Лекција 15: Програмирање у MathCad-у (ради само у верзији Mathcad Professional)

Програми олакшавају извршавање задатака које би било немогуће или врло тешко извршити на било који други начин. То је због тога што програм у Mathcad-у има многе атрибуте који га вежу за програмске језике укључујући условно гранање, цикличне структуре-петље, локално важење варијабли, рад са грешкама, и могућност позивања самог себе рекурзивно.

Шта је програм?

Програм је једноставно израз сачињен од више од једног исказа. Показаћемо како доњи пример изгледа када је написан као програм уместо као јединствен израз.

$$f(x,w) := \log\left(\frac{x}{w}\right)$$

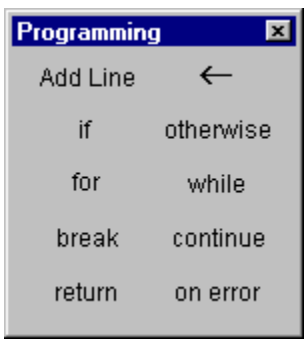
Упркос фундаменталној еквиваленцији између програма и једноставних израза, програми нуде две посебне предности:

- Када употребљавамо контролне структуре као што су петље и условна гранања, програм може постати далеко флексибилнији него што једноставни израз икад може бити.
- Програм сачињен од неколико једноставних корака је често много лакше креирати него еквивалентан, али далеко компликованији израз препун заграда.

Дефинисање програма

Следећи кораци илуструју како дефинисати програм

- $$f(x,w) := \left| \begin{array}{l} z \leftarrow \frac{x}{w} \\ \log(z) \end{array} \right.$$
- Откуцати леву страну дефиниције функције праћену оператором додељивања "**:**".
- Кликнути на Math алатну траку и отворити алатну траку Programming која садржи операторе програмирања.



- Кликнути на дугме "Add Line" или притиснути]. Ово креира вертикалну црту

$$f(x, w) := \left| \begin{array}{l} \blacksquare \\ \blacksquare \end{array} \right.$$

са плејсхолдером за додатне програмске наредбе (програм може имати било који број наредби).

- Кликнути на горњи плејсхолдер, откуцати "z" и кликнути на дугме за локално додељивање. 

(Треба знати да је дефиниција за варијаблу z локална у програму. Она је недефинисана изван програма и нема никаквог ефекта било где осим унутар програма. Не може се употребити Mathcad-ов уобичајени оператор додељивања ":=", унутар програма. Мора се употребити локални оператор додељивања представљен са "<-".)

- Довршити локално додељивање куцајући "*x*/*w*" у плејсхолдер десно од стрелице.

Последњи плејсхолдер треба увек садржавати вредност коју програм враћа. Куцајмо "*log*(*z*)" у тај плејсхолдер. Сада је могуће употребити ову функцију као било коју другу функцију или је израчунати симболички.

Условне наредбе

Употребићемо условну наредбу кад год желимо да се програмска наредба изврши само након испуњења неког услова као у следећем програму

$$f(x) := \left| \begin{array}{ll} 0 & \text{if } |x| > 2 \\ \sqrt{4 - x^2} & \text{otherwise} \end{array} \right.$$

За убацивање условне наредбе треба:
- Кликнути на плејсхолдер у који желимо сместити условну наредбу.
- Кликнути на Math алатну траку да би се отворила Programming алатна трака која садржи операторе програмирања.
- Кликнути "If" дугме или притиснути Shift+]. *Не куцати* реч "if".
- У десни плејсхолдер, куцати boolean израз.
- Кликнути "Add Line" дугме да би се убацили плејсхолдери за додатне наредбе ако је то потребно.
- Кликнути на преостали плејсхолдер и кликнути дугме "otherwise". *Не куцати* реч "otherwise".
- У преостали плејсхолдер куцати вредност коју желимо да програм враћа ако је услов нетачан.

Напомена:

Ако употребимо више од једне "if" наредбе пре "otherwise" наредбе, "otherwise" наредба ће се извршити само када су сви услови нетачни.

