.edu-119091334/ (40%)	./submits_2/milogradov.ma@phystech.edu-119091305/ (34%)
./submits_2/mikheev-zhdanovich.kiu@phystech.edu-119091334/ (40%)	./submits_2/zharkov.iui@phystech.edu-119103334/ (27%)
./submits_2/bessarabov.sd@phystech.edu-119091286/ (20%)	./submits_2/parfentev.ea@phystech.edu-119104346/ (47%)
./submits_2/bessarabov.sd@phystech.edu-119091286/ (20%)	./submits_2/milogradov.ma@phystech.edu-119091305/ (30%)
./submits_2/bessarabov.sd@phystech.edu-119091286/ (20%)	./submits_2/mikheev-zhdanovich.kiu@phystech.edu-119091334/ (35%)
./submits_2/parfentev.ea@phystech.edu-119104346/ (43%)	./submits_2/tretiakov.ev@phystech.edu-119104402/ (93%)
./submits_2/milogradov.ma@phystech.edu-119091305/ (28%)	./submits_2/tretiakov.ev@phystech.edu-119104402/ (93%)
./submits_2/mikheev-zhdanovich.kiu@phystech.edu-119091334/ (33%)	./submits_2/tretiakov.ev@phystech.edu-119104402/ (93%)
./submits_2/anisimov.vs@phystech.edu-119099649/ (91%)	./submits_2/gubarev.gv@phystech.edu-119105228/ (91%)

Использование LLM (ChatGPT/GPT-4, LLaMA 2, Copilot etc.)

Для чего имеет смысл и можно использовать:

1) Получение кратких (но не исчерпывающих) справок по каким-либо темам. Пример: “What’s the difference in memory management in C++ and Python?”
2) Помощь в обработке ошибок. Пример: “I’ve got the “division by zero” error in my Python code. How can I fix it?”

Использовать с осторожностью:

1) Перенос/корректирование кода. С очень высокой вероятностью код будет с ошибками или неэффективный. Чтобы это выявить, нужно уметь разбираться в коде и алгоритмах.
2) Написание кода в малознакомых библиотеках. Например: “Write a code to cut the DataFrame by date in Pandas”

Не использовать:

1) Написание кода с нуля. Сначала научитесь сами, иначе не сможете понять, где можно сделать лучше.
2) Решение алгоритмических и математических задач. Порой искать ошибки в сгенерированных решениях сложнее, чем решить задачу самому.

Общий принцип: **Не пишите то, чего не понимаете.**

Группы и потоки

Перевестись между группами можно по согласованию с отпускающим и принимающим преподавателем, после чего надо также уведомить лектора

Лекция 1

Логические операции. Архитектура ПК.

Историческая справка

1837 – Бэббидж, аналитическая машина, вычисление логарифма
1907 – ламповый триод
1941 – Цузе, Z3, телефонные реле, вычисление квадратного корня, 5Гц
1945 – ENIAC, лампы, вычисление баллистических таблиц, 100 кГц
1947 – биполярный транзистор

1847 – Буль, булева алгебра
1881 – Пирс, стрелка Пирса
1913 – Шеффер, штрих Шеффера (NAND gate)
1927 – Жегалкин, арифметика вычетов по модулю 2, XOR gate

Алгебра логики

Множество значений переменных – 0 (ложь) или 1 (правда)

Три базовых операции:

1) отрицание (¬, ¯)
2) конъюнкция (“И”, *, ∧, &)
3) дизъюнкция (“ИЛИ”, +, ∨, |)

Аксиомы: свойства констант 0 и 1: идемпотентность:	 1+A=1 0*A=0 0+A=A 1*A=A A+A=A A*A=A
Закон исключения третьего: **Закон непротиворечивости:** **Закон отрицания:**	A+ ¬A=1 A* ¬A=0 ¬ (¬A)=A
Законы коммутативности:	A+B=B+A A*B=B*A
Законы ассоциативности:	A+B+C=A+(B+C) A*B*C=A*(B*C)
Законы дистрибутивности:	A*(B+C)=A*B+A*C A+(B*C)=(A+B)*(A+C)
Законы де Моргана:	¬ (A+B)= ¬A* ¬B ¬ (A*B)= ¬A+ ¬B
Законы поглощения:	A+A*B=A A*(A+B)=A

* *На самом деле* раздел, который мы рассматриваем, это лишь двоичная логика. Существуют также многозначные логики (троичная, бесконечнозначная, нечеткая), а также отдельная интуиционистская логика, в которой отсутствует закон исключенного третьего

Таблицы истинности. Прочие операции

Какие еще бывают операции?

