Microsoft Small Basic

Uvod u programiranje

1. poglavlje

Uvod

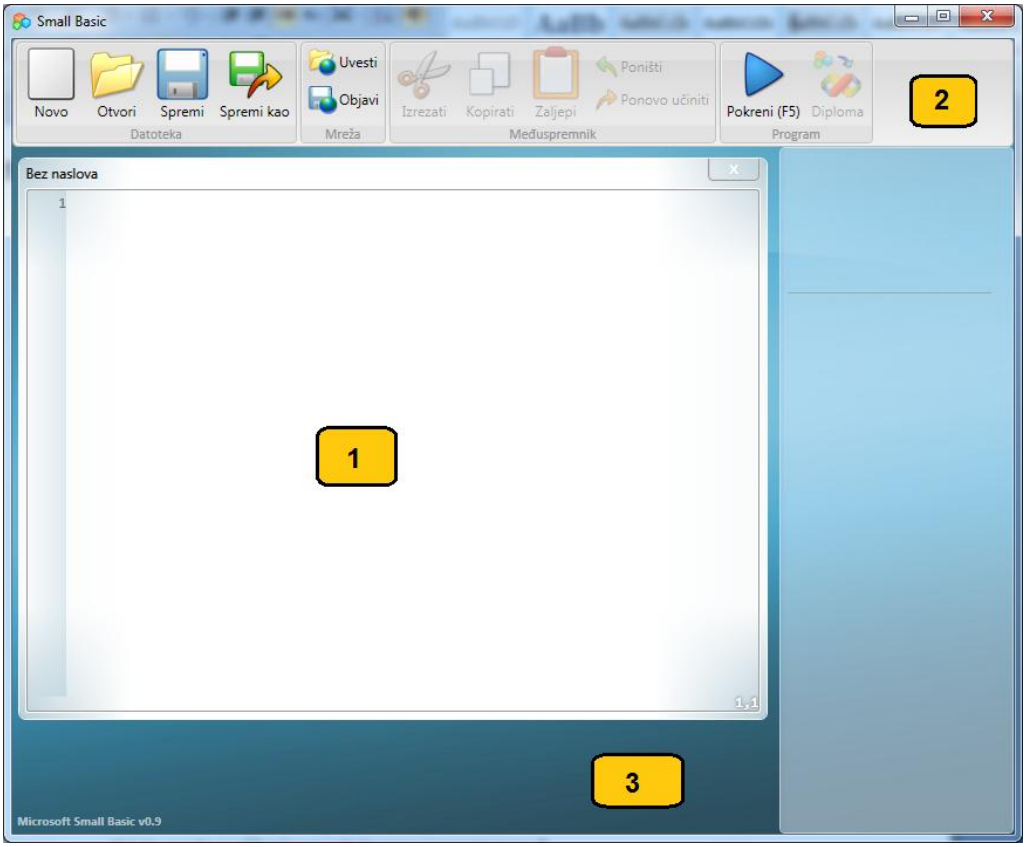
Small Basic i programiranje

Računalno programiranje definira se kao postupak stvaranja računalnog softvera pomoću programskih jezika. Baš kao što mi govorimo i razumijemo engleski, španjolski ili francuski jezik, računala razumiju programe napisane na određenim jezicima. Ti jezici zovu se programski jezici. U početku je postojalo samo nekoliko programskih jezika i bilo ih je lako naučiti i razumjeti. No s vremenom su računala i softver postajali sve složeniji, programski jezici brzo su se razvijali i obuhvaćali sve kompleksnije koncepte. Moderni programski jezici zato su prilično zahtjevni, osobito za početnike. To je mnoge obeshrabrilo i odvratilo od učenja i rada u području računalnog programiranja.

Small Basic je programski jezik osmišljen za iznimno jednostavno programiranje, pristupačan je i zabavan čak i za početnike. Small Basic ruši barijere i služi kao uvod u čudesni svijet računalnog programiranja.

Okruženje programa Small Basic

Krenimo s kratkim uvodom u okruženje Small Basic. Kada pokrenete program Small Basic, otvorit će se prozor prikazan na sljedećoj slici.



Slika 1 – Okruženje programa Small Basic

To je okruženje programa Small Basic u kojem ćemo pisati i izvršavati programe napisane na jeziku Small Basic. Okruženje se sastoji od nekoliko elemenata označenih brojevima.

U **uređivaču**, označenom brojem [1], pišemo programe na jeziku Small Basic. Kada otvorite primjer programa ili prethodno spremljeni program, prikazat će se u uređivaču. Zatim ga možete izmijeniti i spremiti za kasniju upotrebu.

Istodobno možete otvoriti i raditi na više programa. Svaki program na kojem radite prikazat će se u zasebnom uređivaču. Uređivač koji sadrži program na kojem trenutačno radite naziva se *aktivni uređivač*.

Alatna traka, označena brojem [2], upotrebljava se za izdavanje naredbi u *aktivnom uređivaču* ili okruženju. S naredbama na alatnoj traci upoznat ćete se postupno u ovom priručniku.

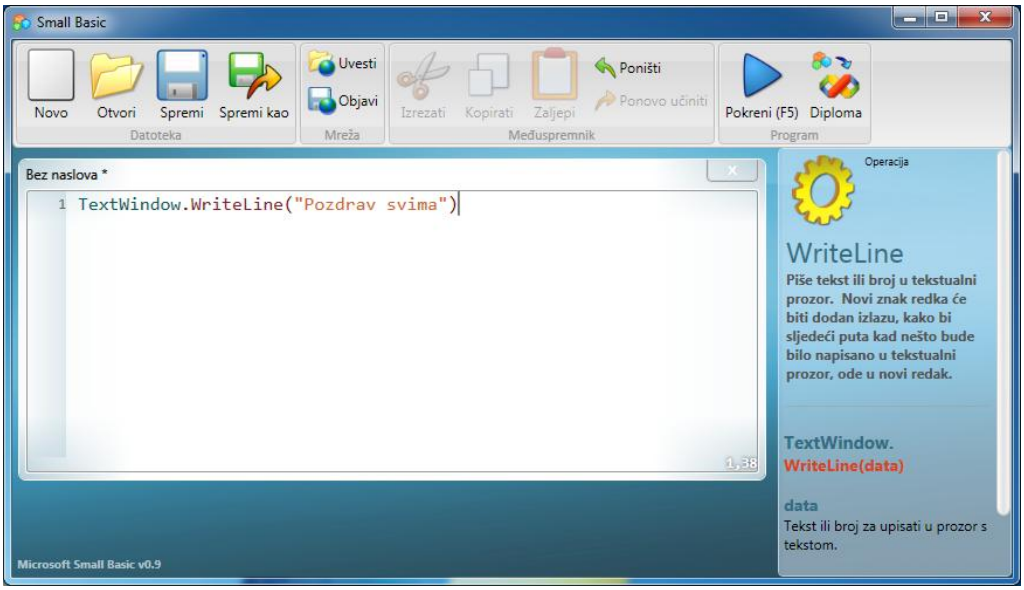
Radna površina, označena brojem [3], mjesto je na kojem se prikazuju svi prozori uređivača.

Naš prvi program

Sada ste se upoznali s okruženjem programa Small Basic pa možemo započeti s programiranjem u njemu. Kao što smo već rekli, programe pišemo u uređivaču. Dakle, upišimo sljedeći redak u uređivač.

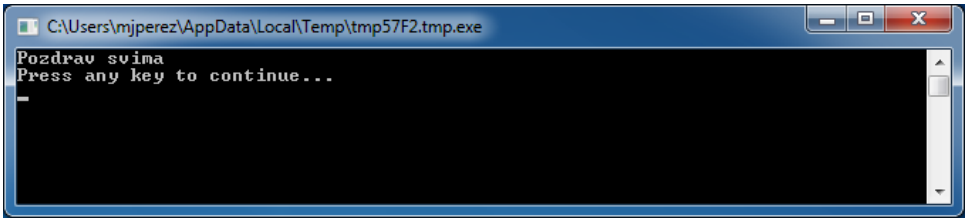
```
TextWindow.WriteLine("Pozdrav svima")
```

To je naš prvi program na jeziku Small Basic. Ako sve točno upišete, trebali biste vidjeti nešto slično kao na slici u nastavku.



Slika 2 – Prvi program

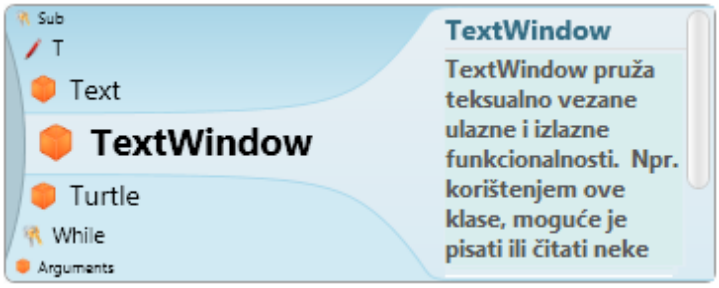
Napisali smo prvi program; pokrenimo ga i pogledajmo što će se dogoditi. Program možemo pokrenuti klikom na gumb *Pokreni* na alatnoj traci ili pritiskom na tipku prečaca, F5, na tipkovnici. Ako sve bude išlo po planu, rezultat pokretanja programa izgledat će kako je prikazano u nastavku.



Slika 3 – Rezultat prvog programa

Čestitamo! Upravo ste napisali i izvršili prvi program na jeziku Small Basic. Program je malen i jednostavan, ali ipak predstavlja vaš prvi veliki korak u svijetu računalnog programiranja!
Prije nego što nastavimo s radom na većim programima, moramo razjasniti još jednu važnu pojedinost. Potrebno je razumjeti što se dogodilo – što smo zapravo rekli računalu i kako je ono znalo što treba učiniti? U sljedećem poglavlju analizirat ćemo program koji smo upravo napisali I odgovoriti na to pitanje.

*Kada ste upisivali svoj prvi program, možda ste primijetili da se pojavio skočni prozor s popisom stavki (*Slika 4*). Ta se značajka zove "intellisense" i omogućuje vam brže pisanje programa. Popis možete pregledavati pritiskom na tipke sa strelicama gore/dolje, a kada nađete željenu stavku, pritisnite tipku Enter za umetanje odabrane stavke u program.*



Slika 4 – Intellisense

Spremanje programa

Ako želite zatvoriti Small Basic i kasnije raditi na programu koji ste upravo napisali, možete ga spremiti. Preporučuje se da povremeno spremite programe kako ne biste izgubili podatke u slučaju nepredviđenog zatvaranja programa ili nestanka struje. Trenutačni program možete spremiti klikom na ikonu "Spremi" na alatnoj traci ili tipkovnim prečacem "Ctrl+S" (pritisnite tipku S dok držite pritisnutom tipku Ctrl).

2. poglavlje

Analiza našeg prvog programa

Što je zapravo računalni program?

Program je niz uputa za računalo. Te upute računalu kažu što točno treba učiniti i ono ih vjerno slijedi. Jednako kao i ljudi, računala mogu slijediti upute samo ako su napisane na jeziku koji razumiju. Ti jezici zovu se programski jezici. Računalo razumije brojne jezike i **Small Basic** je jedan od njih.

Zamislite da razgovarate s prijateljem. Vi i vaš prijatelj upotrebljavate riječi organizirane u rečenice kako biste jedno drugome prenijeli informacije. Jednako tako, programski jezici sadrže zbirke riječi koje se mogu organizirati u rečenice koje računalu prenose informacije. Programi su, dakle, skupovi rečenica (ponekad samo nekoliko njih, ponekad više tisuća) koje razumiju i programer i računalo.

Programi na jeziku Small Basic

Računalo razumije brojne jezike. Java, C++, Python, VB itd. napredni su moderni programski jezici koji se upotrebljavaju za stvaranje jednostavnih i složenih programa.

Program na jeziku Small Basic obično se sastoji od grupe *naredaba*. Svaki je redak programa jedna naredba, a svaka je naredba uputa za računalo. Kada od računala zatražimo da izvrši program na jeziku Small Basic, ono otvara program i čita prvu naredbu. Razumije što od njega tražimo i izvršava našu naredbu. Kada izvrši našu prvu naredbu, vraća se na program te čita i izvršava drugi redak. Nastavlja s tim postupkom dok ne dođe do posljednjeg retka. Tu je kraj programa.

Vratimo se našem prvom programu

Napisali smo sljedeći program:

```
TextWindow.WriteLine("Pozdrav svima")
```

To je vrlo jednostavan program koji se sastoji od samo jedne *naredbe*. Ona nalaže računalu da u tekstnom prozoru ispiše redak teksta **Pozdrav svima**.

Ta naredba na računalnom jeziku znači doslovno:

```
Write Pozdrav svima
```

Možda ste već primijetili da se naredba sastoji od nekoliko manjih dijelova baš kao što se rečenice sastoje od riječi. Prva naredba sastoji se od 3 dijela:

a) TextWindow
b) WriteLine
c) "Pozdrav svima"

Točka, zagrade i navodnici znakovi su koji se moraju postaviti na odgovarajuća mjesta kako bi računalo moglo razumjeti našu naredbu.

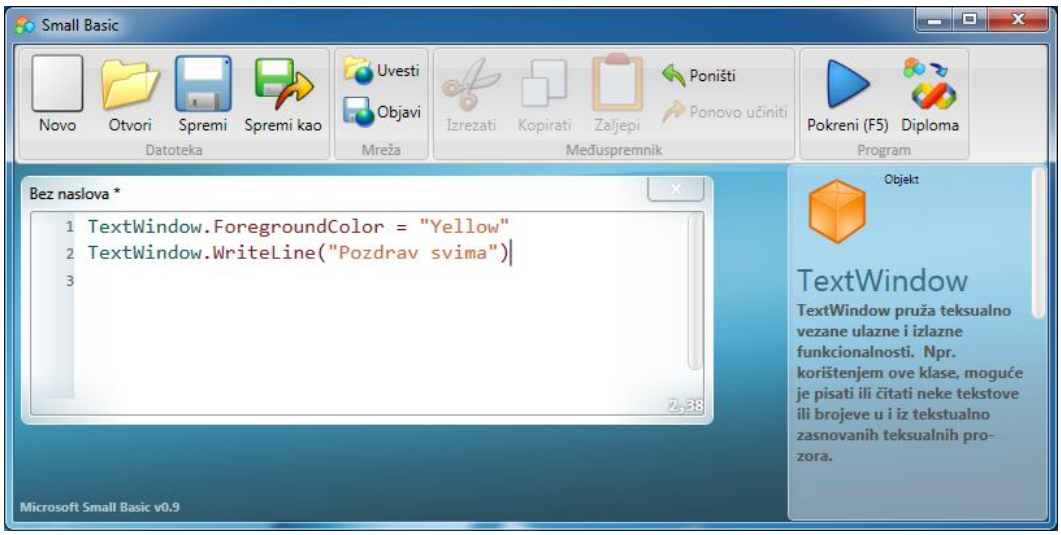
Možda se sjećate crnog prozora koji se pojavio pri pokretanju našeg prvog programa. Taj crni prozor zove se TextWindow, ponekad se upotrebljava i naziv "konzola". U njemu se vide rezultati programa. U našem programu **TextWindow** se zove *objekt*. Programi nam stavljaju na raspolaganje više takvih objekata. Na njima možemo izvršiti nekoliko različitih *operacija*. U svom programu već smo koristili operaciju WriteLine. Možda ste primijetili da nakon operacije WriteLine slijedi tekst **Pozdrav svima** u navodnicima. Taj tekst prosljeđuje se operaciji WriteLine, koja ga ispisuje korisniku. To se zove *unos* za operaciju. Neke operacije zahtijevaju jedan ili više unosa, neke nijedan.

> *Znakovi poput navodnika, točaka i zagrada vrlo su važni u računalnom programu. Njihov položaj i broj može promijeniti značenje izraza.*

Naš drugi program

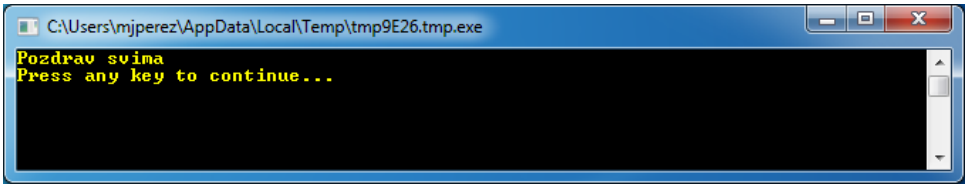
Upoznali smo osnove svog prvog programa i sada ga možemo uljepšati dodavanjem boja.

```
TextWindow.ForegroundColor = "Yellow"
TextWindow.WriteLine("Pozdrav svima")
```



Slika 5 – Dodavanje boja

Kada pokrenete program, izraz "Pozdrav svima" ispisat će u prozoru TextWindow, ali ovaj put žutim slovima umjesto sivih.



