Тема 01. Вступ. Основи програмування

Давидов Вячеслав Вадимович

Що таке програмування?

- Програмування – процес створення комп'ютерних програм. Ключовими безпосередніми завданнями програмування є створення та використання алгоритмів та структур даних.
- У більш широкому значенні під програмуванням (розробкою) розуміють весь спектр діяльності, пов'язаний із створенням та підтримкою в робочому стані програм – програмного забезпечення. Ця інженерно-технічна дисципліна називається програмна інженерія. Сюди входять:
  - аналіз та постановка задачі,
  - проектування програми,
  - побудова алгоритмів,
  - розробка структур даних,
  - написання текстів програм,
  - налагодження та тестування програми (випробування програми),
  - документування,
    - Для себе, для розробника з команди
    - Для кінцевого користувача (doxygen, Javadoc)
    - Для Аудіта (метріки)
  - налаштування (конфігурування),
  - доопрацювання та супровід.
- Програмування ґрунтується на використанні мов програмування, на яких записуються інструкції для комп'ютера. Сучасна програма містить безліч таких інструкцій, пов'язаних між собою.

Мова програмування

Мова програмування - формальна знакова система, призначена для запису комп'ютерних програм. Мова програмування визначає набір лексичних, синтаксичних і семантичних правил, визначальних зовнішній вигляд програми та події, які виконає виконавець (зазвичай — ЕОМ) під керівництвом.

З часу створення перших програмованих машин людство придумало понад вісім тисяч мов програмування (включаючи нестандартні, візуальні та езотеричні мови). Щороку їхня кількість збільшується. Деякими мовами вміє користуватися лише невелика кількість їхніх власних розробників, інші стають відомими мільйонам людей.

Як правило, мова програмування існує у вигляді:

- **стандарту мови** - набору специфікацій, що визначають його синтаксис та семантику; стандарт мови може історично розвиватись (наприклад, ANSI C);
- **втілень (реалізацій) стандарту** - власне програмних засобів, які забезпечують роботу згідно з тим чи іншим варіантом стандарту мови; такі програмні засоби розрізняються за виробником, маркою та варіантом (версією), часом випуску, повнотою втілення стандарту, додатковими можливостями; можуть мати певні помилки або особливості втілення, що впливають на практику використання мови або навіть її стандарт. Наприклад, Borland C, MSVC, GCC

