Programos nustatymai:

1. jei nerodo eilučių numerių:
Tools > Editor options > Display > Line numbers

2. Failo sukūrimas **CodeBlocks**:
File >New > Project:
   - Files
   - C/C++ source > go > next > C++
   - Pasirenkame saugojimui katalogą
   - įrašome failo vardą

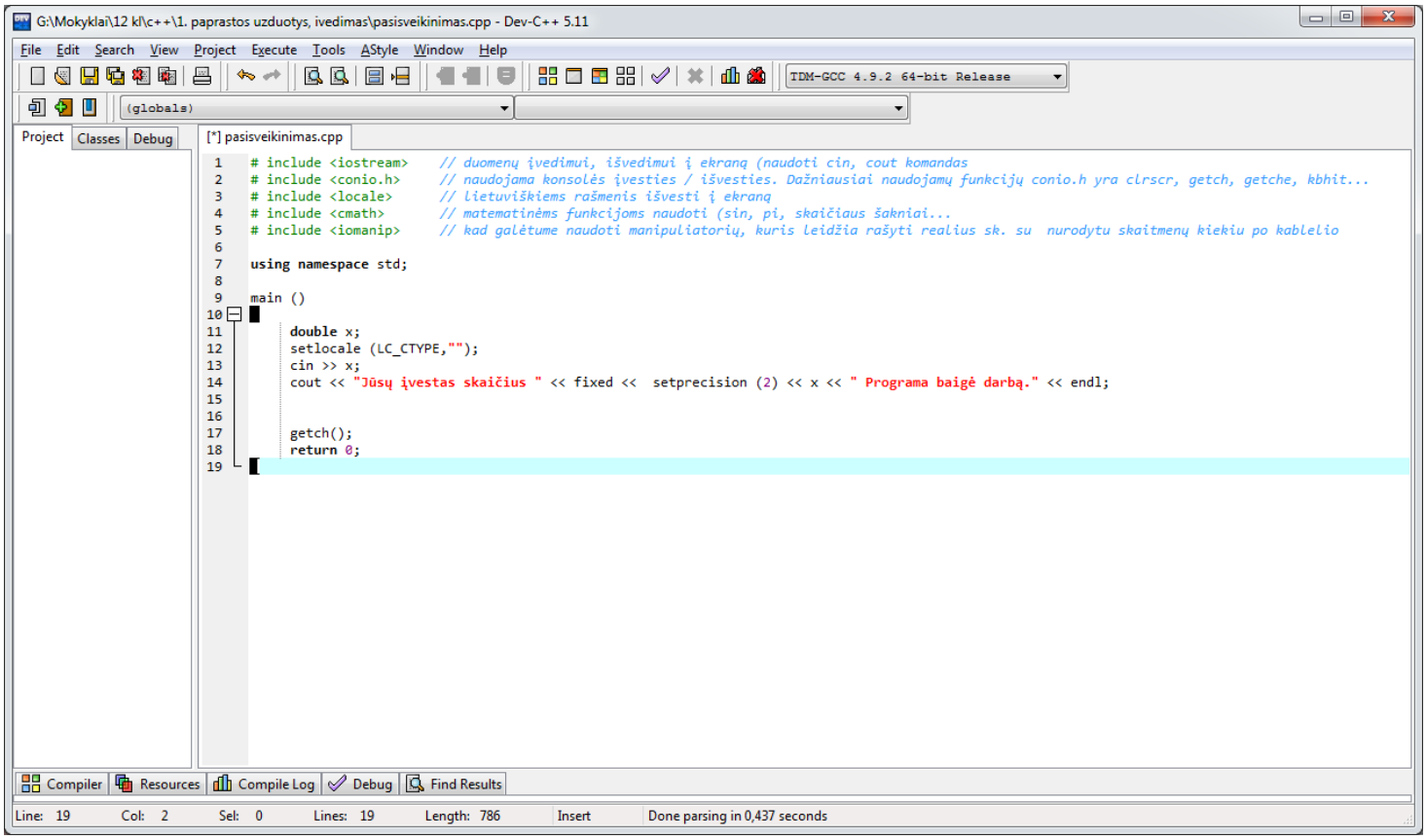
3. Failo sukūrimas **DevC++** programa:
   - File > New > Source File

