Uvod

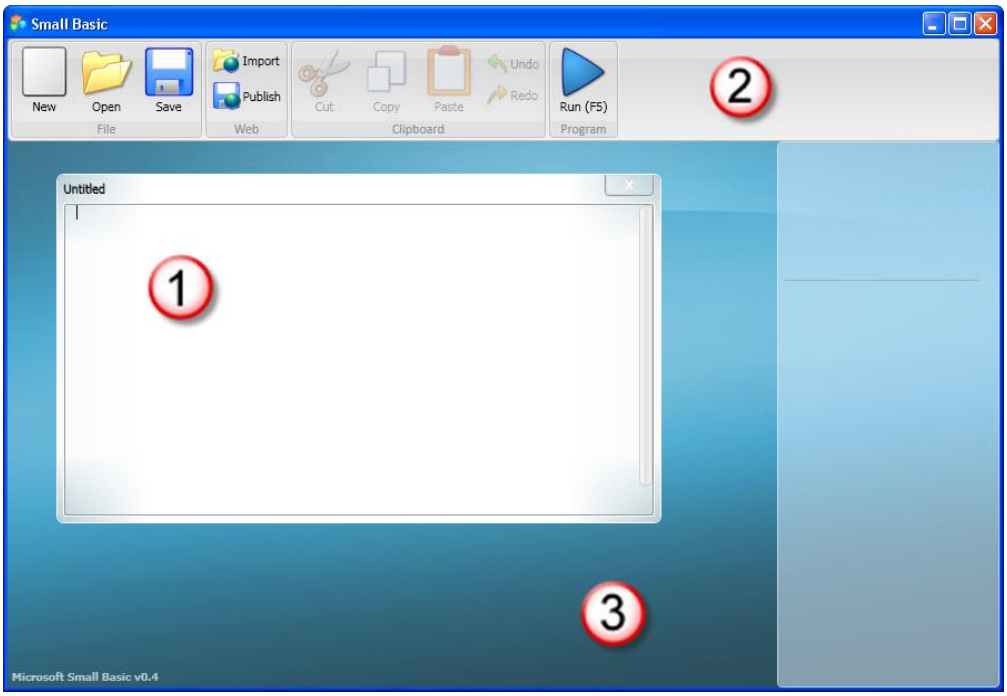
Small Basic i programiranje

Računalno programiranje definira se kao postupak stvaranja računalnog softvera pomoću odgovarajućih programskih jezika. Baš kao što mi govorimo i razumijemo hrvatski, španjolski ili francuski, računala razumiju programe napisane u određenim jezicima. Jezici koje računala razumiju zovu se programski jezici. Nekoć je postojalo samo nekoliko programskih jezika i mogli su se vrlo jednostavno savladati i razumjeti. No kako su računala i softver postajali sve sofisticiraniji, programski su se jezici brzo razvijali obuhvaćajući pritom sve složenije koncepte. Zato su suvremeni programski jezici i njihovi koncepti prilično nerazumljivi početnicima. Ta je činjenica obeshrabrila mnoge koji su željeli naučiti programirati računala i one koji su ih pokušavali programirati.

Small Basic je programski jezik stvoren da programiranje početnicima učini krajnje jednostavnim, pristupačnim i zabavnim. Small Basic je stvoren da ukloni te prepreke i svima omogući ulazak u čudesan svijet računalnog programiranja.

Okruženje Small Basica

Najprije ćemo ukratko opisati okruženje Small Basica. Kada prvi put pokrenete program SmallBasic.exe, pojavit će se prozor koji izgleda kao ovaj na sljedećoj slici.



Slika 1 – okruženje Small Basica

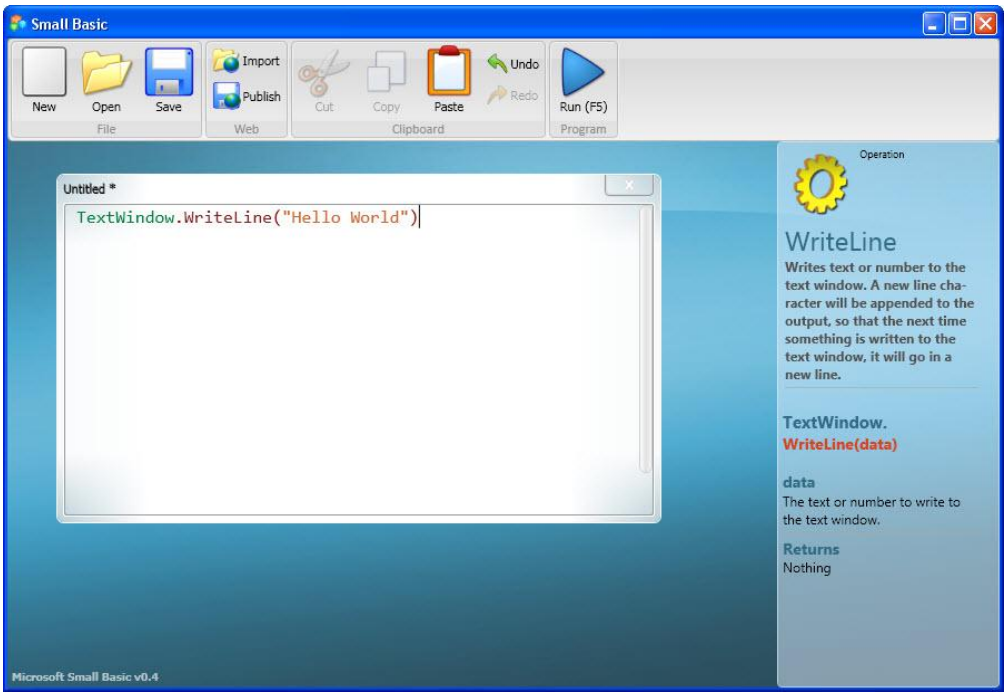
To je okruženje Small Basica, u kojemu ćete pisati i pokretati programe u Small Basicu. Okruženje ima nekoliko elemenata naznačenih brojevima.
U **uređivaču**, naznačenom brojem [1], pisat ćete programe u Small Basicu. Kada otvorite ogledni ili prethodno spremljeni program, on će se pojaviti u uređivaču. Tada ga možete izmijeniti i spremiti za kasnije korištenje.
Istodobno možete otvoriti više programa i raditi s njima. Svaki program s kojim radite prikazat će se u zasebnom uređivaču. Uređivač u kojemu se nalazi program s kojim trenutno radite zove se *aktivni uređivač*.
Alatna traka, naznačena brojem [2], omogućuje upućivanje naredbi *aktivnom uređivaču* ili okruženju.
Razne naredbe dostupne na alatnoj traci upoznat ćete u nastavku.
Na **površini**, naznačenoj brojem [3], nalaze se svi prozori uređivača.

Vaš prvi program

Sada kada ste upoznali okruženje Small Basica, možemo nastaviti i početi programirati u njemu. Kao što je već rečeno, uređivač je mjesto na kojemu pišete programe. Upišite onda sljedeći redak u uređivač.

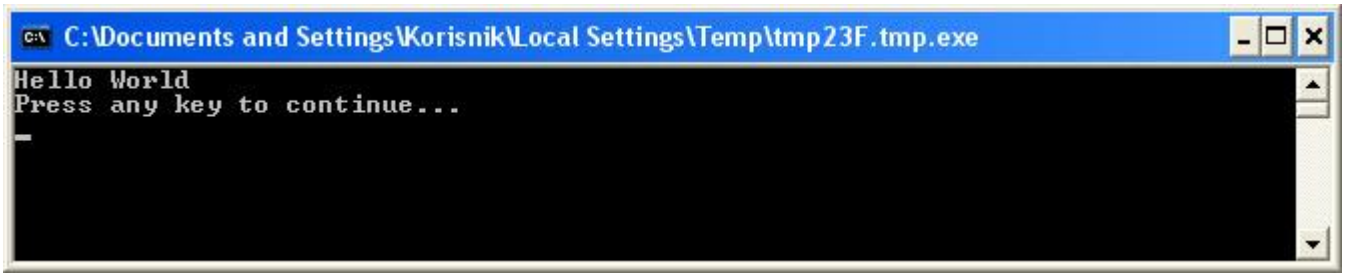
```
TextWindow.WriteLine("Hello World")
```

To je vaš prvi program u Small Basicu. Ako ste ga ispravno upisali, trebali biste vidjeti nešto slično ovome na donjoj slici.



Slika 2 – prvi program

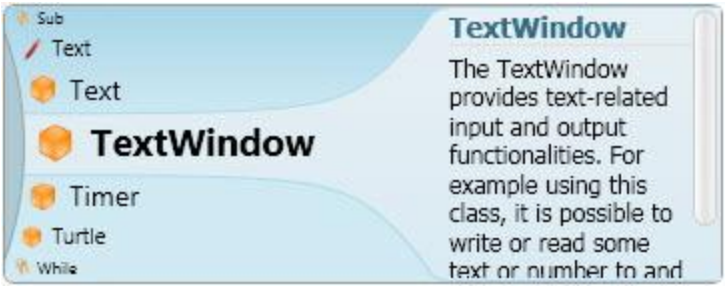
Sad kada ste upisali svoj prvi program, pokrenite ga da biste vidjeli što će se dogoditi. Program možete pokrenuti klikom na gumb *Run (Pokreni)* na alatnoj traci ili pomoću tipkovnog prečaca F5. Ako sve prođe bez poteškoća, program bi se trebao pokrenuti s dolje prikazanim ishodom.



Slika 3 – izlaz prvog programa

Čestitamo! Upravo ste napisali i pokrenuli svoj prvi program u Small Basicu. To je vrlo kratak i jednostavan program, ali je ipak velik korak prema vašoj preobrazbi u pravog računalnog programera! No to je samo jedan korak koju smo morali obaviti prije no što prijeđemo na stvaranje većih programa. Morate razumjeti što se upravo dogodilo – što ste točno naredili računalu i kako je ono znalo što mora učiniti? Zato ćemo u sljedećem poglavlju analizirati program koji ste upravo napisali.

*Dok ste pisali svoj prvi program, vjerojatno ste primijetili da se pojavio skočni prozor s popisom stavki (*slika 4*). Taj se popis zove "intellisense" i omogućuje brže pisanje programa. Krećite se po njemu pomoću tipki sa strelicama gore i dolje, a kada pronađete željenu stavku, pritisnite Enter da biste je umetnuli u program.*



Slika 4 – Intellisense

Spremanje programa

Ako želite zatvoriti okruženje Small Basica i poslije nastaviti raditi na programu koji ste upravo napisali, možete spremiti program. Preporučljivo je povremeno spremati programe da ne biste izgubili podatke u slučaju nehotičnog isključivanja računala ili nestanka električne energije. Trenutni program spremite klikom na ikonu "Save" (Spremi) na alatnoj traci ili pomoću prečaca "Ctrl+S" (pritisnite tipku S držeći pritisnutu tipku Ctrl).

Kako funkcionira vaš prvi program?

Što je zapravo računalni program?

Program je skup naredbi računalu. Te naredbe govore računalu što točno mora učiniti i ono ih uvijek izvršava. Baš kao i ljudi, računala će izvršavati naredbe samo ako su napisane na jeziku koji razumiju. Jezici koje računala razumiju zovu se programski jezici. Računala razumiju brojne jezike, a **Small Basic** je jedan od njih.
Zamislite razgovor između sebe i nekog svog prijatelja. Za međusobni prijenos informacija vi i vaš prijatelj koristit ćete riječi organizirane u rečenice. I programski jezici sadrže zbirke riječi koje se mogu organizirati u rečenice koje prenose informacije računalu. Programi su u osnovi skupovi rečenica (katkad ih sadrže svega nekoliko, a katkad više tisuća) koje kao cjelina imaju smisla i programeru i računalu.

Programi u Small Basicu

Uobičajeni program u Small Basicu sastoji se od brojnih *iskaza*. Svaki redak programa predstavlja iskaz, a svaki je iskaz određena naredba računalu. Kada računalu naredimo da izvrši program u Small Basicu, ono će pročitati prvi iskaz u programu. Računalo će razumjeti što želimo, a zatim izvršiti našu naredbu. Nakon što izvrši prvi iskaz, vratit će se u program te pročitati i izvršiti drugi redak. Nastaviti će to činiti dok ne dođe do kraja programa. Tada se izvođenje programa završava.

Računala razumiju mnoge jezike. Java, C++, Python, VB itd. suvremeni su računalni jezici bogatih mogućnosti, koji se koriste za razvoj softverskih programa različitih razina složenosti.

Još malo o vašem prvom programu

Ovo je prvi program koji ste napisali:

```
TextWindow.WriteLine("Hello World")
```

To je vrlo jednostavan program, koji se sastoji od samo jednog *iskaza*. Taj iskaz naređuje računalu da u prozoru za tekst napiše redak teksta **Hello World**.

Taj iskaz računalo doslovce tumači kao:

Napiši **Hello World**

Vjerojatno ste primijetili da se iskaz može rastaviti na manje dijelove, baš kao što se rečenice mogu rastaviti na riječi. Prvi iskaz ima tri dijela:
a) TextWindow
b) WriteLine
c) "Hello World"
Točka, zagrade i navodnici rečenični su znakovi koje je potrebno umetnuti na odgovarajuća mjesta u iskazu da bi računalo znalo što želimo.
Vjerojatno se sjećate crnog prozora koji se pojavio kada ste pokrenuli svoj prvi program. Taj se crni prozor zove TextWindow (prozor za tekst), a katkad i konzola. U njemu se prikazuje ishod izvođenja programa. **TextWindow** u ovom se programu zove *objekt*. U programima možete koristiti brojne objekte. Na objektima možete obavljati više različitih *operacija*. U svom ste programu već koristili operaciju *WriteLine*. Vjerojatno ste primijetili i da se iza operacije WriteLine nalazi tekst **Hello World** unutar navodnika. Taj se tekst kao ulazna

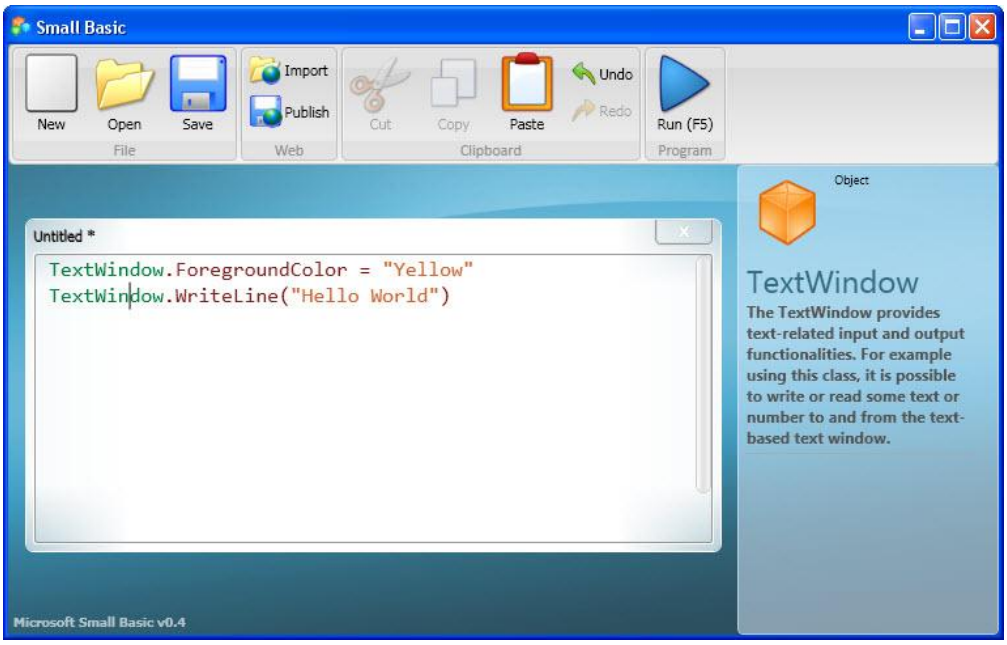
vrijednost prenosi u operaciju WriteLine, a ona ga zatim prikazuje korisniku. To je tzv. *ulazna vrijednost* operacije. Neke operacije prihvaćaju jednu ili više ulaznih vrijednosti, a druge ne prihvaćaju nijednu.

Rečenični znakovi, primjerice navodnici, razmaci i zagrade, vrlo su važni u računalnim programima. Ovisno o njihovu položaju i broju rečenični znakovi mogu promijeniti značenje onoga što želite izraziti.

Vaš drugi program

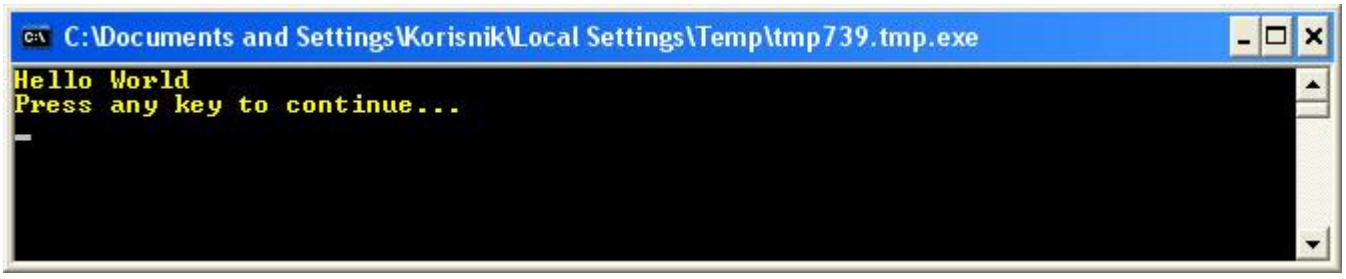
Sada kada razumijete kako funkcionira vaš prvi program, dotjeramo ga malo dodavanjem boja.

```
TextWindow.ForegroundColor = "Yellow"
TextWindow.WriteLine("Hello World")
```



Slika 5 – dodavanje boja

Kada ga pokrenete, gornji će program u prozoru za tekst ponovno prikazati tekst "Hello World", ali ovaj će ga put prikazati u žutoj boji, a ne u sivoj kao prošli put.