Типи мов програмування

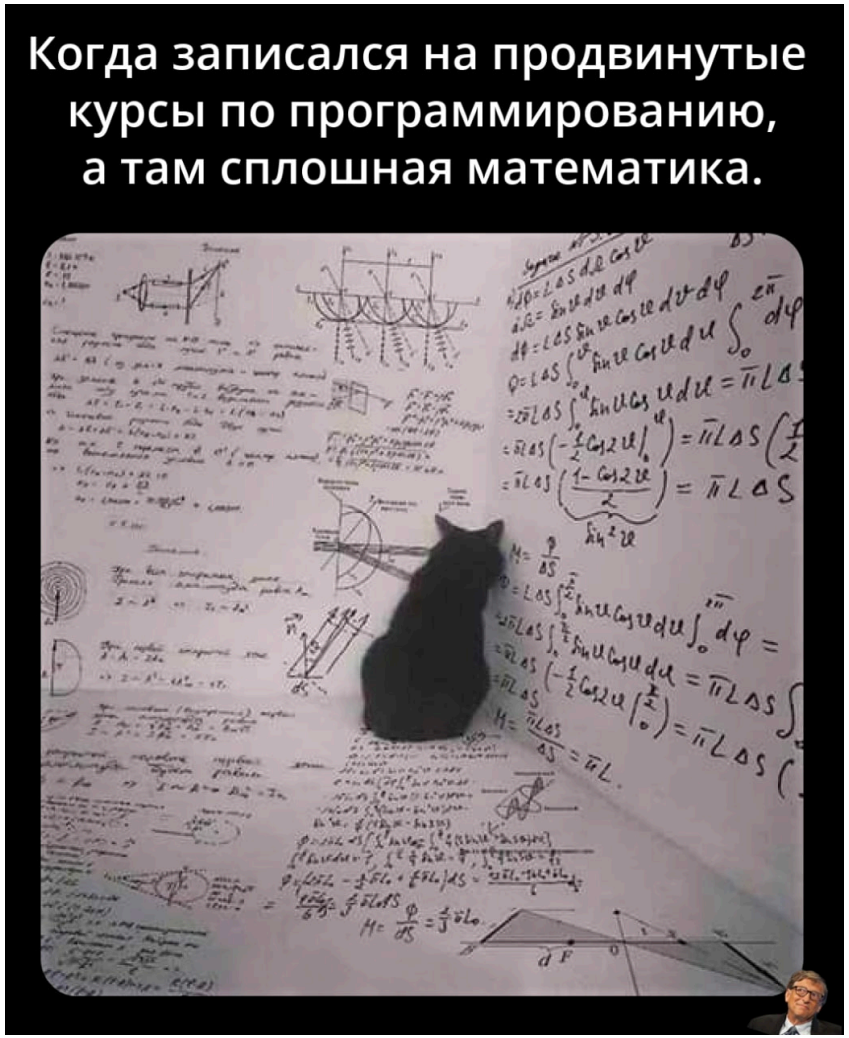
- Процедурні
  - Basic, C, Assembler
- Об'єктно-орієнтовані
  - Java, C++, Delphi, Python, Smalltalk
- Функціональні
  - Haskell, LISP, Prolog, F#, Scala
- Низькорівневі
  - Assembler, machine codes
- Проблемно-орієнтовані
  - TeX, SQL, HTML, Mapple

Чому С?

- Більшість (більша частина) сучасних операційних систем написані на Сі (не Сі++): Unix, Linux, OS X, Android, Windows Phone, Windows
- Сучасні СУБД написані на Сі/Сі++: Oracle, MySQL, MSSQL, PostgreSQL
- Сучасний кінематограф використовує для створення тривимірних фільмів додатки, які здебільшого написані на С та С++. Однією з причин цього є високі вимоги до ефективності та швидкодії таких систем, оскільки вони мають обробляти в одиницю часу величезні обсяги даних.
- Нас оточують безліч звичних приладів та пристроїв, у багатьох з яких також використовується мова С (його урізану версію): електронні будильники, мікрохвильові печі, кавоварки, телевізори, радіоприймачі, системи дистанційного керування тощо.
- Код С виконується дуже швидко і задіює менше пам'яті, ніж у більшості інших мов.
- Програмування на С вважається класичним процедурним програмуванням.
  - Він простіше у розумінні, ніж ЯП інших типів (функціональні, ОО)
  - Він краще описує внутрішню поведінку комп'ютера
  - Після освоєння мови С, будь-яка інша Мова програмування зайде «на ура»

Знання математики як побічний ефект

- **TL DR: Якщо ви плануєте верстати сторінки і використовувати чужі напрацювання не розуміючи їхнього пристрою - вам математика не знадобиться**
- математика - це базис, у якому будується ланцюг алгоритмів, основа будь-якої програми, яку програміст описує. Хороший програміст - розуміє, що робить програма, розбирається в логіці і суті описуваних процесів. Тільки знання математики дозволить написати оптимальну програму.
- математика корисна для розвитку технічного та структурного мислення, необхідного для системного аналізу
- Математика використовується при написанні ігор і, зокрема, при написанні движків до них
- знання математики дозволяє програмісту брати участь у цікавих проектах, створенні чогось нового, а не просто використовувати напрацювання попередників: Data science / Neural networking / Machine Learning / Cryptography / Advanced Game Developing
- Моделювання процесів



