Šolski center Celje
Srednja šola za kemijo, elektrotehniko in računalništvo

PROGRAM ZA DELO S FUNKCIJAMI

raziskovalna naloga

PODROČJE: Računalništvo

Avtorica:
Maša KRANER, R-3.b

Mentor:
mag. Boštjan RESINOVIČ, univ. dipl. inž. rač.

Mestna občina Celje, Mladi za Celje
Celje, 2020

Šolski center Celje

Srednja šola za kemijo, elektrotehniko in računalništvo

PROGRAM ZA DELO S FUNKCIJAMI

raziskovalna naloga

PODROČJE: Računalništvo

Avtorica:

Maša KRANER, R-3.b

Mentor:

mag. Boštjan RESINOVIČ, univ. dipl. inž. rač.

Mestna občina Celje, Mladi za Celje

Celje, 2020

Kazalo vsebine

Kazalo slik

Zahvala

Pri izdelavi raziskovalne naloge se zahvaljujem svojemu mentorju, mag. Boštjanu Resinoviču, za pomoč pri snovanju naloge ter reševanje problemov z vidika programiranja. Zahvaljujem se tudi prof. Nataši Besednjak za pomoč pri reševanju matematičnih problemov. Posebej se zahvaljujem razredniku, Boštjanu Lubeju, ki mi je omogočil izostajanje od pouka.

Zahvaljujem se tudi lektorici, mag. Andreji Tkalec in prof. Ireni Sojč za pregled povzetka v angleščini.

Povzetek

Cilj raziskovalne naloge je bil povezati znanje s področja matematike in računalništva. Matematične funkcije so predpisi, ki vsakemu elementu x množice A priredi natanko določen element y množice B. Ker dijakom predstavljajo veliko težavo pri učenju, sem se odločila narediti aplikacijo ProGrafi, ki riše in računa razne naloge linearne in kvadratne funkcije, ki sta v raziskovalni nalogi podrobneje opisani. Aplikacijo sem naredila v Windows Forms Application, ki je platforma za pisanje grafičnih aplikacij za namizne, prenosne in tablične računalnike. Uporabljala sem ga v programu Visual Studio, ki je tako imenovano integrirano razvojno okolje (IDE), ki ga lahko uporabljamo za pisanje računalniških programov, spletnih mest, spletnih aplikacij in spletnih storitev. Med ustvarjanjem aplikacije sem naletela na mnoga vprašanja in probleme, ki sem jih na koncu uspešno rešila.

KLJUČNE BESEDE: Visual Studio, Windows Forms Application, elementarne funkcije, linearna funkcija, kvadratna funkcija, programiranje, matematika

Abstract

The aim of the research paper has been to integrate the knowledge from the fields of mathematics and computer science. Mathematical functions are rules that assign a specific element y of the set B to each element x of the set A . Since they present a major learning difficulty for students, I decided to create a ProGrafi application that draws and calculates various tasks of two types of functions: a linear and quadratic function; both functions are explained in more detail in this research. The application was created as a Windows Forms Application, which is a graphical platform for writing applications for desktop, laptop and tablet PCs. I used it in a program called Visual Studio, which is an integrated development environment (IDE) that can be used to write computer programs, web sites, web applications and web services. While I was creating the application ProGrafi, I encountered many issues and problems that I eventually solved at the end of my research.

KEY WORDS: Visual Studio, Windows Forms Application, ProGrafi, mathematical functions, linear function, quadratic function, programming, mathematics

1 Uvod

Aplikacijo ProGrafi sem ustvarila z namenom, da bi povezala dva predmeta, ki me najbolj zanimata, in sicer matematiko in programiranje. Ker so elementarne funkcije v srednji šoli ena izmed najtežjih snovi matematiki, sem želela narediti aplikacijo, ki bi risala in računala namesto uporabnika. V nadaljevanju bom opisala dva programa, ki sem ju uporabila za ustvarjanje aplikacije. To je program Visual Studio, v katerem je platforma Windows Forms Application. Prav tako sem bolj podrobneje opisala linearno in kvadratno funkcijo, ki sem ju uporabila v aplikaciji. Opisala bom tudi ustvarjeno aplikacijo ProGrafi, njen namen, kako deluje, opisala bom vsako funkcijo, ki jo izračuna in nariše ter njeno uporabnost. Ker pa more biti vsaka aplikacija dostopna uporabnikom in imeti namen, sem razložila tudi, kako lahko vsak uporabnik dostopa do nje, čeprav nima naloženega Visual Studia.

1.1 Predstavitev problema

Matematika predstavlja velik problem dijakom srednjih šol. Precej problematična snov so elementarne funkcije. Tako sem želela ustvariti aplikacijo, ki bi dijakom in uporabnikom olajšala učenje in razumevanje funkcij. Največji problem dijakov je količina enačb za računanje različnih nalog in risanje grafov. S svojim dosedanjim znanjem, brez poznavanja odvodov, je učenje enačb na pamet žal nujno.

1.2 Hipoteze

Pred začetkom raziskovalne naloge sem si postavila naslednje hipoteze:

H1: Z izdelavo aplikacije bom bolje razumela risanje grafov v računalniškemu okolju.

H2: Windows Forms Application se bo pokazal kot dobra izbira platforme za pisanje programa.

H3: Programski del bo zame predstavljal večjo težavo kot matematični.

H4: Aplikacija bo prikazala vse elementarne funkcije, ki se obravnavajo v srednji šoli.

1.3 Raziskovalne metode

Moje raziskovanje se je začelo z zbiranjem informacij, ki so mi kasneje pomagale pri izdelavi celotne aplikacije. Na začetku sem se usmerila k matematičnemu delu s pomočjo profesorice za matematiko in znanjem iz prejšnjih let, ki sem ga pridobila pri matematiki. Pri tem sem prišla do spoznanja, da bom morala obseg aplikacije zmanjšati, saj še nimam dovolj znanja, da bi lahko aplikacija prikazala vse elementarne funkcije. Tako sem izbor zožila na linearno in kvadratno funkcijo. Ko sem natančno razumela obe funkciji, ki ju bo aplikacija računala in risala, sem se lotila programskega dela aplikacije. Pri raziskovalnem delu sem si pomagala predvsem z mentorjevimi nasveti in brskalnikom. Ko je bila aplikacija končana, sem pričela raziskovati, kako je lahko prijaznejša in na voljo širšemu krogu uporabnikov. Raziskovala sem predvsem s pomočjo brskalnika in s pogovorom s hipotetičnimi uporabniki.

2 Uporabljeni programi

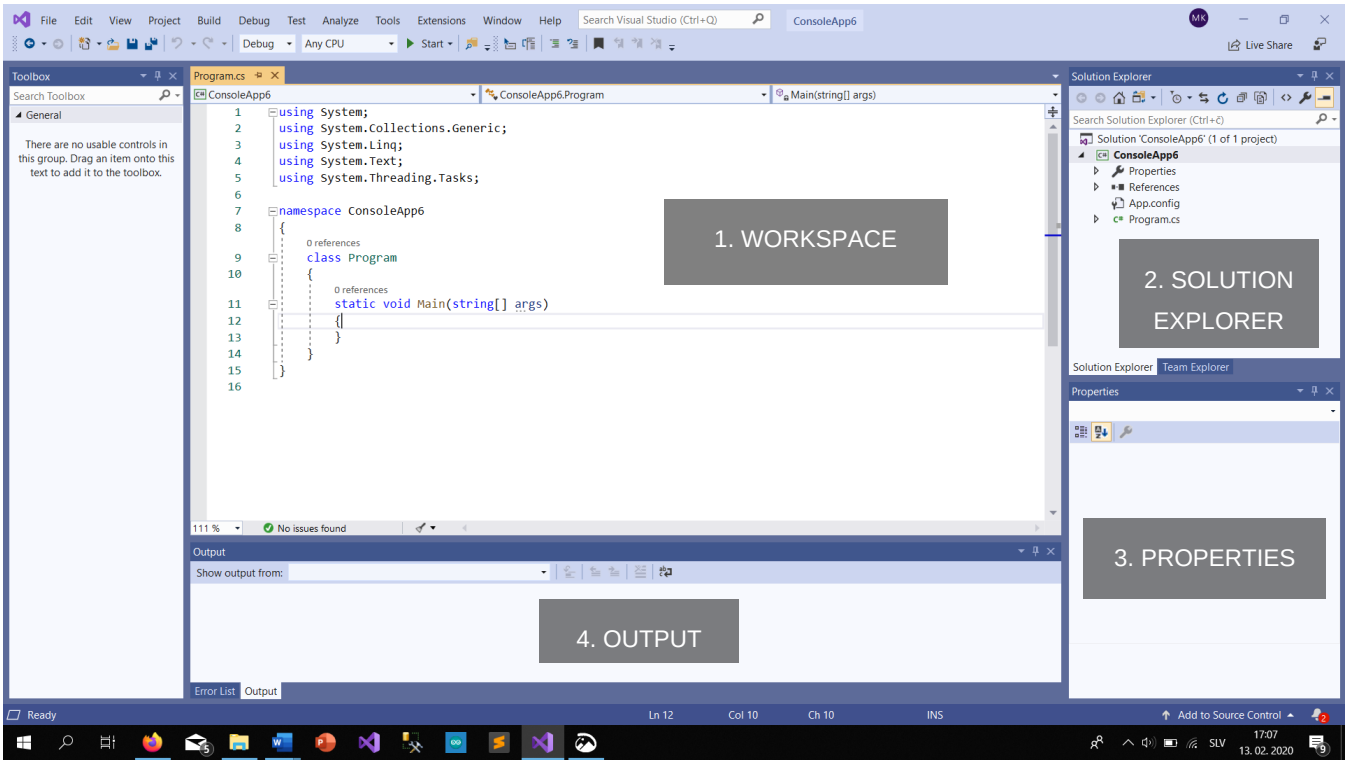
2.1 Visual Studio

Visual Studio je integrirano razvojno okolje (IDE), ki ga lahko uporabljamo za pisanje računalniških programov, spletnih mest, spletnih aplikacij in spletnih storitev. Poleg standardnega urejevalnika in razhroščevalnika, ki ga ponuja večina razvojnih okolij, Visual Studio vključuje še prevajalnike, orodja za dokončanje kode, grafične oblikovalce in številne druge funkcije za lažji postopek razvoja programske opreme. Podpira več programskih jezikov (C, C++, C#, Python, HTML/CSS, JavaScript itd.). Na voljo je v brezplačni različici "Community" in v plačani komercialni različici »Professional«.

