САДРЖАЈ

1. АЛГОРИТАМ

Алгоритам представља скуп акција са дефинисаним редоследом њиховог обављања , који примењен на показани скуп података , доводи до тражених резултата.

У процесу програмирања, скуп акција дефинисан је могућностима рачунара , односно наредбама програмског језика који се користи , док се редослед извршавања акција задаје помогу алгоритамских (програмских) структура.

Алгоритам је скуп правила или правило са своством прецизношћу, једнозначност те обухвата коначан број корака , а сваки корак је описан инструкцијом . Инструкције морају бити изводљиве и једнозначне . Алгоритам описује решавање неког проблема.

Поступак обављања алгоритма је алгоритамски процес. Алгоритам има дефинисане почетне објекте над којима се обављају операције, а исход тога је скуп резултата тј. завршних објеката и он је делотворан.

Да би алгоритам био учинковит резултат се мора добити у прихватљивом или разумном времену. Инструкције се могу извршити неколико пута , те инструкције морају показивати понављање , али за било коју вредност улазних података алгоритам завршава након коначног броја понављања.

Код записивања алгоритма употребљава се програмски језик, реч је о недовршеном коду где су неки низови наредби замењени текстом. Анализа алгоритма подразумева процену времена за извршавање тог алгоритма, а време се поистувећује са бројем операција које одговарајући програм треба обавити и он се изражава као функција.

2. КОНСТРУИСАЊЕ АЛГОРИТМА

Конструисање алгоритма је стваралачки посао.

Ако задатак није сасвим једноставан, а одговарајући алгоритам није тривијалан, онда се при конструисању алгоритма суочавамо са следећим тешкоћама :

1. Где наћи идеје за решавање задатка ?

2. Како конструисати алгоритам, а да у њему не буде грешака ?

Најчешћи извор идеја - наша властита искуства.

Други користан извор идеја - програмерска искуства, искуства из свакодневног живота.

Одговор на друго питање - методологија структурног програмирања.

1. Поступно конструисање алгоритма - почиње се са глобалном, једноставном идејом алгоритма.

Затим се поједини кораци поступно прецизирају. Прецизирања се могу посматрати као потпроблеми првобитног проблема.

2. Основна идеја технике поступног програмирања - приликом прецизирања се увек

усредсређујемо на један проблем, чиме је вероватноћа грешке далеко мања.

3. АНАЛИЗА ПРОБЛЕМА

Поступак примене рачунара у решавању датог задатка можемо поделити у следеће фазе :

1. Дефинисање проблема ;

2. Глобални опис одговарајућег алгоритма ;

3. Поступна детаљна израда алгоритма ;

4. Писање програма за рачунар ;

5. Тестирање програма на рачунару ;

6. Решавање задатка извршавањем програма на рачунару .

3.1. Дефинисање проблема

Циљ прве фазе јесте да тачно одреди каква је основна намена програма, шта ће бити улазни, а шта излазни подаци.У овој фази треба прецизно поставити задатак, уочити и упознати све елементе значајне за решавање задатка. Потребно је критички оценити значај ових елемената и изабрати оне који су битни за решавање постављеног задатка. Дефинисање проблема је у многим случајевима веома једноставно, међутим често се догађа да је пут до дефиниције проблема веома дуг. Од тога како је проблем дефинисан веома зависи какве ће се тешкоће појавити у конструисању алгоритма и каква ће бити употребљивост програма.

3.2. Глобални опис одговарајућег алгоритма

Када је проблем довољно прецизно дефинисан треба глобално описати алгоритам, који ће представљати скелет програма. Ту спада, на пример, избор математичке методе која је примерена

датом проблему. Приликом избора методе треба имати на уму да проблем решавамо на рачунару, а не ручно. Тако се, имајући у виду велику брзину рачунара у извршавању неких врста операција, можемо радије одлучити за поступак који захтева велики број врло простих операција, него за елегантнију и бржу методу која се састоји од " интелигентнијих " операција.

3.3. Поступна детаљна израда алгоритма

Циљ следеће фазе је да се алгоритам довољно подробно изради, како би писање програма било што једноставније. Код сложених проблема је важно да се алгоритам развија систематично и постепено.

3.4. Писање програма за рачунар

Ова фаза представља запис алгоритма у облику прихватљивом за рачунар. Овако записан алгоритам назива се програм. Програм може бити записан на машинском, симболичком или неком вишем програмском језику. Писање програма представља фазу реализације, јер програм оспособљава рачунар за решавање жељеног задатка.