Slika 6 – žuti tekst Hello World

Obratite pozornost na novi iskaz koji smo dodali u izvorni program. Taj iskaz sadrži novu riječ, *ForegroundColor*, koju smo izjednačili s vrijednošću *"Yellow"*.To znači da smo svojstvu *ForegroundColor* dodijelili vrijednost "Yellow". Razlika između svojstva ForegroundColor i operacije WriteLine u tome je što svojstvo ForegroundColor ne prihvaća ulazne vrijednosti i ne treba zagrade. Umjesto toga ono je popraćeno znakom *jednakosti* i određenom riječju. ForegroundColor je *svojstvo* objekta TextWindow. Slijedi popis vrijednosti koje prihvaća svojstvo ForegroundColor. Zamijenite vrijednost "Yellow" nekom od tih vrijednosti i

pogledajte što će se dogoditi (nemojte zaboraviti navodnike – oni su obavezni rečenični znakovi).

```
Black
Blue
Cyan
Gray
Green
Magenta
Red
White
Yellow
DarkBlue
DarkCyan
DarkGray
DarkGreen
DarkMagenta
DarkRed
DarkYellow
```

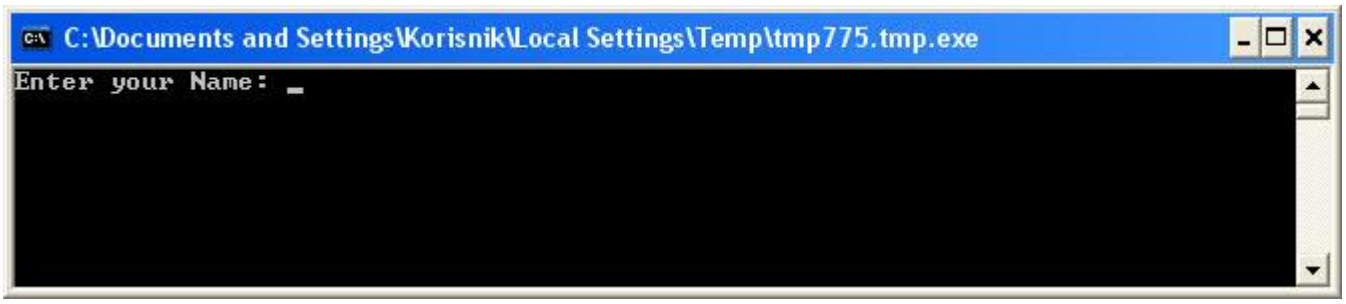
Varijable

Korištenje varijabli u programu

Bilo bi lijepo kada bi vaš program mogao prikazati riječ "Hello" popraćenu korisnikovim imenom umjesto da prikazuje generički tekst "Hello World". Da bismo to postigli, najprije moramo pitati korisnika kako se zove, negdje pohraniti ime, a zatim prikazati tekst "Hello" popraćen korisnikovim imenom. Pogledajmo kako to učiniti:

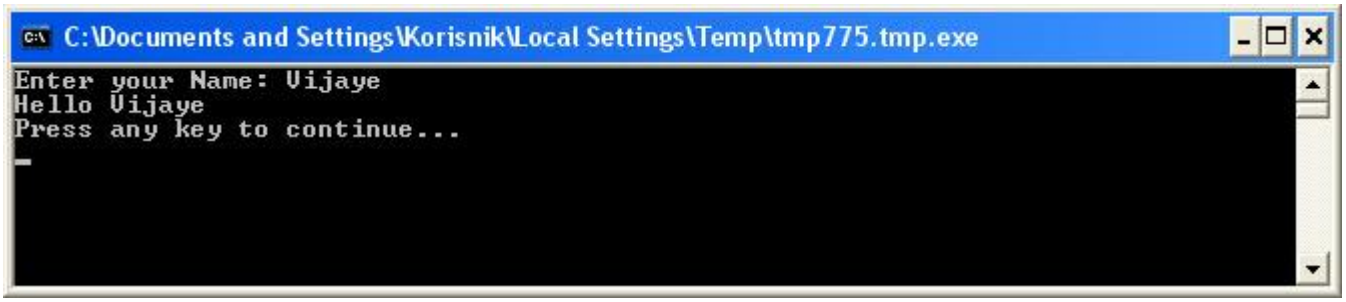
```
TextWindow.Write("Enter your Name: ")
name = TextWindow.Read()
TextWindow.WriteLine("Hello " + name)
```

Kada napišete i pokrenete taj program, pojavit će se ovakav izlaz:



Slika 7 – program traži ime korisnika

A kada upišete ime i pritisnete tipku ENTER, pojavit će se sljedeći izlaz:



Slika 8 – srdačan pozdrav

Ako ga ponovno pokrenete, program će postaviti isto pitanje. Ako upišete neko drugo ime, računalo će prikazati riječ "Hello" popraćenu tim imenom.

Analiza programa

Vjerojatno ste primijetili sljedeći redak programa koji ste upravo pokrenuli:

```
name = TextWindow.Read()
```

Read() izgleda kao *WriteLine()*, ali bez ulaznih vrijednosti. To je operacija koja u osnovi naređuje računalu da pričeka da korisnik nešto upiše i pritisne tipku ENTER. Kada korisnik pritisne tipku ENTER, ta operacija vraća programu ono što je korisnik upisao. Važno je napomenuti da se ono što korisnik upiše sada pohranjuje u *varijablu* **name**. V*arijabla* se definira kao mjesto za privremeno pohranjivanje vrijednosti za naknadnu upotrebu. U gornjem se retku u varijablu **name** pohranjuje ime korisnika.

Zanimljiv je i sljedeći redak:

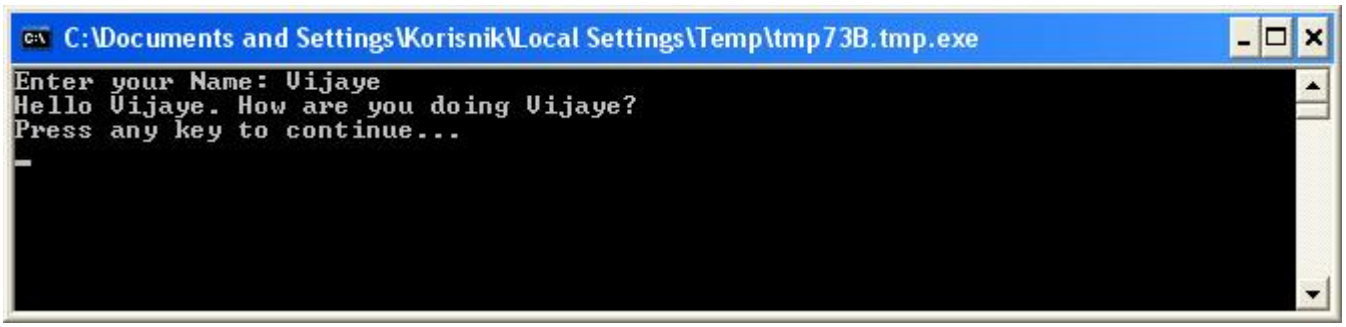
```
TextWindow.WriteLine("Hello " + name)
```

U ovom se retku koristi vrijednost pohranjena u varijablu **name**. Vrijednost pohranjena u varijablu **name** pridodaje se riječi "Hello" i prikazuje u prozoru za tekst.
Kada varijabli dodijelite neku vrijednost, možete je upotrijebiti koliko god puta želite. Možete, primjerice, učiniti sljedeće:

```
TextWindow.Write("Enter your Name: ")
name = TextWindow.Read()
TextWindow.Write("Hello" + name + ".")
TextWindow.WriteLine("How are you doing" + name + "?")
```

Write*, baš kao i* WriteLine*, još je jedna operacija za prozor konzole. Operacija Write omogućuje prikaz teksta u prozoru konzole, ali novi se tekst prikazuje u istom retku kao postojeći.*

Pojavit će se ovakav izlaz:



Slika 9 – ponovno korištenje varijable

Pravila za imenovanje varijabli

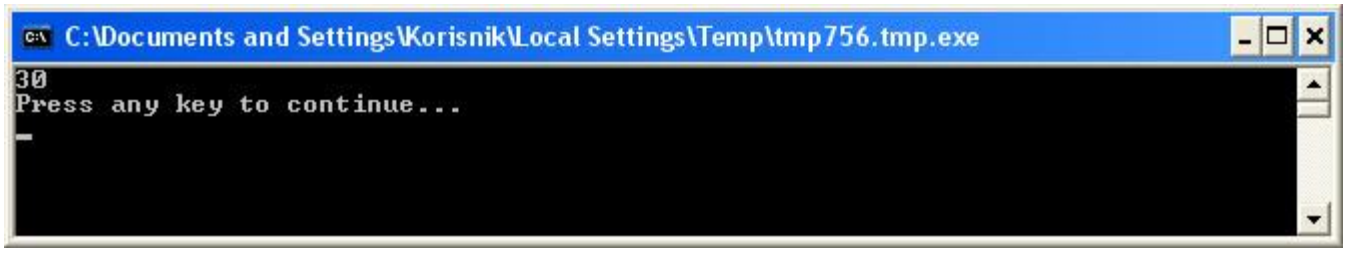
[U IZRADI]

Rad s brojevima

Vidjeli ste kako u varijablu možete pohraniti ime korisnika. U sljedećih ćemo nekoliko programa vidjeti kako se u varijable pohranjuju brojevi i kako se radi s brojevima. Krenimo od vrlo jednostavnog programa:

```
number1 = 10
number2 = 20
number3 = number1 + number2
TextWindow.WriteLine(number3)
```

Kada pokrenete taj program, pojavit će se sljedeći izlaz:



Slika 10 – zbrajanje dvaju brojeva

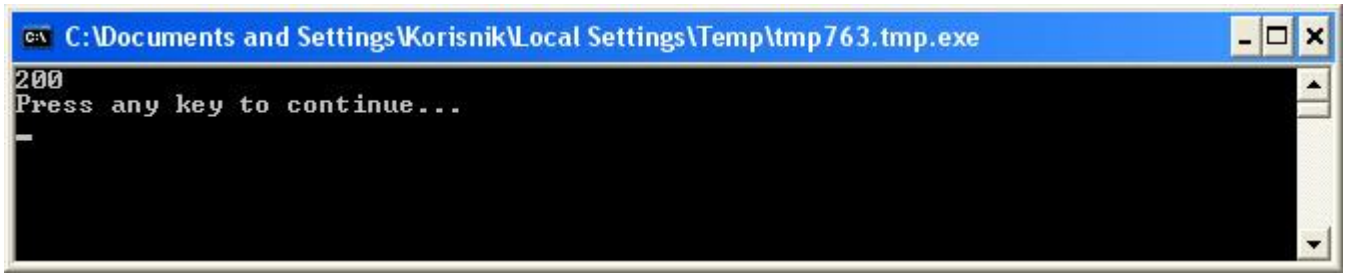
U prvom se retku programa varijabli **number1** dodjeljuje vrijednost 10, a u sljedećem se varijabli **number2** dodjeljuje vrijednost 20. U trećem se retku zbrajaju **number1** i **number2,** a zatim se rezultat zbrajanja dodjeljuje varijabli **number3**. Dakle, varijabla **number3** u ovom slučaju ima vrijednost 30 i upravo se ta vrijednost prikazuje u prozoru za tekst.

Izmijenimo malo program i pogledajmo što će se dogoditi:

```
number1 = 10
number2 = 20
number3 = number1 * number2
TextWindow.WriteLine(number3)
```

Uočite da brojevi nisu navedeni unutar navodnika. Navodnici nisu potrebni za brojeve, već samo kada koristite tekst.

Gornji program množi vrijednost varijable **number1** s vrijednošću varijable **number2** i rezultat pohranjuje u varijablu **number3**. Na donjoj slici prikazano je što će se dogoditi kada pokrenete taj program:



Slika 11 – množenje dvaju brojeva

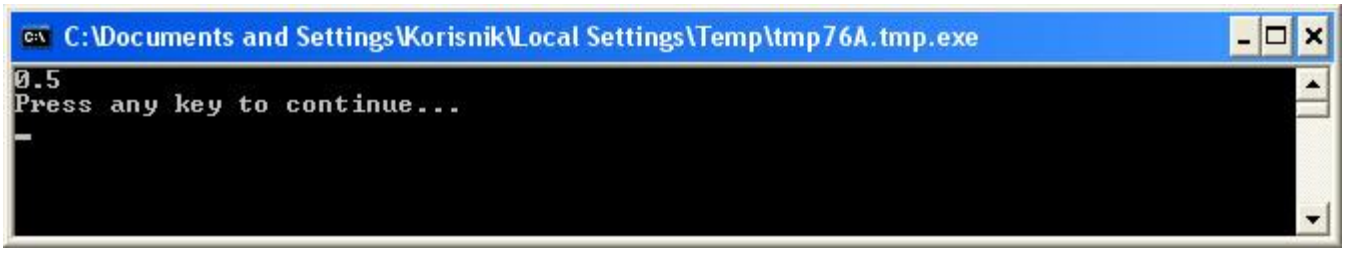
Isto tako možete oduzeti jedan broj od drugog ili podijeliti jedan broj s drugim. Ovo je oduzimanje:

```
number3 = number1 - number2
```

Znak dijeljenja je "/". Program izgleda ovako:

```
number3 = number1 / number2
```

Rezultat dijeljenja je:



Slika 12 – dijeljenje jednog broja s drugim

Jednostavan program za pretvorbu temperaturnih jedinica

$$^{\circ}C = \frac{5(^{\circ}F - 32)}{9}$$

U sljedećem se programu formula koristi za pretvorbu temperature izražene u Fahrenheitovim stupnjevima u temperaturu izraženu u Celzijevim stupnjevima. Najprije se od korisnika traži da unese temperaturu u Fahrenheitovim stupnjevima, a zatim se ta vrijednost pohranjuje u varijablu. Unos brojeva omogućuje operacija **TextWindow.ReadNumber**.

```
TextWindow.Write("Enter temperature in Fahrenheit: ")
fahr = TextWindow.ReadNumber()
```

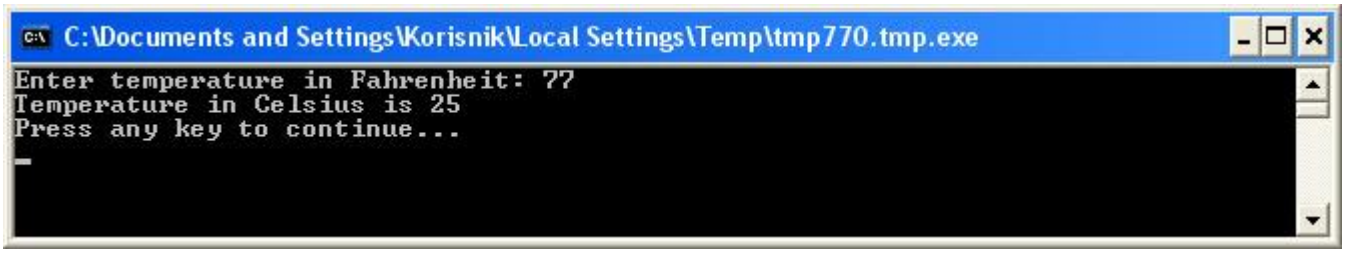
Kada se pohrani u varijablu, temperaturu u Fahrenheitovim stupnjevima lako je pretvoriti u temperaturu u Celzijevim stupnjevima:

```
celsius = 5 * (fahr - 32) / 9
```

Zagrade govore računalu da najprije mora izračunati dio **fahr - 32,** a zatim ostatak izraza. Preostaje još samo prikaz rezultata korisniku. Cijeli program izgleda ovako:

```
TextWindow.Write("Enter temperature in Fahrenheit: ")
fahr = TextWindow.ReadNumber()
celsius = 5 * (fahr - 32) / 9
TextWindow.WriteLine("Temperature in Celsius is " + celsius)
```

Rezultat izvođenja ovog programa je:



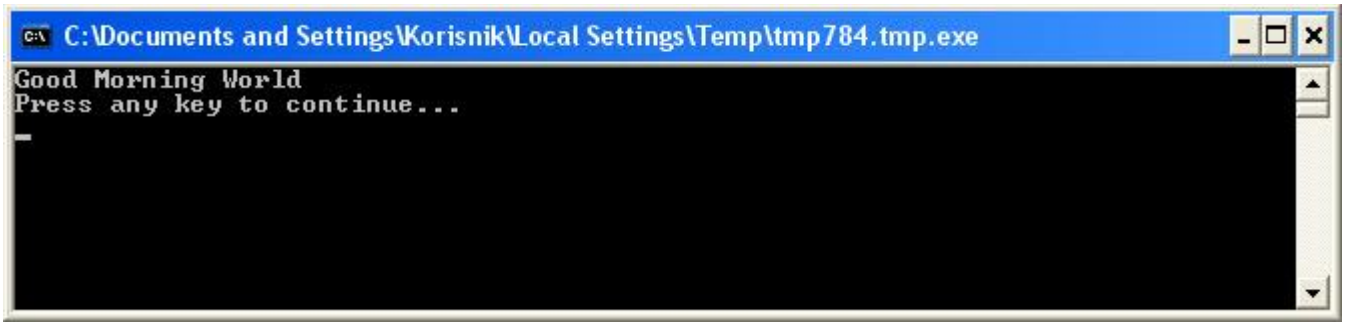
Slika 13 – pretvorba temperaturnih jedinica

Uvjeti i grananje

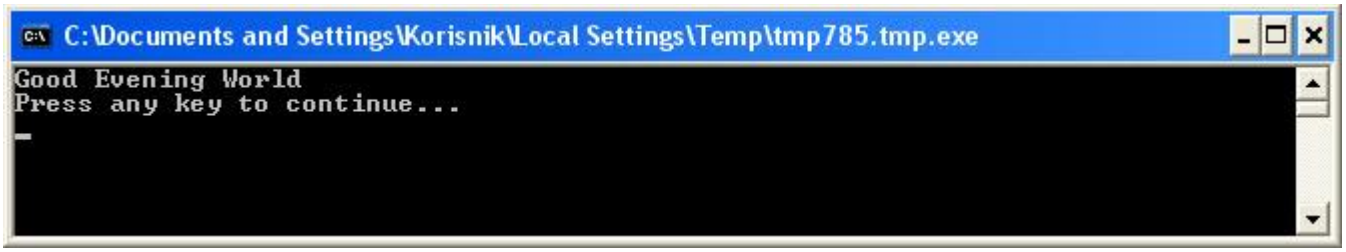
Vratimo se vašem prvom programu. Bilo bi zgodno kada bi se umjesto općenitog teksta *Hello World* ovisno o dobu dana prikazivao tekst *Good Morning World* ili *Good Evening World*. Sljedeći program prije 12.00 sati prikazuje tekst *Good Morning World*, a nakon 12.00 sati tekst *Good Evening World*.

```
If (Clock.Hour < 12) Then
TextWindow.WriteLine("Good Morning World")
EndIf
If (Clock.Hour >= 12) Then
TextWindow.WriteLine("Good Evening World")
EndIf
```

Ovisno o vremenu pokretanja programa pojavit će se sljedeći izlazi:



Slika 14 – Good Morning World



Slika 15 – Good Evening World

Analizirajmo prva tri retka programa. Vjerojatno ste već shvatili da taj redak naređuje računalu da prikaže tekst "Good Morning World" ako je vrijednost svojstva Clock.Hour manja od 12. Riječi **If**, **Then** i **EndIf** posebne su riječi koje računalo razumije prilikom izvođenja programa. Riječ **If** uvijek je popraćena uvjetom, koji je u ovom slučaju (**Clock.Hour < 12)**. Ne zaboravite da su zagrade potrebne da bi računalo znalo što želite. Nakon uvjeta slijedi riječ **then** i operacija koja će se izvršiti. Nakon operacije slijedi riječ **EndIf**. Ta riječ obavještava računalo da je uvjetno izvršavanje gotovo.

