1, 2, 3, кодирја! – Научно објашњење - Алгоритми, језици и програми

Неодређеност глагола « кодирати »

Глагол « кодирати » се веома често користи (можемо га наћи у уводу за предмет информатика у школи у форми « читати, писати, рачунати и кодирати ». Значи ова реч може да има више значења па ћемо их у наставку продискутовати.
Прво од ових значења је најлакше за разјашњење. Реч « кодирати » можемо да употребимо за опис радње током које смо садржај неке поруке учинили нечитљивом за некога ко не зна « скривени код ». Ову активност ћемо, у наставку, означити као « **шифрирање** », односно активност « шифрирања » поруке да би она постала нечитљива оном ко не познаје « шифру. У најопштијем случају реч « кодирати » означава активност током које се дају инструкције машини, односно компјутеру. Ова активност се означава и термином « **програмирање** ». Машина интерпретира и извршава те инструкције дајући неки резултат.
Коначно, кодирање може бити употребљено при **представљању** неке информације помоћу симбола (на пример, писање текста у бинарном коду, помоћу 0 и 1). У овом педагошком приручнику ће, за термини кодирање или декодирање, бити коришћен глагол « кодирати », иако се код неких корисника « кодирати » чешће користи за « програмирати ».

Реч « кодирати » има и друго значење, које није у складу с њеном дефиницијом него се чешће везује за идеју о информатици посредством активности програмирања. Иако се информатика не своди само на програмирање, ипак је та активности највидљивија. Управо то, у свакодневном животу, и даје неки значај програмерима, јер они омогућују функционисање машина, за разлику од оних који их само користе. Карикатурално је и виђење програмера који веома брзо повезује линије кода (lines of code-LOC). Помињемо и слику екрана на ком, током репортаже о неком информатичару или неком документу, у позадини дефилују читаве странице кода (вероватно с циљем да се импресионирају гледаоци и истакне специјалност). Оваквом сликом се доприноси ширењу идеје о инфорамтици као тешкој науци доступној само за елиту или оне који су за то предодређени. На срећу, то није тако. У сваком случају, кодирање је део информатике, а то није и једина њена активност. Кодирање омогућује да конкретизујемо своје идеје. Ипак, ове идеје могу да се створе и нестану коришћењем резоновања као главног покретача, без икаквог позивања на програмирање. Ово резоновање, придружено механичкој природи компјутера и њиховом функционисању, чини **информатичку мисао** *(универзални применљиви скуп ставова и знања које усвајамо и којим знамо да управљамо, п.п.)* и омогућује реализацију алгоритама који данас покрећу велики део наших активности.

Шта је алгоритам?

Свеприсутност алгоритама је очигледна. Наш захтев претраживачу на Гуглу, за тражење одговрајућег текста, покреће више алгоритама истовремено. Када на берзама видимо хиљаде операција у секунди, вероватно нисмо ни свесни да је у питању аутоматско (алгоритамско) трговање високе фреквенце. Када нека влада жели да анализира велику базу података у вези људских активности, с циљем да добије неке информације, позива у помоћ алгоритме. Дакле, алгоритими имају важну улогу у нашем животу, били ми тога свесни или не.

Пример кухињског рецепта је веома добар као илустрација појма алгоритма. Дакле, у питању је серија инструкција које је потребно реализовати да би се добио жељени резултат. Штавише, аналогија између кухињског рецепта и алгоритма може бити и шира. Рецепт може да буде доступан сваком заитересованом али може и да се чува као тајна. Може бити копиран или разглашен причом. Ипак, он остаје на нивоу концепт, све док се не реализује посредством активности у кухињи. Исто тако се и алгоритам у информатици посматра као концепта. Он може бит написна и ручно, на мање или више формалан начин као мешавина објашњења и математичких симбола. Док год је у тој форми, без примене, знамо да неће дати никакав резултат. Моћ неког алгоритма се манифестује интерпретацијом и конкретним људским или машинским деловањем.

Брзина реализације неког алгоритма је нешто сасвим друго... Човек може да интерпретира и примени неки једноставан алгоритам следећи његове етапе и бележећи међурезултате ако их има. Ученици, на пример, науче алгоритам дељења а затим га систематски примењују. Могу да обаве било које дељење строгом применом етапа алгоритма. Бележе међувредности и поново започињу операције све док се не оствари услов за прекид прорачуна. Све ово, ипак, постаје знатно деликатније код компликованог алгоритма а чак и невидљиво код већине алгоритама који се свакодневно реализују око нас. Машине реализују алгоритме све већом брзином, што је остваривo обједињавањем бар два следећа услова:

1. Дизајнирање и писање алгоритма који омогућује остварење предвиђеног резултата.
2. Превођење тог алгоритма у програм који машина може да реализује.