3.5. Тестирање програма на рачунару

У овој фази корисник треба да се увери у исправност састављеног програма. Ово је крупан проблем у програмирању. Обично се дешава да програм није потпуно исправан па је потребно пронаћи грешке. На неке грешке нас упозорава рачунар тиме што за програм јавља дијагностичку поруку која садржи листу грешака.Када отклонимо те грешке још увек не значи да је програм исправан.Програм се тестира тако што се он изврши за одабране скупове улазних величина. Примерима можемо показати да програм није исправан, али не можемо показати да је исправан. Иако је рачунар идеално средство за тестирање исправност програма не можемо доказивати тако што ћемо програм извршити за сваки могући скуп улазних величина.За велики број програма овакво тестирање би трајало годинама.Зато се проверавање врши са подацима за тестирање које треба одабрати тако да се без већих напора могу предвидети исправни резултати. Ако је програм и даље неисправан на то ће упозорити неочекивани резултати.

Овај поступак може бити веома дуготрајан и да од програмера захтева много спретности и досетљивости.

3.6. Решавање задатка извршавањем програма на рачунару

Када смо се уверили у исправност програма, можемо програм користити за решавање задатка за чије решавање је написан.За ову фазу рада је врло важно уз програм припремити детаљну документацију у којој се виде поставка задатка, алгоритам за решавање задатка, читљиво написан и објашњен програм, начин издавања улазних величина и резултата. Детаљна документација о програму омогућиће лако коришћење програма, као и евентуалне касније промене у програму.

4. СТРУКТУРА АЛГОРИТМА

Алгоритам треба прецизирати онолико детаљно колико је потребно с обзиром на то ко ће извршавати алгоритам. То се пре свега односи на то који ћемо програмски преводилац користити. Наредбе алгоритма морају бити такве да их по садржају, али не и према детаљном конкретном запису програмски преводилац разуме.То значи да није довољно само наредбе прилагодити извршиоцу програма већ и типове података тако да са њима уме да рачуна.

Поводом улазно - излазне реакције алгоритма потребно је одговорити на питања која се тичу самог дефинисања проблема. На та питања обавезно морамо да одговоримо пре него што приступимо конструисању самог алгоритма :

1. Шта су улазни подаци нашег програма ?

2. Шта су резултати ?

3. Каква је релација између улазних података и резултата ?

Потребно је да знамо да рачунар не решава проблеме сам, већ само извршава инструкције које му корисник задаје. Према томе, само ако корисник разуме проблем и познаје пут за решавање проблема до најситнијих детаља, биће у стању да представи тај пут преко низа инструкција које рачунар може да прихвати. Тада можемо очекивати од рачунара да тај низ инструкција веома брзо и тачно изврши и да добије резултат. То значи да рачунар не олакшава кориснику рад на решавању проблема, већ само извршава задати програм и у веома кратком времену даје резултате.

Када се процес решавања проблема спроводи без рачунара, о сваком кораку се мисли непосредно у тренутку када тај корак треба да се спроведе. Међутим, када се користи рачунар, тада цео низ корака које процес решавања проблема обухвата мора да буде одређен унапред са свим могућностима које могу да се јаве. Такође, низ корака мора бити прецизно дефинисан у облику алгоритамских веза и логичких услова. То значи да се, пре него што се приступи коришћењу рачунара, мора написати програм.

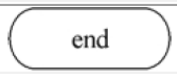
Са написаним програмом и улазним подацима, који одређују вредности низа параметара који се јављају у програму, рачунар може да да резултате задатка који смо желели да решимо.

Из свега претходно реченог очигледно је колико је важно добро написати алгоритам за решавање проблема. Треба имати у виду и карактеристике програмског језика и програмског преводиоца који ћемо користити у поступку решавања проблема уз помоћ рачунара, јер могу битно утицати на структуру алгоритма. Сама структура алгоритма треба да буде јасна и прегледна како би, и у поступку детаљнијег дефинисања алгоритма и у поступку писања програма, имали јасну слику како доћи до решења проблема који нас занима. Због тога је пожељно избегавати у конструисању алгоритма такве структуре које могу смањити прегледност проблема и довести до грешака у његовом решавању. Једна од таквих структура су безусловни скокови.

5. ЗАПИС АЛГОРИТМА ДИЈАГРАМОМ ТОКА

Један од најједноставнијих и често коришћених начина за писање алгоритама јесте дијаграм тока (или блок шема) алгоритма (токовник). Дијаграм тока представља једну варијанту графичког описа алгоритма. Сваки алгоритамски корак представљен је графичким симболом (блоком). Сви блокови су повезани линијама са могућим стрелицама. Најтачније се задаје структура алгоритма и редослед извршавања алгоритамских корака. Облик графичког симбола указује на врсту алгоритамског корака, односно његову функцију у алгоритму. Утабели 1 дати су најчешће коришћени графички симболи и њихова функција.