Назва символів англійською мовою

@	at	+ -	Plus Minus	‘	Apostrophe, single quote
#	hash	=	equal	! ?	Exclamation mark, question mark
*	Asterisk (not asteriks ;))	:	colon	`	Back tick
%	percent	;	Semi colon	_	Underscore
”	Quote, double quote	> <	Greater than, less than	-, –	Dash, Hyphen
^	caret	,	Coma	~	Tilda
&	Ampersand	.	Dot (email) / point (math) / period (full stop)	/ , \	Back and forward slash
( )	Open/close parenthesis	[ ]	Open / close square bracket	{ }	Open / close curly brace

Трохи філософії

- Знання потрібно красти (с) С.Бенедетти
- Вчитель != Викладач
- У мові С усі типи даних – числа
- Чарівних пігулок не існує. Потрібно працювати над собою для досягнення мети
- Якщо у тебе є яблуко і у мене є яблуко, і ми ними обміняємося – у нас залишиться по яблуку. Якщо у мене є ідея і в тебе є ідея і ми ними обміняємося – у нас буде по дві ідеї
- Програмування схоже на кулінарії. Потрібно вкладати душу. Вкладатиму не тільки у функціональну реалізацію, а й у красу коду, а також його документування
- Один із способів примноження та систематизації власних знань – ділитися ними! Для цього нам допоможе OpenSource, тому я практикую викладання
- Зв'язки – наше все. Якщо щось важко зробити одній людині, це не означає, що це важко зробити іншій.
- Можна бути дуже крутим професіоналом, але якщо про це ніхто не знає, користі мало.
- Особиста мотивація - найкоротший шлях до ефективності.
- Списування – невпевненість у собі
- 1/4 навчання - від вчителя, 1/4 - від своїх знань, 1/4 - від одногрупників, 1/4 - з часом (одержання досвіду)

Загальні моменти з дисципліни

- **Лабораторні роботи: 64 бали**
  - Семестр 1. Обов'язкове виконання всіх робіт.
  - Семестр 2. Можна здавати лише ті/стільки робіт, скільки потрібно для отримання бажаної оцінки
- **Практичні заняття: 16 балів**
- **Модульні контролі: 20 балів.** Необов'язкові до здачі.
  - за списування - оцінка за МК йде в мінус на вагу самої роботи
- **Іспит. Необов'язковий: 15 балів**:
  - Іспит сдається лише 1 раз
  - Лабораторні роботи – допуск до іспиту. Складання іспиту без лаб = 15 балів, що явно не вистачає на 3Е (60 балів)
- **Інше**:
  - Якщо є якийсь питання – не соромтесь задавати, бажано вчасно
  - Якщо відсутні засоби для розробки (комп'ютер/ноутбук) – кафедра дає можливість користуватись їх потужностями

Література

- Для душі:
  - Лінус Торвальдс «Заради задоволення»
  - Записки нареченої програміста
  - Історія одного байта
- Для знань:
  - Початковий рівень:
    - Богатырев "Руководство полного идиота по программированию (на языке Си)»
  - Середній рівень (але для себе слід розуміти, що в рамках 75% курсу ми вивчаємо чисте Сі, і набуті знання треба адаптувати)
    - Dawson Michael "Изучаем C++ через программирование игр"
    - Бєлов Ю. А. Вступ до програмування мовою С++. Організація обчислень: навч. посіб. / Ю. А. Бєлов, Т. О. Карнаух, Ю. В. Коваль, А. Б. Ставровський. – К.: Видавничо-поліграфічнийцентр "Київський університет", 2012. – 175 с.
    - Культин Н.Б. C/C++ в задачах и примерах. - СПб.: БХВ-Петербург, 2005. — 288 с : ил. — ISBN 5-94157-029-5.
    - Златопольский Д. 1400 задач по программированию, 2020
    - Марапулец Ю.В. Язык С++, Основы программирования, 2019
    - Jumping into C++, by Alex Allain - Cprogramming.com
  - Рівень вище за середній :
    - Книги Страуструпа, Кернигана, Ричи, Прата
    - Кнут «Искусство программирования»

Термінал

Саме слово **термінал** походить від дієслова terminate (завершити, покласти кінець) і означає «кінцевий пристрій», тобто пристрій, що знаходиться на одному кінці в процесі комунікації з іншим пристроєм (сервером). Завдання терміналу – відправляти текст на сервер, що вводиться з клавіатури, і відображати на дисплеї текстові відповіді від сервера.