Ове две активности су сасвим сигурно основа производње нумеричких објеката. Нивои квалитета неког програма, зависно од искуства прогрмера и познавања коришћеног језика, могу бити веома различити. Неки програм може бити написан на више начина иако се користи исти алгоритам. Неки програмери толико добро познају свој најомиљенији језик да су у стању да напишу програме који су вишеструко бржи и остварују изузетне перформансе у односу на програме које су писали програмери почетници. Међутим, то ипак није увек и довољно да се добије ефикасан програм.

Алгоритми могу бити написани на више начина, јер могу бити и дизајнирани на више начина У наставку ћемо видети да лош дизајн алгоритма није могуће надоместити ефикасним програмирањем, или компјутером високих перформанси. Исто тако, добро замишљен алгоритам, чак и када га програмира почетник на старом компјутеру, може понудити изузетно добре перформансе. Дакле, информатика се, пре свега осталог, заснива на коришћењу одговарајућег метода и резоновања које омогућује продукцију што је могуће ефикаснијег алгоритма. Овим проблемом се бави научна дисциплина позната као алгоритмика. Упознаћемо неколико основних елемената те дисциплине кроз кратак приказ историје информатике.

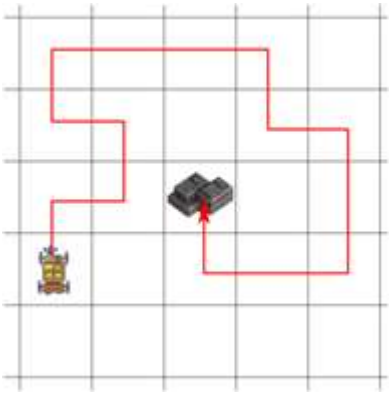
Језици

Програмирање првих компјутера је остваривано посредством перфорираних картица. Чак их је и Бебиџ предвидео за своју аналитичку машину. Грејс Хопер је, желећи да учини програмирање доступним, схватила ограничене могућности перфорираних картица као интерфејса између машине и оних који је програмирају. Превазилажење ових ограничења остварила је креацијом компилатора. Реч је о могућности исказивања програм у језику који је близак људском. Све потешкоће су сад биле сведене на превођење таквог програма на машински језик, што је познато као компилација коју реализује компилатор (програмски преводилац). Компилација неког програма подразумева суочавање са низом изазова попут следећих:

- Препознавање и раздвајање речи и елемената језика да би се направили ентитети с којим ће се радити у наставку. На пример, линија кода ***резултат=a + 3*** омогућује препознавање варијабле ***резултат,*** оператора =, варијабле ***a,*** оператора + и броја ***3.***
- Откривање некохерентности у програму и увереност у коректност програма по питању синтаксе или граматике (у програму је потребно проверити поштовање правила језика у ком је написан).
- Откривање семантичких некохерентности. На пример, ако две варијабле имају исто име онда није могућ превод програма на машински језик (постоји неодређеност која онемогућује компилацију).
- Креирање кода, у машинском језику, помоћу елемената формираних у претходним етапама.

Компилатор има изузетно место у историји информатике. Јер, у програму је знатно лакше користити речи, бројеве, операторе него их приказивати на перфорираним картицама. Данс се та ситауција огледа у присутности мноштва програмских језика међу којим је и *Скрач* (*Scratch),* језик који смо одабрали за увођење ученика у програмирање. Разлике између језика могу бити врло велике. У педагошком пројекту за ученике виших разреда основне школе (циклус 3 у Француској) илустровали смо диверзитет језика који се могу користити за управљање роботом (час 3.1.1) приказујући пример аутоцентричног и алоцентричног језика. (Сајт Pixees вам предлаже « jeu du robot »(сајт је на француском језику *n.n*).

Програмски језици имају екстремно ограничен и врло прецизан речник, без икаквих неодређености. Робот у овом примеру може да буде вођен с четири алоцентричне (апсолутне) инструкције: ***Север*** (што значи « иди једна корак ка Северу »), ***Југу, Истоку, Западу***. Робот, сасвим сигурно, неће извршити инструкцију « избегни препреку », која је нејасна и није директна. У аутоцентричном (релативном) језику роботу се издају три инструкције: ***иди*** (иди један корак напред), ***десно*** (значи « окрени се за 90° на десно»), ***лево***. Последњу инструкција ***лево*** можемо поједноставити са ***десно-десно-десно***, чиме смо коришћени речник свели на само две речи. Могуће је прећи из једног у други језик али та вежба није једноставна па тиме на врло јасан начин илуструјемо важност улоге компилатора.!



Који језик употребити за управљање роботом? (део приказа час 3.1.1)

Пошто укажемо на важност компилатора а тиме и програмских језика, преостаје нам само да пређемо на програмирање! Ипак, пре преласка на програмирање било би веома корисно да се размотри појам алгоритма (а компилатор би затим превео тај програм на машински). Погледајмо како се алгоритми могу направити, од чега су сачињени и нарочито... како то свако до нас може урадити!