Slika 6 – Pozdrav svima u žutoj boji

Pogledajte novu naredbu koju smo dodali izvornom programu. Sadrži novu riječ, ForegroundColor, uz koju nakon znaka jednakosti stoji vrijednost *"Yellow".* To znači da smo objektu ForegroundColor *dodijelili* vrijednost "Yellow". Razlika između svojstva ForegroundColor i operacije WriteLine jest u tome da ForegroundColor ne zahtijeva unose ni zagrade. Umjesto toga nakon svojstva slijedi znak *jednako* i riječ. ForegroundColor definiramo kao *svojstvo* objekta TextWindow. Slijedi popis vrijednosti koje su valjane za svojstvo ForegroundColor. Pokušajte zamijeniti "Yellow" s jednom od sljedećih vrijednosti i pogledajte rezultat – ne zaboravite navodnike, oni su obavezni.

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

3. poglavlje

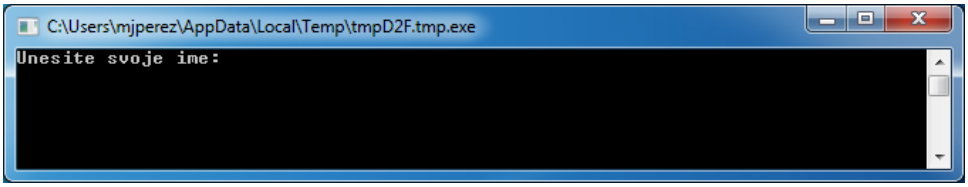
Uvod u varijable

Korištenje varijabli u našem programu

Zar ne bi bilo zgodno kad bi naš program mogao reći "Pozdrav" s imenom korisnika umjesto općenite fraze "Pozdrav svima"? Da bi se to postiglo, najprije od korisnika moramo zatražiti njegovo ime, spremiti ga, a zatim ispisati "Pozdrav" s imenom korisnika. Pogledajmo kako to postići:

```
TextWindow.Write("Unesite svoje ime: ")
ime = TextWindow.Read()
TextWindow.WriteLine("Pozdrav," + ime)
```

Kad upišete i izvršite taj program, prikazat će se rezultat sličan ovom:



Slika 7 – Zatražite od korisnika da unese ime

A kad unesete svoje ime i pritisnete ENTER, prikazat će se sljedeće:

Slika 8 – Topli pozdrav

Ako ponovno pokrenete program, ponovno će vam se postaviti isto pitanje. Možete upisati drugo ime, a računalo će vas pozdraviti s tim imenom.

Analiza programa

U programu koji ste upravo pokrenuli, možda vam je pažnju privukao sljedeći redak:

```
ime = TextWindow.Read()
```

Read() izgleda baš kao *WriteLine()*, ali bez unosa. To je operacija koja upućuje računalo da čeka dok korisnik nešto ne upiše i pritisne tipku ENTER. Kad korisnik pritisne tipku ENTER, operacija uzima unos koji je korisnik upisao i vraća ga u program. Ono što je korisnik upisao sada je spremljeno u *varijabli* pod nazivom **ime**. *Varijabla* se definira kao mjesto gdje možete privremeno spremiti vrijednosti i kasnije ih koristiti. U gornjem retku varijabla **ime** koristi se za spremanje imena korisnika.

> Write*, baš kao i* WriteLine*, još je jedna operacija u prozoru ConsoleWindow. Operacija Write omogućuje vam da upišete nešto u prozor ConsoleWindow, ali omogućuje tekstu u nastavku da se prikaže u istom retku kao trenutni tekst.*

Zanimljiv je i sljedeći redak:

```
TextWindow.WriteLine("Pozdrav," + ime)
```

To je mjesto gdje koristimo vrijednost spremljenu u varijabli **ime**. Uzimamo vrijednost varijable **ime**, dodajemo je izrazu "Pozdrav," i pišemo je u prozoru TextWindow.

Nakon postavljanja varijablu možete koristiti koliko god puta želite. Na primjer, možete napraviti sljedeće:

```
TextWindow.Write("Unesite svoje ime: ")
ime = TextWindow.Read()
TextWindow.Write("Pozdrav," + ime + ".  ")
TextWindow.WriteLine("Kako ste, " + ime + "?")
```

Prikazat će se sljedeći izlazni rezultat:

Slika 9 – Ponovno korištenje varijable

Pravila za davanje naziva varijablama

Varijablama se pridružuju nazivi po kojima ih prepoznajete. Postoje određena jednostavna pravila i neke uistinu dobre smjernice za davanje naziva tim varijablama. To su sljedeća pravila:

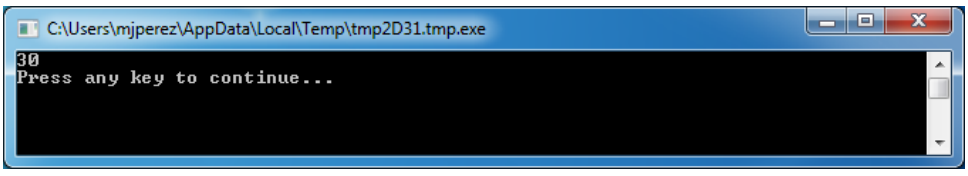
1. Naziv mora započinjati slovom i ne smije se preklapati s nekom od ključnih riječi kao što su **if**, **for**, **then** itd.
2. Naziv može sadržavati bilo koju kombinaciju slova, znamenki i podvlaka.
3. Korisno je da nazivi varijabli imaju odgovarajuće značenje. Budući da duljina naziva varijabli nije ograničena, koristite ih da biste opisali njihovu svrhu.

Igranje s brojevima

Upravo smo vidjeli kako pomoću varijabli spremiti ime korisnika. U nekoliko sljedećih programa vidjet ćemo kako spremiti brojeve u varijablama i kako rukovati njima. Započnimo s vrlo jednostavnim programom:

```
broj1 = 10
broj2 = 20
broj3 = broj1 + broj2
TextWindow.WriteLine(broj3)
```

Kad pokrenete taj program, dobit ćete sljedeći izlazni rezultat:



Slika 10 – Zbrajanje dva broja

U prvom retku programa varijabli **broj1** dodjeljujete vrijednost 10. U drugom retku varijabli **broj2** dodjeljujete vrijednost 20. U trećem retku zbrajate **broj1** i **broj2**, a zatim zbroj dodjeljujete varijabli **broj3**. Dakle, u tom će slučaju **broj3** imati vrijednost 30. A evo što se ispisuje u prozoru TextWindow.

> *Primijetit ćete da brojevi nisu okruženi navodnicima. Navodnici nisu potrebni za brojeve. Oni su potrebni samo ako koristite tekst.*

Izmijenimo malo sada taj program i pogledajmo rezultate:

```
broj1 = 10
broj2 = 20
broj3 = broj1 * broj2
TextWindow.WriteLine(broj3)
```

Gornji program množi varijablu **broj1** s varijablom **broj2**, a zatim sprema rezultat u varijabli **broj3**. Rezultat tog programa možete vidjeti u nastavku:

Slika 11 – Množenje dva broja

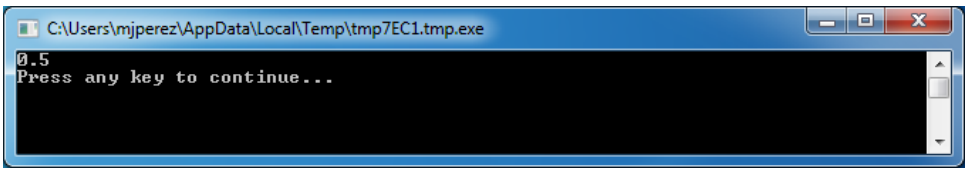
Isto tako možete oduzimati ili dijeliti brojeve. Evo primjera za oduzimanje:

```
broj3 = broj1 - broj2
```

A za dijeljenje se koristi znak "/". Program izgleda ovako:

```
broj3 = broj1 / broj2
```

A ovo je rezultat dijeljenja:



Slika 12 – Dijeljenje dva broja

Jednostavan pretvornik temperature

U sljedećem ćemo programu pomoću formule $℃ = \frac{5(°F-32)}{9}$ pretvoriti temperature u stupnjevima Fahrenheita u temperature u stupnjevima Celzija.

Najprije od korisnika moramo dobiti temperaturu u stupnjevima Fahrenheita te je spremiti u varijabli. Postoji posebna operacija koja nam omogućuje da pročitamo brojeve korisnika, a to je TextWindow.ReadNumber.

```
TextWindow.Write("Unesite temperaturu u fahrenheitima: ")
fahr = TextWindow.ReadNumber()
```

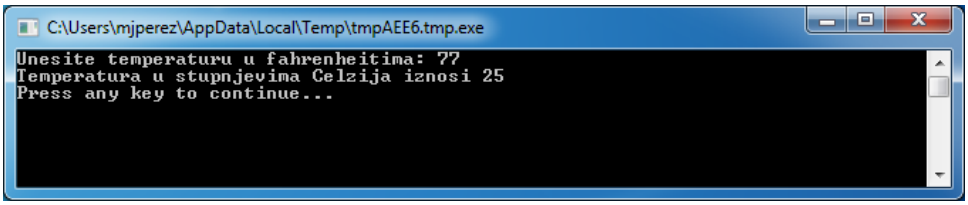
Nakon što smo temperaturu u stupnjevima Fahrenheita spremili u varijabli, možemo je pretvoriti u stupnjeve Celzija na sljedeći način:

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

Zagrade upućuju računalo da najprije izračuna dio **fahr – 32**, a zatim ostatak. Nakon toga samo moramo prikazati rezultat korisniku. Kad sve to spojimo, dobivamo ovaj program:

```
TextWindow.Write("Unesite temperaturu u fahrenheitima: ")
fahr = TextWindow.ReadNumber()
celzij = 5 * (fahr - 32) / 9
TextWindow.WriteLine("Temperatura u stupnjevima Celzija iznosi "
+ celzij)
```

A ovo je rezultat tog programa:



Slika 13 – Pretvorba temperature

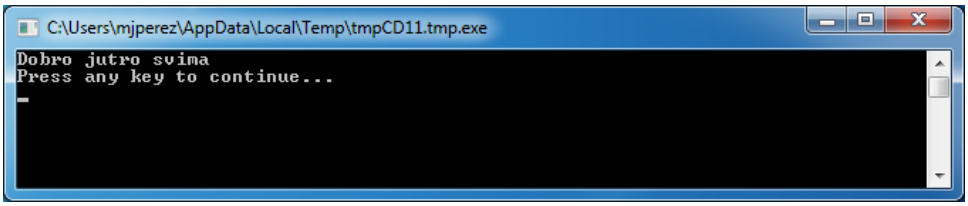
4. poglavlje

Uvjeti i grananje

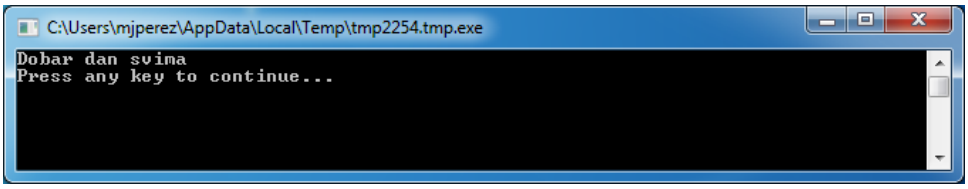
Ako se vratimo na naš prvi program, zar ne bi bilo sjajno da, umjesto općenitog *Pozdrav svima*, možemo reći *Dobro jutro svima* ili *Dobar dan svima* ovisno o dobu dana? U našem sljedećem programu postići ćemo da računalo kaže *Dobro jutro svima* prije 12 sati, a *Dobar dan* poslije 12 sati.

```
If (Clock.Hour < 12) Then
  TextWindow.WriteLine("Dobro jutro svima")
EndIf
If (Clock.Hour >= 12) Then
  TextWindow.WriteLine("Dobar dan svima")
EndIf
```

Ovisno o tome kada se program pokrene, prikazuje se od sljedećih izlaznih rezultata:



Slika 14 – Dobro jutro svima



Slika 15 – Dobar dan svima

Analizirajmo prva tri retka programa. Već ste shvatili da taj redak upućuje računalo da se, ako Clock.Hour prikazuje da je ranije od 12 h, ispiše "Dobro jutro svima". Izrazi **If**, **Then** i **EndIf** posebni su izrazi koje računalo razumije nakon pokretanja programa. Nakon **If** uvijek slijedi uvjet, a to je u ovom slučaju **(Clock.Hour < 12)**. Ne zaboravite da su zagrade potrebne da bi računalo shvatilo vaše namjere. Nakon uvjeta slijedi **then** i stvarna operaciju koju je potrebno izvršiti. A nakon operacije dolazi **EndIf**. To računalo upućuje da je uvjetno izvršenje gotovo.

U programskom jeziku Small Basic za pristupanje trenutnom datumu i vremenu koristi se objekt Clock*. On vam nudi i velik broj svojstava pomoću kojih možete zasebno otvoriti trenutni dan, mjesec, godinu, sat, minute i sekunde.*

Između **then** i **EndIf** može postojati više operacija, a računalo će ih sve izvršiti ako je uvjet valjan. Na primjer, možete napisati nešto poput ovoga:

```
If (Clock.Hour < 12) Then
  TextWindow.Write("Dobro jutro. ")
  TextWindow.WriteLine("Jeste li doručkovali?")
EndIf
```

Izraz Else

U programu na početku ovog poglavlja možda ste primijetili da je drugi uvjet nekako suvišan. Vrijednost **Clock.Hour** može biti manja od 12 ili ne. Zapravo nije ni bilo potrebno izvoditi drugu provjeru. U takvim slučajevima možemo skratiti dvije naredbe **if..then..endif** u jednu pomoću novog izraza **else**.

Kad bismo ponovno napisali taj program pomoću **else**, evo kako bi izgledao:

```
If (Clock.Hour < 12) Then
  TextWindow.WriteLine("Dobro jutro svima")
Else
  TextWindow.WriteLine("Dobar dan svima")
EndIf
```

Taj će program imati potpuno istu funkciju kao onaj prvi pa dolazimo do vrlo važne lekcije u vezi s računalnim programiranjem:

> *u programiranju obično postoji puno načina da se napravi ista stvar. Ponekad jedan način ima više smisla od drugog. Odabir je na programeru. Kako budete pisali sve više programa i stjecali iskustvo, počet ćete primjećivati te različite tehnike i njihove prednosti i nedostatke.*

Uvlake

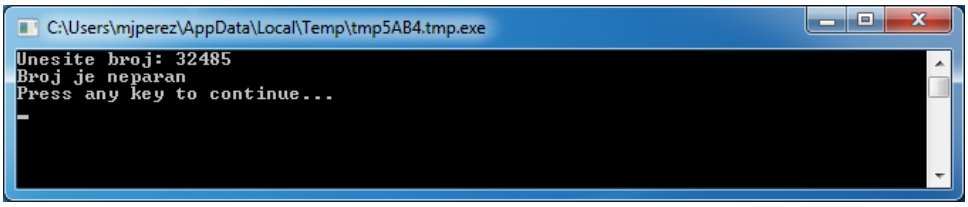
U svim primjerima možete vidjeti da su naredbe između *If,* Else i EndIf uvučene. Uvlake nisu potrebne. Računalo će bez problema razumjeti program i bez njih. No one nam pomažu da lakše vidimo i razumijemo strukturu programa. Stoga se obično smatra dobrom praksom uvući naredbe između takvih blokova.