Перші термінали підключалися телефонними лініями до великих комп'ютерів. Вони являли собою електричні друкарські машинки - телетайпи (teletypewriters, TTY). Команди та відповіді сервера телетайпи, що вводяться, рядково друкували на рулоні паперу.

Наразі в операційних системах терміналами називають програмні емулятори TTY. Це програми, які дозволяють вводити символьні команди, відправляти ці команди іншому процесу і відображати на екрані рядки тексту, що надходять від цього процесу.

Термінал

У комп'ютерній термінології консоль означає пристрій із вбудованою клавіатурою та монітором. Таким чином, консоль – це пристрій, а термінал – це програма комунікації всередині консолі.

У разі персонального комп'ютера терміни "консоль" та "термінал" можна вважати синонімами.

Самі команди, що надходять від терміналу, виконуються спеціальною програмою, яка називається командною оболонкою (command shell). Залежно від отриманої команди оболонка виконує певні дії та генерує символьні рядки, які посилаються назад терміналу для відображення на екрані.

Для кожної операційної системи є різні оболонки, що відрізняються набором команд. У Linux і Mac OS найчастіше користуються оболонками bash, zsh, fish, tsh. До складу Windows входять дві стандартні оболонки: командний рядок cmd та PowerShell.

Важливо розуміти, що оболонки не мають власного інтерфейсу користувача, це не термінали. З тією ж командною оболонкою можна працювати з допомогою різних терміналів, а одному терміналі можна запускати різні оболонки.

Linux howto

- VirtualBox
  - https://www.osboxes.org
- MacOS (no way to use VirtualBox on m1 chip; Parallels)
- Native Linux
- Linux as secondary OS
- Windows Linux Subsystem (win10 only)
  - https://docs.microsoft.com/en-us/windows/wsl/install-win10
  - https://winitpro.ru/index.php/2020/07/13/zapusk-linux-v-windows-wsl-2/
- Windows + mingw / cygwin
- https://labs.play-with-docker.com

Який Linux Використовувати

- Відмінності Linux систем
  - Пакетний менеджер (apt, yum, apk)
    - На продуктивність ніяк не впливає
  - Графічна оболонка за умовчуванням (KDE, Gnome, Mate, XCFE)
    - KDE - найважча, XCFE – найлегша оболонка
  - Спеціалізація ОС:
    - Серверна (Debian, Gentoo/Arch)
    - Десктопна (Ubuntu)
    - Для експериментів, embedded (Alpine)
    - Для тестування безпеки (Kali)

Командна строка Linux

- sudo
- Менеджер пакетів apt (Debian-based):
  - sudo apt-get update
  - sudo apt-get install <package-name>
- Навігація по директорії:
  - Absolute navigation:
    - cd /bin
    - cd ~
  - Relative navigation:
    - cd lab01
    - cd ..
    - cd ../lab01/bin
    - cd <<tab>><<tab>>
  - Execution:
    - ./main.bin
  - <<Up>>/<<down>>

Командна строка Linux

- Створення директорії
  - mkdir programming-labs
  - mkdir “Лаби з програмування”
    - `mkdir Лаби з програмування` создаст 3 папки
- Видалення директорії/файлу
  - rm file.txt
  - rm –rf dir
- Перегляд вмісту каталогу
  - ls
  - ls -la
  - ls -la output
  - tree