Različica Community vsakemu posamezniku omogoča, da razvije svoje brezplačne ali plačljive spletne aplikacije. To različico lahko skupaj uporablja do pet uporabnikov, njen glavni namen pa je zagotoviti večji dostop do razširitev in omogočiti urejanje čim več različnih programskih jezikov. Različica Professional ponuja podporo za urejanje XML in XSLT ter vključuje orodje, kot je raziskovalec strežnikov in integracijo s strežnikom Microsoft SQL. Na voljo je tudi brezplačna poskusna različica te izdaje, ki jo mora uporabnik po poskusnem obdobju plačati, da jo lahko še naprej uporablja. Njeni glavni namen je zagotoviti fleksibilnost (profesionalna orodja za gradnjo katere koli vrste aplikacij), storilnost in sodelovanje med uporabniki.

Preden se uporabnik loti pisanja kode, se mu prikaže začetno okno, ki ga lahko vidite na sliki 1. Na njej so označena glavna okna in njihova orodja, ki so v nadaljevanju opisana:

1. **WORKSPACE** (urejevalnik kode) – mesto, kamor uporabnik piše svojo programsko kodo.
2. **SOLUTION EXPLORER** (raziskovalec rešitev) - prikaže datoteke, s katerimi uporabnik trenutno dela.
3. **PROPERTIES** (lastnosti) – poda dodatne informacijo o izbranih delih trenutnega projekta.
4. **OUTPUT** (izhodno okno) - prikazuje izhode, opozorila prevajalnika, sporočila o napakah in informacije za odpravljanje napak.



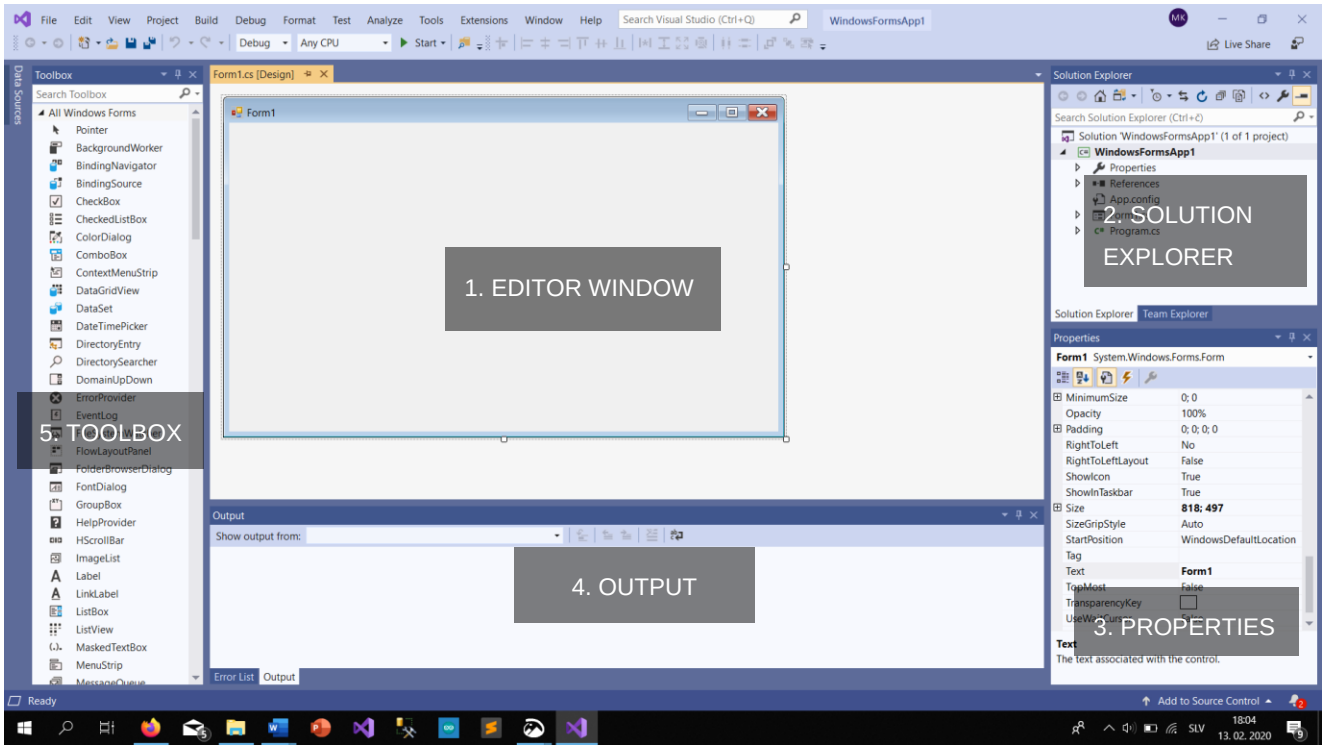
Slika 1: Začetno okno Visual Studia.

2.2 Windows Forms Application

Windows Form Application je programska platforma s knjižnico razredov grafičnega uporabniškega vmesnika (GUI), ki je združena v .Net Framework. Njegov glavni namen je zagotoviti lažji vmesnik za razvoj grafičnih aplikacij za namizne, tablične in osebne računalnike. Imenujejo ga tudi WinForms. Zasnovan je za izdelovanje namiznih aplikacij, ki ne potrebujejo dostopa do spleta. Ponuja različne kontrole, kot so besedilna polja, gumbi, vstavljanje povezav do spletnih strani, možnosti za ustvarjanje kontrol po meri itd. Tabelarični podatki, ki so vezani na XML, bazo podatkov itd., se lahko prikažejo s pomočjo nadzora DataGrid View v obliki vrstic in celic. Vsebuje glavno okno, kjer lahko te kontrolnike razporedimo, kot sami želimo. Z možnostjo dodajanja kode v enem od kontrol se ustvari dogodek. Aplikacija na te dogodke reagira s pomočjo kode in obdeluje dogodke, ko se pojavijo.

Preden začne uporabnik razvijati določeno aplikacijo, se mu pojavi začetno okno, ki ga lahko vidite na sliki 2. Na njej so označena glavna okna in orodja, ki so opisana v nadaljevanju:

1. **EDITOR WINDOW** (urejevalno okno) – mesto, kamor postavljamo razne kontrolnike, ki jih vidimo na levi strani programa (Toolbox). Na začetku je v urejevalnem oknu postavljeno samo eno prazno okno, na katerega lahko dvakrat kliknemo in se nam pojavi okno s kodo.
2. **SOLUTION EXPLORER** (raziskovalec rešitev) – prikazane so lastnosti določenega kontrolnika, ko nanj kliknemo.
3. **PROPERTIES** (lastnosti) – uporablja se za spreminjanje različnih lastnosti izbranega elementa. Prav tako lahko spremenimo lastnosti komponent oz. kontrolnikov.
4. **TOOLBOX** – prikazuje seznam vseh kontrol, ki jih Windows Forms Application vsebuje.
5. **OUTPUT** (izhodno okno) - prikazuje izhode, opozorila prevajalnika, sporočila o napakah in informacije za odpravljanje napak.



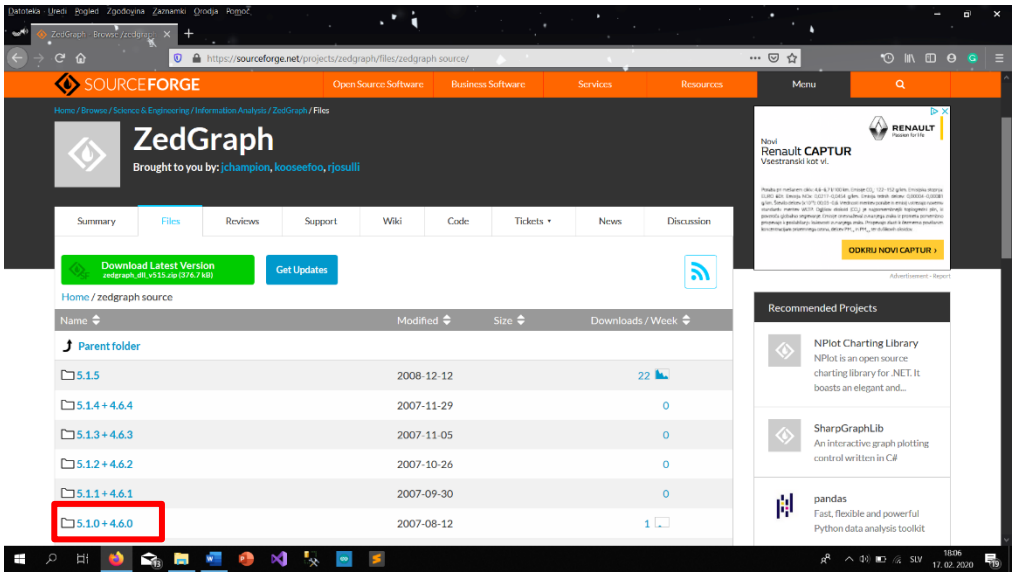
Slika 2: Začetno okno Windows Forms Application.