Parno ili neparno

Sad kad smo svladali naredbu **If..Then..Else..EndIf**, napišimo program koji će odrediti je li neki broj paran ili neparan.

```
TextWindow.Write("Unesite broj: ")
broj = TextWindow.ReadNumber()
ostatak = Math.Remainder(broj, 2)
If (ostatak = 0) Then
  TextWindow.WriteLine("Broj je paran")
Else
  TextWindow.WriteLine("Broj je neparan")
EndIf
```

Kad pokrenete taj program, prikazat će se izlazni rezultat sličan ovom:



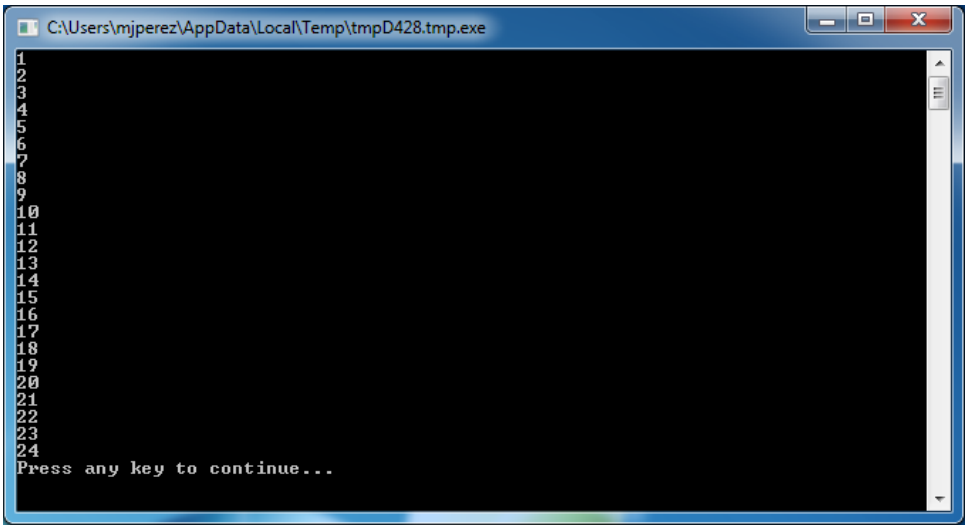
Slika 16 – Parno ili neparno

U tom smo programu uveli još jednu korisnu operaciju, **Math.Remainder**. I da, kao što ste možda već shvatili, **Math.Remainder** dijeli prvi broj s drugim, a zatim vraća ostatak.

Grananje

Ne zaboravite, u drugom poglavlju naučili ste da računalo obrađuje jednu po jednu naredbu programa i to redoslijedom od gore prema dolje. No postoji posebna naredba koja računalo upućuje da preskoči na drugu naredbu bez obzira na redoslijed. Pogledajmo sljedeći program.

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



Slika 17 – Korištenje naredbe Goto

U gornjem programu varijabli **i** dodijelili smo vrijednost 1. Potom smo dodali novu naredbu koja završava dvotočkom (:)

```
start:
```

To se zove *oznaka*. Oznake su kao knjižne oznake koje računalo razumije. Knjižnoj oznaci možete dati kakvo god ime želite i u program možete dodati koliko god oznaka želite, pod uvjetom da sve imaju jedinstveni naziv.

Evo još jedne zanimljive naredbe:

```
i = i + 1
```

Time se samo računalo upućuje da doda 1 varijabli **i** te da taj zbroj ponovno dodijeli varijabli **i**. Dakle, ako je vrijednost varijable **i** bila 1 prije te naredbe, nakon njezina izvođenja te će vrijednost biti 2.

I konačno,

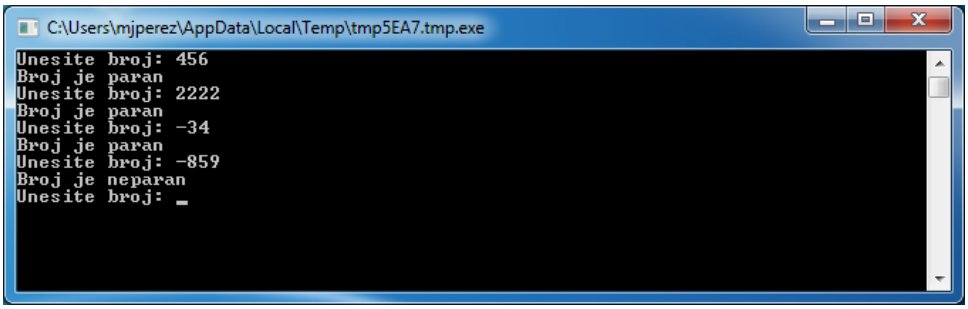
```
If (i < 25) Then
  Goto start
EndIf
```

To je dio koji upućuje računalo da, ako je vrijednost varijable **i** manja od 25, počne izvršavati naredbe iz knjižne oznake **start**.

Neograničeno izvršavanje

Pomoću naredbe **Goto** možete postići da računalo ponavlja nešto neograničen broj puta. Na primjer, možete uzeti program Parno ili neparno te ga izmijeniti na način prikazan u nastavku i program će se neprestano izvoditi. Program možete zaustaviti klikom na gumb Zatvori (X) na alatnoj traci u gornjem desnom kutu prozora.

```
početak:
TextWindow.Write("Unesite broj: ")
broj = TextWindow.ReadNumber()
ostatak = Math.Remainder(broj, 2)
If (ostatak = 0) Then
  TextWindow.WriteLine("Broj je paran")
Else
  TextWindow.WriteLine("Broj je neparan")
EndIf
Goto početak
```



Slika 18 – Neograničeno izvođenje programa Parno ili neparno

5. poglavlje

Petlje

Petlja For

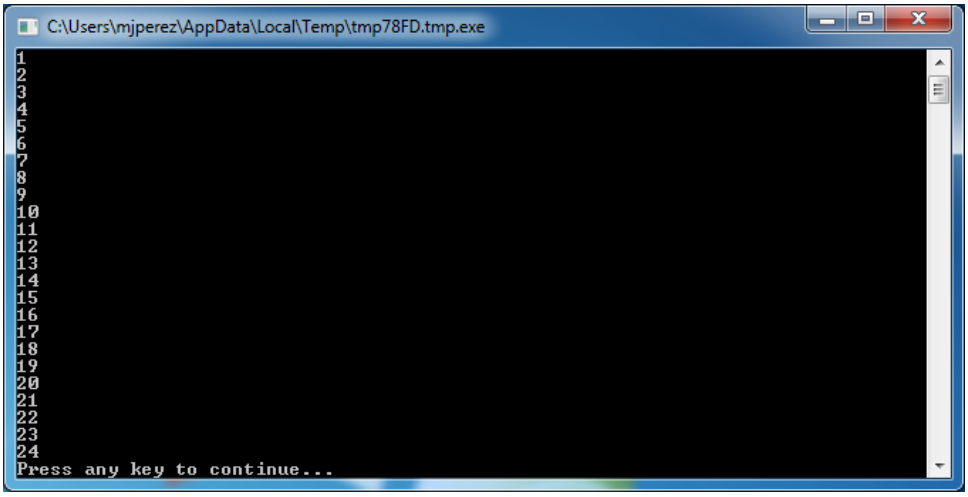
Pogledajmo program koji smo napisali u prethodnom poglavlju.

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

Taj program ispisuje brojeve od 1 do 24 po redu. Taj proces postupnog povećavanja varijable vrlo je uobičajen u programiranju pa programski jezici obično omogućuju jednostavniji način da se to napravi. Gornji program ekvivalentan je programu u nastavku.

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

A izlazni je rezultat sljedeći:



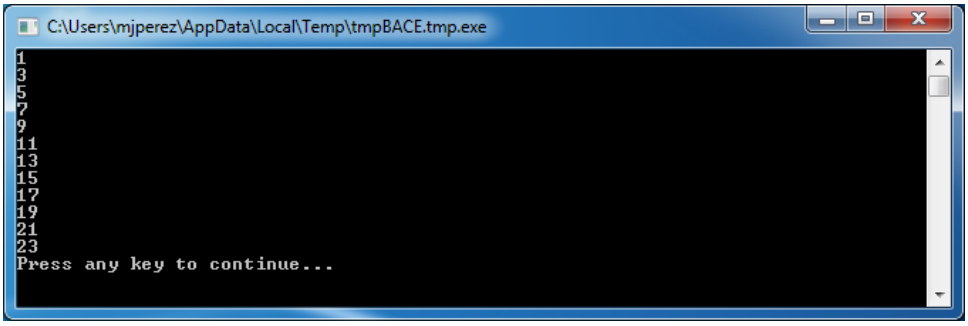
Slika 19 – Korištenje petlje For

Primijetit ćete da smo program od 8 redaka smanjili na program od 4 retka, a da ipak radi potpuno istu stvar kao program od 8 redaka! Sjećate se da smo ranije rekli da obično postoji nekoliko načina da se napravi ista stvar? To je sjajan primjer za to.

U programiranju se **For..EndFor** naziva *petlja*. Ona vam omogućuje da uzmete varijablu, date joj početnu i završnu vrijednost i pustite računalo da umjesto vas postupno povećava tu varijablu. Svaki put kad računalo postupno poveća varijablu, izvodi naredbe između petlji **For** i **EndFor**.

Ali ako želite da se varijabla poveća za 2, a ne za 1 jer, na primjer, želite ispisati sve neparne brojeve između 1 i 24, to možete napraviti pomoću petlje.

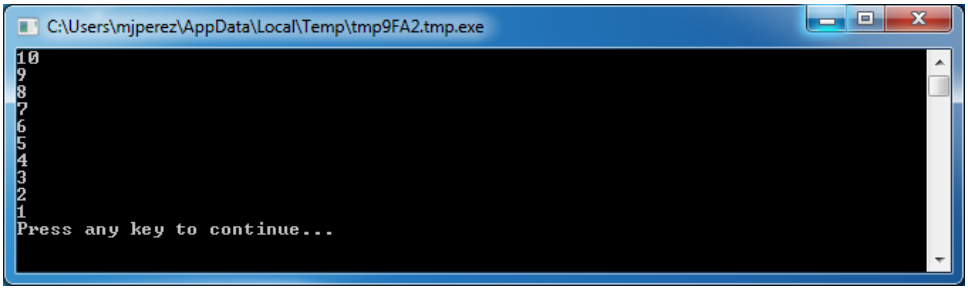
```
For i = 1 To 24 Step 2
  TextWindow.WriteLine(i)
EndFor
```



Slika 20 – Samo neparni brojevi

Dio **Step 2** naredbe **For** upućuje računalo da poveća vrijednost varijable **i** za 2 umjesto uobičajenog 1. Pomoću **Step** možete odrediti svako postupno povećanje koje želite. Možete čak odrediti i negativnu vrijednost za korak te postići da računalo broji unatrag, kao u primjeru u nastavku:

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

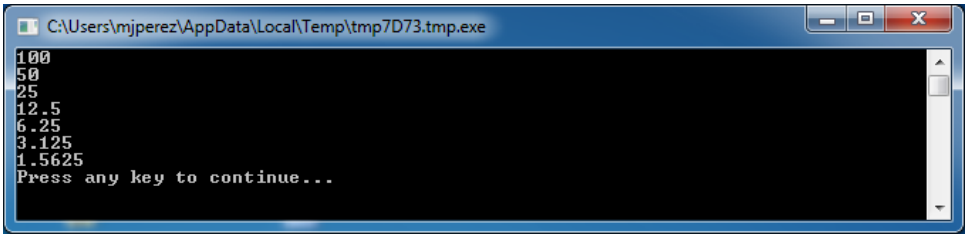


Slika 21 – Brojenje unatrag

Petlja While

Petlja While još je jedna metoda za izradu petlje, posebno ako broj petlje nije unaprijed poznat. Dok se petlja For izvodi unaprijed određeni broj puta, petlja While izvodi se dok se ne ispuni zadani uvjet. U primjeru u nastavku broj dijelimo napola sve dok je rezultat veći od jedan.

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



Slika 22 – Petlja za dijeljenje broja napola

U gornjem programu varijabli *broj* dodjeljujemo vrijednost 100 i izvodimo petlju while sve dok je broj već od 1. Unutar petlje ispisujemo broj, a zatim ga dijelimo s dva, čime ga zapravo dijelimo napola. Kao što se i očekivalo, izlazni rezultat programa brojevi su koji se progresivno jedan za drugim dijele napola.

Bilo bi teško napisati taj program pomoću petlje For jer ne znamo koliko će se puta petlja izvoditi. Pomoću petlje While jednostavno je provjeriti uvjet te zatražiti od računala da nastavi petlju ili da je zaustavi.

Zanimljivo je primijetiti da se svaka petlja While može razviti u naredbu If..Then. Na primjer, gornji program može se ponovno napisati na sljedeći način, a da to ne utječe na krajnji rezultat.

```
broj = 100
startLabel:
TextWindow.WriteLine(broj)
broj = broj / 2

If (broj > 1) Then
  Goto startLabel
EndIf
```

Zapravo, računalo interno ponovno zapisuje svaku petlju While u naredbe koje koriste If..Then zajedno s jednom ili više naredbi Goto.

6. poglavlje

Početak crtanja

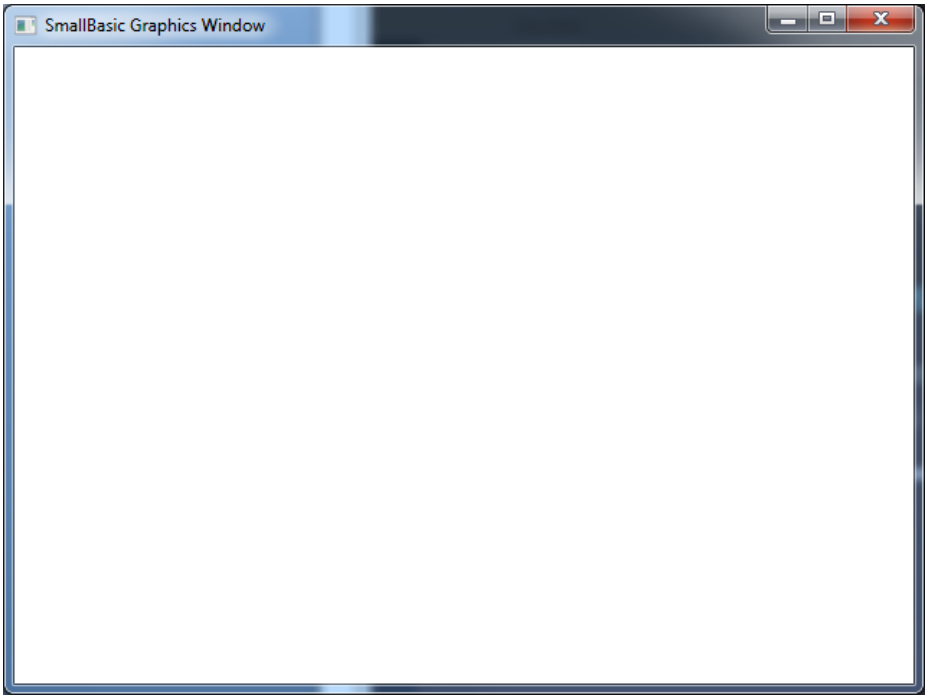
Dosad smo u našim primjerima pomoću prozora TextWindow objašnjavali osnove jezika Small Basic. No Small Basic ima napredan skup mogućnosti crtanja koje ćemo početi istraživati u ovom poglavlju.

Uvod u GraphicsWindow

Baš kao što smo imali prozor TextWindow koji nam je omogućio da radimo s tekstom i brojevima, Small Basic nudi i prozor GraphicsWindow pomoću kojeg možemo crtati. Za početak ćemo prikazati prozor **GraphicsWindow**.

```
GraphicsWindow.Show()
```

Prilikom pokretanja tog programa primijetit ćete da se umjesto uobičajenog crnog prozora s tekstom prikazuje bijeli prozor, kao što je ovaj prikazan u nastavku. Zasad u tom prozoru ne možemo ništa posebno raditi. Ali to će biti osnovni prozor u kojem ćemo raditi u ovom poglavlju. Možete ga zatvoriti klikom na gumb "X" u gornjem desnom kutu prozora.



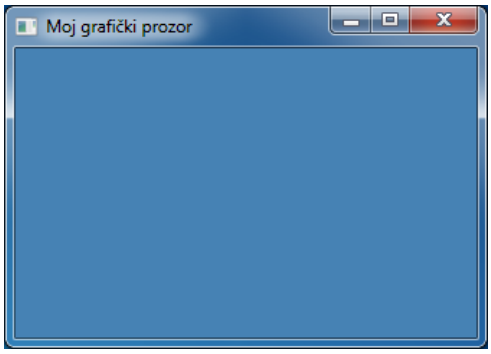
Slika 23 – Prazan grafički prozor

Postavljanje grafičkog prozora

Grafički prozor omogućuje vam da prilagodite njegov izgled prema svojim željama. Možete promijeniti naslov, pozadinu i njegovu veličinu. Izmijenimo ga sada malo, tek toliko da bismo se upoznali s njim.

```
GraphicsWindow.BackgroundColor = "SteelBlue"
GraphicsWindow.Title = "Moj grafički prozor"
GraphicsWindow.Width = 320
GraphicsWindow.Height = 200
GraphicsWindow.Show()
```

Evo kako izgleda prilagođeni grafički prozor. Boju pozadine možete promijeniti u bilo koju od puno vrijednosti navedenih u Dodatku B. Poigrajte se s tim svojstvima da biste vidjeli kako se može izmijeniti izgled prozora.