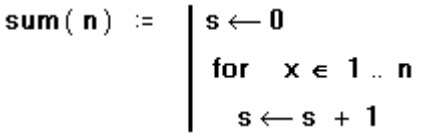
Програмске петље

Петља је програмска наредба која узрокује да се једна или више наредби (тело петље) извршава понављано све док одређени услов не буде испуњен. Постоје две врсте петљи:
- "For" петље су корисне када тачно знамо колико пута тело петље треба да се изврши.
- "While" петље су корисне када желимо да зауставимо извршавање након испуњења услова али не знамо тачно када ће се тај услов испунити.

Када користимо петље, може нам затребати да их прекинемо или да контролишемо одређене итерације.

"FOR" ПЕТЉЕ

Употребићемо "for" петљу када тачно знамо колико пута желимо да се тело петље изврши.

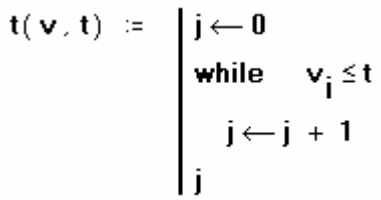


За убацивање "for" петље треба

- Кликнути на плејсхолдер у који желимо сместити "for" петљу.
- Кликнути на Math алатну траку да би се отворила Programming алатна трака која садржи операторе програмирања.
- Кликнути на "for" дугме или притиснути Ctrl+". *Не куцати* реч "for".
- У плејсхолдер лево од "e", унети итерациону варијаблу.
- У плејсхолдер десно од "e", унети опсег вредности који ће заузети итерациона варијабла (иако ћемо најчешће овде употребљавати варијаблу опсега, можемо такође употребити и вектор, листу скалара, и векторе раздвојене запетама).
- Кликнути на "Add Line" дугме да би се убацили плејсхолдери за додатне наредбе ако је то потребно. Ако желимо да се тело петље извршава све док се не испуни услов али не знамо тачно колико пута ће то бити, употребимо уместо овога "while" петљу.

"WHILE" ПЕТЉЕ

Употребићемо "while" петљу кад год желимо да се скуп наредби наставља извршавати све док се услов не испуни. Обавезно негде мора постојати наредба која чини услов нетачним. Иначе петља ће се извршавати бесконачно и мораће се зауставити притиском на тастер **Esc**.



Да би се убацила "while" петља треба:

- Кликнути на плејсхолдер у који желимо сместити "while" петљу.
- Кликнути на Math алатну траку да би се отворила Programming алатна трака која садржи операторе програмирања.
- Кликнути на "while" дугме или притиснути Ctrl+]. *Не куцати* реч "while".
- У плејсхолдер десно од "while", куцати boolean израз. Кликнути на "Add Line" дугме на Programming алатној траци да би се убацили плејсхолдери за додатне наредбе ако је то потребно.
- У плејсхолдер испод "while", унети ону наредбу која треба да се извршава понављано. Употребити "Add Line" дугме за убацивање плејсхолдера за додатне наредбе ако је то потребно.

"While" петље су корисне када желимо да зауставимо извршавање након испуњења услова а не знамо тачно када ће се тај услов испунити. Ако тачно знамо колико итерација желимо, употребићемо уместо овога "for" петљу.

"BREAK" НАРЕДБА

Употребићемо наредбу "break" у петљи кад год желимо да зауставимо извршавање петље.

```
break  if  x ≥ 8
y
```

Да би се убацила наредба "break" треба:

- Кликнути на плејсхолдер у који желимо сместити наредбу "break".
- Кликнути на Math алатну траку да се отвори Programming алатна трака која садржи операторе програмирања.
- Кликнути на "break" дугме или притиснути Ctrl+{. *Не куцати* реч "break".

Када Mathcad наиђе на наредбу "break" у телу “for” или “while” петље:

1. Петља прекида извршавање и враћа последњу израчунату вредност.
2. Извршавање програма се потом наставља са следећом програмском линијом после петље.