2.3 ZedGraph

ZedGraph je knjižnica razredov, napisanih v programskem jeziku C#, ki je dostopna spletnemu nadzoru. Knjižnica je namenjena ustvarjanju različnih 2D grafov, navpičnih ali vodoravnih grafikonov, črtnih grafikonov, tortnih grafov itd. Njegova slabost je, da še ne ustvarja 3D površine ali grafikona. Ustvarjeni grafikoni so lahko sestavljeni z naslovi osi, legendo in puščicami. Ti razredi zagotavljajo visoko stopnjo prilagodljivosti, saj je skoraj vsak graf mogoče uporabniško spremeniti. ZedGraph prav tako vključuje tudi vmesnik UserControl, ki omogoča urejanje »povleci in spusti« v urejevalniku VisualStudio. Do njega lahko dostopamo na spletni strani https://sourceforge.net/projects/zedgraph/files/, kjer namestimo tisto različico knjižnice, ki se nam zdi najbolj uporabna.

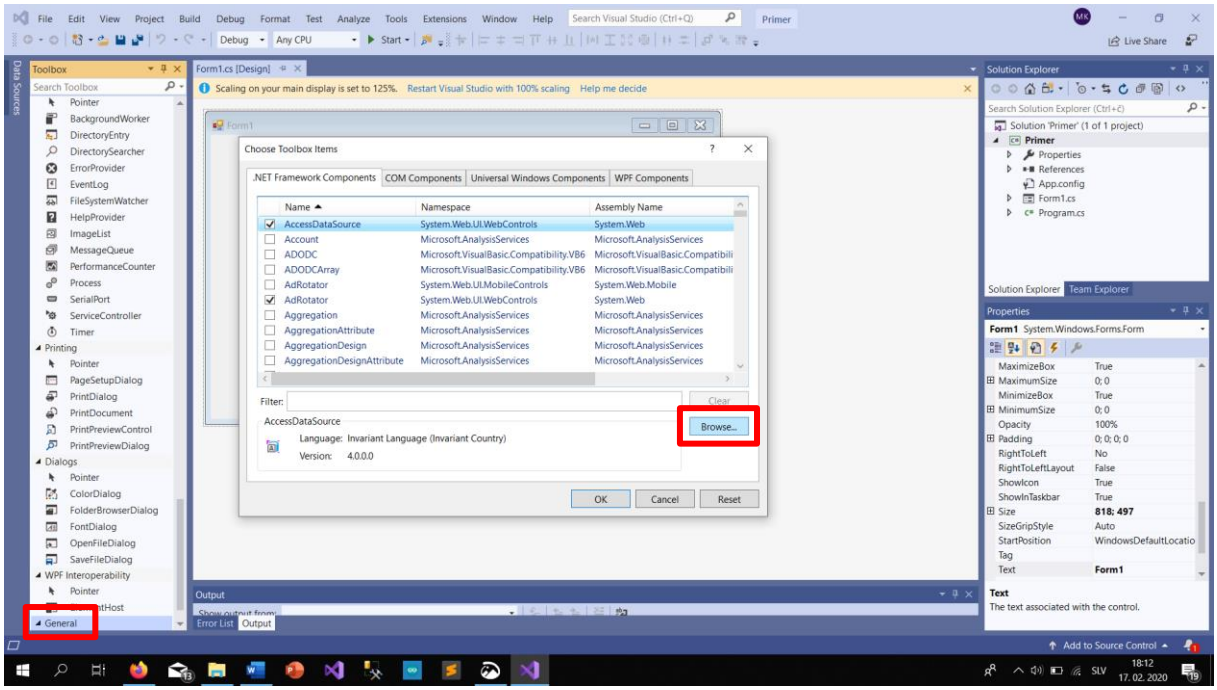
2.3.1 Dodajanje ZedGrapha v Windows Forms Application

ZedGraph bližnjico sem naložila iz spletne strani https://sourceforge.net/projects/zedgraph/files/zedgraph%20source/, ki jo lahko vidite na sliki 3. Jaz sem namestila različico v510.



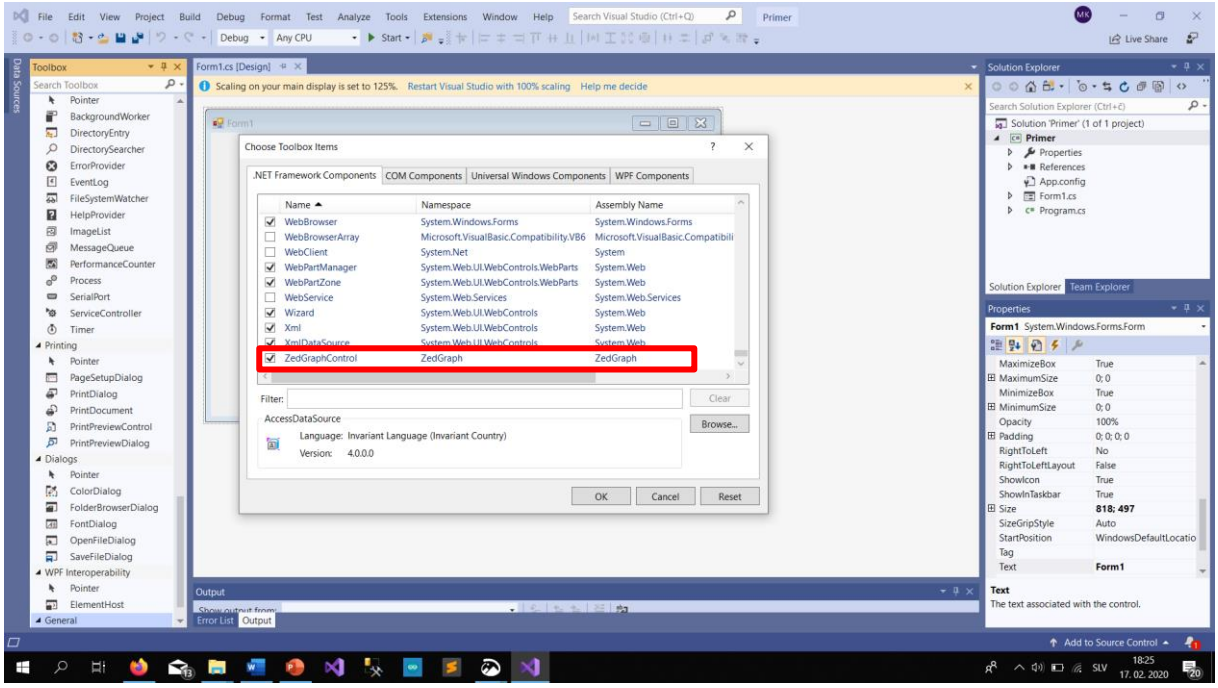
Slika 3: Spletna stran ZedGrapha

Po nameščeni bližnjici sem odprla Visual Studio in nato Windows Forms Application. V Toolboxu sem z desnim klikom na *general*, ki se nahaja na koncu najpogostejših kontrolnikov, odprla okence *choose items*, kjer izberemo gumb *Browse*. Postopek lahko vidite na sliki 4.



Slika 4: Okence Choose items

Po kliku na gumb *Browse* poiščemo mesto, kamor smo shranili različico ZedGrapha, in izberemo datoteko z .dll krajšavo. Postopek lahko vidite v oklepaju (zedgraph_source_v510_460\ZedGraph 5 (.net 2)\source\bin\Debug). Po izboru knjižnice morate obkljukati ZedGraphControl in v Toolboxu boste videli kontrolnike za koordinatni sistem.



Slika 5: Dodana knjižnica ZedGraph.