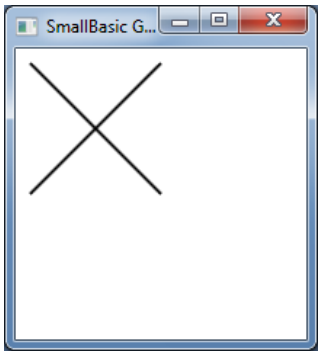


Slika 24 – Prilagođeni grafički prozor

Crtanje linija

Nakon što otvorimo prozorGraphicsWindow, u njemu možemo crtati oblike, tekst, pa čak i slike. Započnimo s crtanjem nekih jednostavnih oblika. Evo programa koji to crta nekoliko linija u grafičkom prozoru.

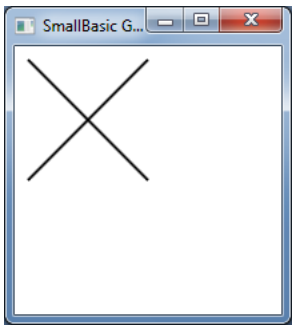
```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Slika 25 – Prekrižene linije

Prva dva retka programa služe za postavljanje prozora, a druga dva retka crtaju prekrižene linije. Prva dva broja koja slijede nakon *DrawLine* određuju početne x i y koordinate, a druga dva određuju završne x i y koordinate. Zanimljiva stvar kod računalnog crtanja to je što koordinate (0, 0) započinju u gornjem lijevom kutu prozora. Zbog toga se u koordinatnom prostoru uzima da se prozor nalazi u drugom kvadrantu.

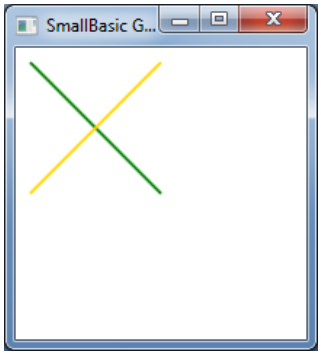
> *Umjesto naziva za boje možete koristiti oznake web-boja (#RRGGBB). Na primjer, #FF0000 označava crvenu, #FFFF00 žutu itd. Više o bojama saznat ćemo u [TODO poglavlju Boje]*



Slika 26 – Koordinatna mapa

Ako se vratimo na program za crtanje linija, zanimljivo je primijetiti da vam Small Basic omogućuje da izmijenite svojstva linije, kao što su njezina boja i debljina. Najprije ćemo izmijeniti boju linija kao što je prikazano u programu u nastavku.

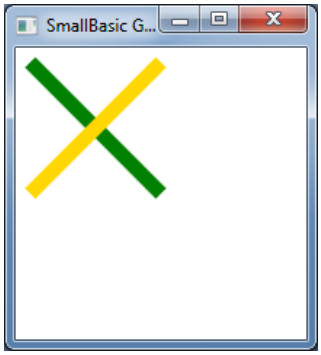
```
GraphicsWindow.Width = 200
GraphicsWindow.Height = 200
GraphicsWindow.PenColor = "Green"
GraphicsWindow.DrawLine(10, 10, 100, 100)
GraphicsWindow.PenColor = "Gold"
GraphicsWindow.DrawLine(10, 100, 100, 10)
```



Slika 27 – Promjena boje linije

Sada ćemo izmijeniti i njezinu veličinu. U programu u nastavku mijenjamo širinu linije u 10, umjesto zadane linije koja je 1.

```
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)
```

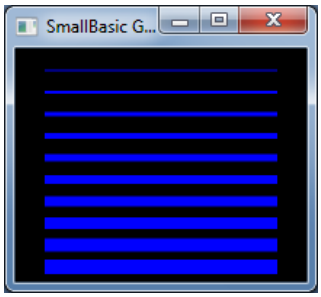


Slika 28 – Debele linije u bojama

Pomoću operacija *PenWidth* i *PenColor* mijenja se olovka kojom se iscrtavaju linije. One ne utječu samo na linije, već i na svaki oblik koji se crta nakon ažuriranja svojstava.

Pomoću naredbi petlji o kojima smo govorili u prethodnim poglavljima možemo jednostavno napisati program koji iscrtava više linija sa sve debljom olovkom.

```
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
```



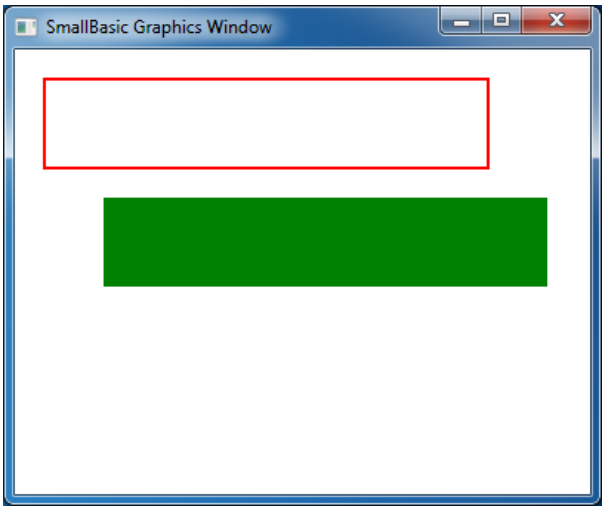
Slika 29 – Različite širine olovke

Zanimljivi dio tog programa jest petlja u kojoj povećavamo PenWidth prilikom svakog pokretanja petlje, a zatim crtamo novu liniju ispod stare.

Crtanje i popunjavanje oblika

Kad je riječ o crtanju oblika, obično postoje dvije vrste operacija za svaki oblik. To su operacije *Draw* (crtanje) i *Fill* (popunjavanje). U operacijama crtanja pomoću olovke se iscrtava obris oblika, a u operacijama popunjavanja oblik se boji pomoću kista. U programu u nastavku, primjerice, imamo dva pravokutnika. Jedan je nacrtan pomoću crvene olovke, a drugi popunjen pomoću zelenog kista.

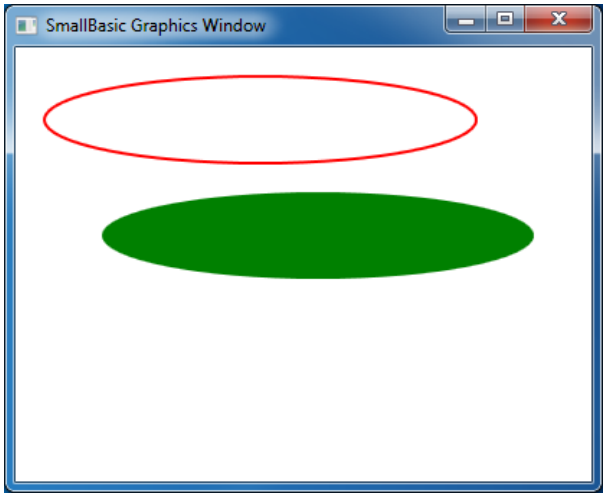
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor= "Red"
GraphicsWindow.DrawRectangle(20, 20, 300, 60)

GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillRectangle(60, 100, 300, 60)
```



Slika 30 – Crtanje i popunjavanje

Da bi se nacrtao ili popunio pravokutnik, potrebna su četiri broja. Prva dva predstavljaju X-koordinatu i Y-koordinatu za gornji lijevi kut pravokutnika. Treći broj određuje širinu pravokutnika, a četvrti njegovu visinu. Zapravo, isto se primjenjuje na crtanje i popunjavanje elipsa, kao što je prikazano u programu u nastavku.

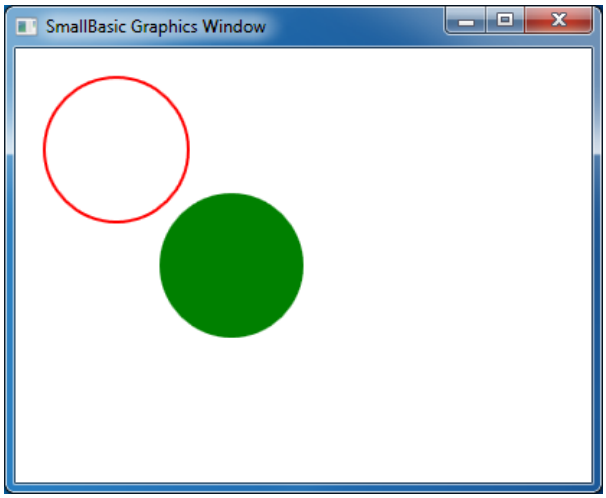
```
GraphicsWindow.Width = 400
GraphicsWindow.Height = 300

GraphicsWindow.PenColor= "Red"
GraphicsWindow.DrawEllipse(20, 20, 300, 60)

GraphicsWindow.BrushColor = "Green"
GraphicsWindow.FillEllipse(60, 100, 300, 60)
```



Slika 31 – Crtanje i popunjavanje elipsa

Elipse su samo varijanta krugova. Ako želite crtati krugove, morate odrediti 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 – Krugovi

7. poglavlje

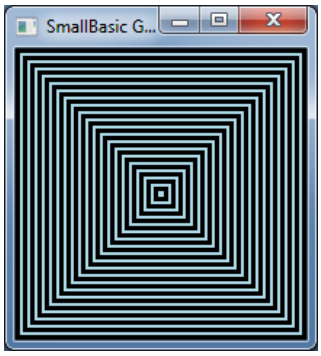
Zabava uz oblike

U ovom ćemo se poglavlju zabaviti uz sve što smo naučili dosad. U njemu se nalaze primjeri koji prikazuju neke zanimljive načine kombiniranja svega naučenog dosad da bi se izradili neki zgodni programi.

Mnoštvo pravokutnika

Ovdje crtamo više pravokutnika u petlji i svaki je veći od prethodnog.

```
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
```

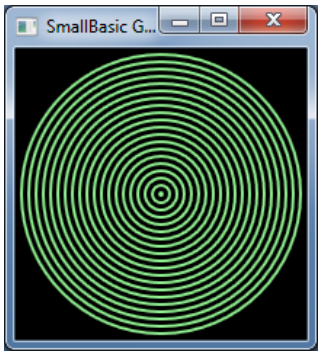


Slika 33 – Mnoštvo pravokutnika

Mnoštvo krugova

Varijanta prethodnog programa u kojoj se umjesto kvadrata crtaju krugovi.

```
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
```

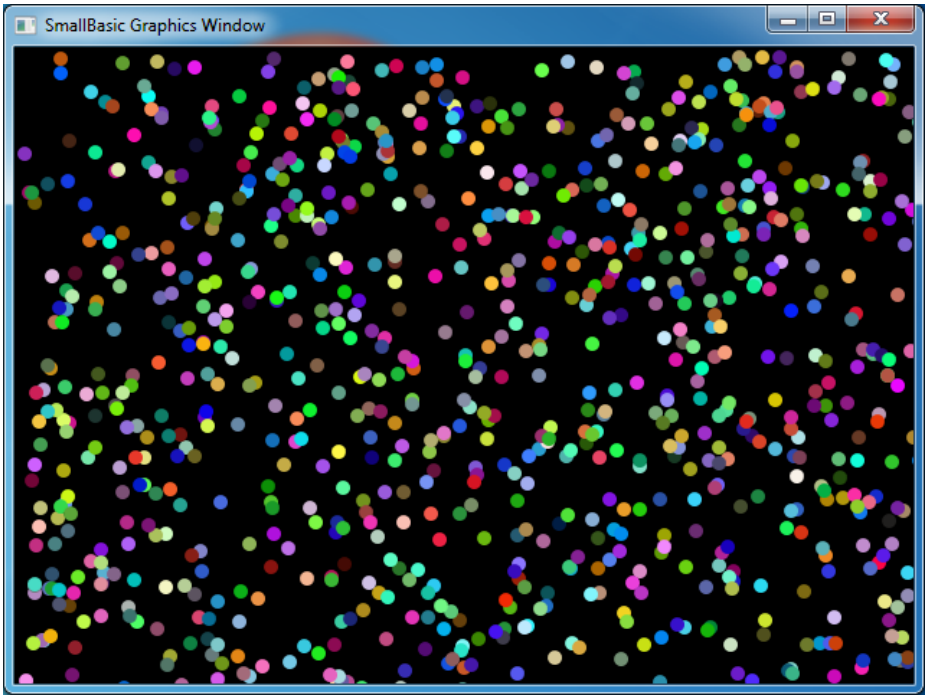


Slika 34 – Mnoštvo krugova

Korištenje slučajne boje

Ovaj program pomoću operacije GraphicsWindow.*GetRandomColor* određuje slučajne boje za kist, a zatim pomoću operacije Math.*GetRandomNumber* određuje x- i y-koordinate za krugove. Te se dvije operacije mogu kombinirati na zanimljive načine da bi se izradili interesantni programi koji daju različite rezultate prilikom svakog izvođenja.

```
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
```

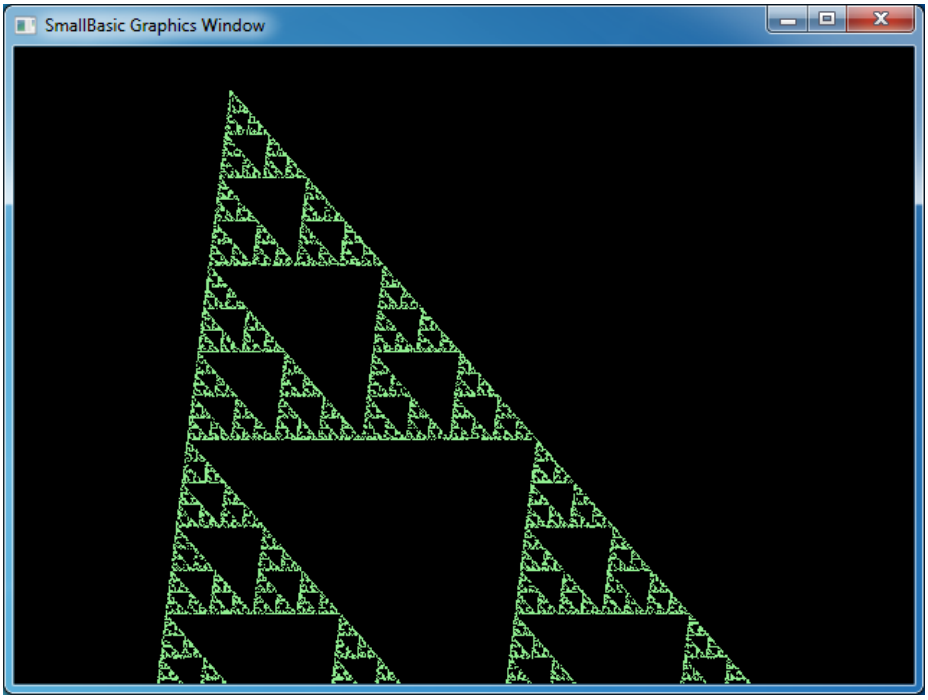


Slika 35 – Korištenje slučajne boje

Fraktali

U sljedećem programu moguće je nacrtati jednostavan fraktal u obliku trokuta pomoću slučajnih brojeva. Fraktal je geometrijski oblik koji je moguće podijeliti na dijelove, pri čemu svaki od dijelova u potpunosti nalikuje nadređenom obliku. U ovom slučaju program crta stotine trokuta od kojih svaki nalikuje svom nadređenom trokutu. A budući da se program izvodi nekoliko sekundi, možete uistinu vidjeti kako trokuti nastaju iz običnih točaka. Samu logiku ponešto je teško opisati pa ćemo vam prepustiti da to sami istražite 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 – Crtanje fraktala u obliku trokuta

Ako uistinu želite vidjeti kako točke polako oblikuju fraktal, možete u petlju uvesti odgodu pomoću operacije **Program.***Delay*. Ova operacija uzima broj kojim se u milisekundama navodi kolika će biti odgoda. Slijedi izmijenjeni program, a izmijenjeni redak prikazan je podebljano.

```
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
```

Povećanjem odgode program će se usporiti. Poigrajte se različitim duljinama trajanja da biste vidjeli što najbolje odgovara vašem ukusu.

