Доклад

По:

Разработка на софтуер

Тема:

Функционално Програмиране

Денислав Димитров, 12А Клас

25.09.22

Парадигма на програмиране:

Разработката на софтуер бизнес процеса на писане на [програмен код](https://bg.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B5%D0%BD_%D0%BA%D0%BE%D0%B4), неговата поддръжка, но в по-широк смисъл включва всичко, което стои между първоначалната концепция за определена програма или желан софтуерен продукт до релийза, понякога това става като планиран процес на разработка. Разработката на софтуер може да включва [изследвания](https://bg.wikipedia.org/wiki/%D0%98%D0%B7%D1%81%D0%BB%D0%B5%D0%B4%D0%B2%D0%B0%D0%BD%D0%B8%D1%8F), нови разработки, създаване на прототипи, модификация, повторно използване, ре-инженеринг, поддръжка и всякакви други дейности, чийто краен резултат е софтуерен продукт.

При разработката на програмния продукт програмираното му е съобразено с нуждите на дадена целева потребителска група, която се интересува от създаването или подобряването на такъв софтуерен продукт и съответства на маркетинга му. Но [софтуерът](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80) може да бъде разработван и по множество причини, като най-общо трите са да отговаря на конкретните нужди на потребителите, на дадена фирма или клиент (на [английски](https://bg.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8_%D0%B5%D0%B7%D0%B8%D0%BA): custom software или bespoke software), да отговаря на възприетите нужди на група от потенциални потребители (често [софтуер с отворен код](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80_%D1%81_%D0%BE%D1%82%D0%B2%D0%BE%D1%80%D0%B5%D0%BD_%D0%BA%D0%BE%D0%B4) или рекламен) или дори за лична употреба (например напредналият програмист или изследовател-учен може да напише програма за автоматизиране на сложни задачи). Понякога се налага разработката и на вграден софтуер (асемблиране), например когато се изисква процесът на разработка да бъде интегриран с разработката на контролиран физически продукт. [Системният софтуер](https://bg.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B5%D0%BD_%D1%81%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80) засяга приложните програми и самия процес на [програмиране](https://bg.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5), поради което често се разработва отделно.

Нуждата от по-добро качество на процеса на софтуерна разработка води до началото на [софтуерното инженерство](https://bg.wikipedia.org/wiki/%D0%A1%D0%BE%D1%84%D1%82%D1%83%D0%B5%D1%80%D0%BD%D0%BE_%D0%B8%D0%BD%D0%B6%D0%B5%D0%BD%D0%B5%D1%80%D1%81%D1%82%D0%B2%D0%BE), което се стреми да приложи систематичния подход, илюстриран в инженерната парадигма, към разработката на софтуер.

Съществуват много подходи към управлението на софтуерни проекти, известни като циклични модели в живота на софтуерната разработка, методологии, процеси или модели. „Моделът на водопада“ на [английски](https://bg.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8_%D0%B5%D0%B7%D0%B8%D0%BA): waterfall е традиционният подход, а т.нар. [гъвкава методология](https://bg.wikipedia.org/wiki/%D0%93%D1%8A%D0%B2%D0%BA%D0%B0%D0%B2%D0%B0_%D0%BC%D0%B5%D1%82%D0%BE%D0%B4%D0%BE%D0%BB%D0%BE%D0%B3%D0%B8%D1%8F) на [английски](https://bg.wikipedia.org/wiki/%D0%90%D0%BD%D0%B3%D0%BB%D0%B8%D0%B9%D1%81%D0%BA%D0%B8_%D0%B5%D0%B7%D0%B8%D0%BA): agile разработка на софтуер е по-модерен подход.

Функционални езици:

Парадигмата на програмиране, парадигма за програмиране или програмна [парадигма](https://bg.wikipedia.org/wiki/%D0%9F%D0%B0%D1%80%D0%B0%D0%B4%D0%B8%D0%B3%D0%BC%D0%B0) представлява фундаменталния стил на [програмиране](https://bg.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5). Има множество програмни парадигми, но основните сред тях са: [обектно ориентирано](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE_%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5), [императивно](https://bg.wikipedia.org/wiki/%D0%98%D0%BC%D0%BF%D0%B5%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5), [функционално](https://bg.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) и [декларативно](https://bg.wikipedia.org/w/index.php?title=%D0%94%D0%B5%D0%BA%D0%BB%D0%B0%D1%80%D0%B0%D1%82%D0%B8%D0%B2%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5&action=edit&redlink=1) .

Парадигмата на програмиране предоставя (и определя) начина, по който програмистът гледа на изпълнението на програмата. Например, в обектно ориентираното програмиране програмистът може да разглежда програмата като съвкупност от взаимодействащи си обекти, докато при функционалното програмиране програмата може да бъде разглеждана като последователност от изчисления на функции, които нямат свои състояния.