3 Matematične funkcije

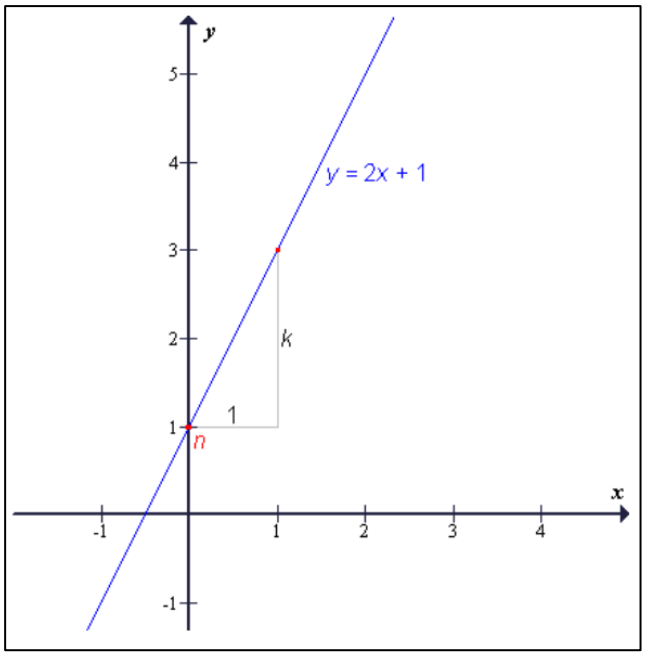
Funkcija f iz množice A v množico B je predpis, ki vsakemu elementu χ množice A priredi natanko določen element γ množice B. Elemente v množici A imenujemo originali, v množici B pa funkcije vrednosti originalov imenujemo slike. Funkcijsko zvezo lahko krajše zapišemo $b = f(a)$. Množico vseh originalov imenujemo definicijsko območje funkcije f in jo označimo z D_f, množico vseh slik pa imenujemo zaloga vrednosti funkcije f in jo označimo z Z_f.

Poznamo več vrst funkcij, in sicer funkcijo realne spremenljivke, ki ima za podatke realna števila, realno funkcijo, ki ima za rezultate realna števila, ter realno funkcijo realne spremenljivke, ki ima za podatke in rezultate realna števila.

3.1 Linearna funkcija

Funkcijo f: $\Re \Rightarrow \Re$ je definirana s predpisom $f(x) = kx + n$ (eksplicitna oblika), kjer sta k in n realni števili imenujemo linearna funkcija.

Število k imenujemo diferenčni količnik med spremembo vrednosti funkcije in spremembo vrednosti neodvisne spremenljivke. Prav tako določa naraščanje ali padanje linearne funkcije. V enačbi premice k imenujemo smerni koeficient premice. Če premica poteka skozi dve dani točki $T1(x_1,y_1)$ in $T2(x_2,y_2)$ lahko smerni koeficient izračunamo po formuli $k = \frac{x_2 - x_1}{y_2 - y_1}$. Če je $k > 0$, funkcija (premica) narašča, če je $k = 0$, je funkcija konstantna (premica je vzporedna osi x) in če je $k < 0$, funkcija (premica) pada.



Slika 6: Primer linearne funkcije.

(http://www2.arnes.si/~mpavle1/mp/lin_f.html)

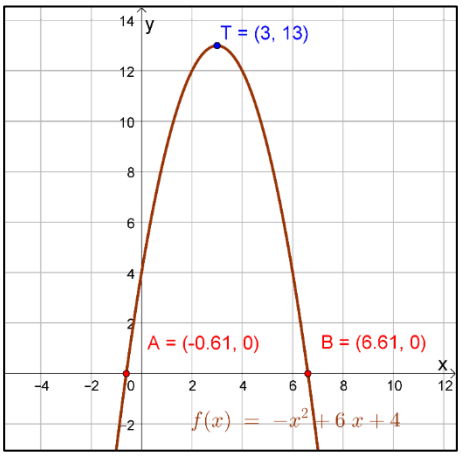
Število n imenujemo začetna vrednost funkcije in določa strmino grafa. Graf linearne funkcije je premica, vendar vsak zapis premice ne predstavlja grafa linearne funkcije. Premico lahko zapišemo v treh oblikah, in sicer v eksplicitni obliki ($y = kx + n$), implicitni obliki ($ax + by + c = 0$) ter odsekovni obliki ($\frac{x}{m} + \frac{y}{n} = 1$). Grafi linearnih funkcij z enakim smernim koeficientom so vzporedni. Vzporednice imenujemo snop premic. Grafi linearnih funkcij z enako začetno vrednostjo oblikujejo šop premic.

3.2 Kvadratna funkcija

Kvadratna funkcija je funkcija, ki jo lahko zapišemo z enačbo oblike $f(x) = ax^2 + bx + c$, kjer so koeficienti a, b in c poljubna realna števila in je vodilni koeficient a različen od 0. Enačbo oblike $f(x) = ax^2 + bx + c$ imenujemo splošna oblika enačbe kvadratne funkcije.

Vsako kvadratno funkcijo lahko zapišemo tudi v temenski obliki $f(x) = a(x - p)^2 + q$, kjer sta števili $p\ in\ q$ koordinati temena kvadratne funkcije. Teme je točka $T(p, q)$, v kateri kvadratna funkcija doseže ekstremno vrednost. Temensko obliko lahko dobimo z dopolnjevanjem splošne oblike do popolnega kvadrata ali pa z izračunom po naslednjih formulah: $p = \frac{-b}{2a}$ in $q = \frac{-D}{4a}$, ki ju dobimo z uporabo odvoda (funkcija doseže največjo oz. najmanjšo vrednost, ko je njen odvod enak 0). Število, ki nastopa v števcu, imenujemo diskriminanta ($D = b^2 - 4ac$), ki nam pove, koliko realnih ničel ima kvadratna funkcija. Če je $D > 0$, sta obe ničli kvadratne funkcije realni, če je $D = 0$, sta števili x_1 in x_1 enaki, torej ima kvadratna funkcija samo eno, dvojno realno ničlo; če je $D < 0, pa$ sta obe ničli kvadratne funkcije kompleksni, kar pomeni, da graf funkcije ne seka abscisne osi v koordinatnem sistemu.

Kvadratno funkcijo lahko zapišemo tudi v ničelni obliki $f(x) = a(x - x_1)(x - x_2)$, kjer sta števili x_1 in x_2 ničli kvadratne funkcije in v splošnem kompleksni števili. Ničelno obliko lahko dobimo iz splošne z razcepom, lahko pa x_1 in x_2 izračunamo po formuli $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$. Graf linearne funkcije je parabola. Narišemo ga postopoma, in sicer najprej izračunamo presečišče parabole z ordinatno osjo in dobimo točko $Py(0, y)$, nato po formuli za p in q izračunamo teme $T(p, q)$ in na koncu izračunamo ničli kvadratne funkcije po zgoraj omenjeni formuli.



Slika 7: Primer kvadratne funkcije.

(https://edutorij.e-skole.hr/share/proxy/alfresco-noauth/edutorij/api/proxy-guest/68015b42-a876-4b53-b0e4-8723181ea03d/html/4095_Nultocke_i_ekstremi_kvadratne_funkcije.html)

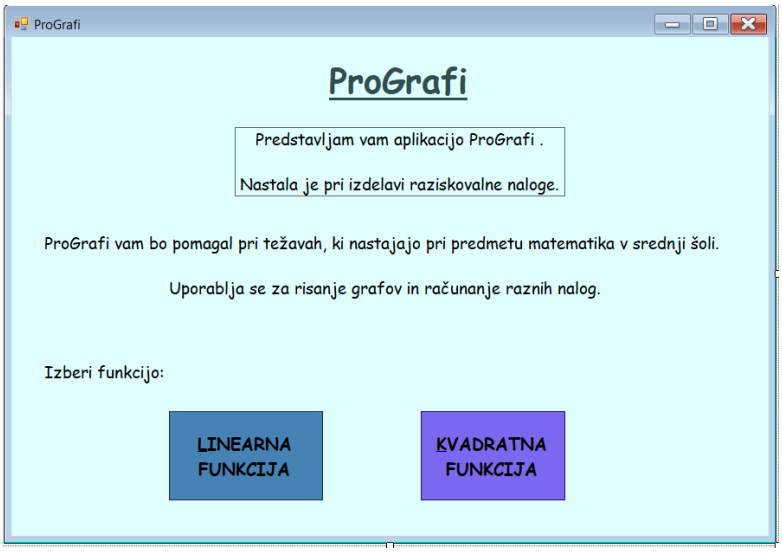
4 Aplikacija ProGrafi

4.1 Namen aplikacije

Aplikacija ProGrafi je zgrajena tako, da namesto uporabnika izračuna različne tipe nalog elementarnih funkcij, ki jih obravnavamo pri pouku matematike. Poleg tega, glede na dane podatke, zraven nariše še graf. Ker je matematičnih funkcij veliko, aplikacija trenutno računa in riše le linearno ter kvadratno funkcijo. Namen programa, poleg reševanja nalog in risanje grafov, je, da lahko dijaki med učenjem in reševanjem nalog preverjajo, če rešujejo in rišejo pravilno.