Još jedna izmjena koju možete napraviti u ovom programu zamjena je sljedećeg retka:

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

ovim retkom

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

Zahvaljujući toj izmjeni program će crtati piksele trokuta koristeći slučajne boje.

8. poglavlje

Crtanje pomoću kornjače

Logo

Sedamdesetih godina dvadesetog stoljeća postojao je vrlo jednostavan, ali napredan programski jezik pod nazivom Logo koji je koristilo nekoliko istraživača. To je bilo tako dok netko programskom jeziku nije dodao nešto što se zove "crtanje pomoću kornjače" te time učinio dostupnom "kornjaču" koja je bila vidljiva na zaslonu i odgovarala na naredbe kao što su *Idi naprijed, Skreni desno, Skreni lijevo* itd. Pomoću kornjače ljudi su mogli crtati zanimljive oblike na zaslonu. To je programski jezik odmah učinilo pristupačnim i privlačnim ljudima svih dobnih skupina te je bilo uvelike odgovorno za njegovu ludu popularnost osamdesetih godina dvadesetog stoljeća.

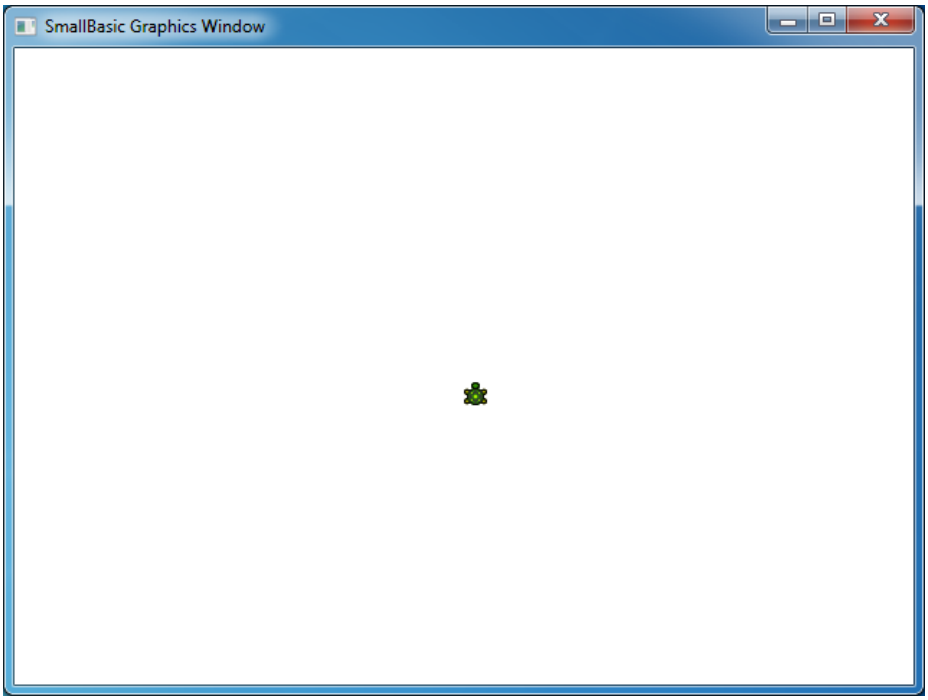
Small Basic ima objekt **Turtle** s velikim brojem naredbi koje se mogu pozivati unutar Small Basic programa. U ovom ćemo poglavlju pomoću kornjače crtati grafičke oblike na zaslonu.

Kornjača

Za početak moramo kornjaču učiniti vidljivom na zaslonu. To se može postići pomoću jednostavnog programa koji se sastoji od jednog retka.

```
Turtle.Show()
```

Prilikom pokretanja tog programa primijetit ćete bijeli prozor, isti kao onaj koji smo vidjeli u prethodnom poglavlju, osim što ovaj ima kornjaču u sredini. Upravo će ta kornjača pratiti naše upute i crtati sve što zatražimo od nje.



Slika 37 – Kornjača je vidljiva

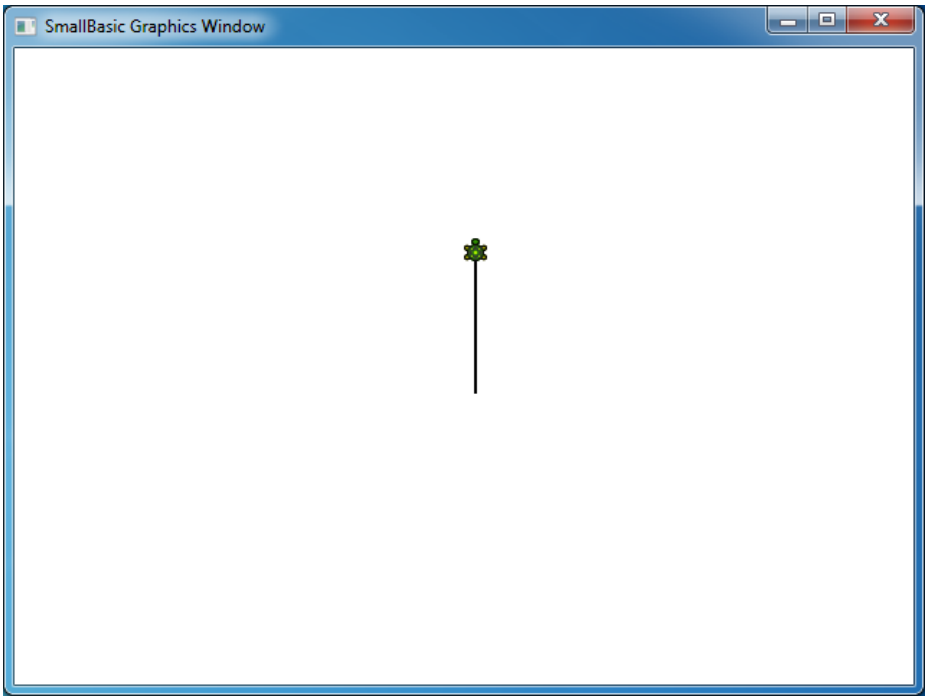
Pomicanje i crtanje

Jedna od naredbi koju kornjača razumije jest **Move**. Ta operacija uzima broj kao ulaznu vrijednost. On upućuje kornjaču koliko se daleko mora pomaknuti. Recimo, u primjeru u nastavku zatražit ćemo od kornjače da se pomakne za 100 piksela.

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

Prilikom pokretanja tog programa uistinu ćete vidjeti kornjaču kako se polako pomiče 100 piksela prema gore. Dok se pomiče, primijetit ćete i kako iza sebe povlači crtu. Nakon što se kornjača prestane pomicati, dobiveni rezultat izgledat će otprilike kao na slici u nastavku.

Prilikom korištenja operacija s kornjačom nije potrebno pozivati naredbu Show(). Kornjača će biti automatski vidljiva pri svakom izvršavanju operacije s kornjačom.



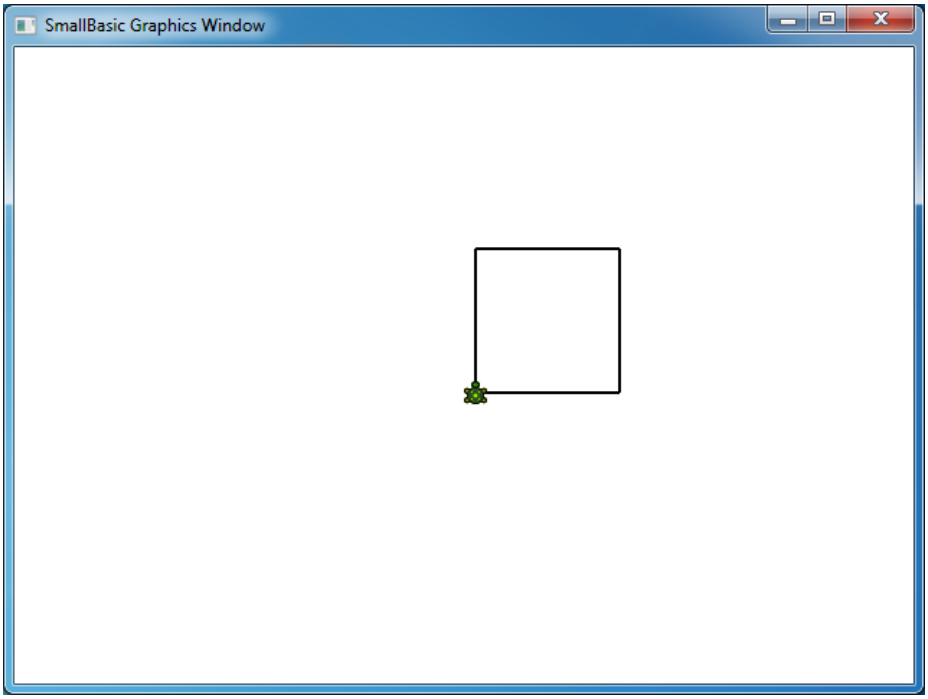
Slika 38 – Pomicanje za stotinu piksela

Crtanje kvadrata

Kvadrat ima četiri stranice, dvije okomite i dvije vodoravne. Da bismo nacrtali kvadrat, potrebno je postići da kornjača povuče liniju, okrene se desno i povuče još jednu liniju, a zatim nastavi dok nisu gotove sve četiri stranice. Evo kako to izgleda prevedeno u program.

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

Kad pokrenete taj program, vidjet ćete da kornjača crta kvadrat, liniju po liniju, a rezultat izgleda kao na slici u nastavku.



Slika 39 – Kornjača crta kvadrat

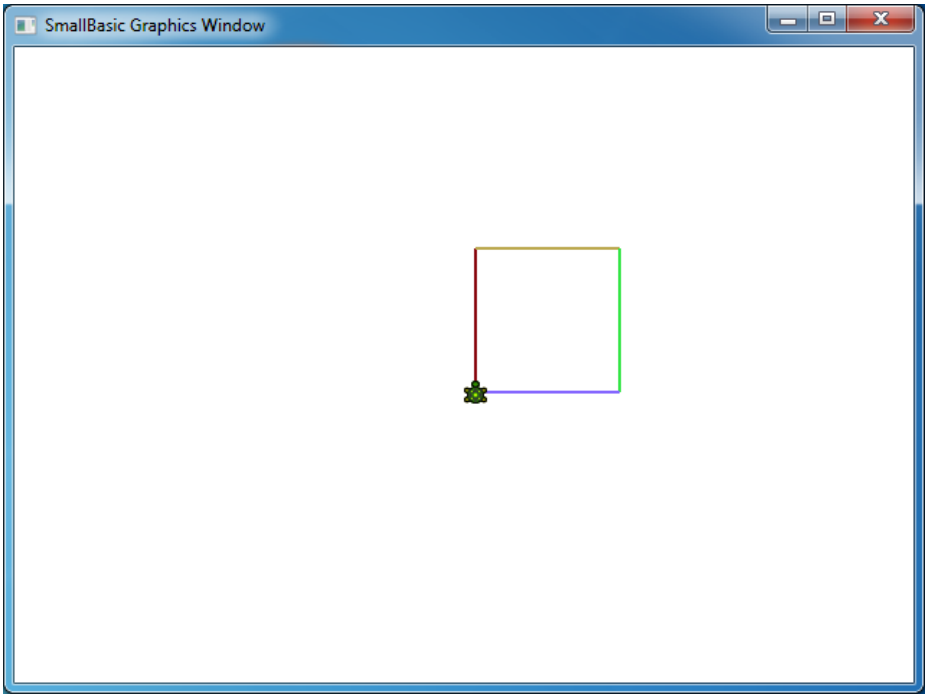
Zanimljivo je napomenuti da opetovano dajemo iste dvije naredbe – točno četiri puta. Već smo naučili da se takve naredbe koje se ponavljaju mogu izvršiti pomoću petlji. Dakle, ako uzmemo gornji program i izmijenimo ga tako da koristi petlju **For..EndFor**, dobit ćemo puno jednostavniji program.

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

Promjena boja

Kornjača crta u potpuno istom prozoru GraphicsWindow koji smo vidjeli u prethodnom poglavlju. To znači da su sve operacije koje smo upoznali u prethodnom poglavlju i dalje valjane ovdje. Na primjer, sljedeći program nacrtat će kvadrat sa svakom stranicom u različitoj boji.

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



Slika 40 – Promjena boja

Crtanje složenijih oblika

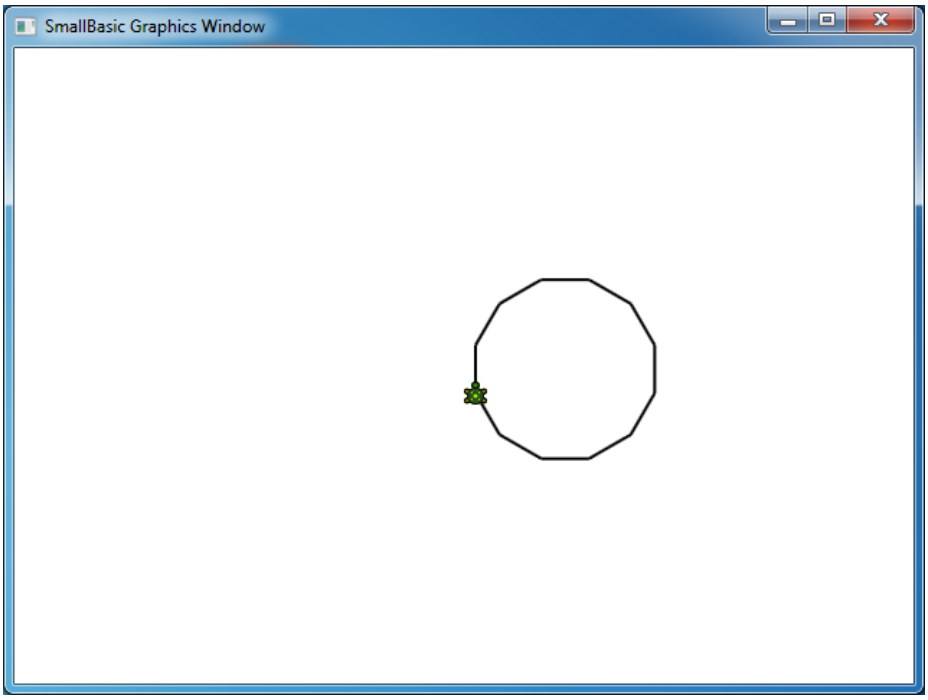
Kornjača, osim operacija **TurnRight** i **TurnLeft** izvršava operaciju **Turn**. Ta operacija uzima ulaznu vrijednost kojom se određuju kut zakretanja. Pomoću te operacije moguće je nacrtati mnogokut s bilo kojim brojem stranica. Sljedeći program crta šesterokut (mnogokut sa šest stranica).

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

Isprobajte taj program da biste vidjeli hoće li uistinu nacrtati šesterokut. Primijetit ćete da koristimo naredbu **Turn(60)** jer je kut između stranica 60 stupnjeva. Za takav mnogokut koji ima sve iste stranice, kut između stranica može se jednostavno dobiti podjelom broja 360 s brojem stranica. Kad znamo taj podatak pomoću varijabli možemo napisati prilično generički program koji može nacrtati mnogokut s bilo kojim brojem stranica.

```
stranice = 12

duljina = 400 / stranice
kut = 360 / stranice

For i = 1 To stranice
  Turtle.Move(duljina)
  Turtle.Turn(kut)
EndFor
```

Pomoću tog programa možete nacrtati bilo koji mnogokut izmjenom varijable **stranice**. Ako tu stavimo 4, dobit ćemo kvadrat s kojim smo započeli. Ako se stavi dovoljno velik broj, na primjer 50, rezultat bi bilo teško razlikovati od kruga.