U Small Basicu objekt Clock *omogućuje pristup datumu i vremenu. Nudi i brojna svojstva pomoću kojih možete dobiti zasebne podatke o danu, mjesecu, godini, satu, minutama i sekundama.*

Između riječi **then** i **EndIf** može se nalaziti više naredbi, koje će računalo izvršiti ako je uvjet ispunjen. Napišite, primjerice, ovakav program:

```
If (Clock.Hour < 12) Then
TextWindow.Write("Good Morning. ")
TextWindow.WriteLine("How was breakfast?")
EndIf
```

Else

Vjerojatno ste primijetili da je u programu na početku ovog poglavlja drugi uvjet suvišan. Vrijednost svojstva **Clock.Hour** može biti ili manja od 12 ili ne biti manja od 12. Druga provjera uopće nije potrebna. U takvim situacijama dva iskaza **if..then..endif** možete skratiti objedinjavanjem u jedan pomoću nove riječi **else**.
Ako ga ponovno napišemo tako da koristi riječ **else**, program će izgledati ovako:

```
If (Clock.Hour < 12) Then
TextWindow.WriteLine("Good Morning World")
Else
TextWindow.WriteLine("Good Evening World")
EndIf
```

Ovaj program čini potpuno istu stvar kao prethodni, što to je dobar povod za vrlo važnu lekciju iz računalnog programiranja:

U programiranju se ista stvar obično može postići na više načina. Katkad jedan način ima više smisla nego neki drugi. Programer odlučuje koji će način koristiti. Kada nakon mnogo napisanih programa steknete određeno iskustvo, počet ćete primjećivati te različite postupke, baš kao i njihove prednosti i nedostatke.

Uvlake

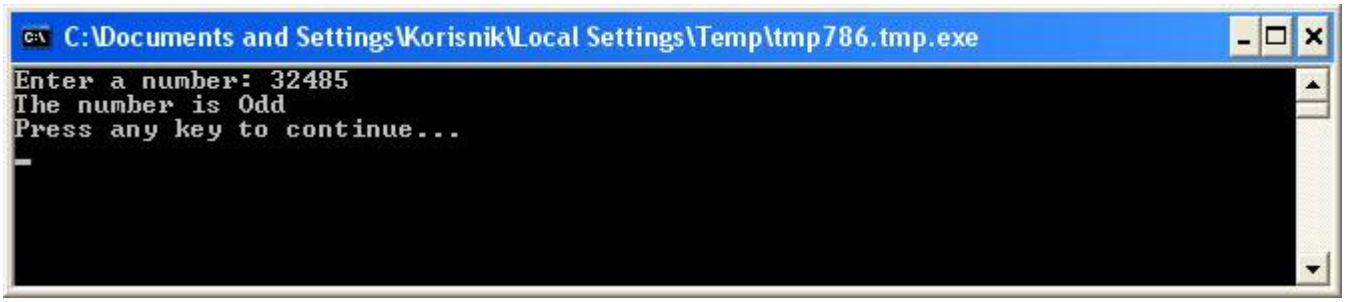
Kao što ste vjerojatno primijetili, u svim su primjerima iskazi između riječi *If, Else* i *EndIf* uvučeni. Te uvlake nisu neophodne. Računalo će razumjeti program i bez njih. No one olakšavaju uvid u strukturu programa i njezino razumijevanje. Zato je obično preporučljivo uvlačiti iskaze između takvih blokova.

Par-nepar

Sada kada ste savladali iskaz **If..Then..Else..EndIf**, napišimo program koji utvrđuje je li uneseni broj paran ili neparan.

```
TextWindow.Write("Enter a number: ")
num = TextWindow.ReadNumber()
remainder = Math.Remainder(num, 2)
If (remainder = 0) Then
TextWindow.WriteLine("The number is Even")
Else
TextWindow.WriteLine("The number is Odd")
EndIf
```

Kada pokrenete taj program, pojavit će se ovakav izlaz:



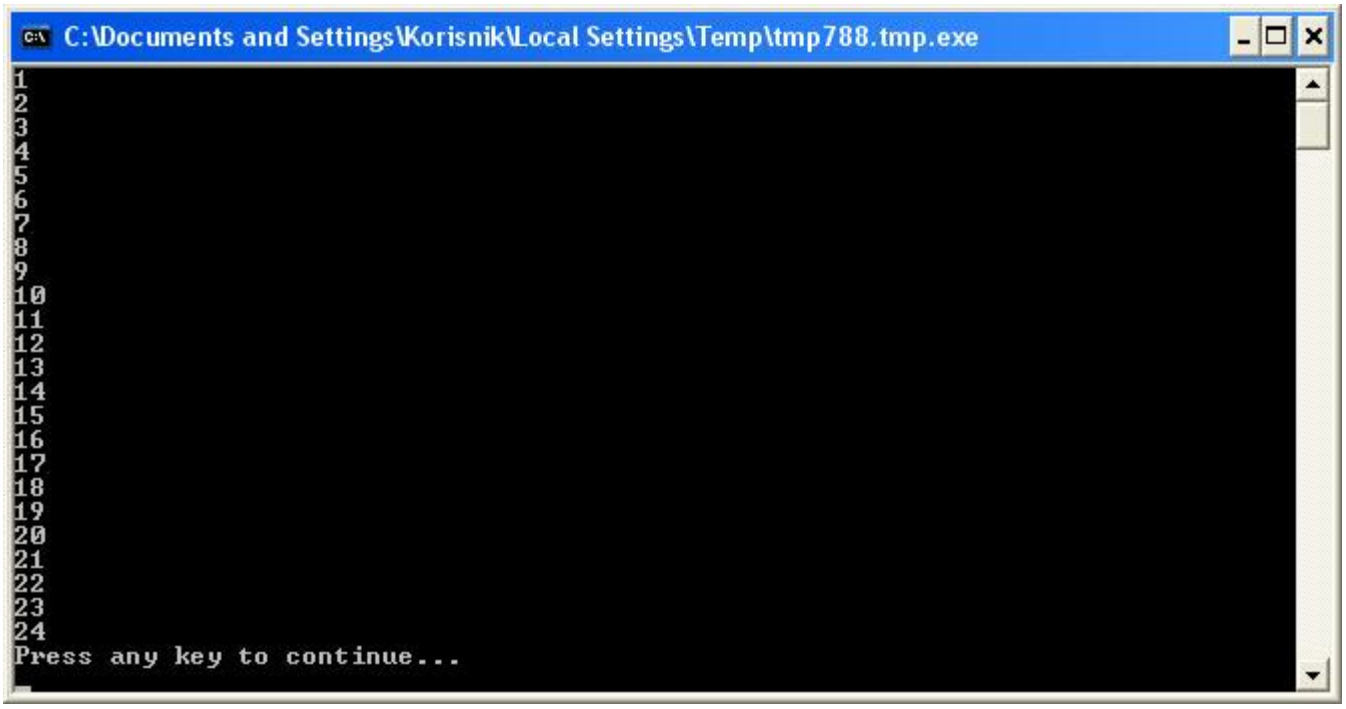
Slika 16 – program koji utvrđuje je li broj paran ili neparan

U ovom smo programu prvi put upotrijebili novu korisnu operaciju **Math.Remainder**. Kao što ste vjerojatno i sami zaključili, operacija **Math.Remainder** dijeli prvi broj drugim, a zatim vraća ostatak.

Grananje

U drugom ste poglavlju naučili da računalo redom obrađuje jedan po jedan iskaz programa od prvog do posljednjeg. No postoji poseban iskaz koji računalu omogućuje skok na neki drugi iskaz, koji nije na redu za izvršavanje. Pogledajmo sljedeći program:

```
i = 1
start:
TextWindow.WriteLine(i)
i = i + 1
If (i < 25) Then
Goto start
EndIf
```



Slika 17 – upotreba iskaza Goto

U gornjem smo programu varijabli **i** dodijelili vrijednost 1, a zatim smo dodali novi iskaz koji završava dvotočkom (:)

```
start:
```

To je *oznaka*. Oznake su svojevrsne knjižne oznake koje računalo razumije. Knjižnoj oznaci možete dodijeliti bilo koji naziv i u program možete dodati koliko god oznaka želite ako im se nazivi međusobno razlikuju.
Drugi zanimljiv iskaz u ovom programu je:

```
i = i + 1
```

Taj iskaz naređuje računalu da vrijednost varijable **i** poveća za 1 i rezultat ponovno dodijeli varijabli **i**. Dakle, ako je vrijednost varijable **i** prije izvođenja ovog iskaza bila 1, nakon njegova izvođenja bit će 2 .
I naposljetku:

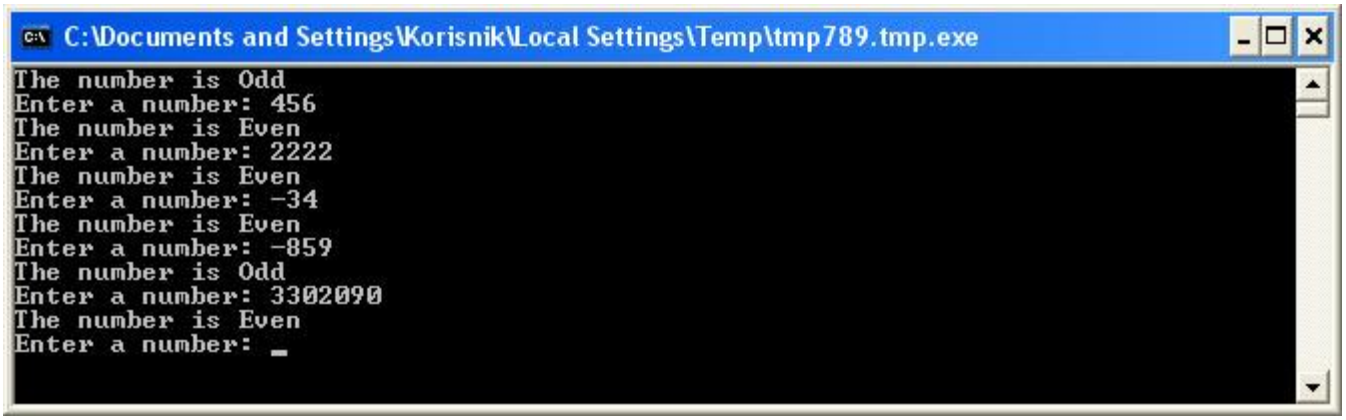
```
If (i < 25) Then
Goto start
EndIf
```

To je dio programa koji računalu naređuje da počne izvršavati iskaze od knjižne oznake **start** ako je vrijednost varijable **i** manja od 25.

Beskonačno izvođenje

Pomoću iskaza **Goto** računalu možete narediti da neku radnju ponovi određeni broj puta. Ako, primjerice, program koji utvrđuje je li broj paran ili neparan izmijenite kao u donjem primjeru, on će se neprekidno izvoditi. Izvođenje programa možete prekinuti klikom na gumb za zatvaranje (X) u gornjem desnom kutu prozora.

```
begin:
TextWindow.Write("Enter a number: ")
num = TextWindow.ReadNumber()
remainder = Math.Remainder(num, 2)
If (remainder = 0) Then
TextWindow.WriteLine("The number is Even")
Else
TextWindow.WriteLine("The number is Odd")
EndIf
Goto begin
```



Slika 18 – program koji utvrđuje je li broj paran ili neparan i koji se neprekidno izvodi

Petlje

Petlja For

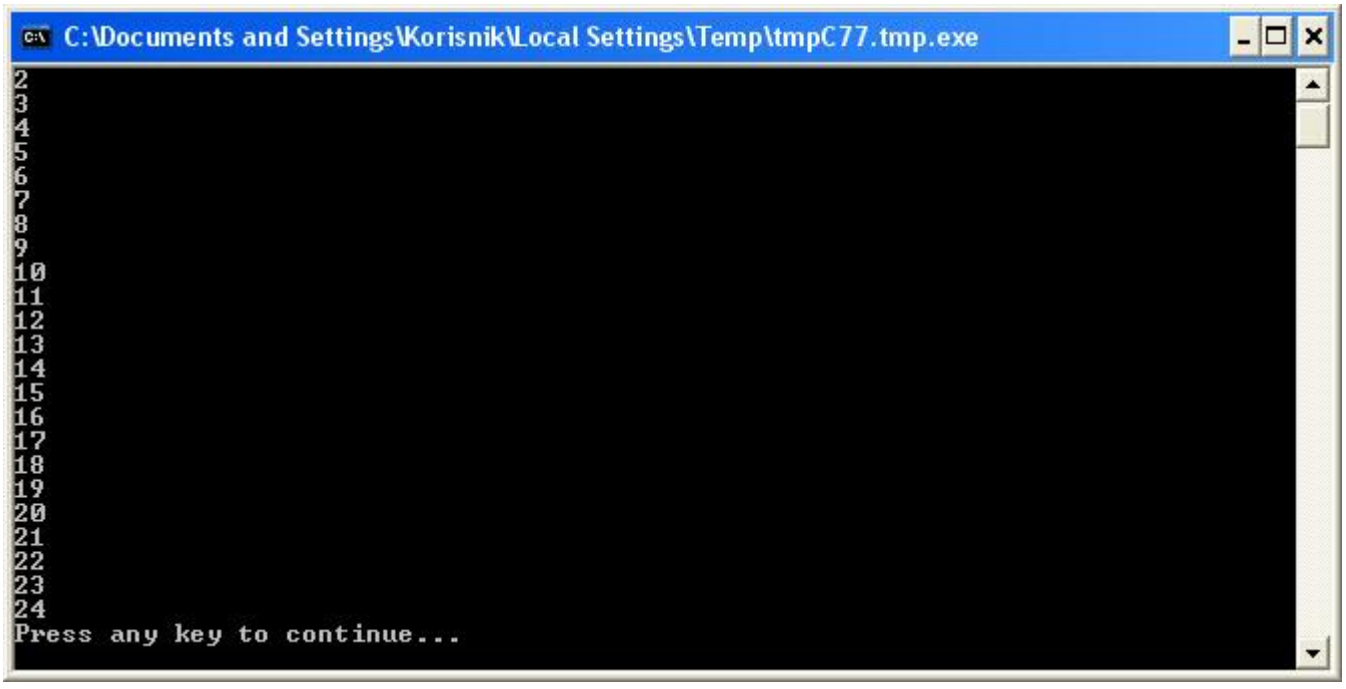
Pogledajmo program koji smo napisali u prethodnom poglavlju.

```
i = 1
start:
TextWindow.WriteLine(i)
i = i + 1
If (i < 25) Then
Goto start
EndIf
```

Taj program redom prikazuje brojeve od 1 do 24. Budući da je ovakav postupak povećavanja vrijednosti varijable uobičajen u programiranju, programski jezici obično nude jednostavniji način na koji se to može učiniti.
Gornji program ima isti učinak kao ovaj:

```
For i = 1 To 24
TextWindow.WriteLine(i)
EndFor
```

A izlaz je:

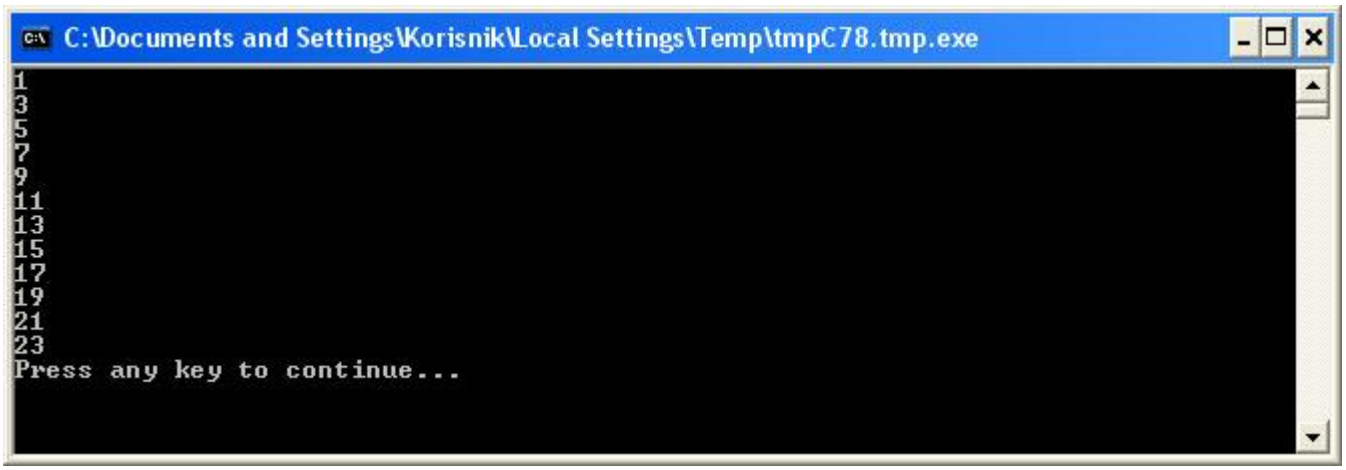


Slika 19 – upotreba petlje For

Uočite da smo program s osam redaka smanjili na program s četiri retka i da unatoč tome ima posve isti učinak kao kada je imao osam redaka! Sjećate li se da smo rekli da se ista stvar obično može postići na više načina? Ovo je odličan primjer.
Iskaz **For..EndFor** u programerskom se žargonu zove *petlja*. Pomoću petlje varijabli možete dodijeliti početnu i završnu vrijednost te postići da računalo povećava vrijednost varijable umjesto vas. Svaki put kada poveća vrijednost varijable, računalo izvršava iskaze između riječi **For** i riječi **EndFor**.

Ako želite da se vrijednost varijable svaki put poveća za 2, a ne za 1 (npr. ako želite prikazati sve neparne brojeve između 1 i 24), i to možete postići pomoću petlje.