4.2 Delovanje aplikacije

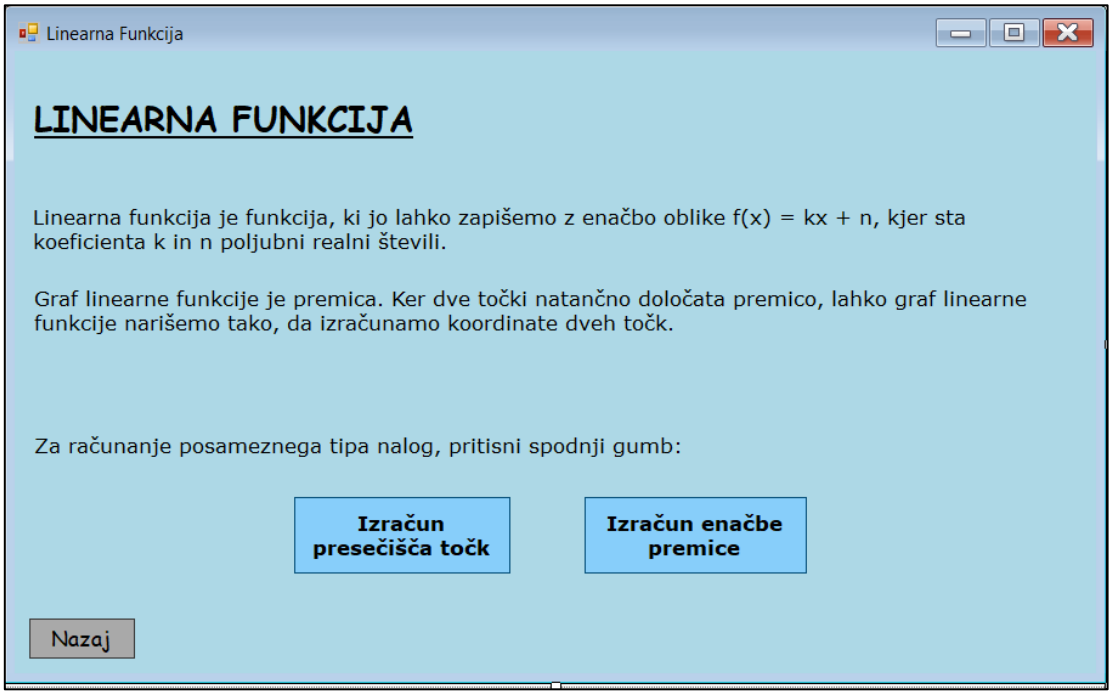
Pri zagonu aplikacije se nam prikaže njena začetna stran (slika 8). Na začetni strani je opisano, kaj aplikacija nudi in počne. Na začetni strani izberemo, katero funkcijo želimo računati. Po kliku se prikaže novo okno glede na to, kaj smo izbrali. Pri izboru linearne funkcije se nam v novem oknu pojavi kratka razlaga o njej. Pod razlago se nahajata dva gumba z dvema različnima tipoma nalog. Prvi tip naloge je, da vpišeš premico in ti program izračuna presečišče N z ordinatno osjo ter presečišče M z abscisno osjo. Pri drugem tipu naloge uporabnik vpiše koordinate dveh točk in program izračuna enačbo premice, ki poteka skozi njiju. Pri izboru kvadratne funkcije, se nam tudi v novem oknu pojavi krajša razlaga o tem, kaj je kvadratna funkcija, poleg tega pa sta zraven še dva gumba. Pri izboru prvega gumba nam program glede na vneseno enačbo izračuna pomembne točke parabole, ki so podrobneje opisane v poglavju 3.2 (kvadratna funkcija). Te pomembne točke so presečišče ordinatne osi, teme ter ničli kvadratne funkcije. Pri izboru drugega gumba nam program glede na vneseno enačbo nariše graf kvadratne funkcije, torej parabolo. V nadaljevanju vam bom podrobneje razložila delovanje in kodo posamezne funkcije:



Slika 8: Začetna stran aplikacije.

4.2.1 Program in koda linearne funkcije

Začetna stran (slika 9), ki se nam prikaže po izboru gumba za linearno funkcijo, ima kratko razlago o tem, kaj je linearna funkcija. Pod razlago lahko uporabnik vidi dva gumba. Klik na enega od gumbov sproži tako imenovan dogodek in odpre novo okno, trenutnega pa zapre. Kodo lahko vidite na sliki 10.

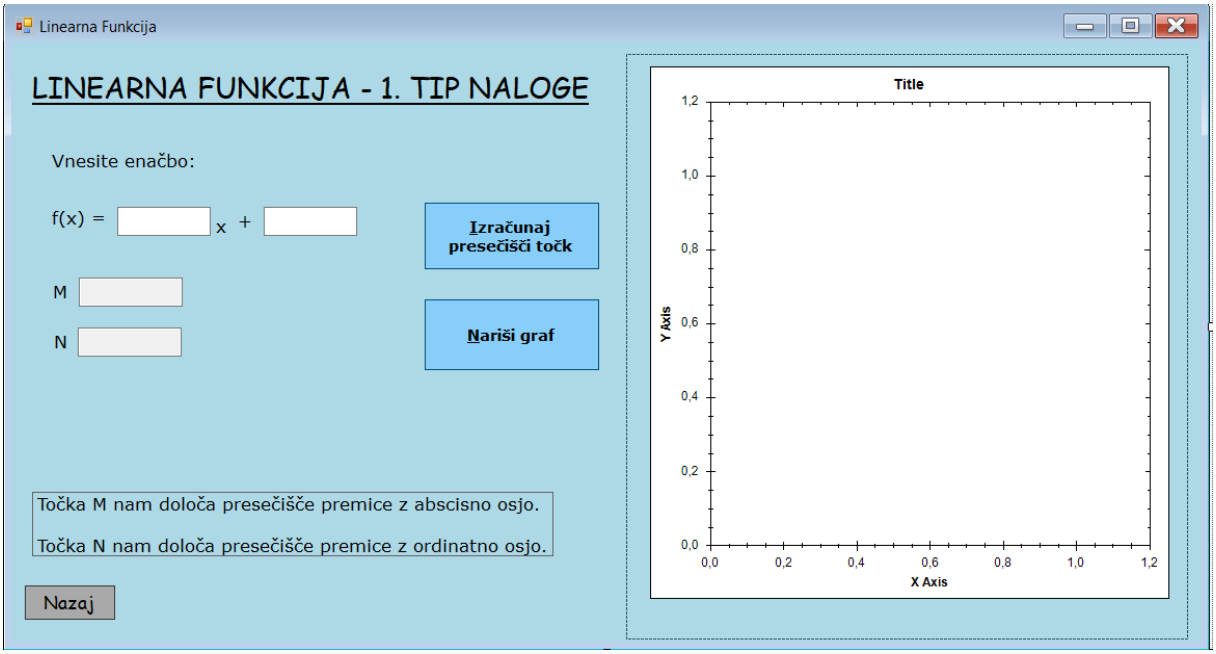


Slika 9: Začetna stran linearne funkcije.

```
1 reference
private void Button1_Click(object sender, EventArgs e) // gumb izračun presečišč funkcije
{
    LinearnaFunkcija2 f2 = new LinearnaFunkcija2();
    this.Close();
    f2.Show();
}

1 reference
private void Button2_Click(object sender, EventArgs e) // gumb izračun enačbe premice
{
    LinearnaFunkcija3 f2 = new LinearnaFunkcija3();
    this.Close();
    f2.Show();
}

1 reference
private void ButtonNazaj_Click(object sender, EventArgs e) // gumb za na začetno stran
{
    ZacetnaStran f2 = new ZacetnaStran();
    this.Close();
    f2.Show();
}
```

Slika 10: Koda za premik med stranmi.

Pri kliku na prvi gumb (*Izračun presečišča točk)* se pojavi novo okno s prvim tipom naloge, ki ga lahko vidite na sliki 11. Ta glede na dano enačbo izračuna presečišče N (z ordinatno osjo) in M (z abscisno osjo). S pritiskom na gumb (*Nariši graf)* program nariše graf glede na dane podatke. Glede na dane podatke, ki jih program dobi od enačbe (koeficient in ničla funkcije), izračuna vsako presečišče posebej. Za točko $N(0, y)$ potrebuje samo še y, ki ga dobi pri ničli funkciji, saj graf tam seka ordinatno os. Za točko $M(X, O)$ potrebuje program samo še x, ki ga izračuna po formuli $x = -\frac{n}{k}$. Graf nariše tako, da program za vsak x izračuna vrednost y po formuli $y = kx + n$ s tako imenovanim tabeliranjem. Postopek računanja presečišč lahko vidite na sliki 12, postopek računanja koeficienta in začetne vrednosti ter risanja grafa pa na sliki 13.



Slika 11: Prvi tip naloge.

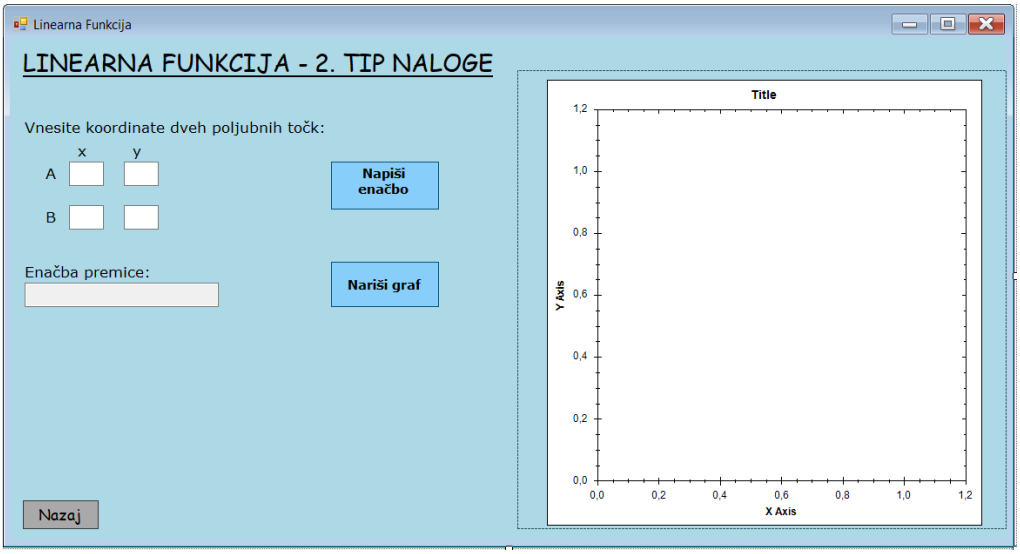
```
1 reference
private void Button1_Click(object sender, EventArgs e) // gumb izračunaj presečišče točk
{
    double k, n, x;
    double.TryParse(TextBox_K.Text, out k);
    double.TryParse(TextBox_n.Text, out n);
    double y = 0;
    x = -(n / k);
    TextBox_M.Text = "(" + x.ToString() + ", " + y.ToString() + ")";
    TextBox_N2.Text = "(" + y.ToString() + ", " + n.ToString() + ")";
}
```