4. Darbo įrašymas:
   - File > Save > katalogą paliekame jau sukurtą > įrašome tik failo vardą

PASTABA: ***kiekvieną kartą kompiliuojant ar startuojant programai pakeitimai patys išsisaugo.***

Programos struktūra

Priemonių aprašymas: ➢ bibliotekos, ➢ konstantos, ➢ kintamieji
Funkcijų prototipai
Pagrindinės funkcijos main() kintamųjų ir veiksmų sakiniai
Funkcijų tekstai



using namespace std; ==> rašomas visada, kai į programą įterpiamas antraštinis failas (pvz.: iostream)

```
int main()                ==> pagrindinė funkcijos antraštė
{

  return 0;               ==> nurodo programai baigti funkcijos darbą
}
```

Bibliotekos

visos bibliotekos prasideda **#include** ==> nurodo, kokių failų tekstai bus įtraukti į programą

CodeBlocks	DevC++
<iostream> ==> duomenų įvedimui, išvedimui į ekraną <fcntl.h> ==> lietuviškiems rašmenis išvesti į ekraną <io.h> ==> lietuviškiems rašmenis išvesti į ekraną **<iomanip>** ==> kad galėtume naudoti manipuliatorių, kuris leidžia rašyti realius sk. su nurodytu skaitmenų kiekiu po kablelio **<conio.h>** ==> naudojama konsolės įvesties / išvesties. Dažniausiai naudojamų funkcijų conio.h yra clrscr, getch, getche, kbhit t.t. Funkcijos conio.h gali būti naudojamas išvalyti ekraną, pakeisti spalvą teksto ir fono, perkelti tekstą. **<fstream>** ==> naudojama įvesti / išvesti duomenų srautus į failą **<string>** ==> darbui su simbolių eilutėmis	**<iostream>** (**iostream.h** – senesnei versijai) ==> duomenų įvedimui, išvedimui į ekraną <locale> ==> lietuviškiems rašmenis išvesti į ekraną **<iomanip>** ==> kad galėtume naudoti manipuliatorių, kuris leidžia rašyti realius sk. su nurodytu skaitmenų kiekiu po kablelio **<conio.h>** ==> naudojama konsolės įvesties / išvesties. Dažniausiai naudojamų funkcijų conio.h yra clrscr, getch, getche, kbhit t.t. Funkcijos conio.h gali būti naudojamas išvalyti ekraną, pakeisti spalvą teksto ir fono, perkelti tekstą. **<fstream>** ==> naudojama įvesti / išvesti duomenų srautus į failą **<string>** ==> darbui su simbolių eilutėmis

Duomenų išvedimas

cout<<" Rašome, tekstą, kurį norim pamatyti ekrane**" << endl;** **cout** << "* Labas. Tai mūsų pirmoji C++ programa *" << endl; **left** == > išvedant duomenis, jie lygiuojami pagal kairį kraštą **right** == > išvedant duomenis, jie lygiuojami pagal dešinį kraštą **setw (n)** == > nustato išvedamų duomenų lauko plotį n **setprecision (n)** == > nustato realiojo skaičiaus išvedamų skaitmenų skaičių n	**endl;** == > naudojame, norėdami, kad kitas pranešimas būtų rašomas naujoje eilutėje int a = 45; double b = 123.258; cout << a; cout << setw (5) << a <<" "<< fixed << setprecision (2) << b << endl; **Ekrane matysite:** 4 5 4 5 1 2 3 . 2 6 Paskutinis skaičius suapvalintas, nes nurodytų pozicijų skaičius per mažas

Duomenų įvedimas

cin>>kintamojo vardas (vardai, atskirti >> ženklu);	**cin >>** a >> b >> c;

http://www.youtube.com/playlist?list=PL4F3C96A7A402E9DB&feature=plcp

Kintamųjų tipai:

Kintamojo tipas		Mažiausia reikšmė	Didžiausia reikšmė	Ilgis baitais
bool	Loginė reikšmė: true (1) arba false (0)	false	true	1
char	Vienas simbolis `A`, `8` arba skaičius [-128; 127]	-128	128	1
int	Sveikasis skaičius	-2147483648	2147483647	4
long	Sveikasis skaičius (didesnis skaičių intervalas)	-2^{31}	$2^{31}-1$	4
double	Realusis skaičius	$-2.23*10^{-308}$	$1,79*10^{308}$	8 (15 ženklų tikslumas)
float	Realusis skaičius (tik duoda mažiau skaitmenų po kablelio)	$-3.4*10^{38}$	$3.4*10^{38}$	4 (6 ženklų tikslumas)
long double	Realusis skaičius (tik duoda daugiau skaitmenų po kablelio)	$-1,7*10^{-308}$	$1,7*10^{308}$	10 (20 ženklų tikslumas)