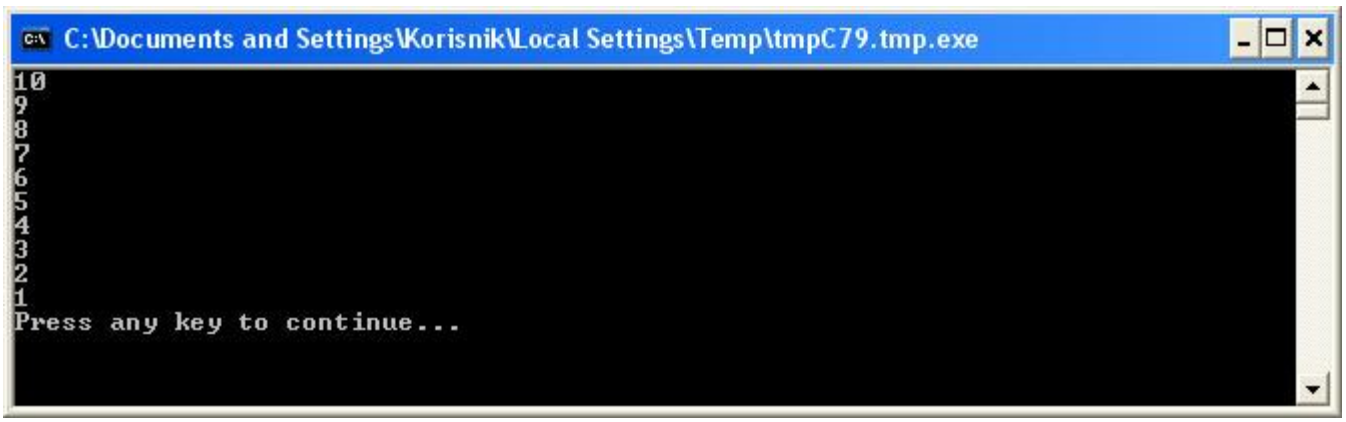
```
For i = 1 To 24 Step 2
TextWindow.WriteLine(i)
EndFor
```



Slika 20 – samo neparni brojevi

Dio **Step 2** iskaza **For** naređuje računalu da vrijednost varijable **i** povećava za 2 umjesto za uobičajenu vrijednost 1. Pomoću riječi **Step** možete odrediti bilo koji prirast. Čak možete navesti negativnu vrijednost za korak ako želite da računalo broji unatrag, kao u donjem primjeru:

```
For i = 10 To 1 Step -1
TextWindow.WriteLine(i)
EndFor
```

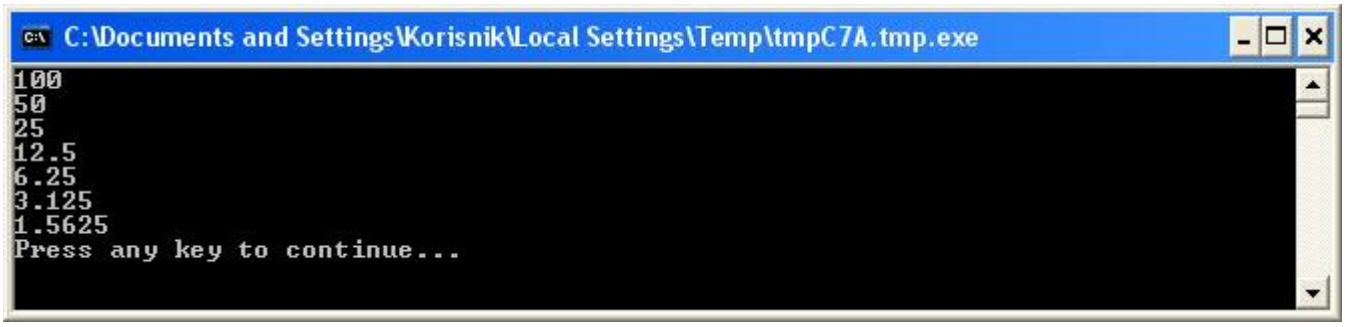


Slika 21 – brojenje unatrag

Petlja While

Petlja While drugi je način ponavljanja, koji je posebice koristan kada broj ponavljanja nije unaprijed poznat. Dok se petlja For izvodi unaprijed definirani broj puta, petlja While izvodi se dok god je ispunjen određeni uvjet. U donjem se primjeru broj dijeli s 2 dok god je rezultat veći od 1.

```
number = 100
While (number > 1)
TextWindow.WriteLine(number)
number = number / 2
EndWhile
```



Slika 22 – petlja za prepolovljivanje

U gornjem se programu varijabli *number* dodjeljuje vrijednost 100, a petlja while izvodi se dok god je vrijednost varijable number veća od 1. Unutar petlje prikazuju se brojevi, a zatim se dijele s 2, odnosno prepolovljuju se. I kao što se moglo očekivati, izlaz programa je niz brojeva u kojemu je svaki sljedeći broj dvostruko manji od prethodnog.
Ovaj bi program bilo vrlo teško napisati pomoću petlje For jer ne znamo koliko se puta petlja mora izvesti. Pomoću petlje While jednostavno je provjeriti je li ispunjen neki uvjet i narediti računalu da nastavi izvoditi petlju ili da prekine njezino izvođenje.
Valja napomenuti da se svaka petlja While može razviti u iskaz If..Then. Gornji program, primjerice, može se na sljedeći način prepraviti bez utjecaja na konačni rezultat.

```
number = 100
startLabel:
TextWindow.WriteLine(number)
number = number / 2
If (number > 1) Then
Goto startLabel
EndIf
```

Zapravo, računalo interno prepravlja svaku petlju While u iskaze koji koriste If..Then i jedan ili više iskaza Goto.

Osnove grafike

Dosad smo u svim primjerima koristili objekt TextWindow da bismo objasnili osnove jezika Small Basic. No Small Basic nudi napredan skup grafičkih mogućnosti, koje ćemo početi upoznavati u ovom poglavlju.

Prozor za grafiku

Osim prozora za tekst, koji omogućuje rad s tekstom i brojevima, Small Basic nudi i **prozor za grafiku**,koji omogućuje crtanje. Prikažimo za početak prozor za grafiku.

```
GraphicsWindow.Show()
```

Kada pokrenete ovaj program, primijetit ćete da se umjesto uobičajenog crnog prozora za tekst pojavio bijeli prozor kao što je ovaj na donjoj slici. U tom prozoru zasad ne možemo učiniti mnogo. No to će biti osnovni prozor koji ćemo koristiti u ovom poglavlju. Taj prozor možete zatvoriti klikom na gumb "X" u njegovu gornjem desnom kutu.

Slika 23 – prazan prozor za grafiku

Postavljanje prozora za grafiku

Izgled prozora za grafiku možete prilagoditi po svom ukusu. Možete mu promijeniti naslov, pozadinu i veličinu. Izmijenimo ga onda malo da biste ga bolje upoznali.

```
GraphicsWindow.BackgroundColor = "SteelBlue"
GraphicsWindow.Title = "My Graphics Window"
GraphicsWindow.Width = 320
GraphicsWindow.Height = 200
GraphicsWindow.Show()
```

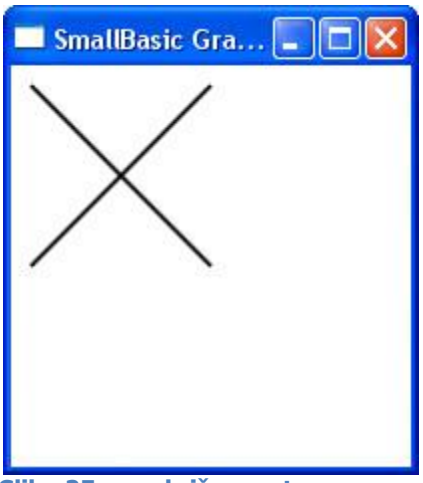
Evo kako izgleda prilagođeni prozor za grafiku. Boju možete promijeniti u jednu od brojnih vrijednosti navedenih na popisu u dodatku B. Eksperimentirajte s tim svojstvima da biste vidjeli kako sve možete promijeniti izgled prozora.

Slika 24 – prilagođeni prozor za grafiku

Crtanje crta

Kada otvorite prozor za grafiku, u njemu možete crtati likove, tekst, pa čak i slike. Nacrtajmo najprije nekoliko jednostavnih likova. Ovako izgleda program koji u prozoru za grafiku crta dvije crte:

```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Slika 25 – prekrižene crte

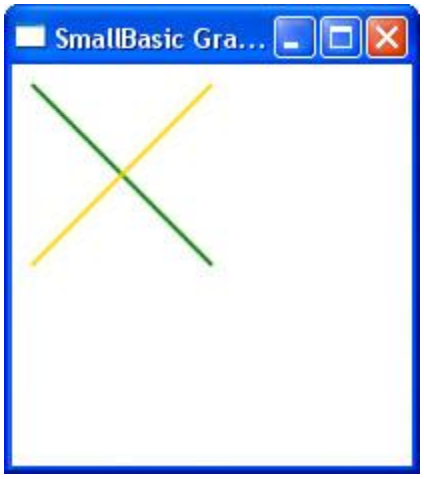
Umjesto naziva boja možete koristiti web-notaciju za boje (#RRGGBB). Primjerice, #FF0000 označava crvenu, #FFFF00 žutu itd. O bojama će biti više riječi u poglavljima [poglavlja o bojama U IZRADI].

Prva dva retka programa postavljaju prozor, a sljedeća dva crtaju prekrižene crte. Prva dva broja iza riječi *DrawLine* određuju koordinate x i y početne točke crte, a druga dva koordinate x i y završne točke crte. Zanimljivo je da se u računalnoj grafici koordinate (0, 0) odnose na točku u gornjem lijevom kutu prozora. Zato se, prema koordinatnom sustavu, prozor nalazi u drugom kvadrantu.
[U IZRADI: umetnuti sliku kvadranta]

Slika 26 – koordinatna karta

Ako se vratimo programu za crtanje crta, valja napomenuti da Small Basic omogućuje promjenu svojstava crte, primjerice njezine boje i debljine. Promijenimo najprije boju crta na način prikazan u donjem programu.

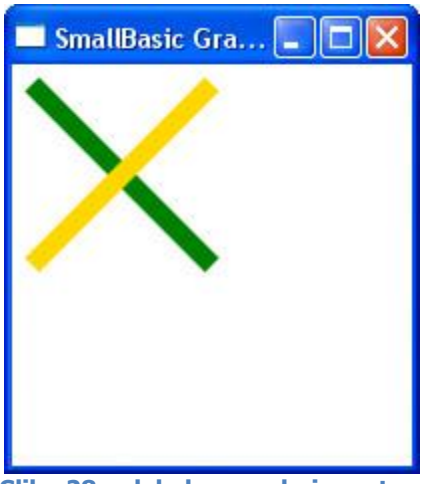
```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Slika 27 – promjena boje crte

Promijenimo im i debljinu. U donjem se programu zadana vrijednost 1 za debljinu crte mijenja u 10.

```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenWidth = 10
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Slika 28 – debele raznobojne crte

Svojstva *PenWidth* i *PenColor* mijenjaju olovku kojom se te crte crtaju. Ona ne samo što utječu na crte, već i na sve likove nacrtane nakon njihova ažuriranja.

Pomoću iskaza za stvaranje petlji o kojima je bilo riječi u prethodnim poglavljima možemo jednostavno napisati program koji crta više crta povećavajući debljinu olovke za svaku sljedeću crtu.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 160
GraphicsWindow.PenColor = "Blue"
For i = 1 To 10
GraphicsWindow.PenWidth = i
GraphicsWindow.DrawLine(20, i * 15, 180, i * 15)
endfor
```

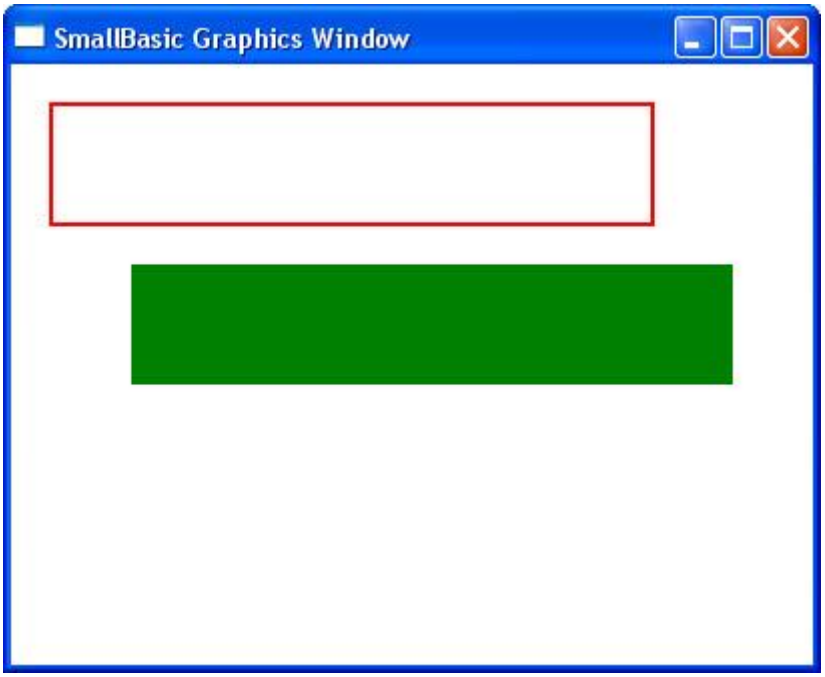
Slika 29 – različite debljine olovke

Zanimljiv dio ovog programa je petlja, u kojoj se vrijednost svojstva *PenWidth* povećava pri svakom izvođenju petlje, nakon čega se crta nova crta ispod prethodne.

Crtanje i ispunjavanje likova

Kada je riječ o crtanju likova, na svakom se liku obično mogu obavljati dvije vrste operacija. Riječ je o operacijama crtanja (*Draw*) i o operacijama ispunjavanja (*Fill*). Operacije crtanja omogućuju crtanje ocrta lika olovkom, a operacije ispunjavanja bojenje lika kistom. U donjem se programu, primjerice, stvaraju dva pravokutnika – jedan se crta crvenom olovkom, a drugi se ispunjava zelenim kistom.

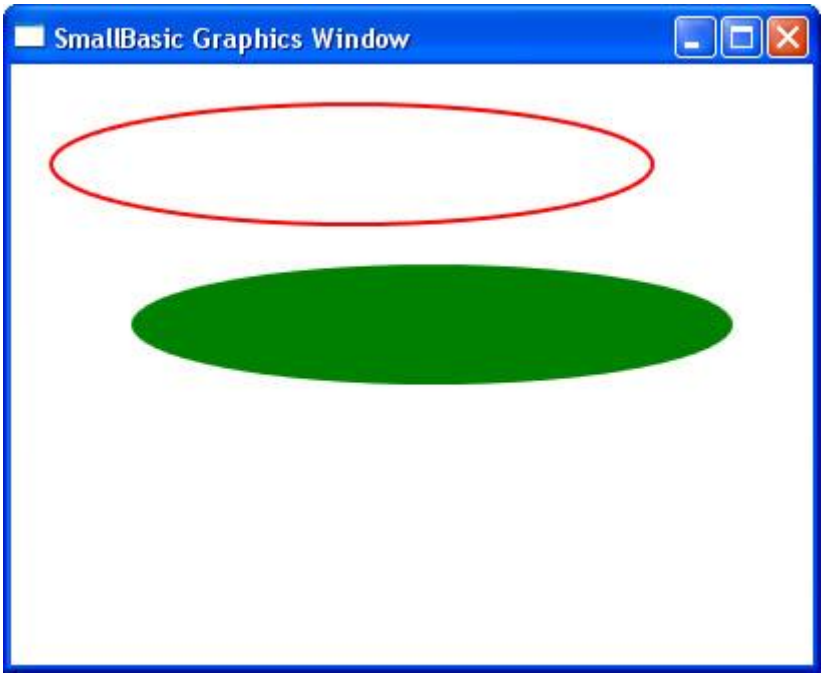
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300
GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawRectangle(20, 20, 300, 60)
GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillRectangle(60, 100, 300, 60)
```



Slika 30 – crtanje i ispunjavanje

Za crtanje ili ispunjavanje pravokutnika potrebna su četiri broja. Prva dva broja predstavljaju koordinate X i Y gornjeg lijevog kuta pravokutnika. Treći broj određuje širinu pravokutnika, a četvrti njegovu visinu. Kao što možete vidjeti u donjem programu, isto vrijedi i za crtanje i ispunjavanje elipsa.

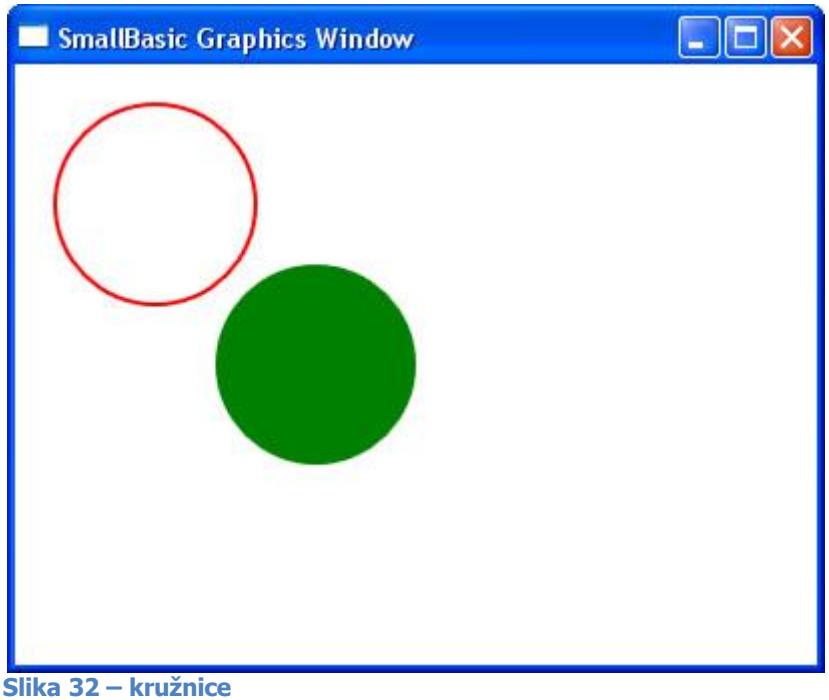
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300
GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawEllipse(20, 20, 300, 60)
GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillEllipse(60, 100, 300, 60)
```



Slika 31 – stvaranje i crtanje elipsa

Elipse su samo općenita kategorija kružnica. Ako želite nacrtati kružnice, morate navesti istu širinu i visinu.

```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300
GraphicsWindow.PenColor = "Red"
GraphicsWindow.DrawEllipse(20, 20, 100, 100)
GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillEllipse(100, 100, 100, 100)
```



Slika 32 – kružnice

Zabava s likovima

U ovom ćemo se poglavlju malo zabaviti primjenjujući sva dosad stečena znanja. Ovo poglavlje sadrži zanimljive primjere koji pokazuju kako primjenom svih dosad stečenih znanja možete stvarati atraktivne programe.