Табела 1. Графички симболи дијаграма тока

	početak
	ulaz podataka
	deklaracija varijabli i konstanti; postavljanje na početnu vrijednost; obrada podataka
	izlaz podataka
	kraj
	spojna točka, radi lakšeg praćenja toka podataka obično se u spojnu točku upisuju brojevi koji su veze između različitih dijelova algoritma

Произвољна алгоритамска шема се у општем случају може разложити на елементарне шеме или елементарне алгоритамске структуре.

Постоје следеће врсте елементарних шема:

- прост линијски алгоритам - сваки алгоритамски корак се остварује тачно једном од горе на доле,
- разгранати линијски алгоритам - сваки алгоритамски корак се остварује највише једном, тј. има и корака који се неће остваривати за дате полазне величине,
- циклични алгоритам - постоје алгоритамски кораци који се остварују више пута,
- селективни алгоритам - у зависности од датог критеријума бирају се следећи алгоритамски кораци.

Показује се да је довољно имати на располагању једну линијску, једну разгранату и једну цикличну елементарну шему да би се реализовала произвољна алгоритамска структура. У пракси се често користе различити облици разгранатих и цикличних елементарних шема. На следећим сликама дата је структура елементарних шема у структурном програмирању.

6. ОСНОВЕ АЛГОРИТМИЗАЦИЈЕ

Појам програмирања подразумева дуг процес размишљања о задатку. У том процесу решавања датог проблема користимо познате методе и правила или сами одређујемо поступке решења. Када се заврши мисаони процес решавања проблема, програм је готов. Сада је процес записивања програма само нужна формалност коју не мора да обави програмер. Довољно је да се идеја решења запише у погодном облику да би на основу тог записа нека, за то квалификована особа, кодирала програм - уписала га у меморију рачунара и извршила. Да би се идејно решење проблема могло записати тако да га свако за то квалификован, може разумети морају се поштовати нека прописана и формализована правила понашања. Тај коначан скуп строго формулисаних правила којима је одређен низ радњи за решавање одређеног проблема чини алгоритам.

Постоји много начина да се алгоритам прецизније дефинише, али за нас је довољна и оваква, не престрога, дефиниција:

Алгоритам је повезани низ елементарних правила, односно, алгоритамских корака у којима се поступно трансформишу улазне величине све док се не добије коначно решење.

Сваки алгоритам красе следеће карактеристике:

одређеност - сваки алгоритамски корак мора да буде прецизно дефинисан и једнозначан,

коначност - алгоритам се мора завршити после коначно много алгоритамских корака,

масовност - алгоритам се прави тако да се може применити за решавање читаве групе сродних проблема без обзира на различите полазне величине,

ефикасност - алгоритам треба да доведе до решења за што краће време и у што мањем броју алгоритамских корака.

Да би се логичко одвијање процеса решавања проблема лакше сагледало, алгоритам се представља дијаграмом тока, тј. графичком алгоритамском шемом.

Међутим, код решавања неког проблема, програмер нема пред собом обавезу да направи један од строго одређених типова алгоритама. Најчешће се дешава да се за решење проблема морају употребити различите, готово све поменуте структуре алгоритама у појединим сегментима програма, зато ову поделу алгоритама треба схватити само као теоријску.

ЛИТЕРАТУРА

1. С. Стојковић, Н. Стојановић, Д. стојановић: увод у рачунарство, Ниш, 2014
2. Ж.Тошић: Основи рачунарске технике, Ниш, 1994
3. Д. Прокин, В.Петровић, М. Мијалковић: Збирка задатака из Основа рачунарске технике, Висока техничка школа електротехнике и рачунарства струковних студија, Београд, 2013.

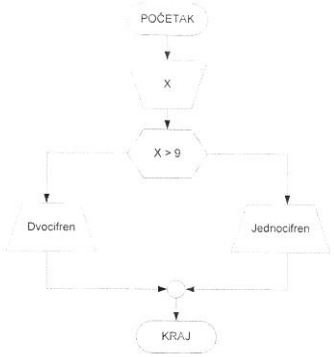
7. ЗАДАЦИ

Primer 1. Napraviti algoritam za sabiranje dva broja.

Rešenje:

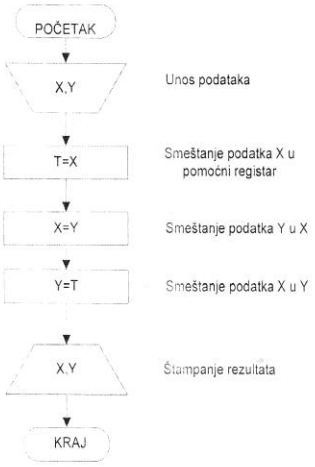
Primer 2. Napisati algoritam koji za uneti prirodni broj do 100 ispisuje da li je broj dvocifren ili jednocifren.

Rešenje:



Primer 3. Napraviti algoritam koji učitava brojeve X i Y i vrši zamenu njihovih vrednosti.