Kintamųjų aprašymas:

```
double pi = 3.1415;
int a = 5, b = 3;
char a = 'A';
bool statusis = false;
```

Kintamojo galiojimo pradžia – jo aprašymo vieta. Kintamasis galioja tame programos bloke, kuriame jis yra aprašytas. Bloką sudaro programavimo kalbos sakinių seka, parašyta tarp riestinių skliaustų { }

Aprašant kintamuosius, reikia prisiminti:

- Kintamieji, aprašyti prieš pagrindinę funkciją **main()**, vadinami globaliaisiais. Jie galioja visoje programoje.
- Kintamieji, aprašyti funkcijoje, vadinami lokaliaisiais. Jie galioja tik toje funkcijoje, išėjus už funkcijos
- ribų, šių kintamųjų reikšmės neišsaugomos.
- Jei kintamasis tokiu pačiu vardu aprašytas prieš pagrindinę funkciją **main()** ir funkcijoje, pirmenybė suteikiama lokaliajam, t. y. funkcijoje globalusis kintamasis negalioja.

PRISKYRIMO SAKINYS

Priskyrimo sakinys naudojamas, kai kintamajam reikia suteikti reikšmę programos tekste.

Priskyrimo sakinio struktūra tokia: **kintamojoVardas = Reiškinys**;

Čia simbolis = žymi priskyrimo operaciją, o Reiškinys nurodo, kokius veiksmus, kokia tvarka ir su kokiais argumentais reikia atlikti. Kairiojoje priskyrimo operacijos pusėje įrašytas kintamojo vardas nurodo, kam atmintyje suteikiama apskaičiuota reiškinio reikšmė.

Pavyzdžiui, priskyrimo sakinys
y = x * x;
reiškia, kad argumento **x** reikšmė keliama kvadratu ir rezultatas priskiriamas kintamajam **y**

Sudėtinis sakinys – tai tarp riestinių skliaustų { } parašyta sakinių seka
Pavyzdžiui

```
{
    n=n+1;
    suma = suma+n;
}
```

Aritmetinės operacijos

+ sudėtis
- atimtis
***** daugyba
/ sveikų skaičių dalyba (skaičius iki kablelio)
% sveikų skaičių dalyba (sveikoji liekana)

n=n+k;	**n+=k;**
n=n-k;	**n-=k;**
n=n*k;	**n*=k;**
n=n / k; (fps > div)	**n/=k;**
n=n % k; (fps > mod)	**n%=k;**
n=n+1;	**++n; arba n++;**
n=n-1;	**--n; arba n--;**

I	**+ - * / %**	**I**	**=**	**I**
R	**+ - * / %**	**I**	**=**	**R**

&& loginiam reiškiniui – IR (loginė daugyba)
|| loginiam reiškiniui – ARBA (loginė sudėtis)
! loginiam reiškiniui – NE
> DAUGIAU;
< MAŽIAU;
>= DAUGIAU ARBA LYGU;
<= MAŽIAU ARBA LYGU;
!= NELYGU;
== loginiam reiškiniui – LYGU

Kintamųjų reikšmės		Loginių reiškinių rezultatai		
a	**b**	**a && b**	**a || b**	**! a**
true	true	true	true	false
true	false	false	true	false
false	true	false	true	true
false	false	false	false	true

SĄLYGINIS SAKINYS

if (sąlyga **)** *pirmas sakinys;*
else *antras sakinys;*

```
if (a > 0)   ats = sqrt(a);
  else cout << " Šaknies iš neigiamo skaičiaus ištraukti negalima" ;
```

Sąlyginiu sakiniu programoje keičiama nuosekli sakinių atlikimo tvarka:
- jei Sąlyga tenkinama, atliekamas Pirmas Sakinys,
- jei ne – po else esantis Antras Sakinys.

Jeigu reikia atlikti kelis sakinius, kai Sąlyga tenkinama arba netenkinama, tai jie rašomi tarp { ir }.