Koncentrični pravokutnici

U ovom ćemo primjeru pomoću petlje nacrtati više pravokutnika povećavajući svaki sljedeći.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightBlue"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
For i = 1 To 100 Step 5
GraphicsWindow.DrawRectangle(100 - i, 100 - i, i * 2, i
* 2)
EndFor
```

Slika 33 – koncentrični pravokutnici

Koncentrične kružnice

Verzija prethodnog programa koja umjesto pravokutnika crta kružnice.

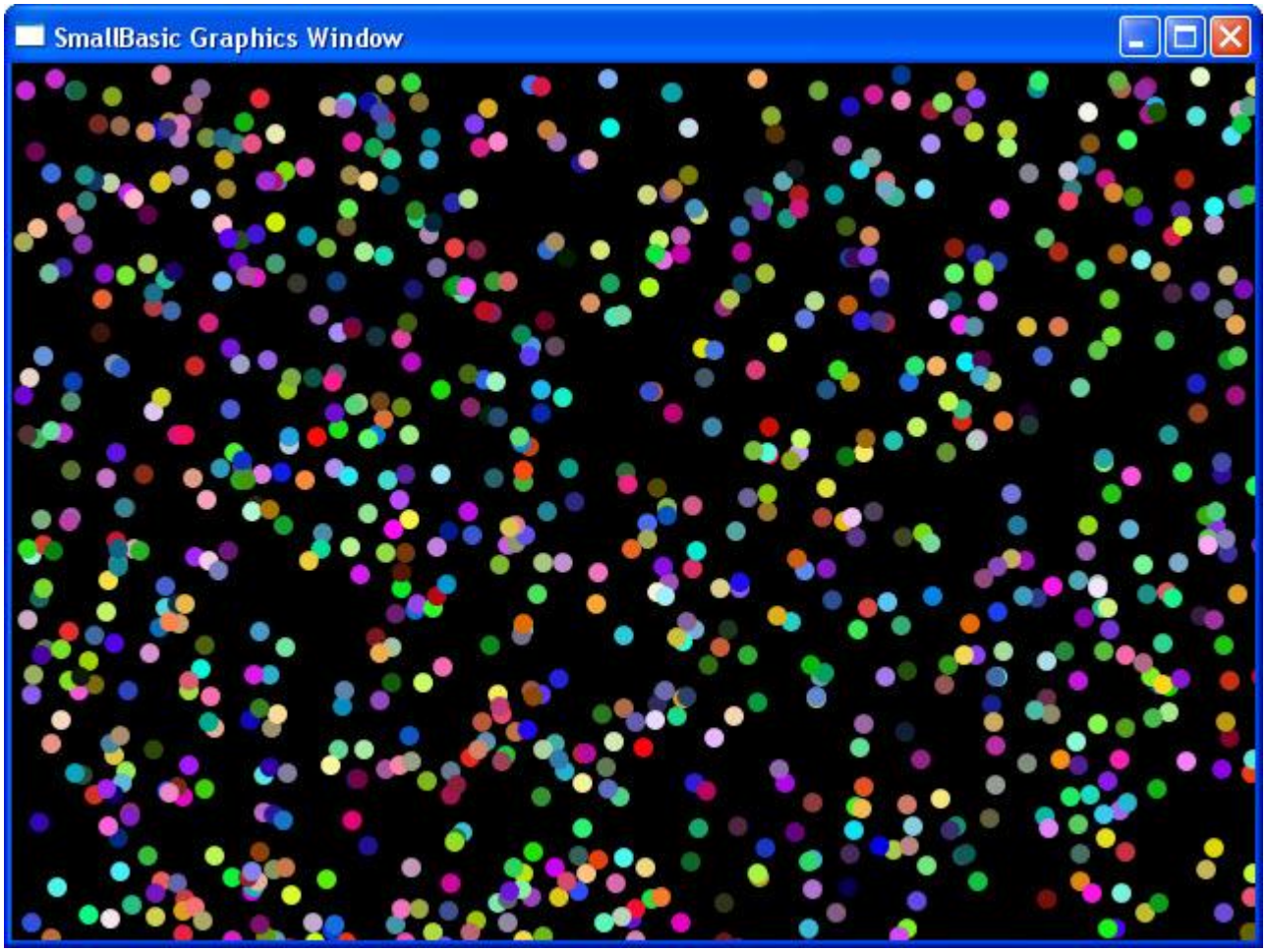
```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightGreen"
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
For i = 1 To 100 Step 5
GraphicsWindow.DrawEllipse(100 - i, 100 - i, i * 2, i *
2)
EndFor
```

Slika 34 – koncentrične kružnice

Nasumično crtanje

Ovaj program pomoću operacije GraphicsWindow.*GetRandomColor* nasumično bira boju kista, a zatim pomoću operacije Math.*GetRandomNumber* određuje koordinate x i y kružića. Kreativnim kombiniranjem tih dviju operacija možete stvarati zanimljive programe čiji se ishod mijenja sa svakim izvođenjem.

```
GraphicsWindow.BackgroundColor = "Black"
For i = 1 To 1000
GraphicsWindow.BrushColor =
GraphicsWindow.GetRandomColor()
x = Math.GetRandomNumber(640)
y = Math.GetRandomNumber(480)
GraphicsWindow.FillEllipse(x, y, 10, 10)
EndFor
```

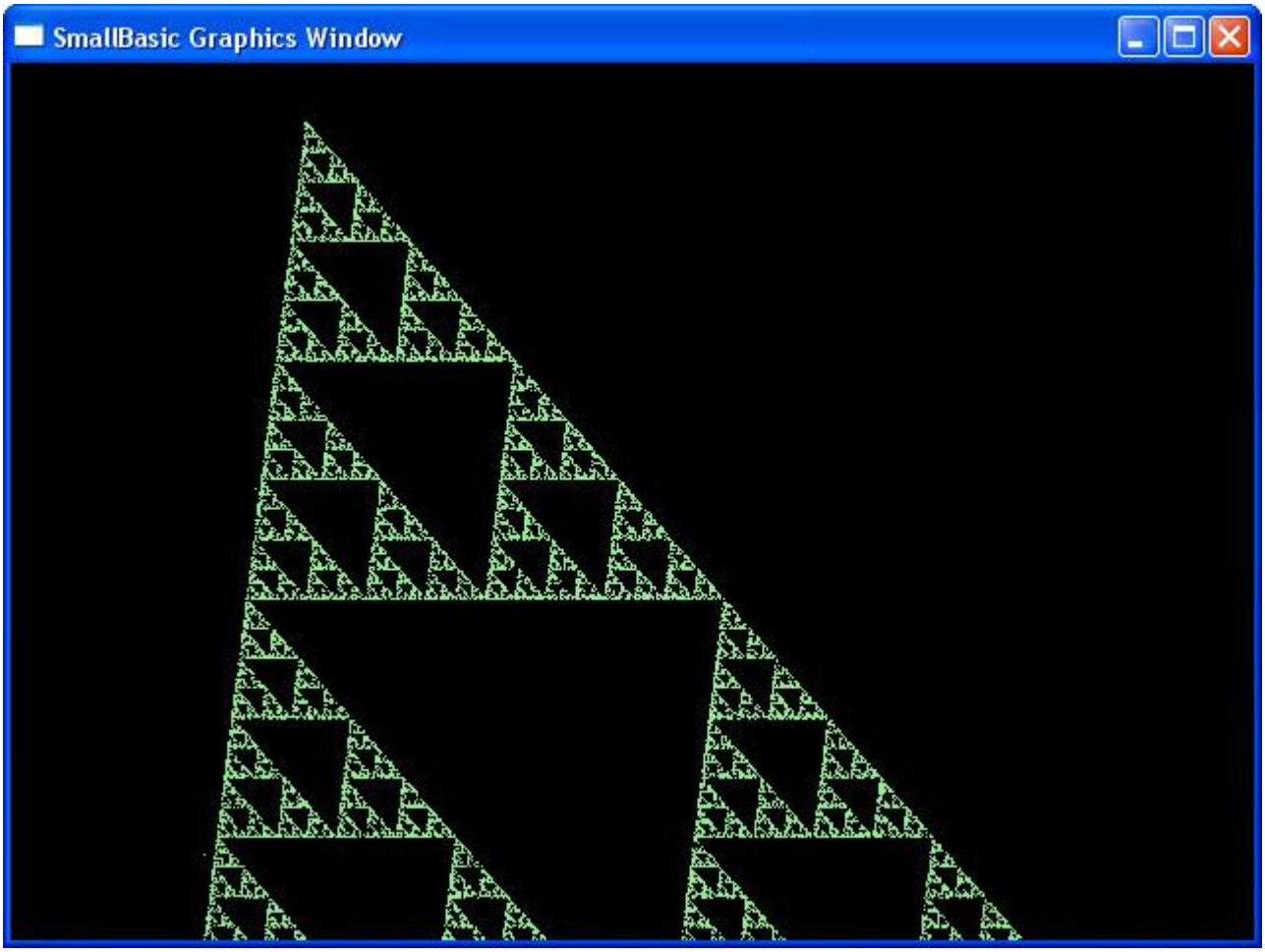


Slika 35 – nasumično crtanje

Fraktali

Sljedeći program crta jednostavan trokutasti fraktal pomoću nasumično odabranih brojeva. Fraktal je geometrijski lik koji se može podijeliti na manje dijelove od kojih je svaki identičan nadređenom liku. Program u ovom primjeru crta stotinu trokuta od kojih je svaki identičan nadređenom trokutu. Budući da izvođenje programa traje nekoliko sekundi, moći ćete vidjeti kako trokuti polako nastaju iz običnih točaka. Logiku programa nije lako opisati i njezino ću vam proučavanje prepustiti kao zadatak za vježbu.

```
GraphicsWindow.BackgroundColor = "Black"
x = 100
y = 100
For i = 1 To 100000
r = Math.GetRandomNumber(3)
ux = 150
uy = 30
If (r = 1) then
ux = 30
uy = 1000
EndIf
If (r = 2) Then
ux = 1000
uy = 1000
EndIf
x = (x + ux) / 2
y = (y + uy) / 2
GraphicsWindow.SetPixel(x, y, "LightGreen")
EndFor
```



Slika 36 – trokutasti fraktal

Ako želite bolje vidjeti kako točke polako stvaraju fraktal, u petlju možete dodati odgodu pomoću operacije **Program.***Delay.* Ta operacija prihvaća broj koji određuje trajanje odgode u milisekundama. Ovo je izmijenjeni program s podebljanim izmijenjenim retkom:

```
GraphicsWindow.BackgroundColor = "Black"
x = 100
y = 100
For i = 1 To 100000
r = Math.GetRandomNumber(3)
ux = 150
uy = 30
If (r = 1) then
ux = 30
uy = 1000
EndIf
If (r = 2) Then
ux = 1000
uy = 1000
EndIf
x = (x + ux) / 2
y = (y + uy) / 2
GraphicsWindow.SetPixel(x, y, "LightGreen")
Program.Delay(2)
EndFor
```

Produljivanjem odgode usporit ćete izvođenje programa. Eksperimentirajte s brojevima da biste vidjeli koji daje najbolje rezultate.
Taj program možete izmijeniti i zamjenom retka

```
GraphicsWindow.SetPixel(x, y, "LightGreen")
```

GraphicsWindow.SetPixel(x, y, "LightGreen")

retkom

```
color = GraphicsWindow.GetRandomColor()
GraphicsWindow.SetPixel(x, y, color)
```

Tako ćete narediti programu da piksele trokuta crta pomoću nasumično odabranih boja.

Crtanje pomoću kornjače

Logo

U sedamdesetim godinama 20. stoljeća nastao je vrlo jednostavan, ali mogućnostima bogat programski jezik zvan Logo, koji je tada koristilo samo nekoliko znanstvenika. Situacija se promijenila kada je netko obogatio jezik tzv. "crtanjem pomoću kornjače" (Turtle Graphics) i na zaslonu učinio dostupnom "kornjaču", koja je reagirala na naredbe kao što su *Move Forward* (pomakni se prema naprijed), *Turn Right* (okreni se udesno), *Turn Left* (okreni se ulijevo) itd. Kornjača je korisnicima omogućivala crtanje zanimljivih likova na zaslonu. Ta je mogućnost programski jezik Logo odmah učinila pristupačnim i privlačnim korisnicima svih uzrasta te je velikim dijelom bila zaslužna za njegovu ogromnu popularnost osamdesetih godina.
Small Basic nudi objekt **Turtle** s brojnim naredbama koje se mogu pozivati iz programa u Small Basicu. U ovom ćemo poglavlju pomoću kornjače crtati grafiku na zaslonu.

Kornjača

Najprije moramo prikazati kornjaču na zaslonu. To je izvedivo pomoću jednostavnog programa s jednim retkom.

```
Turtle.Show()
```

Kada pokrenete taj program, prikazat će se bijeli prozor sličan onome iz prethodnog poglavlja, samo što se u središtu tog prozora nalazi kornjača. Ta će kornjača izvršavati naše naredbe i crtati sve što joj naredimo.

Slika 37 – kornjača je prikazana

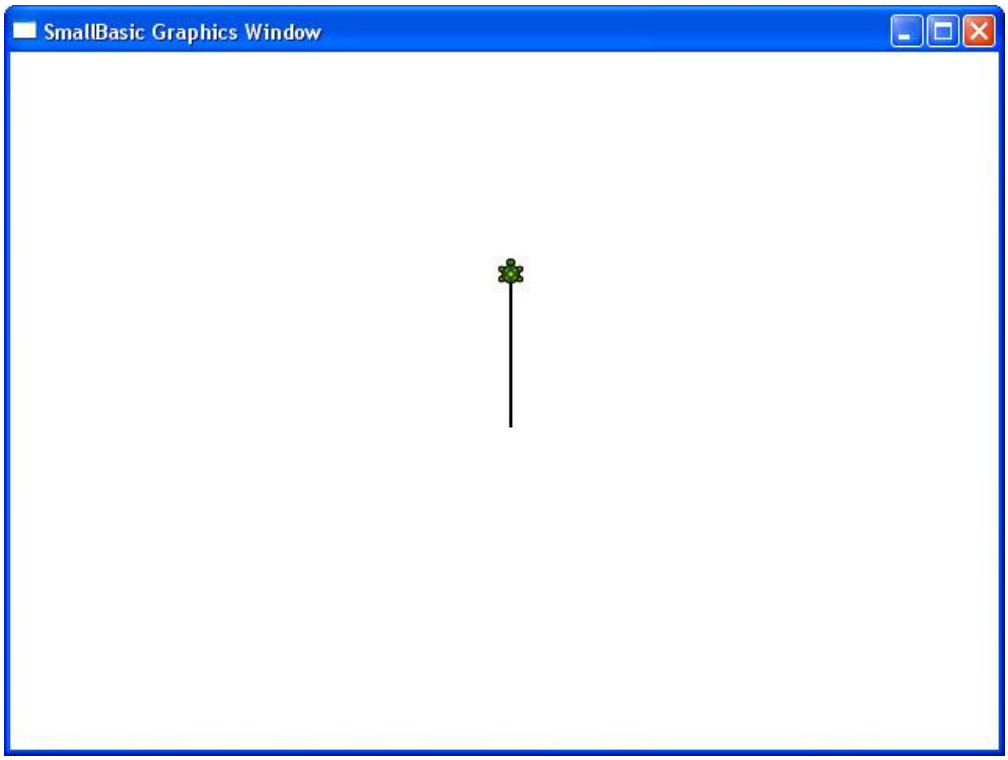
Pomicanje i crtanje

Jedna od naredbi koju kornjača razumije jest **Move** (pomakni se). Ta operacija prihvaća broj kao ulaznu vrijednost. Taj broj govori kornjači koliko se daleko mora pomaknuti. U donjem ćemo primjeru narediti kornjači da se pomakne za 100 piksela.

```
Turtle.Move(100)
```

Kada pokrenete taj program, vidjet ćete kako se kornjača polako pomiče za 100 piksela prema gore. Kako se bude pomicala, kornjača će crtati crtu. Kada se kornjača zaustavi, rezultat njezina pomicanja izgledat će kao na donjoj slici.

Kada koristite operacije na kornjači, ne morate pozivati operaciju Show(). Kornjača će se automatski prikazati kad god se izvrši neka uz nju vezana operacija.



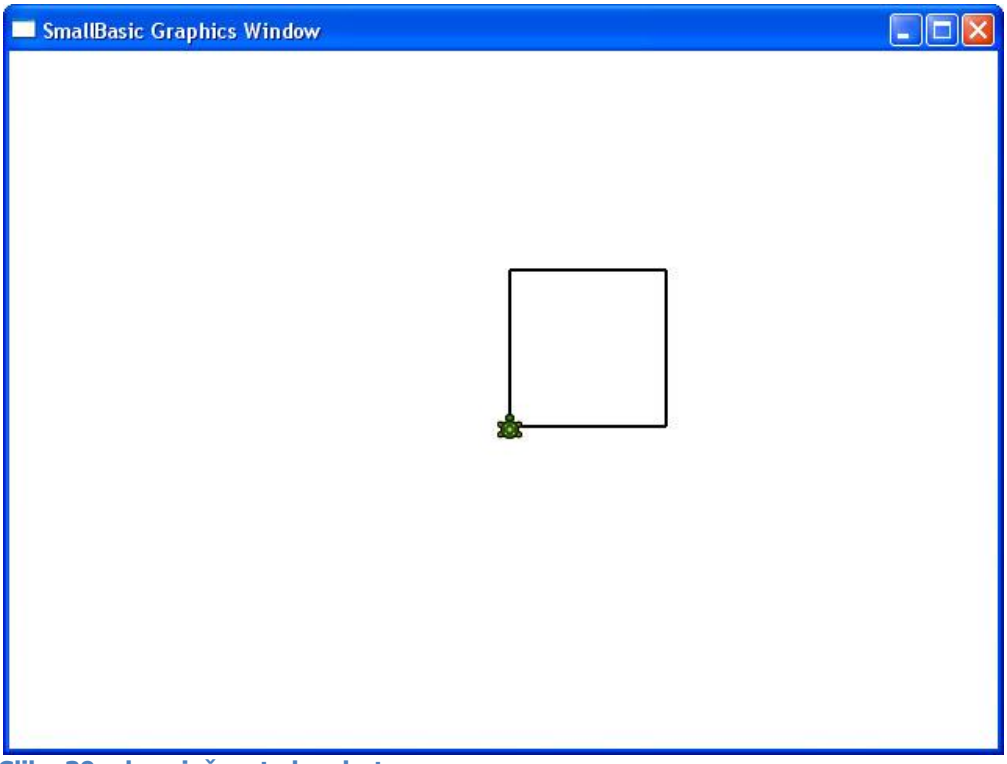
Slika 38 – pomicanje za stotinu piksela

Crtanje kvadrata

Kvadrat ima četiri stranice – dvije okomite i dvije vodoravne. Da bismo nacrtali kvadrat, moramo narediti kornjači da nacrta crtu, okrene se udesno, nacrta sljedeću crtu i nastavi to činiti dok ne nacrta sve četiri stranice. Ako to pretvorimo u program, on će izgledati ovako:

```
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
Turtle.Move(100)
Turtle.TurnRight()
```

Kada pokrenete taj program, vidjet ćete da kornjača crta kvadrat jednu po jednu crtu, a rezultat će izgledati kao na donjoj slici.



Slika 39 – kornjača crta kvadrat

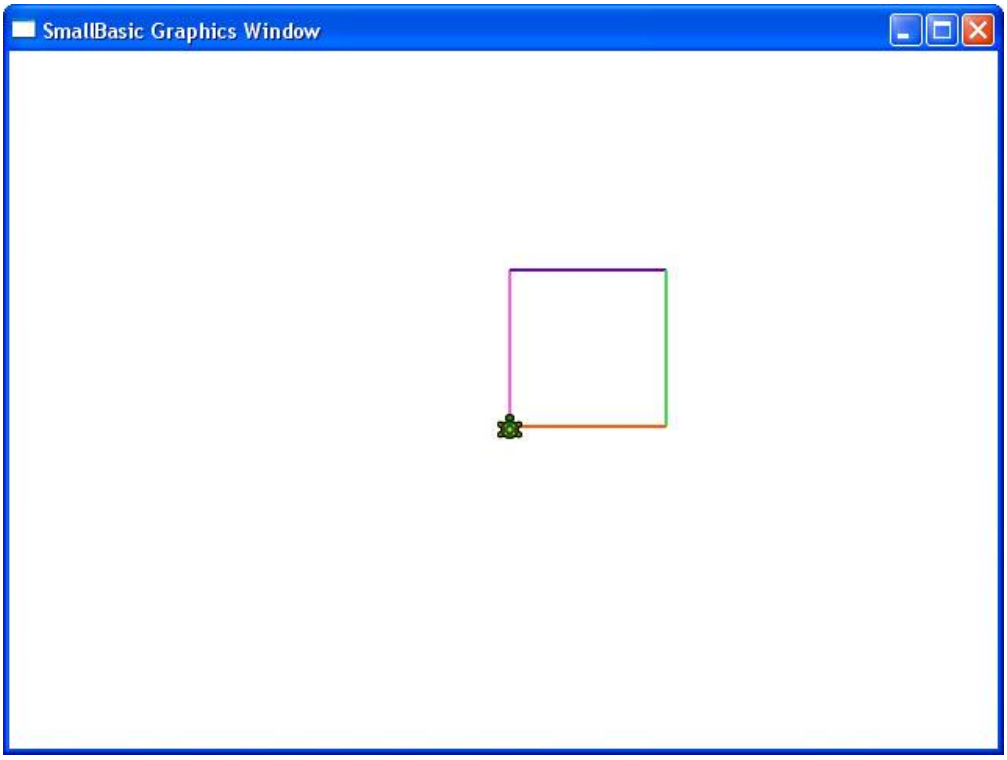
Uočite da smo više puta uputili iste naredbe – točnije četiri puta. A već ste naučili da se naredbe koje se ponavljaju mogu izvoditi pomoću petlji. Ako gornji program izmijenimo tako da koristi petlju **For..EndFor,** dobit ćemo mnogo jednostavniji program.

```
For i = 1 To 4
Turtle.Move(100)
Turtle.TurnRight()
EndFor
```

Promjena boja

Kornjača crta u istom prozoru za grafiku koji ste vidjeli u prethodnom poglavlju. To znači da se sve operacije opisane u prethodnom poglavlju mogu koristiti i ovdje. Sljedeći će program, primjerice, nacrtati kvadrat kojemu je svaka stranica drukčije boje.

```
For i = 1 To 4
GraphicsWindow.PenColor =
GraphicsWindow.GetRandomColor()
Turtle.Move(100)
Turtle.TurnRight()
EndFor
```



Slika 40 – promjena boja

Crtanje složenijih likova

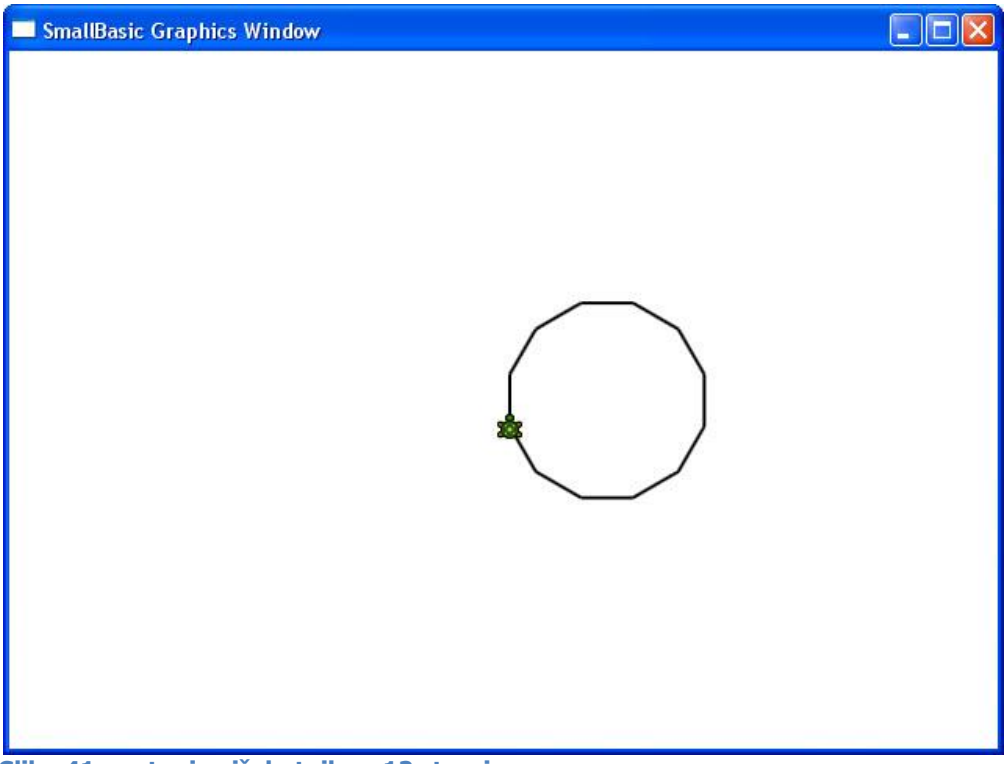
Osim operacija **TurnRight** i **TurnLeft**,kornjača podržava i operaciju **Turn**. Ta operacija prihvaća jednu ulaznu vrijednost koja određuje kut zakretanja. Pomoću te operacije možete nacrtati višekutnik s bilo kojim brojem stranica. Sljedeći program crta šesterokut (višekutnik sa šest stranica).

```
For i = 1 To 6
Turtle.Move(100)
Turtle.Turn(60)
EndFor
```

Iskušajte taj program da biste vidjeli crta li zaista šesterokut. Uočite da smo, budući da je kut između stranica šesterokuta 60 stupnjeva, koristili operaciju **Turn(60)**. Za takve višekutnike, kojima su sve stranice iste dužine, kut između stranica može se jednostavno izračunati tako da 360 podijelite s brojem stranica. Oboružani tim informacijama i uz pomoć varijabli možemo napisati prilično generički program koji crta višekutnik s bilo kojim brojem stranica.

```
sides = 12
length = 400 / sides
angle = 360 / sides
For i = 1 To sides
Turtle.Move(length)
Turtle.Turn(angle)
EndFor
```

Pomoću tog programa možete nacrtati višekutnik s bilo kojim brojem stranica – dovoljno je promijeniti vrijednost varijable **sides**. Ako joj dodijelite vrijednost 4, dobit ćete kvadrat, od kojeg smo krenuli. Ako joj dodijelite neku dovoljno veliku vrijednost, recimo 50, nacrtani se višekutnik neće razlikovati od kružnice.

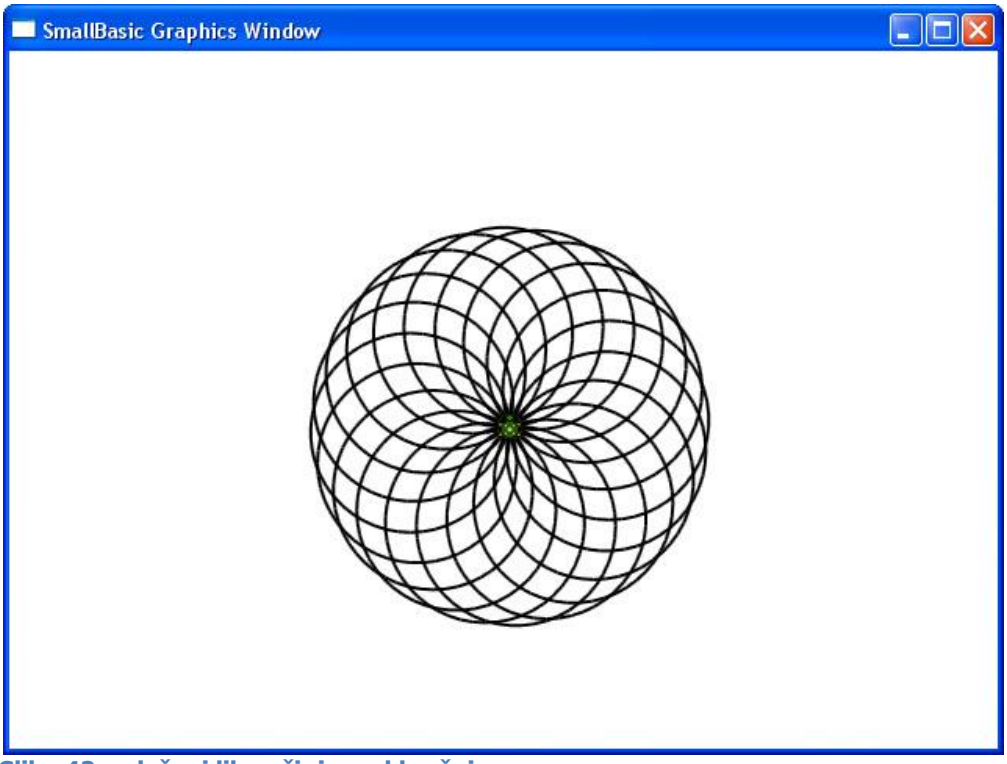


Slika 41 – crtanje višekutnika s 12 stranica

Pomoću postupka koji ste upravo naučili kornjači možemo narediti da nacrta više kružnica i da svaku malo pomakne u odnosu na prethodnu, što će imati zanimljiv ishod.

```
sides = 50
length = 400 / sides
angle = 360 / sides
Turtle.Speed = 9
For j = 1 To 20
For i = 1 To sides
Turtle.Move(length)
Turtle.Turn(angle)
EndFor
Turtle.Turn(18)
EndFor
```

Gornji program koristi dvije petlje **For..EndFor**, jednu unutar druge. Unutrašnja petlja (*i = 1 to sides*) slična je onoj u programu za crtanje višekutnika i zadužena je za crtanje kružnice. Vanjska petlja (*j = 1 to 20*) zadužena je za neznatno zakretanje kornjače pri crtanju svake kružnice. Ta petlja naređuje kornjači da nacrta 20 kružnica. Ishod izvođenja cijelog programa vrlo je zanimljiv uzorak, prikazan na donjoj slici.



Slika 42 – složeni lik načinjen od kružnica

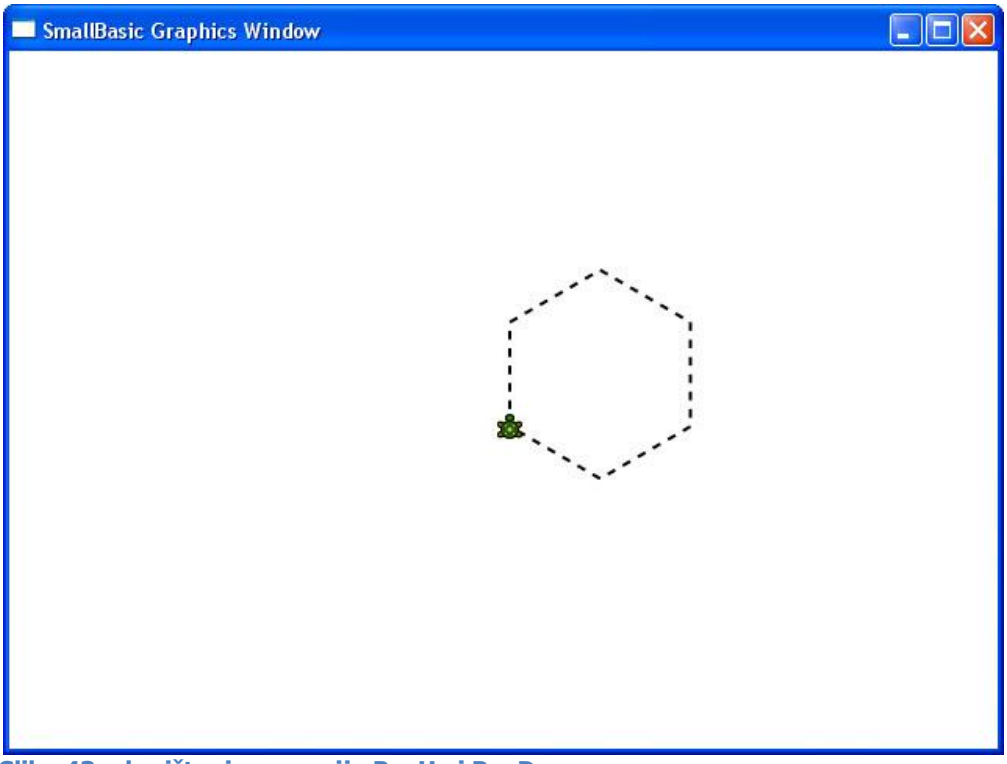
U gornjem smo programu odredili da se kornjača brže kreće tako što smo svojstvu Speed dodijelili vrijednost 9. Tom svojstvu možete dodijeliti bilo koju vrijednost između 1 i 10 da biste odredili koliko će se brzo kornjača kretati.

Pomicanje bez crtanja

Kornjači možete onemogućiti crtanje pozivanjem operacije **PenUp**. Tako možete pomaknuti kornjaču u bilo koji dio zaslona, a da pritom ne nacrtate crtu. Pozivanjem operacije **PenDown** kornjači ćete ponovno omogućiti crtanje. Tako možete dobiti zanimljive efekte, primjerice isprekidanu crtu. Sljedeći program koristi tu mogućost za crtanje višekutnika čije su stranice isprekidane crte.

```
sides = 6
length = 400 / sides
angle = 360 / sides
For i = 1 To sides
For j = 1 To 6
Turtle.Move(length / 12)
Turtle.PenUp()
Turtle.Move(length / 12)
Turtle.PenDown()
EndFor
Turtle.Turn(angle)
EndFor
```

I ovaj program ima dvije petlje. Unutrašnja petlja crta jednu isprekidanu crtu, a vanjska određuje koliko će crta biti nacrtano. Budući da smo u ovom primjeru varijabli **sides** dodijelili vrijednost 6, dobili smo šesterokut čije su stranice isprekidane crte, kao što je ovaj na donjoj slici.



Slika 43 – korištenje operacija PenUp i PenDown

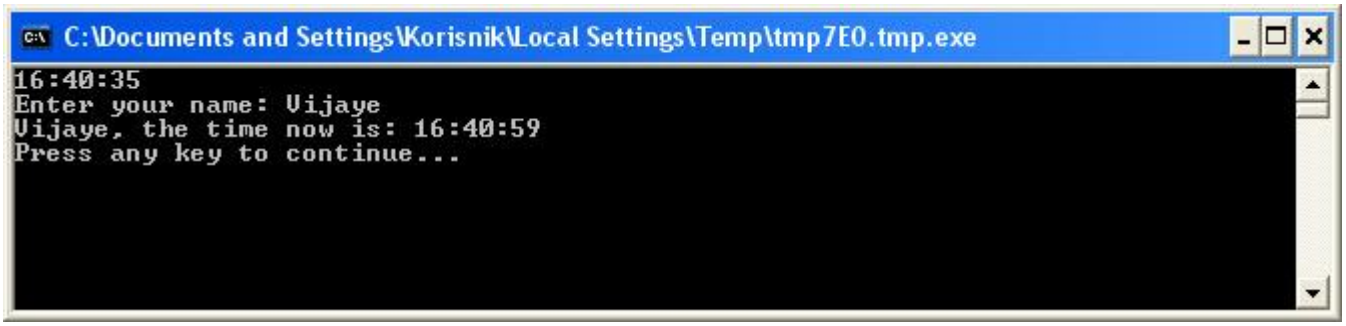
Potprogrami

Pri pisanju programa česte su situacije u kojima je potrebno više puta ponoviti isti skup koraka. U takvim slučajevima nema smisla više puta ponovno pisati iste iskaze. Tada su korisni *potprogrami*.
Potprogram je dio koda unutar većeg programa koji obično čini nešto vrlo specifično i koji se može pozvati iz bilo kojeg dijela programa. Potprogrami se prepoznaju po nazivu iza ključne riječi **Sub** i po tome što završavaju ključnom riječju **EndSub**. Sljedeći dio koda predstavlja potprogram pod nazivom *PrintTime*, koji prikazuje trenutno vrijeme u prozoru za tekst.

```
Sub PrintTime
TextWindow.WriteLine(Clock.Time)
EndSub
```

Donji program sadrži potprogram i poziva ga s različitih mjesta.