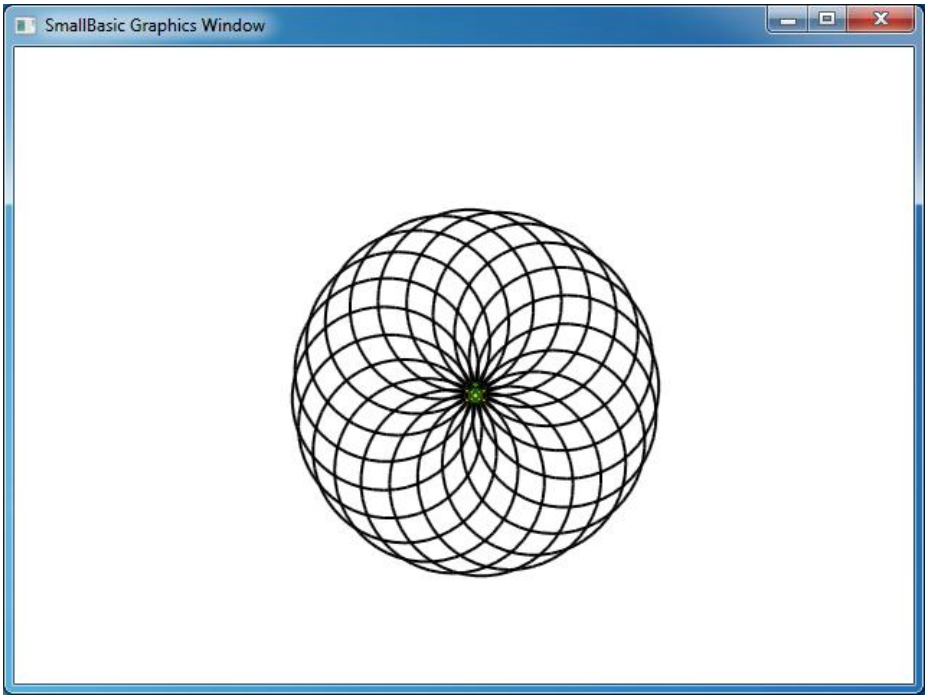
Slika 41 – Crtanje mnogokuta s 12 stranica

Pomoću tehnike koju smo upravo naučili, možemo postići da kornjača crta višestruke krugove, ali svaki put s malim pomakom, što daje zanimljiv rezultat.

```
stranice = 50
duljina = 400 / stranice
kut = 360 / stranice

Turtle.Speed = 9

For j = 1 To 20
  For i = 1 To stranice
    Turtle.Move(duljina)
    Turtle.Turn(kut)
  EndFor
  Turtle.Turn(18)
EndFor
```

Gornji program ima dvije **For..EndFor** petlje, jednu unutar druge. Unutarnja petlja (*i = 1 do stranica*) slična je programu za mnogokut i služi za crtanje kruga. Vanjska petlja (*j = 1 do 20*) služi za neznatno okretanje kornjače za svaki nacrtani krug. To kornjači govori da nacrta 20 krugova. Kad se sve to koristi zajedno, ovaj program rezultira vrlo zanimljivim uzorkom, kao što je onaj prikazan u nastavku.

U gornjem programu natjerali smo kornjaču da se brže kreće postavljanjem brzine na 9. To je svojstvo moguće postaviti na bilo koju vrijednost između 1 i 10 da bi se kretala onoliko brzo koliko želite.



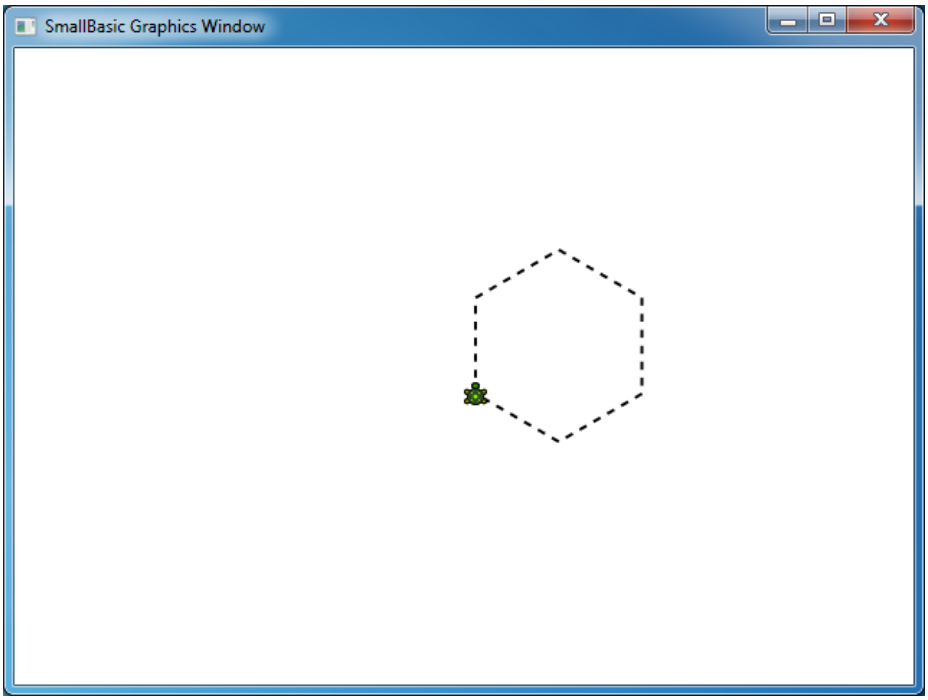
Slika 42 – Vrtnja u krug

Pomicanje uokolo

Ako ne želite da kornjača crta, možete pozvati operaciju **PenUp**. To vam omogućuje da pomaknete kornjaču bilo gdje na zaslonu, a da se pritom ne povlači linija. Pozivanjem naredbe **PenDown** kornjača će ponovno početi crtati. To se može koristiti za postizanje nekih zanimljivih efekata, kao što su istočkane linije. Evo programa koji to koristi za crtanje poligona s istočkanim linijama.

```
stranice = 6

duljina = 400 / stranice
kut = 360 / stranice

For i = 1 To stranice
  For j = 1 To 6
    Turtle.Move(duljina / 12)
    Turtle.PenUp()
    Turtle.Move(duljina / 12)
    Turtle.PenDown()
  EndFor
  Turtle.Turn(kut)
EndFor
```

Taj program ponovno ima dvije petlje. Unutarnja petlja crta jednu istočkanu liniju, a vanjska petlja određuje koliko će se linija iscrtati. U našem primjeru koristili smo 6 za varijablu **stranice** te smo stoga dobili šesterokut s istočkanim linijama, kao u nastavku.



Slika 43 – Korištenje naredbi PenUp i PenDown

9. poglavlje

Potprogrami

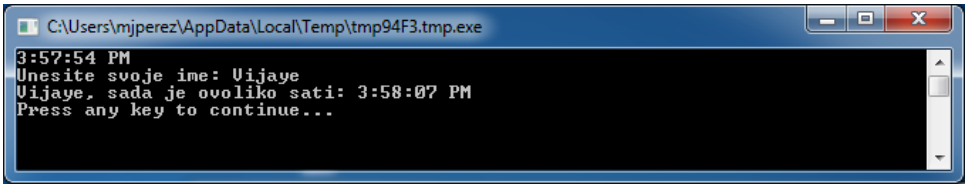
Često se prilikom pisanja programa dogodi da moramo uvijek iznova izvršavati isti skup koraka. U tim slučajevima vjerojatno ne bi imalo smisla više puta pisati iste izjave. Tada dobro dođu *potprogrami*.

Potprogram je dio koda unutar većeg programa koji obično izvršava određenu funkciju i koji se može pozvati bilo gdje u programu. Potprogrami se prepoznaju po nazivu koji slijedi iza ključne riječi **Sub**, a završavaju ključnom riječi **EndSub**. Na primjer, sljedeći isječak koda predstavlja potprogram pod nazivom *PrintTime*, a izvršava zadatak ispisa trenutnog vremena za TextWindow.

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

U nastavku je prikazan program koji obuhvaća potprogram i poziva ga ss različitih mjesta.

```
PrintTime()
TextWindow.Write("Unesite svoje ime: ")
ime = TextWindow.Read()
TextWindow.Write(ime + ", sada je ovoliko sati: ")
PrintTime()

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



Slika 44 – Pozivanje jednostavnog potprograma

Potprogram se izvršava pozivanjem naredbe SubroutineName(). Kao i obično, interpunkcijski znakovi "()" potrebni su da bi se računalu reklo da želite izvršiti potprogram.

Prednosti korištenja potprograma

Kao što smo upravo vidjeli gore, potprogrami omogućuju da se smanji količina koda koju je potrebno unijeti. Nakon što ste napisali potprogram *PrintTime*, možete ga pozvati bilo gdje u programu i on će ispisati trenutno vrijeme.

Ne zaboravite da potprogram programa SmallBasic možete pozvati samo unutar istog programa. Potprogram nije moguće pozvati unutar drugog programa.

Osim toga, potprogrami omogućuju rastavljanje složenih problema na jednostavnije dijelove. Ako, primjerice, morate riješiti složenu jednadžbu, morate napisati nekoliko potprograma koji rješavaju manje dijelove složene jednadžbe. Zatim možete spojiti rezultate da biste dobili rješenje izvorne složene jednadžbe.

Potprogrami mogu i poboljšati čitljivost programa. Drugim riječima, ako imate potprograme s odgovarajućim nazivima za dijelove programa koji se obično izvode, program će biti lako čitljiv i razumljiv. To je vrlo važno ako želite shvatiti program koji je napisao netko drugi ili ako želite da drugi shvate vaš program. Ponekad je to korisno čak i ako želite pročitati vlastiti program, primjerice tjedan dana nakon što ste ga napisali.

Korištenje varijabli

Unutar potprograma možete pristupiti svim varijablama u programu te ih koristiti. Sljedeći program, primjerice, prihvaća dva broja i ispisuje veći od ta dva broja. Primijetit ćete da se varijabla *maksimalno* koristi unutar potprograma i izvan njega.

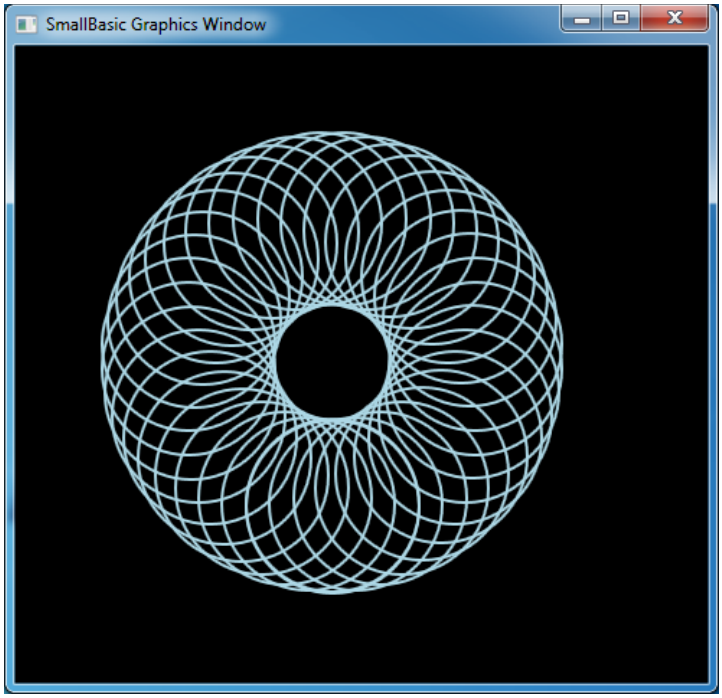
```
TextWindow.Write("Unesite prvi broj: ")
broj1 = TextWindow.ReadNumber()
TextWindow.Write("Unesite drugi broj: ")
broj2 = TextWindow.ReadNumber()

FindMax()
TextWindow.WriteLine("Maksimalni je broj: " + maksimalno)

Sub FindMax
  If (broj1 > broj2) Then
    maksimalno = broj1
  Else
    maksimalno = broj2
  EndIf
EndSub
```

A izlazni rezultat tog programa izgleda ovako.

Slika 45 – Određivanje većeg od dva broja pomoću potprograma

Pogledajmo još jedan primjer koji ilustrira korištenje potprograma. Ovaj ćemo puta koristiti grafički program koji izračunava različite točke koje će spremati u obliku varijabli *x* i *y*. Zatim poziva potprogram **DrawCircleUsingCenter** koji je odgovoran za crtanje kruga pomoću *x* i *y* kao središta.

```
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
```



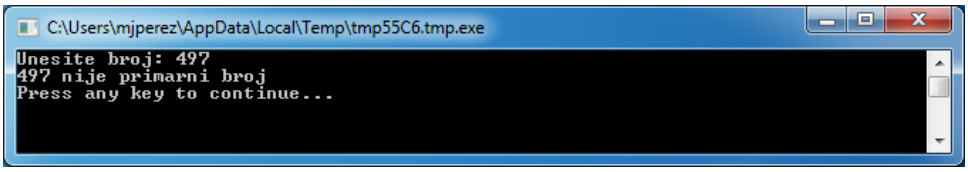
Slika 46 – Grafički primjer potprograma

Pozivanje potprograma unutar petlji

Ponekad se potprogrami pozivaju unutar petlje, a tijekom tog postupka izvršavaju isti skup izjava, ali s različitim vrijednostima u jednoj ili više varijabli. Na primjer, ako imate potprogram pod nazivom *PrimeCheck* i taj potprogram određuje je li broj primarni. Možete napisati program koji korisniku omogućuje da unese vrijednost, a zatim pomoću tog potprograma možete odrediti je li taj broj primaran. To ilustrira program u nastavku.

```
TextWindow.Write("Unesite broj: ")
i = TextWindow.ReadNumber()
isPrime = "True"
PrimeCheck()
If (isPrime = "True") Then
  TextWindow.WriteLine(i + " je primarni broj")
Else
  TextWindow.WriteLine(i + " nije primarni broj")
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 uzima vrijednost *i* i pokušava je podijeliti manjim brojevima. Ako je *i* djeljiv s brojem bez ostatka, to znači da *i* nije primarni broj. U tom trenutku potprogram postavlja vrijednost za *isPrime* na "False" i zatvara se. Ako broj nije djeljiv pomoću manjih brojeva, onda vrijednost *isPrime* ostaje "True".

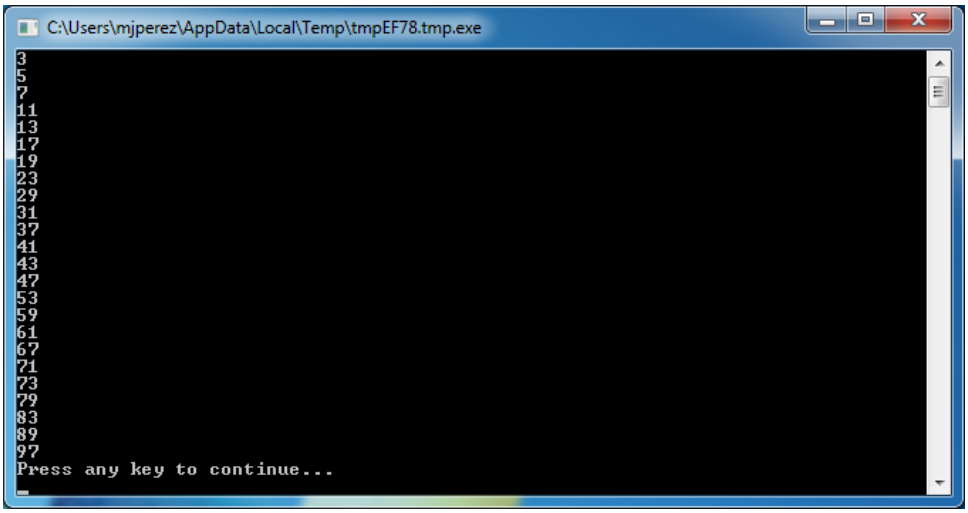


Slika 47 – Provjera primarnih brojeva

Sada kad imate potprogram koji može za vas izvršiti provjeru primarnih brojeva, pomoću njega možete, na primjer, napraviti popis svih primarnih brojeva manjih od 100. Gornji program uistinu je jednostavno izmijeniti i postići da se naredba *PrimeCheck* poziva iz petlje. To potprogramu omogućuje da prilikom svakog izvođenja petlje izračunava drugačiju vrijednost. Na primjeru u nastavku pogledajmo kako se to radi.

```
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 programu vrijednost za *i* ažurira se prilikom svakog izvođenja petlje. Unutar petlje izvodi se pozivanje potprograma *PrimeCheck*. Potprogram *PrimeCheck* potom uzima vrijednost *i* te izračunava je li *i* primarni broj. Taj se rezultat sprema u varijabli *isPrime* kojom se potom pristupa putem petlje izvan potprograma. Vrijednost za *i* potom se ispisuje ako je riječ o primarnom broju. Budući da petlja započinje od 3 i nastavlja se sve do 100, dobivamo popis svih primarnih brojeva u rasponu od 3 do 100. U nastavku je prikazan rezultat programa.



Slika 48 – Primarni brojevi

10. poglavlje

Polja

Sad ste se već majstor za korištenje varijabli – na koncu konca dospjeli ste čak do ovdje, a još se uvijek zabavljate, zar ne?

Idemo se na trenutak vratiti na prvi program koji smo napisali pomoću varijabli:

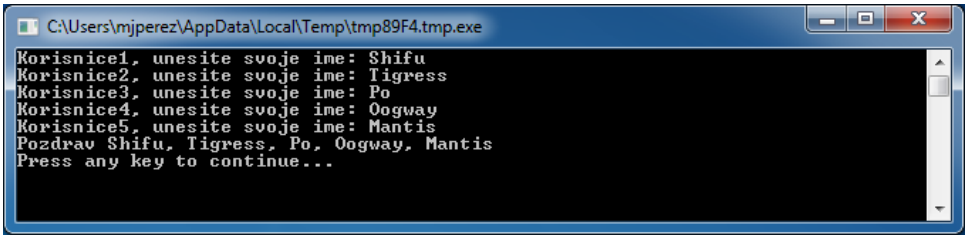
```
TextWindow.Write("Unesite svoje ime: ")
ime = TextWindow.Read()
TextWindow.WriteLine("Pozdrav" + ime)
```

U tom smo programu primili i spremili naziv korisnika u varijabli koja se zove **ime**. Zatim smo kasnije "pozdravili" korisnika. Pretpostavimo da postoji više korisnika, primjerice 5 korisnika. Kako bismo spremili sva njihova imena? To se može napraviti ovako:

```
TextWindow.Write("Korisniče1, unesite svoje ime: ")
ime1 = TextWindow.Read()
TextWindow.Write("Korisniče2, unesite svoje ime: ")
ime2 = TextWindow.Read()
TextWindow.Write("Korisniče3, unesite svoje ime: ")
ime3 = TextWindow.Read()
TextWindow.Write("Korisniče4, unesite svoje ime: ")
ime4 = TextWindow.Read()
TextWindow.Write("Korisniče5, unesite svoje ime: ")
ime5 = TextWindow.Read()
```

```
TextWindow.Write("Pozdrav ")
TextWindow.Write(ime1 + ", ")
TextWindow.Write(ime2 + ", ")
TextWindow.Write(ime3 + ", ")
TextWindow.Write(ime4 + ", ")
TextWindow.WriteLine(ime5)
```

Kad pokrenete taj potprogram, dobit ćete sljedeći rezultat:



Slika 49 – Nekorištenje polja

Očito mora postojati bolji način za pisanje tako jednostavnog programa, zar ne? Posebno zato što su računala uistinu dobra za obavljanje opetovanih zadataka pa zašto bismo se mučili i uvijek iznova pisali isti kod za svakog novog korisnika? Trik je u tome da se spremi i dohvati više od jednog imena korisnika pomoću iste varijable. Ako to možemo učiniti, onda možemo koristiti petlju **For** o kojoj smo govorili u prethodnim poglavljima. Tu nam priskaču u pomoć polja.

Što je polje?

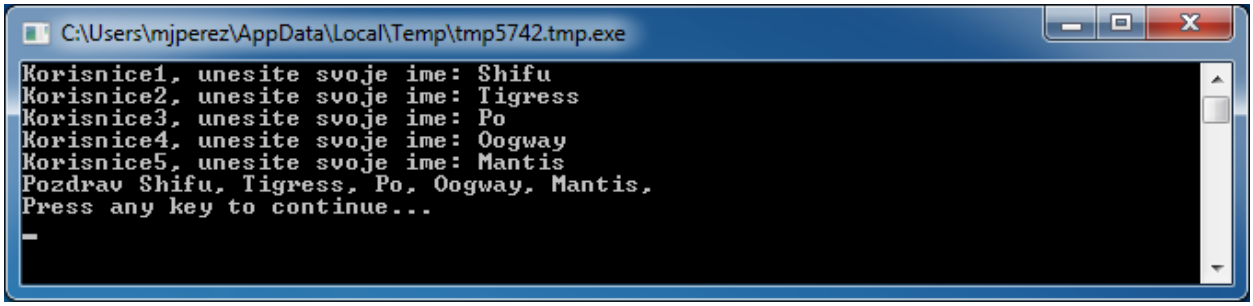
Polje je posebna vrsta varijable koja može istovremeno sadržavati više vrijednosti. To zapravo znači da bismo, umjesto izrade varijabli **ime1, ime2, ime3, ime4** i **ime5** za spremanje pet korisničkih imena, mogli koristiti samo varijablu **ime** za spremanje svih pet korisničkih imena. Više vrijednosti spremamo pomoću nečeg što zovemo "indeks". Na primjer, svaka od varijabli **ime[1], ime[2], ime[3], ime[4]** i **ime[5]** može spremiti vrijednost. Brojevi 1, 2, 3, 4 i 5 zovu se *indeksi* za polje.

Iako **ime[1]**, **ime[2]**, **ime[3]**, **ime[4]** i **ime[5]** izgledaju kao različite varijable, to je zapravo samo jedna varijabla. Možda ćete se zapitati koja je prednost toga. Najbolji dio spremanja vrijednosti u polje to je što možete odrediti indeks pomoću druge varijable, što nam omogućuje da jednostavno pristupimo poljima unutar petlji.

Pogledajmo kako možemo upotrijebiti svoje novostečeno znanje za ponovno pisanje prethodnog programa pomoću polja.

```
For i = 1 To 5
  TextWindow.Write("Korisniče" + i + ", unesite svoje ime: ")
  ime[i] = TextWindow.Read()
EndFor

TextWindow.Write("Pozdrav ")
For i = 1 To 5
  TextWindow.Write(ime[i] + ", ")
EndFor
TextWindow.WriteLine("")
```

Puno je lakše za čitanje, zar ne? Obratite pozornost na dva podebljana retka. Prvi sprema vrijednost u polju, a drugi je čita iz polja. Na vrijednost koju spremite u varijabli **ime[1]** ne utječe ono što spremite u varijabli **ime[2]**. Stoga **ime[1]** i **ime[2]** možete uglavnom tretirati kao dvije zasebne varijable s istim identitetom.



Slika 50 – Korištenje polja

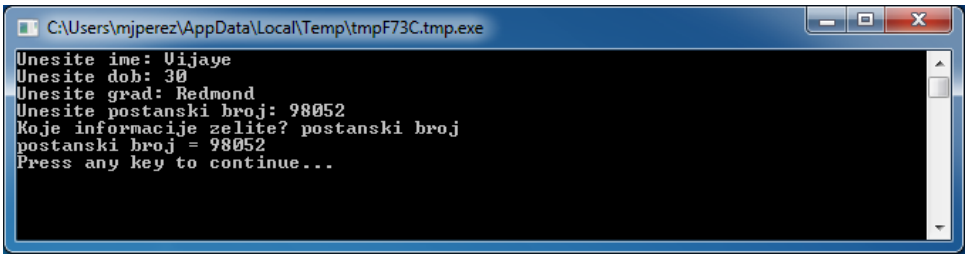
Gornji program daje gotovo potpuno isti rezultat kao onaj bez polja, osim zareza na kraju *mantise*. To možete popraviti tako da petlju za ispis napišete u ovom obliku:

```
TextWindow.Write("Pozdrav ")
For i = 1 To 5
  TextWindow.Write(ime[i])
  If i < 5 Then
    TextWindow.Write(", ")
  EndIf
EndFor
TextWindow.WriteLine("")
```

Indeksiranje polja

U našem prethodnom programu vidjeli ste da smo koristili brojeve kao indekse za spremanje i dohvaćanje vrijednosti iz polja. Ispada da indeksi nisu ograničeni samo na brojeve, a u praksi je vrlo korisno koristiti i tekstualne indekse. Na primjer, u sljedećem programu tražimo i spremamo različite informacije o korisniku, a zatim ispisujemo informacije koje je korisnik tražio.

```
TextWindow.Write("Unesite ime: ")
korisnik["ime"] = TextWindow.Read()
TextWindow.Write("Unesite dob: ")
korisnik["dob"] = TextWindow.Read()
TextWindow.Write("Unesite grad: ")
korisnik["grad"] = TextWindow.Read()
TextWindow.Write("Unesite postanski broj: ")
korisnik["postanski broj"] = TextWindow.Read()

TextWindow.Write("Koje informacije želite? ")
indeks = TextWindow.Read()
TextWindow.WriteLine(indeks + " = " + korisnik[indeks])
```



Slika 51 – Korištenje nenumeričkih indeksa

Više od jedne dimenzije

Pretpostavimo da želite spremiti ime i telefonski broj svih svojih prijatelja, a zatim moći potražiti njihove telefonske brojeve uvijek kad je to potrebno – nešto kao telefonski imenik. Kako napisati takav program?

U tom slučaju riječ je o dva skupa indeksa (to se još naziva dimenzijom polja). Pretpostavimo da svakog prijatelja prepoznajemo po nadimku. To postaje naš prvi indeks u polju. Nakon što pomoću prvog indeksa dohvatimo varijablu našeg prijatelja, drugi indeksi **ime** i **telefonski broj** omogućit će nam da dohvatimo stvarno ime I telefonski broj tog prijatelja.

Indeksi polja ne razlikuju veliko i malo slovo. Baš kao kod običnih varijabli, u pogocima indeksa polja ne moraju se točno poklapati velika i mala slova.

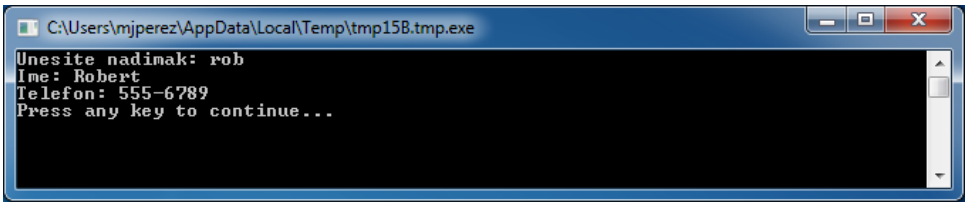
Način na koji spremamo te podatke izgleda ovako:

```
prijatelji["Rob"]["Ime"] = "Robert"
prijatelji["Rob"]["Telefon"] = "555-6789"

prijatelji["VJ"]["Ime"] = "Vijaye"
prijatelji["VJ"]["Telefon"] = "555-4567"

prijatelji["Ash"]["Ime"] = "Ashley"
prijatelji["Ash"]["Telefon"] = "555-2345"
```

Budući da imamo dva indeksa u istom polju **prijatelji**, to se polje naziva dvodimenzionalnim poljem.

Nakon što postavimo taj program, kao ulaznu vrijednost možemo uzeti nadimak prijatelja, a zatim ispisati informacije koje smo spremili o njima. Evo cijelog programa koji to radi:

```
prijatelji["Rob"]["Ime"] = "Robert"
prijatelji["Rob"]["Telefon"] = "555-6789"

prijatelji["VJ"]["Ime"] = "Vijaye"
prijatelji["VJ"]["Telefon"] = "555-4567"

prijatelji["Ash"]["Ime"] = "Ashley"
prijatelji["Ash"]["Telefon"] = "555-2345"

TextWindow.Write("Unesite nadimak: ")
nadimak = TextWindow.Read()

TextWindow.WriteLine("Ime: " + prijatelji[nadimak]["Ime"])
TextWindow.WriteLine("Telefon: " + prijatelji[nadimak]["Telefon"])
```



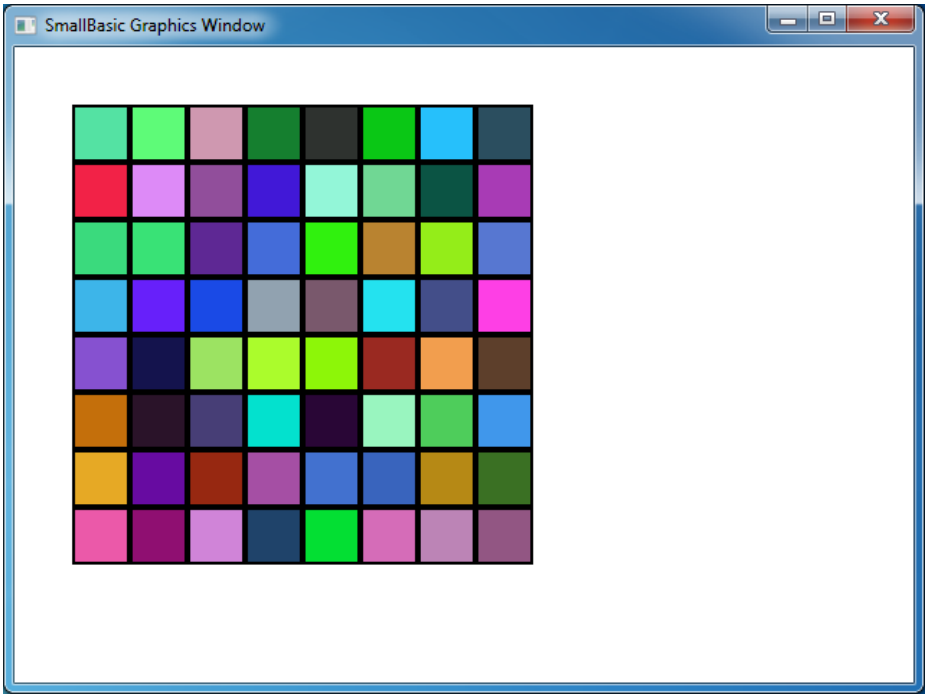
Slika 52 – Jednostavan imenik

Prikaz mreža pomoću polja

Višedimenzionalna polja obično se koriste za prikaz mreža/tablica. Mreže imaju retke i stupce koji se mogu uklopiti u dvodimenzionalno polje. U nastavku je naveden jednostavan program koji prikazuje okvire u mreži:

```
reci = 8
stupci = 8
veličina = 40

For r = 1 To reci
  For c = 1 To stupci
    GraphicsWindow.BrushColor = GraphicsWindow.GetRandomColor()
    boxes[r][c] = Shapes.AddRectangle(veličina, veličina)
    Shapes.Move(boxes[r][c], c * veličina, r * veličina)
  EndFor
EndFor
```

Taj program dodaje pravokutnike i postavlja ih u obliku mreže 8x8. Osim što slaže te okvire, on ih i sprema u polje. To pojednostavnjuje praćenje tih okvira i njihovo ponovno korištenje po potrebi.

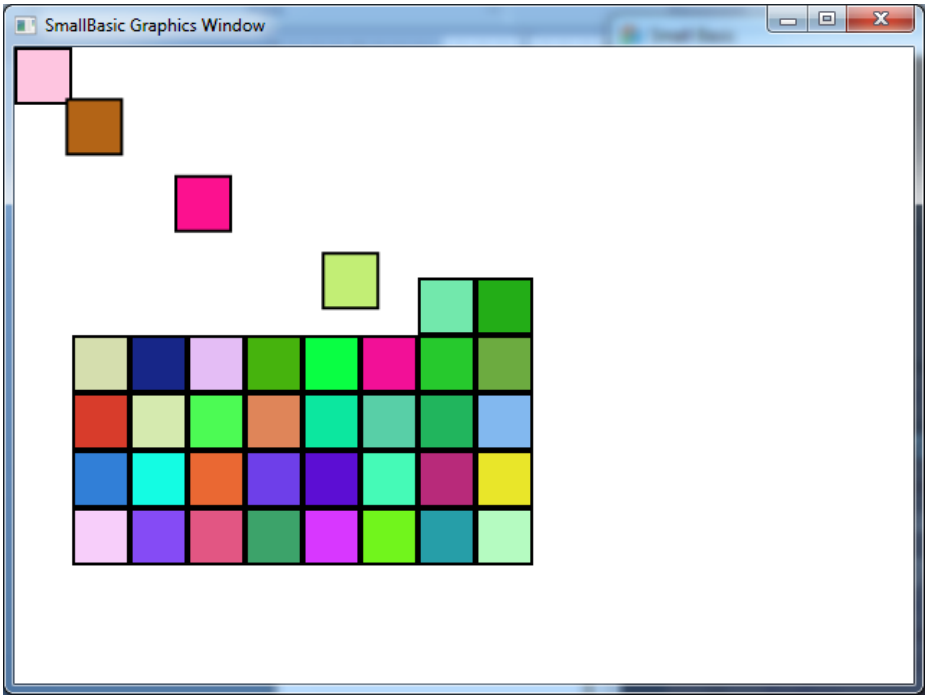


Slika 53 – Slaganje okvira u mrežu

Primjerice, dodavanjem sljedećeg koda na kraj prethodnog programa postići će se da se ti okviri animiraju do gornjeg lijevog kuta.