1) Импликация, =>, “если A, то B”
2) Эквивалентность, ⇔, “A тогда и только тогда, когда B”
3) Исключающее “ИЛИ”, XOR, ⊕
4) Стрелка Пирса, “ИЛИ-НЕ”, NOR, ↓
5) Штрих Шеффера, “И-НЕ”, NAND, ↑ или |

Таблица истинности показывает результат логической функции в зависимости от значения аргументов

A	B	A∨B	A∧B	A => B	A <=> B	A XOR B	A NOR B	A NAND B
0	0	0	0	1	1	0	1	1
1	0	1	0	0	0	1	0	1
0	1	1	0	1	0	1	0	1
1	1	1	1	1	1	0	0	0

Утверждение: используя только NAND (или NOR), можно построить любые логические операции

$$X \downarrow X \equiv \neg X$$

$$(X \downarrow X) \downarrow (Y \downarrow Y) \equiv X \wedge Y$$

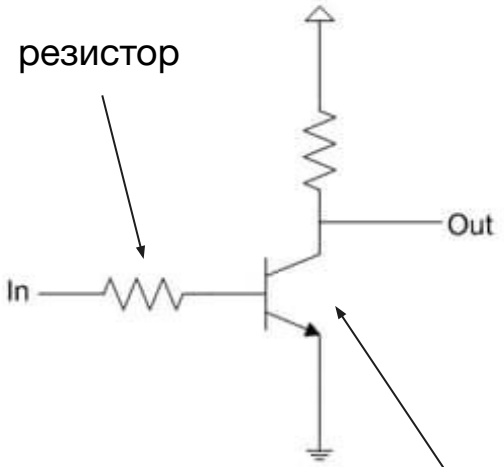
$$(X \downarrow Y) \downarrow (X \downarrow Y) \equiv X \vee Y$$

$$X \mid X = \neg X$$

$$(X \mid X) \mid (Y \mid Y) = X \vee Y$$

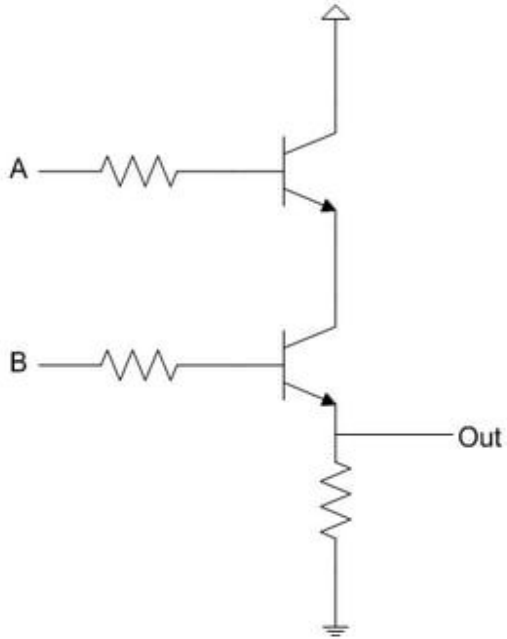
$$(X \mid Y) \mid (X \mid Y) = X \wedge Y$$