І ще немного про Linux

- Редактор:
  - Рекомендації:
    - Beginner Level: nano
    - Advanced level: vim
  - В принципі, можна користуватися будь-якими графічними засобами розробки (IDE), але в цьому випадку не «діставати» викладачів щодо особливостей їх використання
- Якщо ви не плануєте переходити на лінукс - не варто завантажувати важку гарну версію ☺

Робота з git

- Система контролю за версіями. Цілі:
  - Бекап коду
  - Шаринг коду з іншими (зокрема, викладачем)
  - Дивитись зміни між поточним та останнім «стабільним» станом з можливістю відкочування
- Documentation: https://git-scm.com/book/ru/v2
- Recommend git service: giltab.com . Необхідно в цій системі створити обліковий запис. Після цього буде доступне створення репозиторіїв (бажано приватних)
- Клонування репозиторію
  - git clone <url> [destination]
  - git clone https://github.com/davydov-vyacheslav/sample_project.git lab01
- Налаштування інформації про користувача:
  - git config --global user.name "Name Surname"
  - git config --global user.email "mailbox@example.com"
- Перегляд налаштувань
  - git config --list
  - git config --global --list

Робота з git. Basic flow

- $ cd ~/dev/project_linear/
- $ git status
- ... Тут ми бачимо, що жоден файл не проіндексован ....
- $ git add Makefile
- $ git add src/main.c
- $ git status ...
  new file: Makefile
  new file: src/main.c
- $ git commit -m "Lab03 initial implementation”
- $ git status
  ... Тут ми бачимо, що жоден файл вже не проіндексован ... ... **(**усі файли закомічені **)**
- $ git push origin master
- **TODO: advanced: .gitignore**

Робота з git

- https://github.com/davydov-vyacheslav/sample_project
  - github.com – тип git провайдера (github / gitlab / gitea / bitbucket)
  - Davydov-vyacheslav - ім’я користувача (його логін) або організації
  - Sample_project – назва проекту

Отладка (Debug). lldb

- **Мета** – побачити, що відбувається «всередині» вашої програми, поки вона виконується, або що робила програма в момент перед аварійним (?) завершенням. Побічна мета – побачити значення змінних перед завершенням, тому що у дисципліні Програмування виведення на екран буде «не скоро»
- Можливості відладчика:
  - Зупинити програму за певних умов або на певному етапі
  - Визначити стан програми / значення змінних у заданий момент роботи програми
  - Модифікація змінних у режимі реального часу (без зміни у коді з наступною перекомпіляцією та перезапуском)
  - Покрокове виконання програми з метою визначення алгоритму (поточної послідовності дій) роботи програми

Debug sample

```
% clang -g test.c -o test.bin
% lldb test.bin
(lldb) l
   1       int main() {
   2                   int a = 5 + 10;
   3
   4                   return 0;
(lldb) b 4
Breakpoint 1: where = test.bin`main + 20 at test.c:4:2, address = 0x0000000100003fac
(lldb) r
* thread #1, queue =`main-thread', stop reason = breakpoint 1.1
    frame #0: 0x0000000100003fac test.bin`main at test.c:4:2
   1       int main() {
   2                   int a = 5 + 10;
   3
-> 4                   return 0;
   5       }
(lldb) p a
(int) $0 = 15
```

Складання проекту. Компіляція

- sudo apt install **clang**
- clang -g src/main.c -o ./dist/main.bin
- Опції компіляції:
  - Режими компіляції
    - ”Debug” (з налагоджувальною інформацією) (**-g**)
    - “Release” (без налагоджувальної інформації)
  - other options:
    - -Wall -Wextra -Werror -Wformat-security -Wfloat-equal
    - -Wshadow -Wconversion -Wlogical-not-parentheses -Wnull-dereference
- Запуск: ./dist/main.bin

Складання проекту. Makefile

- sudo apt install make

Makefile:

```
clean:
	rm -rf dist
prep:
	mkdir dist
compile:
	gcc -g src/main.c -o ./dist/main.bin
run:
	./dist/main.bin
all: clean prep compile run
```

Execute:

```
make clean prep compile run
make all
```