Различните [програмни езици](https://bg.wikipedia.org/wiki/%D0%95%D0%B7%D0%B8%D0%BA_%D0%B7%D0%B0_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) препоръчват различни парадигми на програмиране. Някои езици са проектирани да поддържат някоя определена парадигма на програмиране – [SmallTalk](https://bg.wikipedia.org/w/index.php?title=SmallTalk&action=edit&redlink=1) и [Java](https://bg.wikipedia.org/wiki/Java) поддържат [обектно ориентираното програмиране](https://bg.wikipedia.org/wiki/%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE_%D0%BE%D1%80%D0%B8%D0%B5%D0%BD%D1%82%D0%B8%D1%80%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5), докато [Lisp](https://bg.wikipedia.org/wiki/Lisp) поддържа [функционалното програмиране](https://bg.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5). Някои езици за програмиране поддържат повече от една парадигма – [C++](https://bg.wikipedia.org/wiki/C_plus_plus) поддържа и обектно ориентирано програмиране, и [процедурно програмиране](https://bg.wikipedia.org/w/index.php?title=%D0%9F%D1%80%D0%BE%D1%86%D0%B5%D0%B4%D1%83%D1%80%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5&action=edit&redlink=1).

Много парадигми на програмиране са познати повече с това какво забраняват, отколкото с това какво препоръчват. Например функционалното програмиране забранява използването на странични ефекти, [структурното програмиране](https://bg.wikipedia.org/wiki/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5) забранява употребата на goto. Заради това новите парадигми обикновено се приемат като твърде сурови и ограничаващи от програмистите, свикнали вече с някакъв предишен стил на програмиране. Но трябва да се отбележи, че избягването на определени техники може да подпомогне доказването на коректността на програмата или просто да улесни разбирането на поведението ѝ, като при това не ограничава приложимостта на програмния език.

Отношението между парадигмите на програмиране и програмните езици може да бъде доста сложно, тъй като един език може да поддържа повече от една парадигма. Например [C++](https://bg.wikipedia.org/wiki/C_plus_plus) е проектиран да поддържа елементи от процедурното програмиране, обектно ориентираното програмиране, [обектно базираното програмиране](https://bg.wikipedia.org/w/index.php?title=%D0%9E%D0%B1%D0%B5%D0%BA%D1%82%D0%BD%D0%BE_%D0%B1%D0%B0%D0%B7%D0%B8%D1%80%D0%B0%D0%BD%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5&action=edit&redlink=1) и [родовото програмиране](https://bg.wikipedia.org/w/index.php?title=%D0%A0%D0%BE%D0%B4%D0%BE%D0%B2%D0%BE_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%B8%D1%80%D0%B0%D0%BD%D0%B5&action=edit&redlink=1). Един C++ програмист може да напише чисто процедурна програма, или чисто обектно ориентирана програма, а може да напише и програма, която обединява елементи и от двете парадигми.

Състояние на програма:

Думата за състояние на програмата е регистър, който изпълнява функцията на регистър на състоянието и програмен брояч, а понякога и повече. Терминът се прилага и за копие на PSW на склад. Тази статия обсъжда само PSW в IBM System/360 и неговите наследници и следва конвенцията на IBM за номериране на битове, започващи с 0 като най-левия (най-значимия) бит.

Въпреки че някои полета в рамките на PSW могат да бъдат тествани или зададени чрез използване на непривилегировани инструкции, тестването или настройката на останалите полета може да бъде извършено само чрез използване на привилегировани инструкции.

В рамките на PSW се съдържа двубитов код на условие, представляващ нула, положителен, отрицателен, препълващ и подобни флагове на регистрите за състояние на други архитектури. Инструкциите за условно разклоняване тестват това, кодирано като четирибитова стойност, като всеки бит представлява тест на една от четирите стойности на кода на условието, 23 + 22 + 21 + 20. (Тъй като IBM използва битово номериране с голям порядък, стойността на маската 8 избира кода 0, стойност на маска 4 избира код 1, стойност на маска 2 избира код 2, а стойност на маска 1 избира код 3.)

Функций:

Една функция е просто „парче“ код, който можете да използвате отново и отново, вместо да го пишете многократно. Функциите позволяват на програмистите да разбият или разложат даден проблем на по-малки части, всяка от които изпълнява определена задача. Веднъж създадена функция, подробностите за това как тя работи могат почти да бъдат забравени. По този начин детайлите се абстрахират, което позволява на програмиста да се съсредоточи върху по-голямата картина.