```
For r = 1 To reci
  For c = 1 To stupci
    Shapes.Animate(boxes[r][c], 0, 0, 1000)
```

```
    Program.Delay(300)
  EndFor
EndFor
```



Slika 54 – Praćenje okvira u mreži

11. poglavlje

Događaji i interaktivnost

U prva dva poglavlja predstavili smo objekte s varijablama *svojstva* i *operacije*. Osim svojstava i operacija, neki objekti imaju varijablu koja se zove ***događaji***. Događaji su kao signali koji se prikazuju, na primjer, kao odgovor na korisničke radnje, na primjer pomicanje miša ili klik mišem. Događaji su donekle suprotnost operacijama. U slučaju operacija vi kao programer pozivate ih da bi računalo napravilo nešto, dok vam u slučaju događaja računalo javlja kad se dogodi nešto zanimljivo.

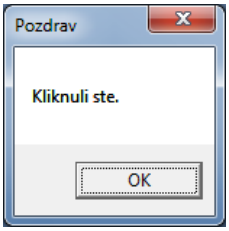
Koliko su korisni događaji?

Događaji su od ključne važnosti za uvođenje interaktivnosti u program. Ako želite dopustiti korisniku da stupi u interakciju s vašim programom, koristit ćete događaje. Na primjer, ako želite napisati program za igru križić-kružić. Želite dopustiti korisniku da odabere želi li biti križić ili kružić, zar ne? Za to služe događaji − korisničke ulazne informacije unutar programa primate pomoću događaja. Ako vam to nije previše jasno, ne brinite. Pogledat ćemo vrlo jednostavan primjer pomoću kojeg ćete shvatiti što su događaji i kako se mogu koristiti.

U nastavku je prikazan vrlo jednostavan program koji se sastoji od samo jedne naredbe i jednog potprograma. Potprogram koristi operaciju *ShowMessage* da bi na GraphicsWindow objektu prikazao okvir s porukom za korisnika.

```
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
  GraphicsWindow.ShowMessage("Kliknuli ste.", "Pozdrav")
EndSub
```

Jedna zanimljivost koju je moguće primijetiti u vezi s gornjim programom jest redak u kojem se dodjeljuje naziv potprograma za događaj **MouseDown** objekta GraphicsWindow. Vidjet ćete da MouseDown u velikoj mjeri izgleda kao svojstvo – osim što mu se umjesto vrijednosti dodjeljuje potprogram *OnMouseDown*. To je posebnost događaja – kad se dogodi, automatski se poziva potprogram. U ovom se slučaju potprogram *OnMouseDown* poziva svaki put kad korisnik mišem klikne na GraphicsWindow. Samo naprijed, pokrenite program i isprobajte ga. Svaki put kad mišem kliknete na GraphicsWindow, prikazat će se okvir s porukom kao što je onaj prikazan na slici u nastavku.

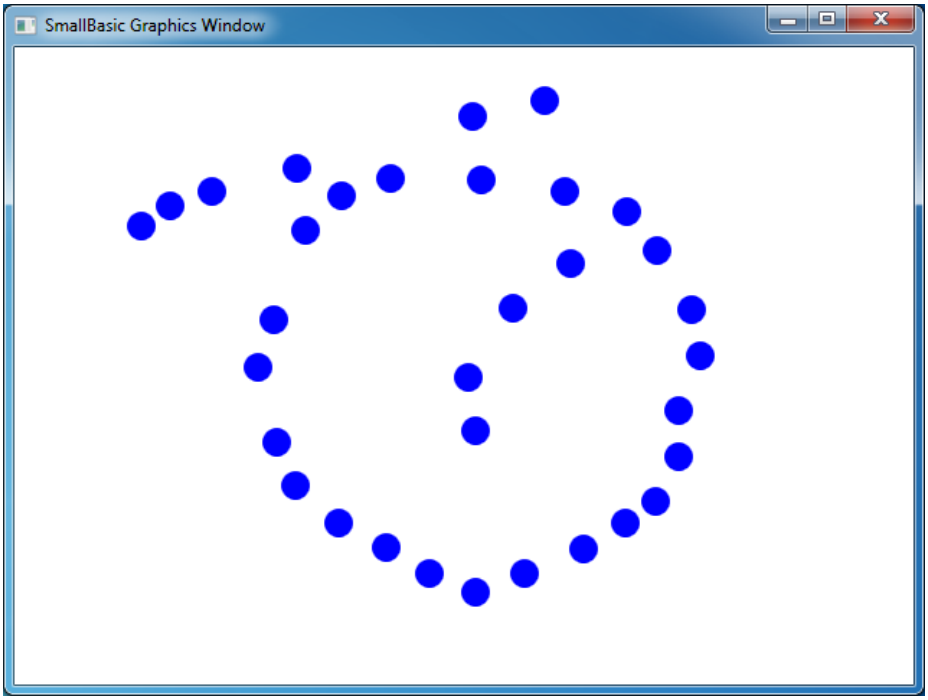


Slika 55 – Odgovor na događaj

Takva vrsta upravljanja događajem vrlo je napredna i omogućuje prilično kreativne i zanimljive programe. Programi napisani na taj način često se nazivaju programima koje pokreću događaji.

Potprogram *OnMouseDown* možete izmijeniti tako da radi druge stvari osim prikaza skočnog okvira s porukom. Na primjer, kao u programu u nastavku, možete nacrtati velike plave točke na mjestu gdje korisnik klikne mišem.

```
GraphicsWindow.BrushColor = "Blue"
GraphicsWindow.MouseDown = OnMouseDown

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



Slika 56 – Upravljanje događajem pomicanja miša prema dolje

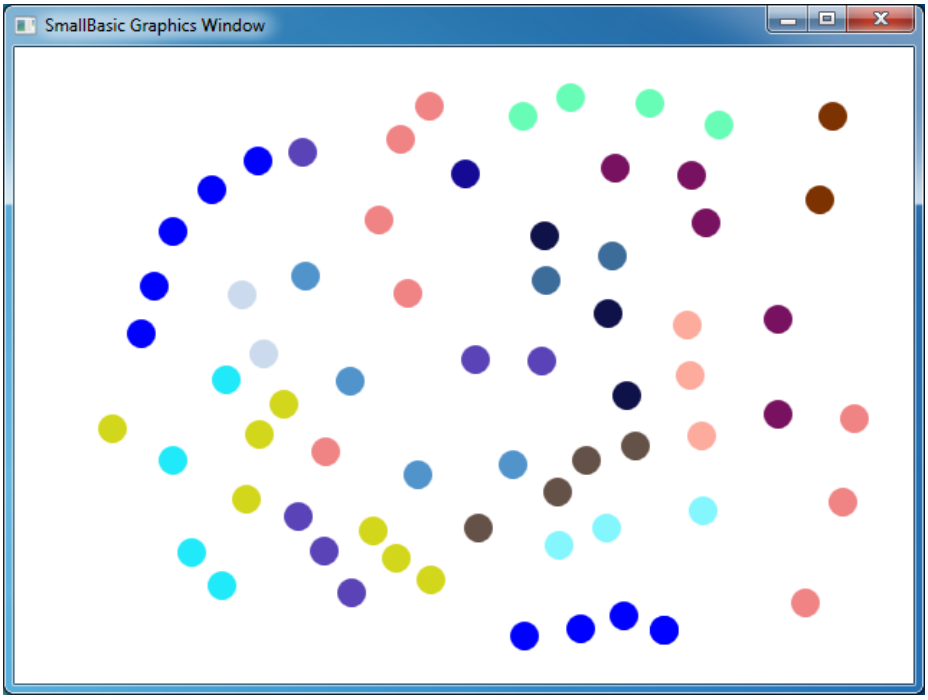
Napominjemo da smo u gornjem programu koristili *MouseX* i *MouseY* da bismo dohvatili koordinate miša. Potom to koristimo da bismo nacrtali krug pomoću koordinata miša kao središta kruga.

Upravljanje s više događaja

Zapravo ne postoje ograničenja u vezi s brojem događaja kojim se može upravljati. Možete čak imati jedan potprogram koji upravlja s više događaja. No događajem možete upravljati samo jednom. Ako istom događaju želite dodijeliti dva potprograma, prednost ima onaj drugi.

Da bismo to dokazali, uzmimo prethodni primjer i dodajmo potprogram kojim se upravlja pritiskom na tipke. Isto tako, postići ćemo da novi potprogram mijenja boju kista pa ćete nakon klika mišem dobiti točku drugačije 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 57 – Upravljanje s više događaja

Ako pokrenete taj program i kliknete prozor, prikazat će se plava točka. Ako sada jednom pritisnete bilo koju tipku i ponovno kliknete, dobit ćete točku drugačije boje. Nakon pritiska na tipku izvršava se potprogram *OnKeyDown*, a time se boja kista mijenja u slučajnu boju. Kad nakon toga kliknete mišem, iscrtava se krug pomoću novopostavljene boje, zbog čega nastaju točke slučajnih boja.

Program za bojanje

Naoružani događajima i potprogramima sad možemo napisati program koji korisniku omogućuje crtanje u prozoru. Takav je program iznenađujuće jednostavno napisati, pod uvjetom da problem podijelimo na manje dijelove. U prvom koraku napisat ćemo programu koji omogućuje korisnicima da pomiču miša bilo gdje u grafičkom prozoru te da ostave trag kamo god pomaknu miša.

```
GraphicsWindow.MouseMove = OnMouseMove

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

No kad pokrenete taj program, prva linija uvijek počinje u gornjem lijevom rubu prozora (0, 0). Taj problem možemo riješiti upravljanjem događajem *MouseDown* i bilježenjem vrijednosti *prevX* i *prevY* kad se to dogodi.

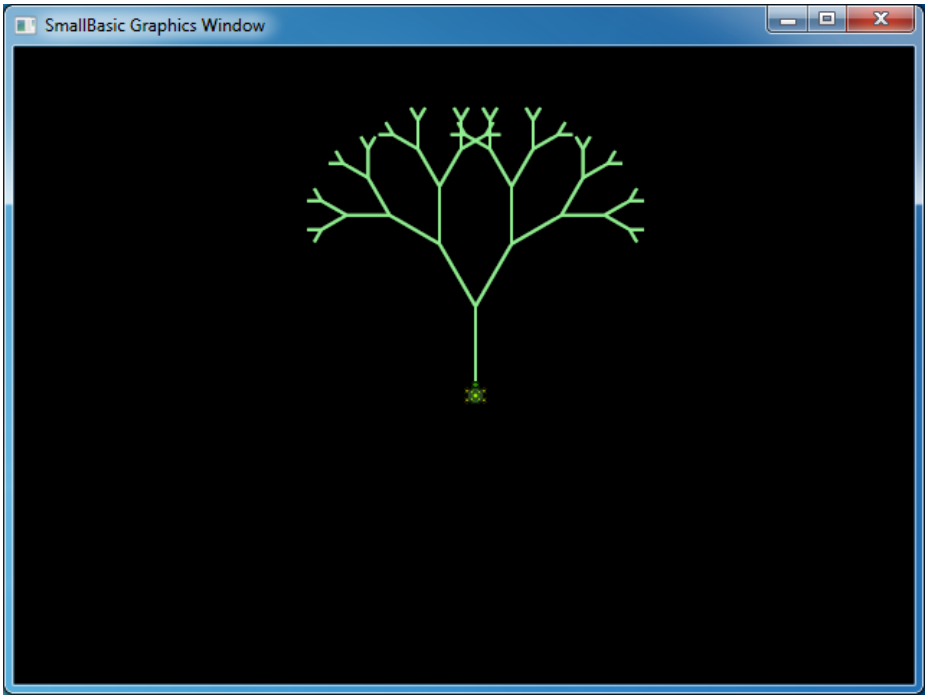
Isto tako, potrebno nam je da se iscrtavanje izvodi samo ako korisnik drži pritisnutu tipku miša. Ostatak vremena liniju nije potrebno iscrtavati. Da bismo to postigli, koristit ćemo svojstvo *IsLeftButtonDown* na objektu **Mouse**. To svojstvo govori drži li korisnik pritisnut lijevi gumb. Ako je ta vrijednost istinita, onda ćemo povući crtu, a ako nije, crtu ćemo preskočiti.

```
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 58 – kornjača crta fraktal u obliku stabla

```
kut = 30
delta = 10
udaljenost = 60
Turtle.Speed = 9
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.PenColor = "LightGreen"
DrawTree()

Sub DrawTree
  If (udaljenost > 0) Then
    Turtle.Move(udaljenost)
    Turtle.Turn(kut)

    Stack.PushValue("udaljenost", udaljenost)
    udaljenost = udaljenost - delta
    DrawTree()
    Turtle.Turn(-kut * 2)
    DrawTree()
    Turtle.Turn(kut)
    udaljenost = Stack.PopValue("udaljenost")

    Turtle.Move(-udaljenost)
  EndIf
EndSub
```

Fotografije s web-mjesta Flickr

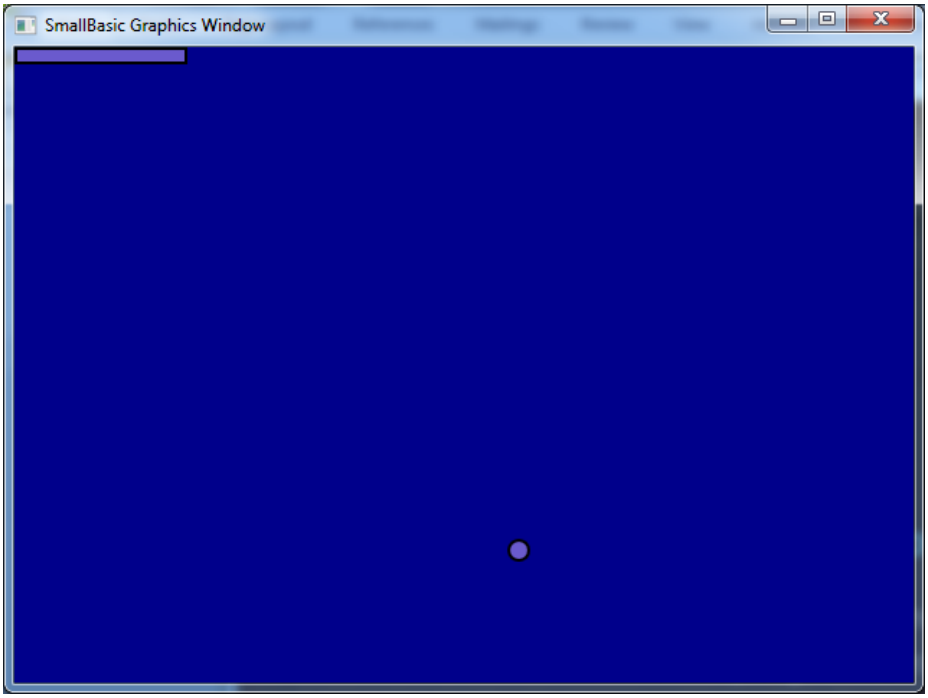
Slika 59 – dohvaćanje slika s web-mjesta Flickr

```
GraphicsWindow.BackgroundColor = "Black"
GraphicsWindow.MouseDown = OnMouseDown

Sub OnMouseDown
    slika = Flickr.GetRandomPicture("planine, rijeke")
    GraphicsWindow.DrawResizedImage(slika, 0, 0, 640, 480)
EndSub
```

Dinamična pozadina radne površine

```
Za i = 1 do 10
  slika = Flickr.GetRandomPicture("planine")
  Desktop.SetWallPaper(slika)
  Program.Delay(10000)
EndFor
```

Igra s lopticom



Slika 60 – igra s palicom

```
GraphicsWindow.BackgroundColor = "DarkBlue"
palica = Shapes.AddRectangle(120, 12)
loptica = 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 (palica)
  If (y = gh - 28 and x >= padX and x <= padX + 120) Then
    deltaY = -deltaY
  EndIf

  Shapes.Move(loptica, x, y)
  Program.Delay(5)

  If (y < gh) Then
    Goto RunLoop
  EndIf

GraphicsWindow.ShowMessage("Izgubili ste", "Palica")

Sub OnMouseMove
  paddleX = GraphicsWindow.MouseX
  Shapes.Move(palica, paddleX - 60, GraphicsWindow.Height - 12)
EndSub
```

Dodatak B

Boje

Slijedi popis 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 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 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 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 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 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