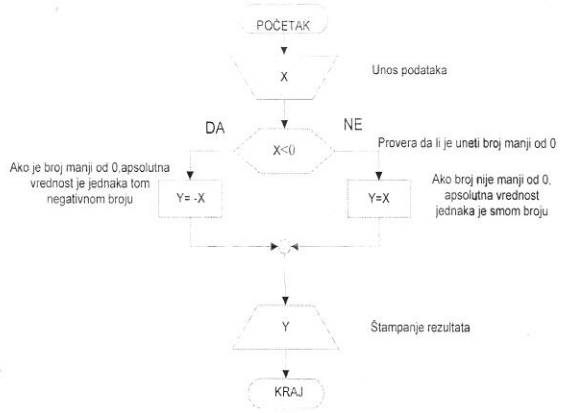
Rešenje:



Primer 4. Napraviti algoritam koji određuje apsolutnu vrednost broja X.

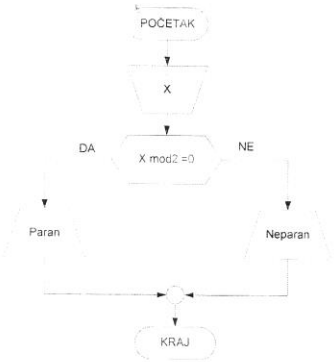
$$y = |x| = \begin{Bmatrix} -x; x < 0 \\ x; x \geq 0 \end{Bmatrix}$$

Rešenje:



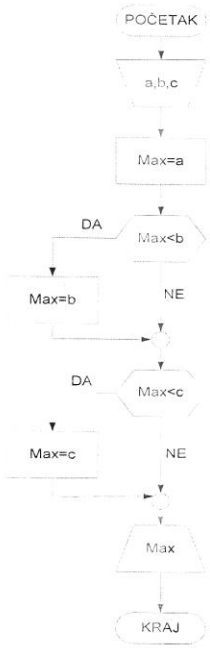
Primer 5. Napisati algoritam koji ispituje da li je broj paran ili neparan

Rešenje:



Primer 6. Napraviti algoritam koji promenljivoj max dodeljuje najvecu vrednost tri zadata broja a, b, c

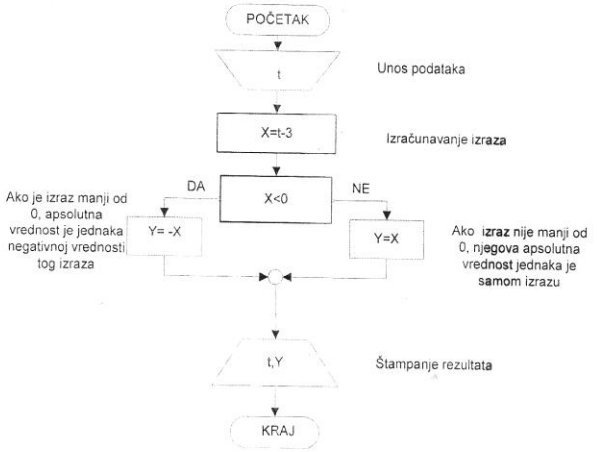
Rešenje:



111

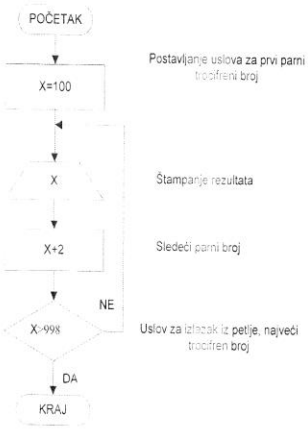
Primer 7. Napraviti algoritam koji izračunava izraz |t-3| i štampa uneti podatak t i apsolutnu vrednost izraza .

Rešenje:



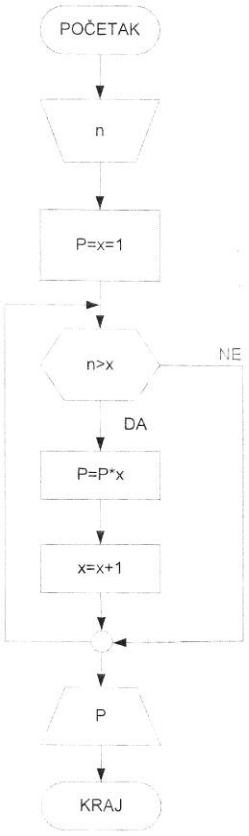
Primer 8. Napraviti algoritam koji štampa sve parne trocifrene brojeve.

Rešenje:



Primer 9. Napisati algoritam za izračunavanje faktorijala.

Rešenje:



Primer 10. Napraviti algoritam za unošenje dva prirodna broja (x,y) i izračunavanje izraza $z = x/y$. Štampati vrednost izraza, z.