Slika 12: Koda za računanje presečišč.

```
1 reference
private double[] YPODATKI()
{
    // izračuna vrednosti Y in jih shranjuje v polje
    double[] Ypodatki = new double[2000];
    double x, y;
    double x1, x2, y1, y2, k, n;
    double.TryParse(TextBox_K.Text, out k);
    double.TryParse(TextBox_n.Text, out n);
    x1 = -(n / k);
    y1 = 0;
    x2 = 0;
    y2 = n;
    int i = 0;
    for (x = -1000; x < 1000; x++)
    {
        y = k * x + n;
        Ypodatki[i] = y;
        i++;
    }
    return Ypodatki;

}

1 reference
private void Button2_Click_1(object sender, EventArgs e)
{
    // nariše premico z podatki x in y
    PointPairList krnekiPairList = new PointPairList();
    GraphPane myPane = zedGraphControl1.GraphPane;

    double[] data1 = YPODATKI();
    int j = 0;
    for (int i = -1000; i < 1000; i++)
    {
        krnekiPairList.Add(i, data1[j]);
        j++;
    }
    LineItem krnekiACurve = myPane.AddCurve("graf", krnekiPairList, Color.Blue, SymbolType.None);

}
```

Slika 13: Koda za risanje grafa.

Pri kliku na drugi gumb (*Izračun enačbe premice)* se pojavi novo okno z drugim tipom naloge, ki ga lahko vidite na sliki 14. Ta glede na koordinate dveh točk, ki jih vpiše uporabnik, izračuna enačbo premice, ki poteka skozi dve točki, ki ju je vnesel uporabnik. Ker sta smerni koeficient (k) in začetna vrednost (n) neznana, ju program izračuna s formulami $k = \frac{x_2 - x_1}{y_2 - y_1}$. in $n = y_1 - kx_1$. Aplikacija graf nariše tako, da program za vsak x izračuna vrednost y po formuli $y = kx + n$ s tako imenovanim tabeliranjem. Postopek računanja enačbe lahko vidite na sliki 15, postopek računanja koeficienta in začetne vrednosti ter risanja grafa pa na sliki 16.



Slika 14: Drugi tip naloge.

```
1 reference
private void Button2_Click(object sender, EventArgs e) // gumb Napiši enačbo
{
    // izračuna vrednosti k in n
    double x1, x2, y1, y2, k, n;
    double.TryParse(TextBox_X1.Text, out x1);
    double.TryParse(TextBox_X2.Text, out x2);
    double.TryParse(TextBox_Y1.Text, out y1);
    double.TryParse(TextBox_Y2.Text, out y2);
    k = (y2 - y1) / (x2 - x1);
    n = y1 - k * x1;

    //izpiše premico
    TextBox_EnacbaPremice.Text = "f(x) = " + Math.Round(k, 3, MidpointRounding.ToEven) + " x + " + Math.Round(n, 3, MidpointRounding.ToEven);


}
```

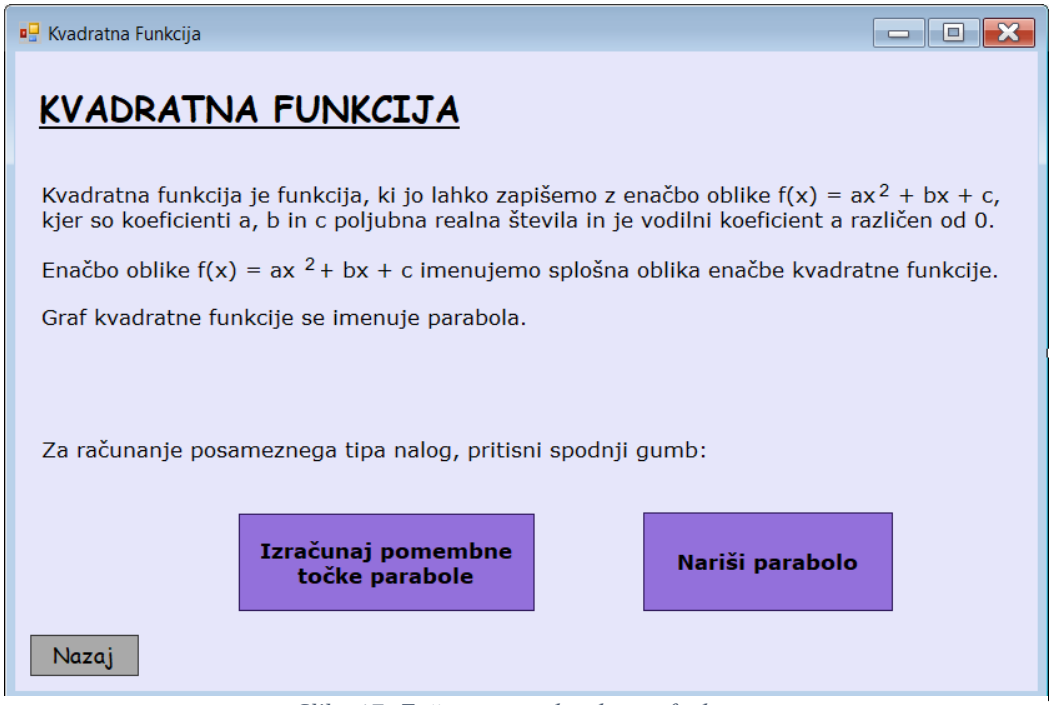
Slika 15: Koda za računanje enačbe.

```
1 reference
private double[] YPODATKI()
{
    // izračuna vrednosti Y in jih shranjuje v polje
    double[] ypodatki = new double[2000];
    double x, y;
    double x1, x2, y1, y2, k, n;
    double.TryParse(TextBox_X1.Text, out x1);
    double.TryParse(TextBox_X2.Text, out x2);
    double.TryParse(TextBox_Y1.Text, out y1);
    double.TryParse(TextBox_Y2.Text, out y2);
    k = (y2 - y1) / (x2 - x1);
    n = y1 - (k * x1);
    int i = 0;

    for (x = -1000; x < 1000; x++)
    {
        y = k * x + n;
        ypodatki[i] = y;
        i++;
    }
    return ypodatki;
}

1 reference
private void Button1_Click(object sender, EventArgs e) // gumb Nariši graf
{

    // nariše premico
    PointPairList krnekiPairList = new PointPairList();
    GraphPane myPane = zedGraphControl1.GraphPane;

    double[] data1 = YPODATKI();
    int j = 0;
    for (int i = -1000; i < 1000; i++)
    {
        krnekiPairList.Add(i, data1[j]);
        j++;
    }
    LineItem krnekiACurve = myPane.AddCurve("graf1", krnekiPairList, Color.Red, SymbolType.None); // nariše graf

}
```

Slika 16: Enačba za risanje grafov.

4.2.2 Program in koda kvadratne funkcije

Začetna stran (slika 17), ki se nam prikaže po izboru gumba za kvadratno funkcijo, ima kratko razlago o tem, kaj je kvadratna funkcija. Pod razlago lahko uporabnik vidi dva gumba. Klik na enega od gumbov sproži tako imenovan dogodek in odpre novo okno, trenutnega pa zapre. Kodo lahko vidite na sliki 18.