Pavyzdžiai	Sąlyga		Ekrane matysite
`x = 4;` `if (x < 9) y = x + 3;` `else y = x + 5;` `cout << y;`	4 < 9	true	7
`x = 14;` `if (x < 9) y = x + 3;` `else y = x + 5;` `cout << y;`	14 < 9	false	19
`x = 4;` `if (x < 9) {` `  x = x + 2;` `  y = x * 3;` `}` `else {` `  x = x - 5;` `  y = x * 5;` `}` `cout << x << "  " << y;`	4 < 9	true	6 18

Galima rašyti sutrumpintą sąlyginį sakinį, kuriame yra tik Pirmas Sakinys ir veiksmai atliekami tik tuomet, kai Sąlyga tenkinama.

if (*Sąlyga*) *PirmasSakinys;*	**if** (*Sąlyga*) { *Sakiniai, kurie atliekami, kai Sąlyga tenkinama;* }

Sąlyginio sakinio šakose galima užrašyti bet kokius sakinius.
Bet kurioje sąlyginio sakinio šakoje galima užrašyti dar vieną sąlyginį sakinį, pastarojo šakose – dar po vieną ir t. t. Toks sakinys vadinamas **sudėtingu sąlyginiu sakiniu**.

if (*Sąlyga1*) **if** (*Sąlyga2*) *PirmasSakinys;* **else** *AntrasSakinys;* **else** *TrečiasSakinys;*	**if** (*Sąlyga1*) **if** (*Sąlyga2*) **if** (*Sąlyga3*) *PirmasSakinys;* **else if** (*Sąlyga4*) *AntrasSakinys;*

CIKLO SAKINYS WHILE

Ciklas yra naudojamas pasikartojantiems veiksmams atlikti. Labai dažnai veiksmai kartojami tol, kol tenkinama nurodyta sąlyga. Tokiais atvejais naudojamas ciklo sakinys **while**.

`while (Sąlyga)` `  KartojamasSakinys;`	`while (Sąlyga) {` `  KartojamiSakiniai;` `};`

Vykdant sakinį **while** pirmiausia patikrinama **Sąlyga**:

- jei jos reikšmė yra **true** (tiesa), tuomet atliekamas KartojamasSakinys,
- priešingu atveju jis praleidžiamas ir atliekami toliau už ciklo esantys sakiniai.
- Jei kartoti reikia ne vieną sakinį, rašomas sudėtinis sakinys { }.

Naudojant nežinomo kartojimų skaičiaus ciklą **while**, reikia **atkreipti dėmesį** į šiuos dalykus:

- jeigu ciklo viduje reikia įvykdyti kelis sakinius, tai jie turi būti rašomi tarp { ir };
- jei prieš pradedant ciklą Sąlyga yra netenkinama (false), KartojamasSakinys (ar sakinių grupė) nevykdomas nė karto;
- kintamiesiems, kurie yra sąlygos reiškinyje, prieš sakinį **while** esančioje programos dalyje **turi būti suteiktos pradinės reikšmės**.
- Ciklo viduje šių kintamųjų reikšmės turi būti keičiamos taip, kad kada nors Sąlyga taptų **false**.
- Kitu atveju ciklas bus vykdomas be galo daug kartų. (Tai amžinasis ciklas.)

CIKLO SAKINYS FOR

Kai iš anksto žinoma, kiek kartų reikia kartoti veiksmus, naudojamas žinomo kartojimų skaičiaus ciklas **for**. Ciklo **for** antraštė valdo tik vieno sakinio kartojimą. Jeigu reikia kartoti kelis sakinius, tai jie rašomi tarp { ir }.

`for (R1; Sąlyga; R2)` `  KartojamasSakinys;`	`for (R1; Sąlyga; R2) {` `  KartojamiSakiniai;` `}`

```
m = 10; n = 12;
for (i = m; i <= n; i = i + 1)
  cout << i << endl;
```

Ciklo žingsniai	Kintamojo *i* reikšmė	Sąlyga: *i* <= 12		Ekrane matysite
1	10	10 <= 12	true	10
2	11	11 <= 12	true	11
3	12	12 <= 12	true	12
4	13	13 <= 12	false	

Ciklas atliekamas 3 kartus.