```
PrintTime()
TextWindow.Write("Enter your name: ")
name = TextWindow.Read()
TextWindow.Write(name + ", the time now is: ")
PrintTime()
Sub PrintTime
TextWindow.WriteLine(Clock.Time)
EndSub
```



Slika 44 – pozivanje jednostavnog potprograma

Potprogram se izvršava pozivanjem operacije *NazivPotprograma()*. Kao i obično, rečenični znakovi "()" potrebni su da bi računalo znalo da želite izvršiti potprogram.

Prednosti korištenja potprograma

Kao što ste vidjeli u prethodnom primjeru, potprogrami smanjuju količinu koda koji je potrebno upisati. Kada napišete potprogram *PrintTime*, možete ga pozvati iz bilo kojeg dijela programa i on će prikazati trenutno vrijeme.
Uz to, potprogrami omogućuju rastavljanje složenih problema na jednostavnije dijelove. Primjerice, ako morate riješiti neku složenu jednadžbu, možete napisati nekoliko potprograma koji rješavaju njezine manje dijelove. Zatim možete međusobno povezati rezultate da biste dobili rješenje izvorne složene jednadžbe. Potprogrami mogu pridonijeti i čitljivosti programa. Drugim riječima, ako napišete primjereno imenovane potprograme za dijelove potprograma koji se često izvode, olakšat ćete čitanje i razumijevanje programa. To je vrlo važno ako želite razumjeti program koji je napisao netko drugi ili ako želite da vaše

programe razumiju drugi korisnici. Katkad su potprogrami korisni čak i ako želite pročitati vlastiti program, primjerice tjedan dana nakon što ga napišete.

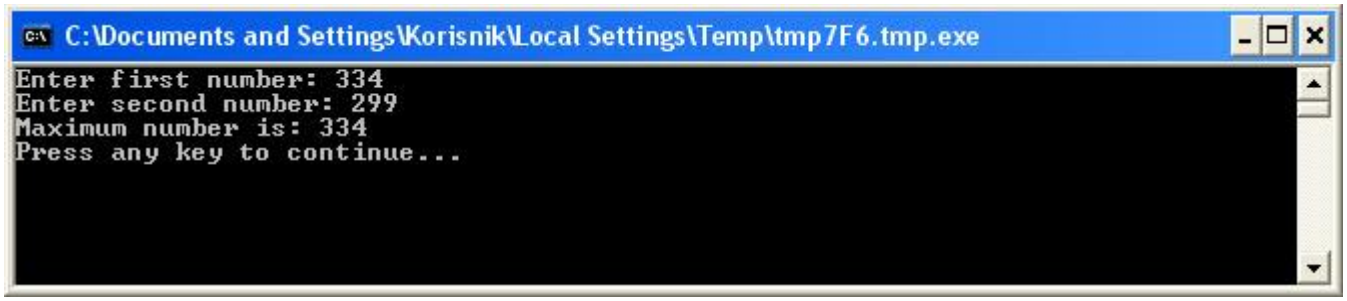
Imajte u vidu da potprograme u Small Basicu možete pozivati samo unutar istog programa. Ne možete pozvati potprogram iz nekog drugog programa.

Korištenje varijabli

Varijablama korištenima u programu možete pristupiti i koristiti ih unutar potprograma. Sljedeći program, na primjer, prihvaća dva broja i prikazuje veći broj. Uočite da se varijabla *max* koristi i unutar i izvan potprograma.

```
TextWindow.Write("Enter first number: ")
num1 = TextWindow.ReadNumber()
TextWindow.Write("Enter second number: ")
num2 = TextWindow.ReadNumber()
FindMax()
TextWindow.WriteLine("Maximum number is: " + max)
Sub FindMax
If (num1 > num2) Then
max = num1
Else
max = num2
EndIf
EndSub
```

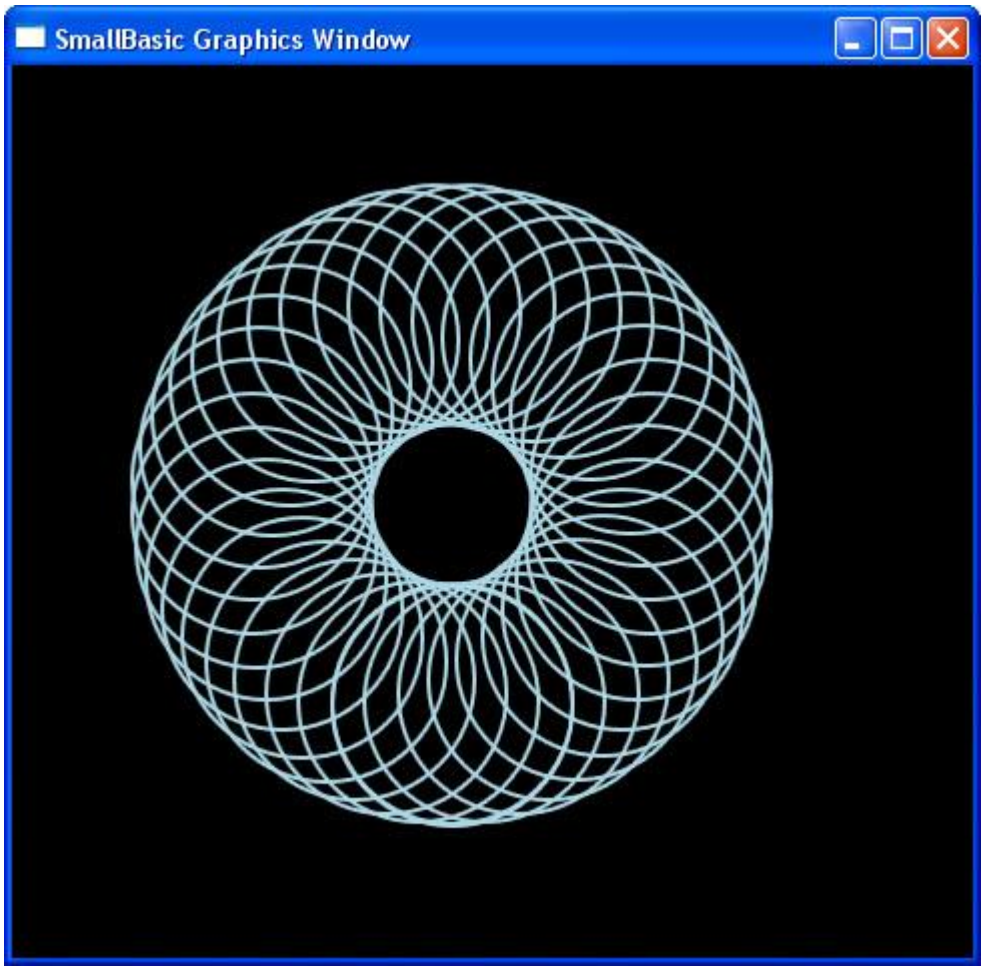
Izlaz tog programa izgleda ovako.



Slika 45 – prikaz većeg od dvaju brojeva pomoću potprograma

Pogledajmo drugi primjer koji ilustrira korištenje potprograma. Ovaj put ćemo koristiti grafički program koji izračunava koordinate raznih točaka i pohranjuje ih u varijable *x* i *y*. Zatim poziva potprogram **DrawCircleUsingCenter**, koji je zadužen za crtanje kružnice čije središte ima koordinate pohranjene u varijable *x* i *y*.

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightBlue"
GraphicsWindow.Width = 480
For i = 0 To 6.4 Step 0.17
x = Math.Sin(i) * 100 + 200
y = Math.Cos(i) * 100 + 200
DrawCircleUsingCenter()
EndFor
Sub DrawCircleUsingCenter
startX = x - 40
startY = y - 40
GraphicsWindow.DrawEllipse(startX, startY, 120, 120)
EndSub
```



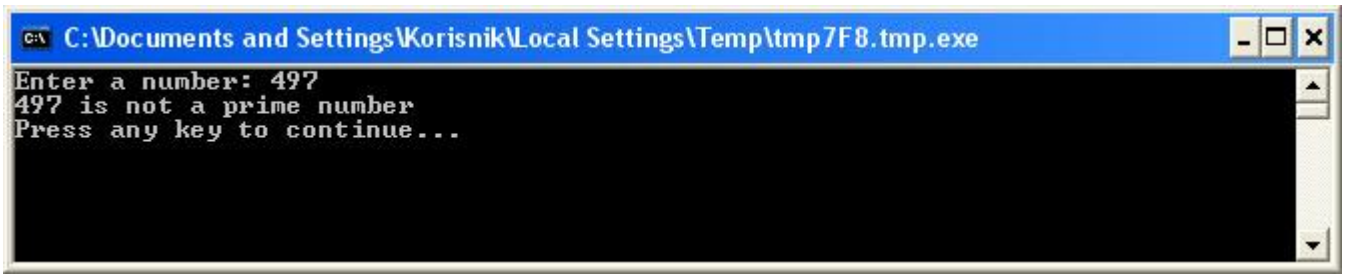
Slika 46 – grafički primjer korištenja potprograma

Pozivanje potprograma unutar petlji

Potprogrami se katkad pozivaju unutar petlje, tijekom čega izvršavaju isti skup iskaza, ali s drukčijim vrijednostima u jednoj ili više varijabli. Recimo da potprogram pod nazivom *PrimeCheck* utvrđuje je li neki broj prost. Ako napišete program koji korisniku omogućuje unos određene vrijednosti, pomoću tog potprograma možete utvrditi je li unesena vrijednost prosti broj. To ilustrira donji program.

```
TextWindow.Write("Enter a number: ")
i = TextWindow.ReadNumber()
isPrime = "True"
PrimeCheck()
If (isPrime = "True") Then
TextWindow.WriteLine(i + " is a prime number")
Else
TextWindow.WriteLine(i + " is not a prime number")
EndIf
Sub PrimeCheck
For j = 2 To Math.SquareRoot(i)
If (Math.Remainder(i, j) = 0) Then
isPrime = "False"
Goto EndLoop
EndIf
Endfor
EndLoop:
EndSub
```

Potprogram PrimeCheck prihvaća vrijednost varijable *i* i pokušava je podijeliti s manjim brojevima. Ako nakon dijeljenja vrijednosti varijable *i* s nekim brojem nema ostatka, vrijednost varijable *i* nije prosti broj. Tada potprogram varijabli *isPrime* dodjeljuje vrijednost "False" i završava izvođenje. Ako broj nije djeljiv s nijednim manjim brojem, vrijednost varijable *isPrime* ostaje "True".

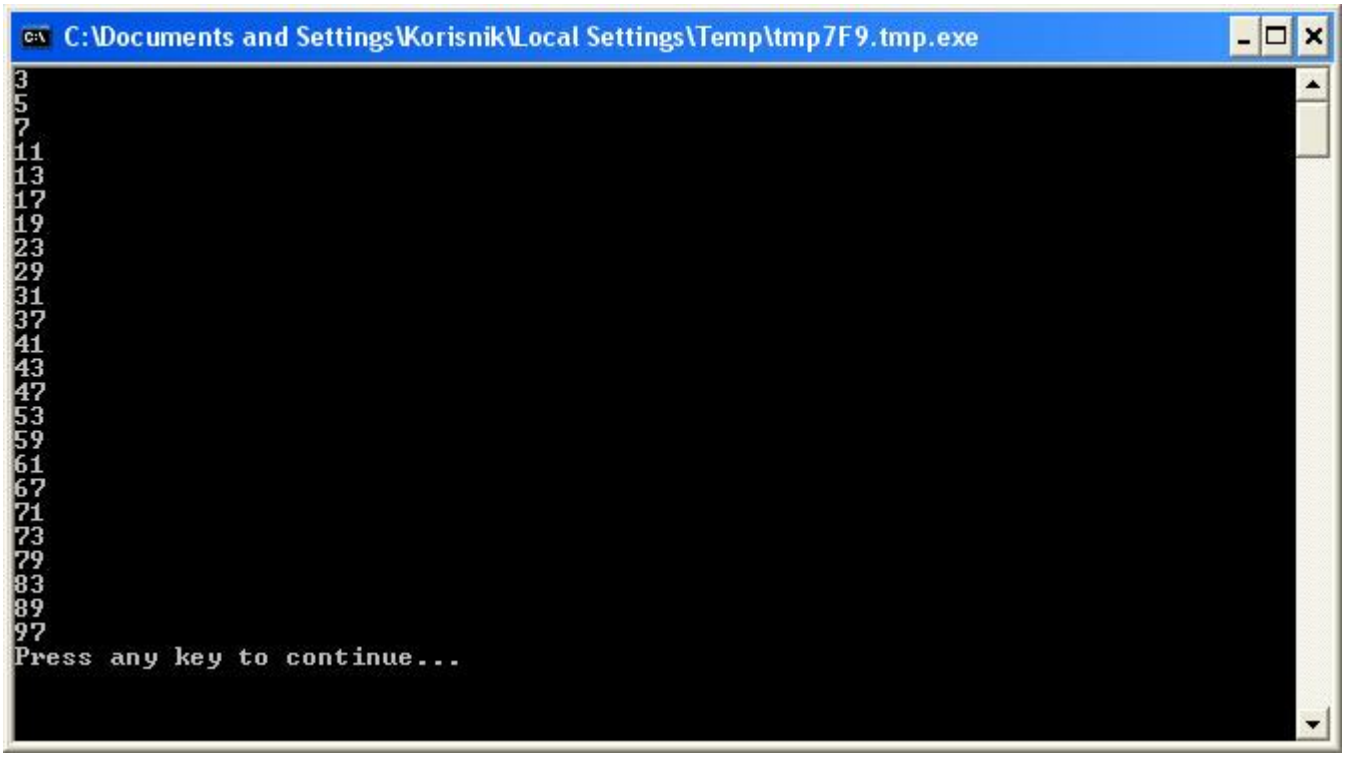


Slika 47 – provjera je li broj prost

Sada kada imate potprogram koji provjerava je li broj prost, pomoću njega možete prikazati popis svih prostih brojeva manjih od, recimo, 100. Vrlo je jednostavno izmijeniti gornji program i pozvati potprogram *PrimeCheck* iz petlje. Tako će potprogram dobivati drukčiju vrijednost za računanje kad god se petlja izvede. U donjem primjeru možete vidjeti kako to postići.

```
For i = 3 To 100
isPrime = "True"
PrimeCheck()
If (isPrime = "True") Then
TextWindow.WriteLine(i)
EndIf
EndFor
Sub PrimeCheck
For j = 2 To Math.SquareRoot(i)
If (Math.Remainder(i, j) = 0) Then
isPrime = "False"
Goto EndLoop
EndIf
Endfor
EndLoop:
EndSub
```

U gornjem se programu vrijednost varijable *i* ažurira pri svakom izvođenju petlje. Unutar petlje poziva se potprogram *PrimeCheck*. Potprogram *PrimeCheck* zatim uzima vrijednost varijable *i* te izračunava je li to prost broj. Rezultat se pohranjuje u varijablu *isPrime*, kojoj zatim pristupa petlja izvan potprograma. Vrijednost varijable *i* zatim se prikazuje ako riječ o prostom broju. Budući da petlja kreće od broja 3 i završava brojem 100, dobit ćemo popis svih prostih brojeva između 3 i 100. Na donjoj slici prikazan je rezultat izvođenja programa.



Slika 48 – prosti brojevi

Događaji i interaktivnost

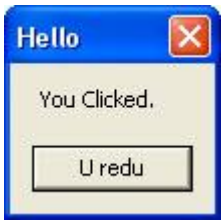
U prva dva poglavlja govorili smo o objektima, koji imaju *svojstva* i *operacije*. Osim svojstava i operacija, neki objekti imaju i tzv. ***događaje***. Događaji su svojevrsni signali koji se aktiviraju, primjerice, određenim korisničkim akcijama, kao što su pomicanje miša ili klikanje njime. Događaji su na određeni način suprotnost operacijama. Kada koristite operacije, vi ih kao programer pozivate da biste računalu naredili da nešto učini, a kada koristite događaje, računalo vas obavještava kada se dogodi nešto zanimljivo.

Zašto su događaji korisni?

Događaji su ključni za omogućivanje interaktivnosti u programu. Ako korisniku želite omogućiti interakciju s programom, koristit ćete događaje. Recimo da pišete program za igru *križić-kružić*. Korisniku želite omogućiti da odabere potez koji želi odigrati, zar ne? Tu na scenu stupaju događaji – u programu informacije od korisnika dobivate putem događaja. Ako vam to nije posve jasno, ne brinite – slijedi vrlo jednostavan primjer koji će vam pojasniti što su događaji i kako ih koristiti.
U nastavku je prikazan vrlo jednostavan program sa samo jednim iskazom i jednim potprogramom. Potprogram koristi operaciju *ShowMessage* na objektu GraphicsWindow za prikaz okvira s porukom korisniku.

```
GraphicsWindow.MouseDown = OnMouseDown
Sub OnMouseDown
GraphicsWindow.ShowMessage("You Clicked.", "Hello")
EndSub
```

Uočite da se u jednom retku gornjeg programa naziv potprograma dodjeljuje događaju **MouseDown** objekta GraphicsWindow. Vjerojatno ste primijetili da je događaj MouseDown vrlo sličan svojstvu – samo što mu se umjesto vrijednosti dodjeljuje potprogram *OnMouseDown*. To je posebnost događaja – kada se dogodi događaj, automatski se poziva potprogram. U ovom se primjeru potprogram *OnMouseDown* poziva kad god korisnik mišem klikne prozor za grafiku. Pokrenite program i iskušajte ga. Kad god mišem kliknete prozor za grafiku, pojavit će se okvir s porukom, kao što je ovaj na donjoj slici.