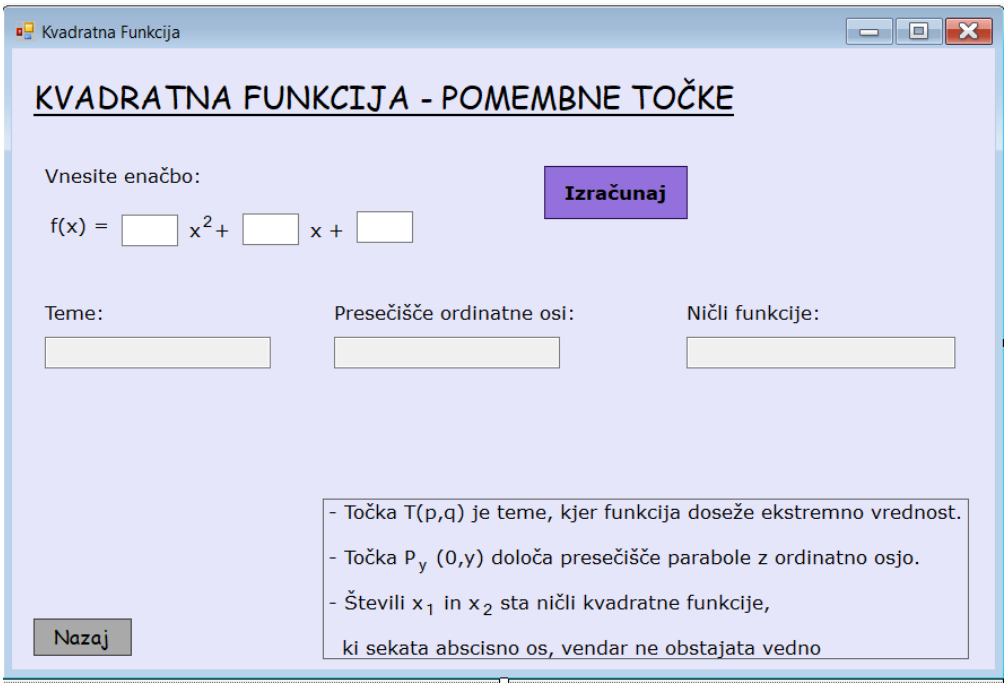


Slika 17: Začetna stran kvadratne funkcije.

```
1 reference
private void ButtonNazaj_Click(object sender, EventArgs e) // gumb Nazaj
{
    ZacetnaStran f1 = new ZacetnaStran();
    this.Close();
    f1.Show();
}
1 reference
private void Button1_Click(object sender, EventArgs e) // gumb Nariši parabolo
{
    KvadratnaFunkcija2 f1 = new KvadratnaFunkcija2();
    this.Close();
    f1.Show();
}
1 reference
private void Button2_Click(object sender, EventArgs e) // gumb Izračunaj pomembne točke parabole
{
    KvadratnaFunkcija3 f1 = new KvadratnaFunkcija3();
    this.Close();
    f1.Show();
}
```

Slika 18: Koda za premik med stranmi.

Pri kliku na prvi gumb (*Izračunaj pomembne točke parabole)* se pojavi novo okno z računanjem pomembnih presečišč parabole, ki ga lahko vidite na sliki 19. Program glede na dano enačbo, izračuna presečišče z ordinatno osjo $Py(0, y)$, kjer je neznanka y sicer število c, ki je podano v enačbi. Nato izračuna točko T(p, q) oziroma teme s formulama $p = \frac{-b}{2a}$ in $q = \frac{-D}{4a}$. Za izračun q je neznanka diskriminanta (D), zato jo posebej izračuna po formuli $D = b^2 - 4ac$. Zadnjo pomembno presečišče, to sta ničli kvadratne funkcije, izračuna po formuli $x_{1,2} = \frac{-b \pm \sqrt{D}}{2a}$. Ker pa ničli ne obstajata vedno, vsebuje koda še "if stavke", s katerimi preverja, če je D večji, manjši ali enak 0. Če je $D > 0$, sta obe ničli kvadratne funkcije realni, če je $D = 0$, sta števili x_1 in x_1 enaki, torej ima kvadratna funkcija samo eno, dvojno realno ničlo, če je $D < 0, pa$ sta obe ničli kvadratne funkcije kompleksni, kar pomeni, da graf funkcije ne seka abscisne osi v realnem koordinatnem sistemu. Postopek računanja pomembnih presečišč parabole lahko vidite na sliki 20.

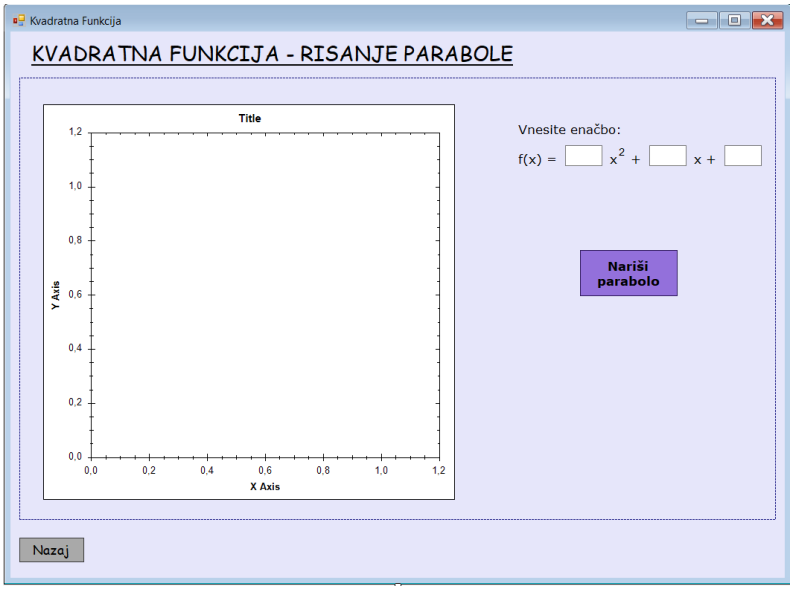


Slika 19: Začetna stran prve naloge

```
1 reference
private void Button1_Click(object sender, EventArgs e) // gumb Izračunaj
{
    double p, q, px, py, d, x1, x2;
    double a, b, c;
    double.TryParse(textBox_A.Text, out a);
    double.TryParse(textBox_B.Text, out b);
    double.TryParse(textBox_C.Text, out c);
    //Izračuna D
    d = (Math.Pow(b, 2)) - (4 * a * c);
    if (d >0)
    {
        x1 =( ((-b) + Math.Sqrt(d))) /( 2 * a);
        x2 = (((-b) - Math.Sqrt(d))) / (2 * a);
        textBox_Nicli.Text = "x1 = " + x1.ToString() + ", x2 = " + x2.ToString();

    }
    if (d==0)
    {
        x1 = ((-b) - Math.Sqrt(d)) / 2 * a;
        textBox_Nicli.Text = "ničla je samo ena - " + x1.ToString();
    }
    if (d<0)
    {
        textBox_Nicli.Text = "ničli ne obstajata";
    }
    p = -(b) / (2 * a);
    q = (-d) / (4 * a);
    py = c;
    px = 0;
    //Izpiše vrednosti
    textBox_P.Text = "T("+ p.ToString() + ", " + q.ToString()+ ")";
    textBox_Px.Text ="Py(" + px.ToString() + ", " + py.ToString() + ")";
}
```

Slika 20: Koda za računanje pomembnih presečišč.

Pri kliku na drugi gumb (*Nariši parabolo)* se pojavi novo okno za risanje grafa kvadratne funkcije oziroma parabole, ki ga lahko vidite na sliki 21. Program glede na dano izračuna za vsak x vrednost y po formuli $y = ax^2 + bx + c$ s tako imenovanim tabeliranjem. Kodo za nastavljanje grafa in tabeliranje lahko vidite na sliki 22.



Slika 21: Začetna stran druge naloge.

```
private double[] YPODATKI()
{
    // izračuna vrednosti Y in jih shranjuje v polje
    double[] ypodatki = new double[20000];
    double x, y;
    double a, b, c;
    double.TryParse(textBox_A.Text, out a);
    double.TryParse(textBox_B.Text, out b);
    double.TryParse(textBox_C.Text, out c);
    int i = 0;

    for (x = -1000; x < 1000; x = x + 0.1)
    {
        y = a * Math.Pow(x, 2) + b * x + c;
        ypodatki[i] = y;
        i++;
    }
    return ypodatki;


}

1 reference
private void Button1_Click_1(object sender, EventArgs e) // gumb Nariši parabolo
{
    // nariše premico
    PointPairList krnekiPairList = new PointPairList();
    GraphPane myPane = zedGraphControl1.GraphPane;

    double[] data1 = YPODATKI();
    int j = 0;
    for (double i = -1000; i < 1000; i = i + 0.1)
    {
        krnekiPairList.Add(i, data1[j]);
        j++;
    }
    LineItem krnekiACurve = myPane.AddCurve("graf1", krnekiPairList, Color.Red, SymbolType.None); // nariše graf

}
```

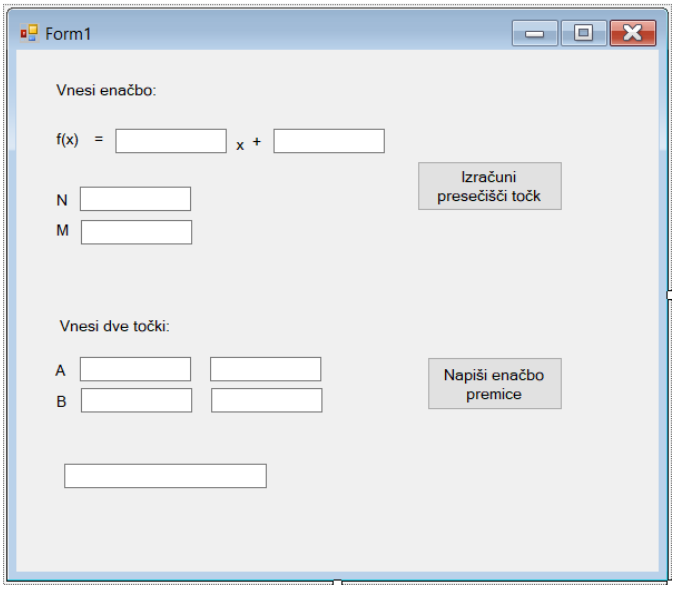
Slika 22: Koda za nastavljanje grafa.