Formaliai ciklo **for** veiksmus galima aprašyti taip:

1. Vykdomas sakinys R1. Paprastai jis naudojamas ciklo kintamojo (kintamasis, kuris vartojamas ciklo pabaigai apibrėžti) pradinei reikšmei priskirti. Sakinys R1 įvykdomas tik pirmą kartą.
2. Patikrinama Sąlyga. Jei jos reikšmė yra true, tuomet atliekamas KartojamasSakinys (arba KartojamiSakiniai). Priešingu atveju ciklas nutraukiamas.
3. Vykdomas sakinys R2. Jis nurodo, kaip turi būti apskaičiuojama ciklo kintamojo reikšmė. Veiksmai kartojami nuo 2 žingsnio.

Naudojant žinomo kartojimų skaičiaus ciklą, reikia atkreipti dėmesį į šiuos dalykus:

- jeigu ciklo viduje reikia įvykdyti kelis sakinius, tai jie turi būti rašomi tarp { ir };
- jei prieš pradedant ciklą Sąlyga yra netenkinama (false), KartojamasSakinys (ar sakinių grupė) nevykdomas nė karto;

- kintamiesiems, kurie yra sąlygos reiškinyje, prieš patikrinant Sąlygą pirmą kartą turi būti suteiktos pradinės reikšmės. Jos gali būti nurodomos prieš ciklą arba sakinyje R1 priskyrimo sakiniu, kurie atskiriami kableliu.
- Ciklo viduje kintamųjų, kurie yra sąlygos reiškinyje, reikšmės turi būti keičiamos taip, kad kada nors Sąlyga taptų false. Kitu atveju ciklas bus vykdomas be galo daug kartų. Toks ciklas vadinamas **amžinuoju ciklu**.

NAUDOJAMŲ IR / AR REKOMENDUOJAMŲ MATEMATINIŲ FUNKCIJŲ SĄRAŠAS

Funkcija	Paskirtis
`int abs(int x);`	Grąžina sveikojo skaičiaus x absoliutųjį didumą. Esant teigiamam ar neigiamam funkcijos parametrui, rezultatas yra teigiamas. Pavyzdys: `int x = -10;` `cout << "Sveikojo skaičiaus " << x` `<< " modulis yra " << abs(x) << endl;` Ekrane matysite: `Sveikojo skaičiaus -10 modulis yra 10`
`double fabs(double x);`	Grąžina realiojo skaičiaus x absoliutųjį didumą. Pavyzdys: `double x = -15.5;` `cout << "Realiojo skaičiaus " << x` `<< " modulis yra " << fabs(x) << endl;` Ekrane matysite: `Realiojo skaičiaus -15.5 modulis yra 15.5`
`double pow(double x, double y);`	Skaičių x kelia laipsniu y. (Jei y yra 0.5, tai funkcija skaičiuoja neneigiamo argumento x kvadratinę šaknį.) Pavyzdys: `int x = 15, y = 3;` `cout << "Skaičius " << x` `<< " pakeltas laipsniu " << y` `<< ": " << pow(x, y) << endl;` Ekrane matysite: `Skaičius 15 pakeltas laipsniu 3: 3325`
`double sqrt(double x);`	Skaičiuoja neneigiamo argumento x kvadratinę šaknį. Pavyzdys: `double x = 15.5;` `cout << "Kvadratinė šaknis iš realiojo skaičiaus "` `<< x << " yra "` `<< sqrt(x) << endl;` Ekrane matysite: `Kvadratinė šaknis iš realiojo skaičiaus 15.5 yra 3.937`
`int random(int Riba);`	Grąžina atsitiktinį sveikąjį skaičių iš intervalo `[0; Riba)`. Pavyzdys: `cout << "Atsitiktinis skaičius intervale 0-99: "` `<< random(100) << endl;` Ekrane matysite: `Atsitiktinis skaičius intervale 0-99: 84`
`int rand(void);`	Grąžina atsitiktinį sveikąjį skaičių iš intervalo `[0; RAND_MAX]`. Konstanta `RAND_MAX` dažniausiai lygi 32767. Norint pasinaudoti konstanta `RAND_MAX`, reikia prie programos prijungti failą `stdlib.h` sakiniu `#include <stdlib.h>` Pavyzdys: `cout << "Atsitiktinis skaičius: " <<` `<< rand() << endl;` Ekrane matysite: `Atsitiktinis skaičius: 130`