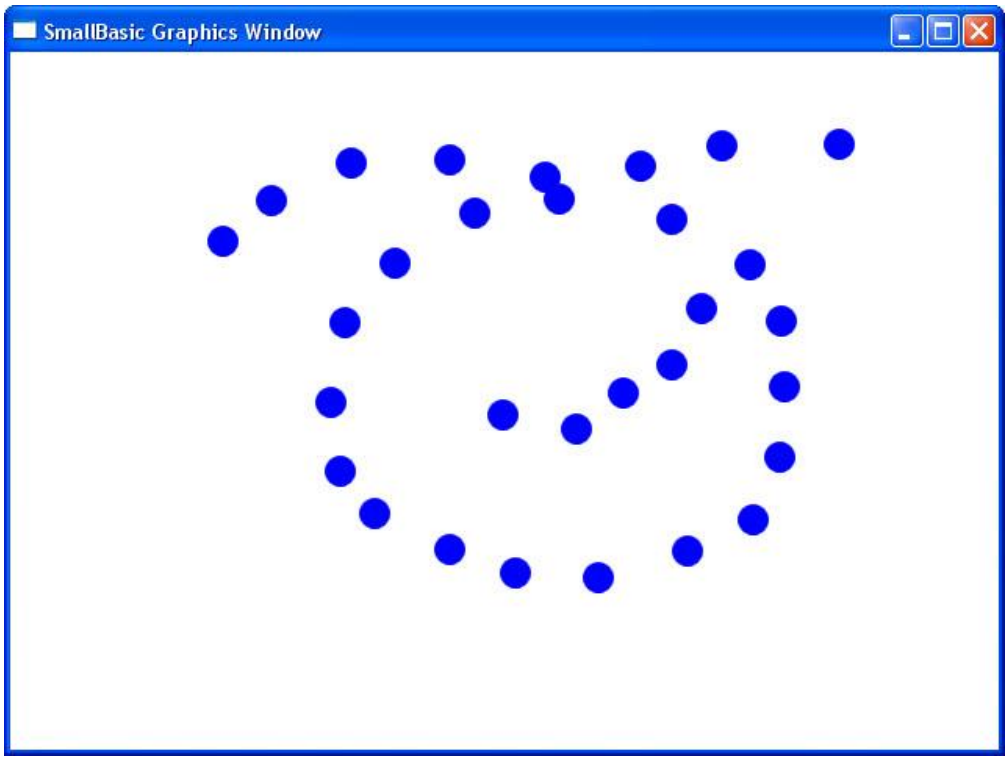


Slika 49 – reakcija na događaj

Takvo korištenje događaja vrlo je moćno i omogućuje stvaranje vrlo maštovitih i zanimljivih programa. Tako napisani programi često se zovu programi upravljani događajima.
Potprogram *OnMouseDown* možete izmijeniti tako da obavlja druge zadatke umjesto da prikazuje okvir s porukom. Možete, kao u donjem programu, odrediti da se na mjestu koje korisnik klikne pojavi velika plava točka.

```
GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.MouseDown = OnMouseDown
Sub OnMouseDown
x = GraphicsWindow.MouseX - 10
```

```
y = GraphicsWindow.MouseY - 10
GraphicsWindow.FillEllipse(x, y, 20, 20)
EndSub
```



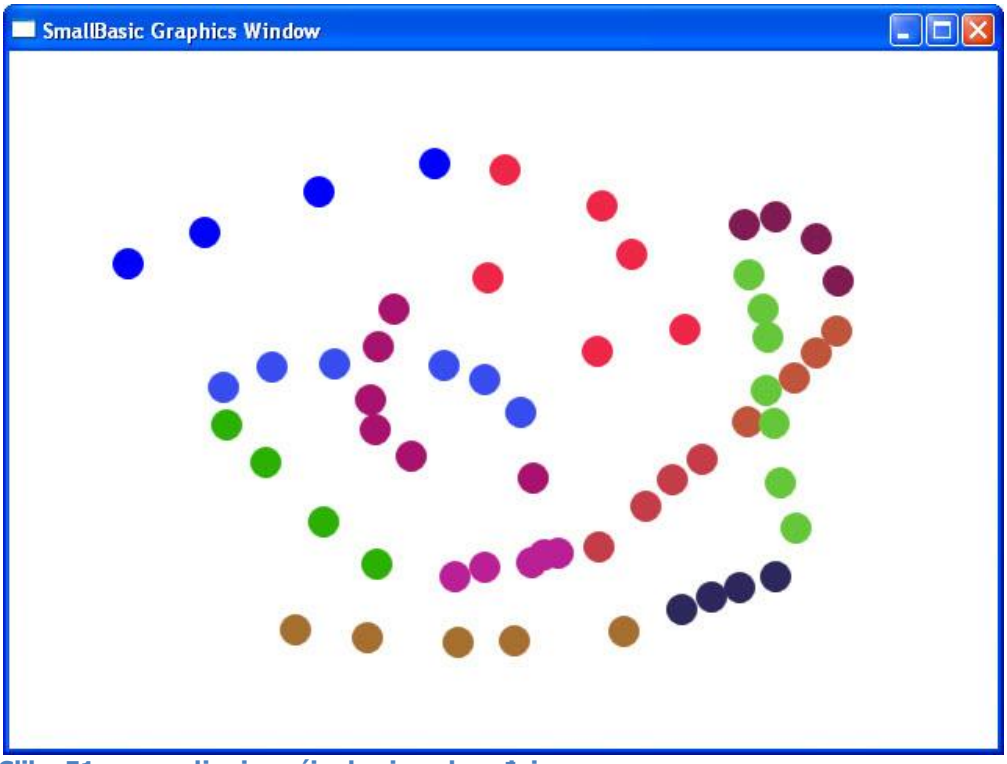
Slika 50 – korištenje događaja MouseDown

Uočite da se u gornjem programu za dohvaćanje koordinata pokazivača miša koriste operacije *MouseX* i *MouseY*. Zatim se crta kružnica sa središtem na mjestu koje određuju koordinate pokazivača miša.

Upravljanje većim brojem događaja

Broj događaja kojim možete upravljati nije ograničen. Čak možete odrediti da jedan potprogram upravlja većim brojem događaja. No događajem možete upravljati samo jedanput. Ako istom događaju pokušate dodijeliti dva potprograma, u obzir će se uzeti samo drugi.
Dodajmo za ilustraciju u prethodni primjer potprogram zadužen za pritiske na tipke. Odredimo i da taj novi potprogram mijenja boju kista da biste svakim klikom prikazali točku neke druge boje.

```
GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.MouseDown = OnMouseDown
GraphicsWindow.KeyDown = OnKeyDown
Sub OnKeyDown
GraphicsWindow.BrushColor =
GraphicsWindow.GetRandomColor()
EndSub
Sub OnMouseDown
x = GraphicsWindow.MouseX - 10
y = GraphicsWindow.MouseY - 10
GraphicsWindow.FillEllipse(x, y, 20, 20)
EndSub
```



Slika 51 – upravljanje većim brojem događaja

Ako pokrenete taj program i kliknete prozor, pojavit će se plava točka. Ako jedanput pritisnete bilo koju tipku, a zatim ponovno kliknete, pojavit će se točka neke druge boje. Kada pritisnete tipku, izvodi se potprogram *OnKeyDown*, koji mijenja boju kista u neku nasumično odabranu boju. Kada kliknete mišem, crta se kružnica ispunjena novom bojom – tako se pojavljuju raznobojne točke.

Program za bojenje

Oboružani događajima i potprogramima, možemo napisati program koji korisnicima omogućuje crtanje u prozoru. Takav je program iznenađujuće jednostavno napisati ako problem rastavimo na manje dijelove. Napišimo najprije program koji korisnicima omogućuje da pomicanjem miša ostavljaju trag u prozoru za grafiku.

```
GraphicsWindow.MouseMove = OnMouseMove
Sub OnMouseDown
x = GraphicsWindow.MouseX
y = GraphicsWindow.MouseY
GraphicsWindow.DrawLine(prevX, prevY, x, y)
prevX = x
prevY = y
EndSub
```

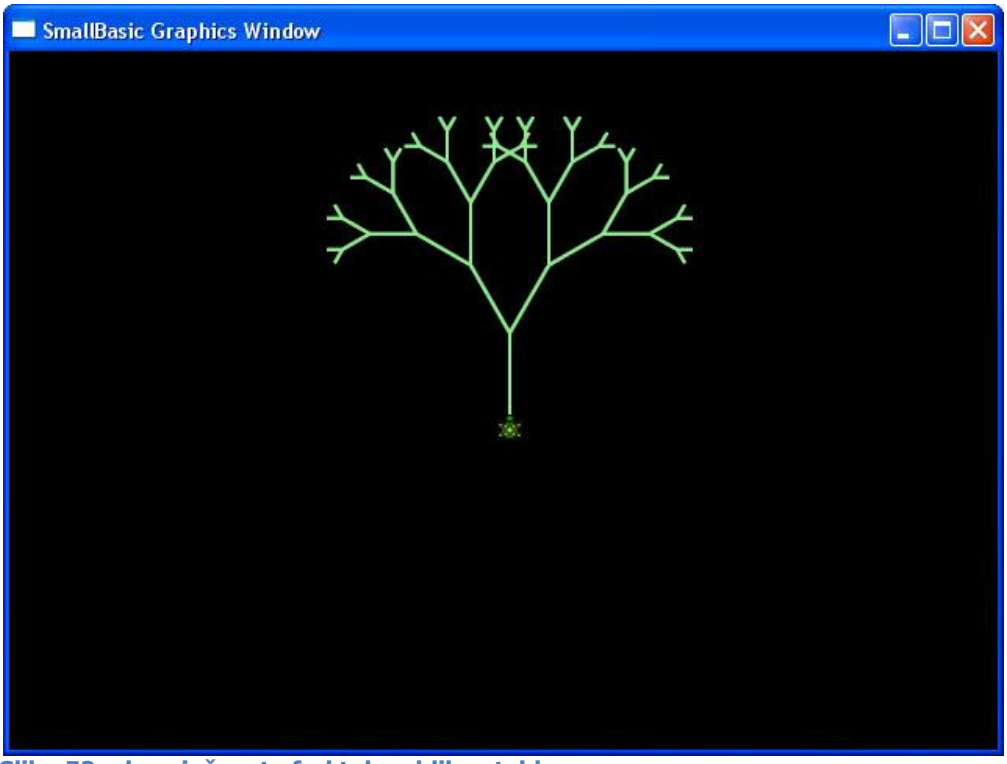
No kada pokrenete taj program, prva crta uvijek počinje od gornjeg lijevog kuta prozora (0, 0). Taj problem možete riješiti korištenjem događaja *MouseDown* i uzimanjem vrijednosti *prevX* i *prevY* kada se dogodi događaj.
Uz to, pokazivač miša trebao bi ostavljati trag samo kada korisnik drži pritisnutu tipku miša. Kada ne držimo pritisnutu tipku miša, ne bismo trebali povlačiti crtu. Da bismo to postigli, koristit ćemo svojstvo *IsLeftButtonDown* objekta **Mouse**. To svojstvo govori je li pritisnuta lijeva tipka miša. Ako je vrijednost tog svojstva istinita, povlačit ćemo crtu, a ako nije, nećemo je povlačiti.

```
GraphicsWindow.MouseMove = OnMouseMove
GraphicsWindow.MouseDown = OnMouseDown
Sub OnMouseDown
prevX = GraphicsWindow.MouseX
prevY = GraphicsWindow.MouseY
EndSub
Sub OnMouseMove
x = GraphicsWindow.MouseX
y = GraphicsWindow.MouseY
If (Mouse.IsLeftButtonDown) Then
GraphicsWindow.DrawLine(prevX, prevY, x, y)
EndIf
prevX = x
prevY = y
EndSub
```

Dodatak A

Zabavni primjeri

Crtanje fraktala pomoću kornjače



Slika 52 – kornjača crta fraktal u obliku stabla

```
angle = 30
delta = 10
distance = 60
Turtle.Speed = 9
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightGreen"
DrawTree()
Sub DrawTree
If (distance > 0) Then
Turtle.Move(distance)
Turtle.Turn(angle)
Stack.PushValue("distance", distance)
distance = distance - delta
DrawTree()
Turtle.Turn(-angle * 2)
DrawTree()
Turtle.Turn(angle)
distance = Stack.PopValue("distance")
Turtle.Move(-distance)
EndIf
EndSub
```

Fotografije s web-mjesta Flickr

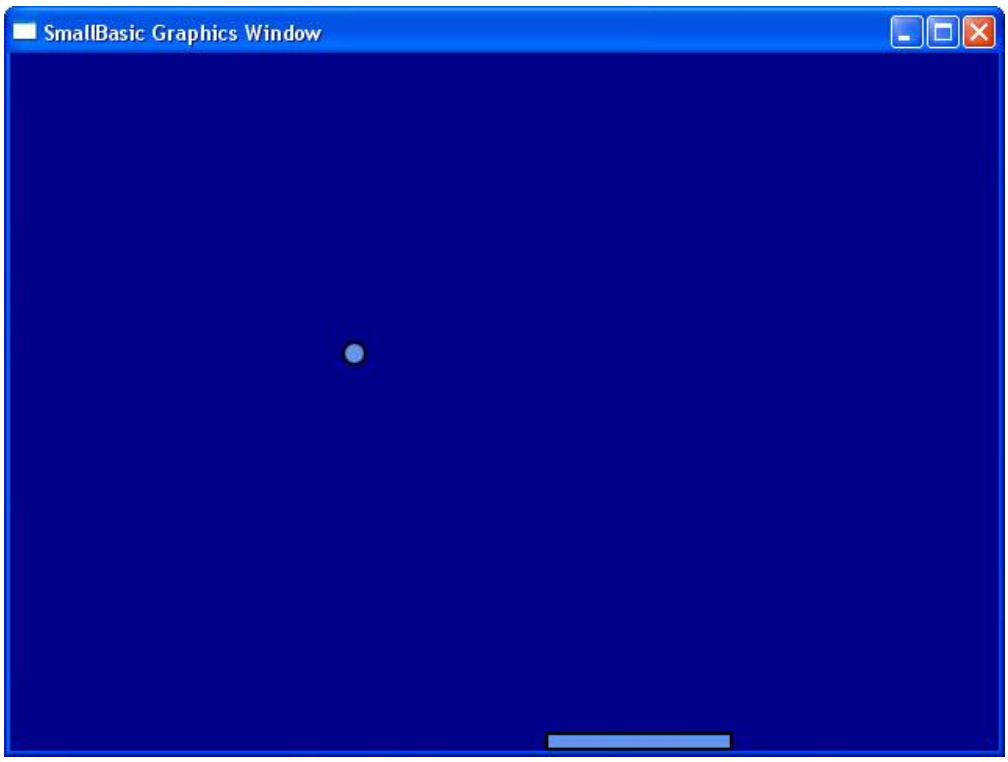
Slika 53 – dohvaćanje slika s web-mjesta Flickr

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.MouseDown = OnMouseDown
Sub OnMouseDown
pic = Flickr.GetRandomPicture("mountains, river")
GraphicsWindow.DrawResizedImage(pic, 0, 0, 640, 480)
EndSub
```

Dinamična pozadina radne površine

```
For i = 1 To 10
pic = Flickr.GetRandomPicture("mountains")
Desktop.SetWallPaper(pic)
Program.Delay(10000)
EndFor
```

Igra s lopticom



Slika 54 – igra s lopticom

```
GraphicsWindow.BackgroundColor = "DarkBlue"
paddle = Shapes.AddRectangle(120, 12)
ball = Shapes.AddEllipse(16, 16)
GraphicsWindow.MouseMove = OnMouseMove
x = 0
y = 0
deltaX = 1
deltaY = 1
RunLoop:
x = x + deltaX
y = y + deltaY
gw = GraphicsWindow.Width
gh = GraphicsWindow.Height
If (x >= gw - 16 or x <= 0) Then
deltaX = -deltaX
EndIf
If (y <= 0) Then
deltaY = -deltaY
EndIf
padX = Shapes.GetLeft (paddle)
If (y = gh - 28 and x >= padX and x <= padX + 120) Then
deltaY = -deltaY
EndIf
Shapes.Move(ball, x, y)
Program.Delay(5)
If (y < gh) Then
Goto RunLoop
EndIf
GraphicsWindow.ShowMessage("You Lose", "Paddle")
Sub OnMouseMove
paddleX = GraphicsWindow.MouseX
Shapes.Move(paddle, paddleX - 60, GraphicsWindow.Height
- 12)
EndSub
```

Dodatak B

Boje

U IZRADI: opisi i heksadecimalni kodovi boja
Slijedi popis naziva boja koje podržava Small Basic, kategoriziranih prema osnovnoj boji.

Nijanse crvene boje

IndianRed	#CD5C5C
LightCoral	#F08080
Salmon	#FA8072
DarkSalmon	#E9967A
LightSalmon	#FFA07A
Crimson	#DC143C
Red	#FF0000
FireBrick	#B22222
DarkRed	#8B0000

Nijanse ružičaste boje

Pink	#FFC0CB
LightPink	#FFB6C1
HotPink	#FF69B4
DeepPink	#FF1493
MediumVioletRed	#C71585
PaleVioletRed	#DB7093

Nijanse narančaste boje

LightSalmon	#FFA07A
Coral	#FF7F50
Tomato	#FF6347
OrangeRed	#FF4500
DarkOrange	#FF8C00
Orange	#FFA500

Nijanse plave boje

Aqua	#00FFFF
Cyan	#00FFFF
LightCyan	#E0FFFF
PaleTurquoise	#AFEEEE
Aquamarine	#7FFFD4
Turquoise	#40E0D0
MediumTurquoise	#48D1CC
DarkTurquoise	#00CED1
CadetBlue	#5F9EA0
SteelBlue	#4682B4
LightSteelBlue	#B0C4DE
PowderBlue	#B0E0E6
LightBlue	#ADD8E6
SkyBlue	#87CEEB
LightSkyBlue	#87CEFA
DeepSkyBlue	#00BFFF
DodgerBlue	#1E90FF
CornflowerBlue	#6495ED
MediumSlateBlue	#7B68EE
RoyalBlue	#4169E1
Blue	#0000FF
MediumBlue	#0000CD
DarkBlue	#00008B
Navy	#000080
MidnightBlue	#191970

Nijanse žute boje

Gold	#FFD700
Yellow	#FFFF00
LightYellow	#FFFFE0
LemonChiffon	#FFFACD
LightGoldenrodYellow	#FAFAD2
PapayaWhip	#FFEFD5
Moccasin	#FFE4B5
PeachPuff	#FFDAB9
PaleGoldenrod	#EEE8AA
Khaki	#F0E68C
DarkKhaki	#BDB76B

Nijanse ljubičaste boje

Lavender	#E6E6FA
Thistle	#D8BFD8
Plum	#DDA0DD
Violet	#EE82EE
Orchid	#DA70D6
Fuchsia	#FF00FF
Magenta	#FF00FF
MediumOrchid	#BA55D3
MediumPurple	#9370DB
BlueViolet	#8A2BE2
DarkViolet	#9400D3
DarkOrchid	#9932CC
DarkMagenta	#8B008B
Purple	#800080
Indigo	#4B0082
SlateBlue	#6A5ACD
DarkSlateBlue	#483D8B
MediumSlateBlue	#7B68EE

Nijanse smeđe boje

Cornsilk	#FFF8DC
BlanchedAlmond	#FFEBCD
Bisque	#FFE4C4
NavajoWhite	#FFDEAD
Wheat	#F5DEB3
BurlyWood	#DEB887
Tan	#D2B48C
RosyBrown	#BC8F8F
SandyBrown	#F4A460
Goldenrod	#DAA520
DarkGoldenrod	#B8860B
Peru	#CD853F
Chocolate	#D2691E
SaddleBrown	#8B4513
Sienna	#A0522D
Brown	#A52A2A
Maroon	#800000

Nijanse zelene boje

GreenYellow	#ADFF2F
Chartreuse	#7FFF00
LawnGreen	#7CFC00
Lime	#00FF00
LimeGreen	#32CD32
PaleGreen	#98FB98
LightGreen	#90EE90
MediumSpringGreen	#00FA9A
SpringGreen	#00FF7F
MediumSeaGreen	#3CB371
SeaGreen	#2E8B57
ForestGreen	#228B22
Green	#008000
DarkGreen	#006400
YellowGreen	#9ACD32
OliveDrab	#6B8E23
Olive	#808000
DarkOliveGreen	#556B2F
MediumAquamarine	#66CDAA
DarkSeaGreen	#8FBC8F
LightSeaGreen	#20B2AA
DarkCyan	#008B8B
Teal	#008080

Nijanse bijele boje

White	#FFFFFF
Snow	#FFFAFA
Honeydew	#F0FFF0
MintCream	#F5FFFA
Azure	#F0FFFF
AliceBlue	#F0F8FF
GhostWhite	#F8F8FF
WhiteSmoke	#F5F5F5
Seashell	#FFF5EE
Beige	#F5F5DC
OldLace	#FDF5E6
FloralWhite	#FFFAF0
Ivory	#FFFFF0
AntiqueWhite	#FAEBD7
Linen	#FAF0E6
LavenderBlush	#FFF0F5
MistyRose	#FFE4E1

Nijanse sive boje

Gainsboro	#DCDCDC
LightGray	#D3D3D3
Silver	#C0C0C0
DarkGray	#A9A9A9
Gray	#808080
DimGray	#696969
LightSlateGray	#778899
SlateGray	#708090
DarkSlateGray	#2F4F4F
Black	#000000