4.3 Težave pri ustvarjanju aplikacije

Prvi cilj je bil, da bi ustvarila aplikacijo, ki bi risala in računala več vrst elementarni funkcij. Predvsem sem želela risati bolj zahtevnejše funkcije, kot so npr. eksponentna, logaritemska, polinomska itd. Po posvetu s profesorico matematiko, sem prišla do spoznanja, da me lahko pri tem ovira pomanjkanje znanja. Tako sem morala zmanjšati izbor na samo dve funkciji, in sicer linearno in kvadratno. Ker program računa in riše samo dve vrsti funkcij, sem zaradi tega pri vsaki funkciji naredila dve nalogi. Ti nalogi se najpogosteje uporabljata pri pouku matematike.

Naslednja težava je bila izbrati tip programa oziroma platformo, ki bi mi ustrezal glede na znanje, ki sem ga pridobila v treh letih srednje strokovne šole za računalništvo. Po posvetu z mentorjem sem izbrala tip programa Windows Forms Application, saj smo ga že obravnavali pri strokovnem predmetu. Želela sem nadgraditi osnovno znanje programa, ki sem ga pridobila pri pouku.

Sprva sem se lotila vstavljanja osnovnih gradnikov v eno okence in napisala kodo za vse tipe nalog pri linearni funkciji, ki sem jih uporabila. Prvotni izgled aplikacije lahko vidite na sliki 23. Pri tem ni bilo večjih težav, saj smo osnovne gradnike, ki so v aplikaciji, opisali in uporabili pri pouku. Potem sem ugotovila, da programi tipa Windows Forms Application nimajo gradnika za koordinatni sistem, ki sem ga potrebovala za risanje grafov. Tako sem s pomočjo mentorja spoznala, da bom morala uporabiti knjižnico, ki vsebuje potrebni koordinatni sistem. Uporabila sem spletno stran ZedGraph, kjer je tudi nekaj primerov, kako se knjižnica uporablja. Naučiti sem se morala, kako dodaš bližnjico v Toolbox, da lahko koordinatni sistem vstaviš v okence.



Slika 23: Prvotni izgled aplikacije.

Po dodanem koordinatnem sistemu iz najnovejše različice (v515) sem ugotovila, da knjižnica ni priročna za risanje grafov, ki gredo v negativno smer po obeh oseh. Tako sem morala izbrisati najnovejšo verzijo ZedGrapha in naložiti starejšo različico (v510). Po uspešni namestitvi bližnjice v Windows Forms Application sem s pomočjo znanja matematike narisala grafe linearne in kvadratne funkcije s tako imenovanim tabeliranjem.

Ko je bila aplikacija dokončno oblikovana in sprogramirana, je bila naslednja težava določanje imena, saj mora imeti vsaka aplikacija ime. Želela sem povezati besedi programiranje in grafi ali funkcije. Tako je nastalo ime ProGrafi.

4.4 Uporabnost aplikacije

Kot sem že omenila, je aplikacija napisana kot Windows Forms Application v Visual Studio. Program ima možnost, da lahko vsako dodano kodo in vsak dodan gradnik aplikacijo testiraš in vidiš kako jo bodo videli drugi uporabniki ter morebiti kaj popraviš. Nato sem se vprašala, ali lahko aplikacijo odpre tudi nek drugi uporabnik, ki na računalniku nima prednameščenega Visual Studia in odgovor je bil seveda da. Izvorno kodo je mogoče odpreti v katerem koli urejevalniku besedil, program s končnico .exe, ki ga ustvari Visual Studio ob prevajanju, pa je mogoče zagnati na vsakem računalniku z nameščenim ogrodjem .NET Framework, torej na vsakem računalniku z operacijskim sistemom Windows. Visual Studio potrebujemo le za ustvarjanje novih različic .programa (.exe datotek) ne pa tudi za njihov zagon.

Exe datoteka je datoteka, ki se uporablja v operacijskih sistemih Windows in vsebuje izvršljivo kodo. To so programi, ki jih največkrat odpremo tako, da na njih kliknemo dvakrat. Na takšen način odpiramo tudi aplikacijo ProGrafi.

5 Možnosti nadgradnje v prihodnosti

Kot sem že omenila, je bil prvotni namen ustvariti aplikacijo z več različnimi matematičnimi funkcijami, ampak sem prišla do ugotovitve, da moje znanje pri matematiki še ni na stopnji, ko bi to lahko naredila. V prihodnosti bi rada program nadgradila. Profesorica za matematiko mi je svetovala, da nadgradnjo izvedem prihodnje leto, saj bomo takrat obravnavali bolj kompleksne funkcije in odvode. Nadgradila bi tudi kodo, in sicer z bolj zahtevnimi metodami, in bolj optimalno oblikovala izgled aplikacije.

6 Ugotovitve

Na začetku sem si postavila 4 hipoteze, v nadaljevanju so opisane ugotovitve v zvezi z njimi:

H1: *Z izdelavo aplikacije bom bolj razumela risanje grafov v računalniškemu okolju*. Hipotezo sem potrdila, saj sedaj razumem, kako deluje risanje grafov v programu Windows Forms Application.

H2: *Windows Forms Application se bo pokazal kot dobra izbira platforme za pisanje programa*. Hipotezo sem potrdila, saj sem uporabila platformo, ki sem jo uporabljala že prej.

H3: *Programski del mi bo predstavljal večjo težavo kot matematični*. Hipotezo sem potrdila, saj je bil največji problem ugotoviti, kako Windows Forms Application deluje s pomočjo ZedGrapha.

H4: *Aplikacija bo prikazala vse elementarne funkcije, ki se obravnavajo v srednji šoli*. Hipoteze ne morem potrditi, saj nimam dovolj znanja pri matematikI, da bi lahko aplikacija prikazala vse elementarne funkcije.

7 Zaključek

Za to raziskovalno nalogo je bilo potrebnega veliko znanja in poznavanja matematike ter programiranja. Vesela sem, da mi je uspelo narediti medpredmetno povezavo in nadgraditi svoje znanje. Spoznala sem se z uporabo matematike v programiranju, pripomogla k širšemu razumevanju linearne in kvadratne funkcije ter ugotovila, da je razvijanje aplikacije pravzaprav dolgotrajen proces, ki vsebuje marsikatere podrobnosti, za katere prej nisem vedela, da obstajajo. Čeprav je bilo za nalogo porabljenega veliko prostega časa, sem vesela, da mi je uspelo, in že komaj čakam, da bom aplikacijo lahko nadgradila.

8 Viri in literatura

Mateja Škrlec (2019), MATEMATIKA 1, Zbirka nalog za gimnazije. Ljubljana: DZS, založništvo in trgovina, d. d.

Mateja Škrlec (2019), MATEMATIKA 2, Zbirka nalog za gimnazije. Ljubljana: DZS, založništvo in trgovina, d. d.

Geeks for geeks. (2017). Pridobljeno 12. januar 2020 iz https://www.geeksforgeeks.org/introduction-to-visual-studio/

Osvoji znanje. (2016). Pridobljeno 12. januar 2020 iz

https://osvojiznanje.weebly.com/matematika1/category/funkcije#

Scott Lilly. (2015). Pridobljeno 12. januar 2020 iz

https://scottlilly.com/learn-c-by-building-a-simple-rpg-index/lesson-00-3-the-parts-of-visual studio/

Wikipedia. (2004). Pridobljeno 12. januar 2020 iz

https://en.wikipedia.org/wiki/Windows_Forms

ZedGraph. (2005). Pridobljeno 12. januar 2020 iz

http://zedgraph.sourceforge.net/samples.html

Večina slik je iz lastnega arhiva. Slike pridobljene s spleta, imajo poleg imena še povezavo na kateri so dostopne.

9 Izjava

IZJAVA*

Mentor Boštjan Resinovič v skladu z 20. členom Pravilnika o organizaciji mladinske raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje, zagotavljam, da je v raziskovalni nalogi z naslovom Program za delo s funkcijami, katere avtorica je/so Maša Kraner:

- besedilo v tiskani in elektronski obliki istovetno,
- pri raziskovanju uporabljeno gradivo navedeno v seznamu uporabljene literature,
- da je za objavo fotografij v nalogi pridobljeno avtorjevo dovoljenje in je hranjeno v šolskem arhivu,
- da sme Osrednja knjižnica Celje objaviti raziskovalno nalogo v polnem besedilu na knjižničnih portalih z navedbo, da je raziskovalna naloga nastala v okviru projekta Mladi za Celje,
- da je raziskovalno nalogo dovoljeno uporabiti za izobraževalne in raziskovalne namene s povzemanjem misli, idej, konceptov oziroma besedil iz naloge ob upoštevanju avtorstva in korektnem citiranju,
- da smo seznanjeni z razpisni pogoji projekta Mladi za Celje.

Celje, 2.6.2020

ŠOLSKI CENTER CELJE SREDNJA ŠOLA ZA KEMIJO, ELEKTROTEHNIKO IN RAČUNALNIŠTVO CELJE, Pot na Lavo 22 1

Žig šole

Podpis mentorja

Podpis odgovorne osebe

•

POJASNILO

V skladu z 20. členom Pravilnika raziskovalne dejavnosti »Mladi za Celje« Mestne občine Celje je potrebno podpisano izjavo mentorja (-ice) in odgovorne osebe šole vključiti v izvod za knjižnico, dovoljenje za objavo avtorja (-ice) fotografskega gradiva, katerega ni avtor (-ica) raziskovalne naloge, pa hrani šola v svojem arhivu.