Контрола итерација у петљи

Програмске петље су дизајниране тако да се извршавају све док не буде испуњен неки услов или до одређеног броја итерација. Међутим, можемо пожелети да зауставимо петљу за време одређене итерације и наставимо је са следећом итерацијом. Да би се то урадило, употребићемо наредбу “continue”.
Да би се убацила наредба "continue" треба:

-

```
for i ∈ 0..4
  for j ∈ 0..3
    continue  if mod(j, 2) = 0
    a_{i,j} ← 1
a
```

$$= \begin{bmatrix} 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

- Кликнути на плејсхолдер у који желимо сместити наредбу "continue".
- Кликнути на Math алатну траку да се отвори Programming алатна трака која садржи операторе програмирања.
- Кликнути на "continue" дугме или притиснути Ctrl+[. *Не куцати* реч "continue".
- Када програм наиђе на “continue,” он зауставља итерацију, одлази на најближу ванјску петљу, и наставља са следећом итерацијом.

Враћање вредности из програма

Подразумевано, програм враћа оно што је на последњој линији. Међутим, можемо вратити вредност лоцирану другде у програму помоћу наредбе “return”.

$x := 0$

if $x=0$ $= 2$
  $a \leftarrow 2$
  return a
for $i \in 0..x$
  $t \leftarrow t + x$
t

Да би се убацила наредба "return" треба:

- Кликнути на плејсхолдер у који желимо сместити наредбу "return".
- Кликнути на Math алатну траку да се отвори Programming алатна трака која садржи операторе програмирања.
- Кликнути на "return" дугме или притиснути Ctrl+|. Не куцати реч "return".
- У плејсхолдер десно од "return",откуцати оно што желимо да буде враћено.

"Return" наредбе су корисне када желимо да вратимо вредност из одређене петље.

Лов на грешке у програму

Да би се вратила алтернативна вредност када се наиђе на грешку у изразу, користићемо “on error” оператор

$$f(x) := \infty \text{ on error } \frac{1}{2 - x}$$

$$f(1) = 1$$

$$f(2) = 1 \cdot 10^{307}$$

програмирања:

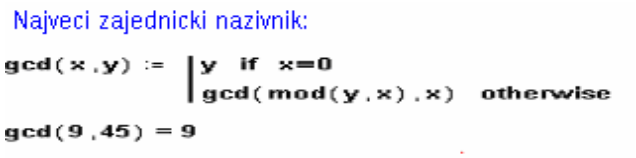
- Кликнути на плејсхолдер у који желимо сместити наредбу "on error".
- Кликнути на Math алатну траку да се отвори Programming алатна трака која садржи операторе програмирања.
- Кликнути на "on error" дугме или притиснути Ctrl+'. *Не куцати* речи "on error".
- У плејсхолдер десно од "on error", откуцати оно што желимо да се врати, под претпоставком да се то може успешно израчунати.
- У плејсхолдер лево од "on error", откуцати оно што желимо да се врати ако подразумевани израз који је требало вратити не може бити израчунат. Користити "Add Line" дугме за убацивање плејсхолдера за додатне наредбе ако је то потребно.

Десни израз је израчунат и враћен ако се није појавила ниједна грешка. Ако се грешка појави, вратиће се леви аргумент.

Рекурзија

Рекурзија је моћно програмско средство које укључује дефинисање варијабле изражене преко саме себе како је показано у овом примеру:

Дефиниције рекурзивних функција морају увек имати бар два дела:



1. Почетни услов који ће спречити да рекурзија траје непрестано, и
2. Дефиниција функције изражена преко претходне вредности функције.

Идеја је слична оној која лежи у основи појма математичке индукције: ако се може добити $f(n+1)$ из $f(n)$, и знамо $f(0)$, онда знамо све што се може знати о f. Међутим, треба имати на уму, да рекурзивне дефиниције функција нису ипак, упркос својој елеганцији и концизности, увек компутационо најефикасније дефиниције. Могуће је да ће нека еквивалентна дефиниција која употребљава неку од итеративних петљи рачунати брже.

Симболичко рачунање програма

Када смо једном дефинисали програм за којег желимо да рачуна симболички:
(1) Откуцајмо име функције или име варијабле које је дефинисано преко програма.
(2) Притиснимо **Ctrl +Тачка** да убацимо десну стрелицу која представља симболички знак једнакости.