Логические вентили

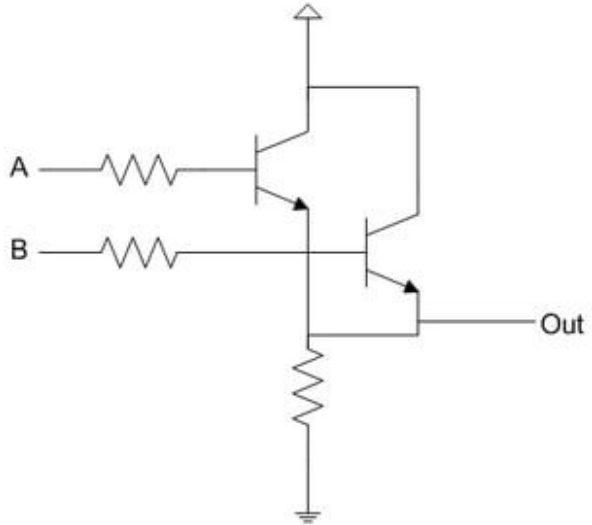


NOT

транзистор

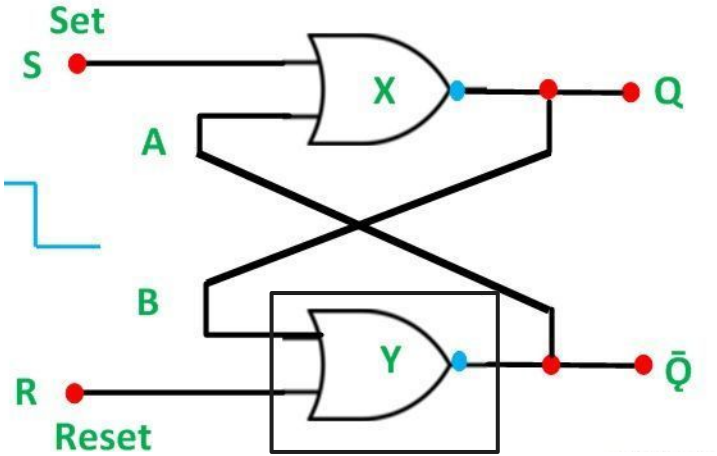


AND



OR

NOR RS-switch

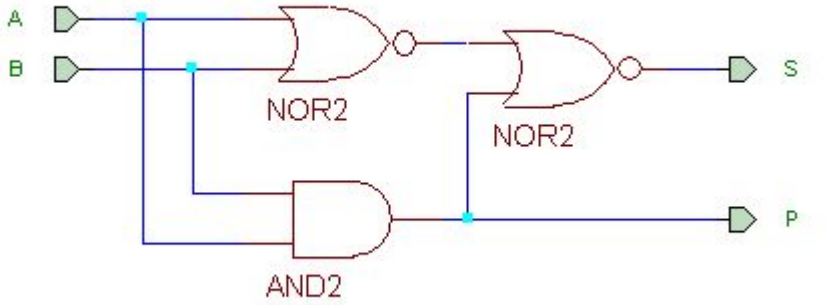


NOR gate

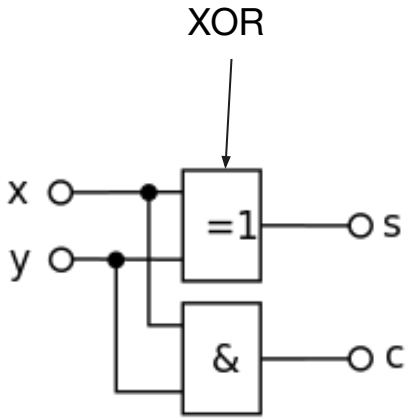
S	R	Q	$\bar{Q}$
0	0	No Change	No Change
0	1	0	1
1	0	1	0
1	1	0	0

Элемент памяти размером 1 бит (хранит 0 или 1)

[полу]Сумматор



На NOR

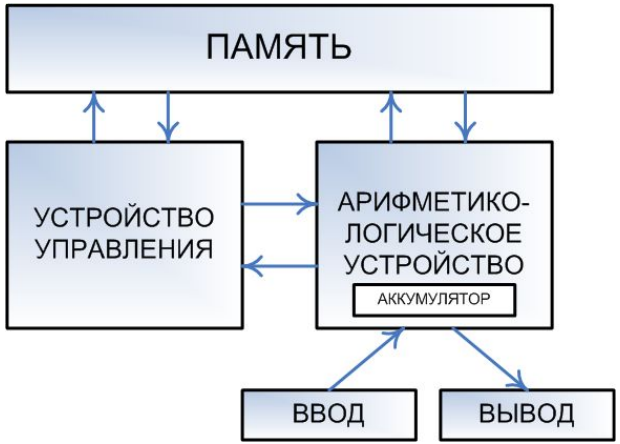


На XOR

$$S = \mathbf{f}(x_1, x_0) = (\overline{x_1} \cdot x_0) \vee (x_1 \cdot \overline{x_0})$$

Архитектура фон Неймана

- Принцип **двоичного кодирования данных**
- Принцип **однородности памяти**
  - Код программы и ее данные хранятся в одной и той же памяти. Поэтому ЭВМ не различает, что хранится в данной ячейке памяти—число, текст или команда. Над командами можно выполнять такие же действия, как и над данными.
- Принцип **адресуемости памяти**
  - Основная память структурно состоит из линейно пронумерованных ячеек; процессору в произвольный момент времени доступна любая ячейка -> Возможность использования переменных.
- Принцип **последовательного программного управления**
  - Предполагает, что программа состоит из набора команд, которые выполняются процессором автоматически друг за другом в определенной последовательности. Для изменения прямолинейного хода необходимы специальные команды перехода.
- Принцип **жесткости архитектуры**
  - Неизменяемость в процессе работы топологии, архитектуры, списка команд.



Альтернативы:

- Гарвардская архитектура
- Нейроморфная архитектура

Интересные ссылки

https://simulator.io/

https://habr.com/ru/articles/755638

https://habr.com/ru/companies/timeweb/articles/653159/

https://habr.com/ru/articles/505360/

https://benchmarksgame-team.pages.debian.net/benchmarksgame/index.html

https://www.sciencedirect.com/science/article/pii/S0167642321000022