Programmēšanas valoda C++

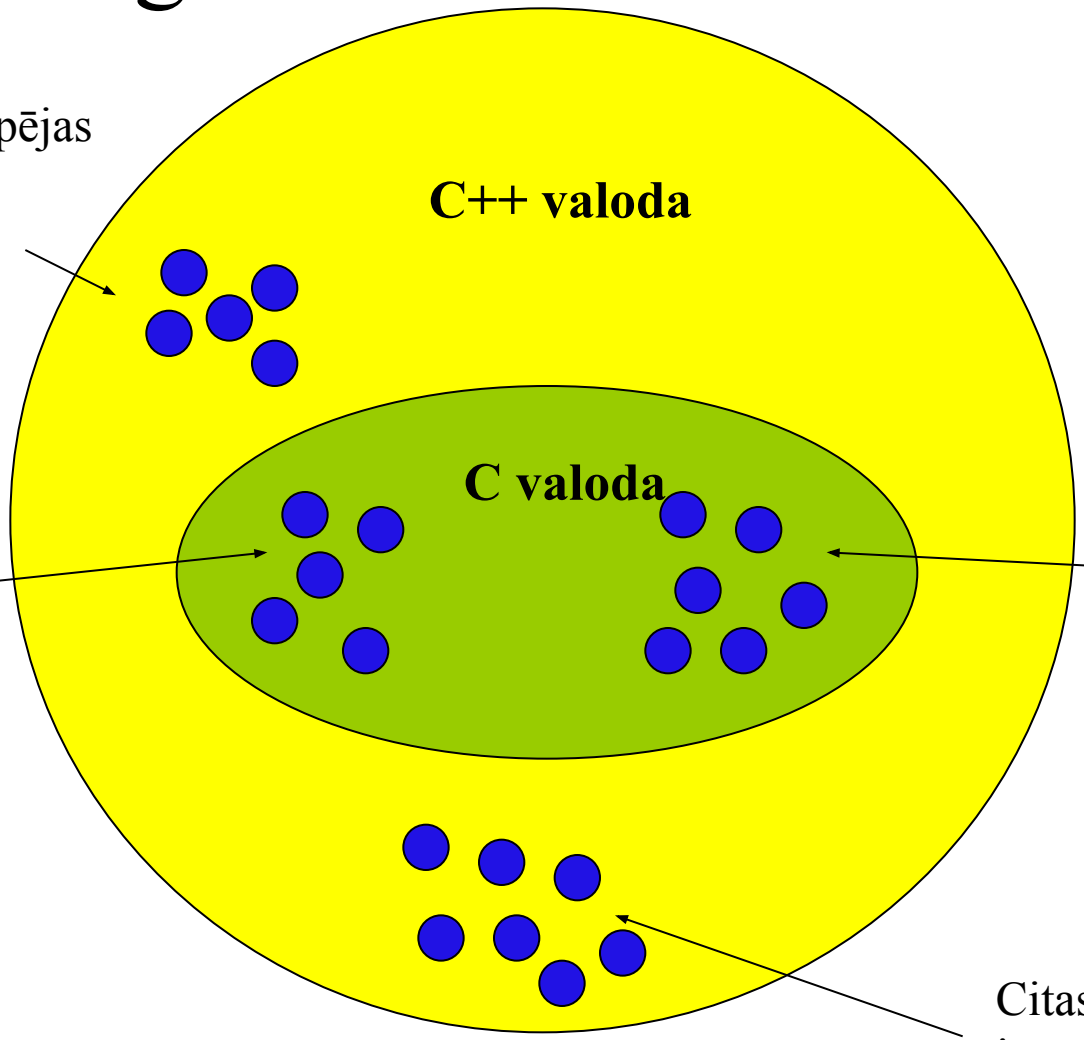
Programmēšanas valoda C

- Programmēšanas valodu C 1970. gadā izstrādāja *AT&A Bell Laboratories* līdzstrādnieks **Deniss Ričijs** (*Dennis Ritchie*).
- Valoda C uzskatāma par kaut ko, kas ir pa vidu starp ļoti augsta un zema līmeņa programmēšanas valodām.
- C valodas trūkumi – tajā rakstītās programmas nav tik viegli uztveramas un saprotamas, nav daudzu iebūvētu automātiskās pārbaudes iespēju, nav augsta līmeņa konstrukciju.

Programmēšanas valoda C++

- Valoda C++ ir izcēlusies no valodas C
- Programmēšanas valodu C++ 20. gs. 70. gadu beigās – 80. gadu sākumā izstrādāja AT&A Bell Laboratories līdzstrādnieks **Bjerns Stroustrups** (*Bjarne Stroustrup*)
- Ieviestas daudzas augsta līmeņa konstrukcijas, kas atvieglo programmētāja darbu.
- C++ ir saglabājis savietojamību ar valodu C
- Uz valodu C++ vēl lielākā mērā attiecas tas, ko savulaik attiecināja uz valodu C – tajā ir kopā apvienotas gan zema, gan ļoti augsta līmeņa konstrukcijas

Programmēšanas valoda C++



Funkcionālas iespējas objektorientētas programmēšanas nodrošināšanai

Funkcionālas iespējas kas ir kopīgas C un C++

Funkcionālas iespējas kas parasti neizmanto C++

Citas funkcionālas iespējas

Programmēšanas valoda C++

C++ valoda ir C valoda uz kuras ir uzlikts krusts!... DIVREIZ!!!

:)

Pirmā programma uz C++

Ievadam programmas tekstu, rindiņu numurus nav jāievada.

C++ valoda ir jūtīga burtu reģistram – aizvietot main ar Main vai MAIN nedrīkst!

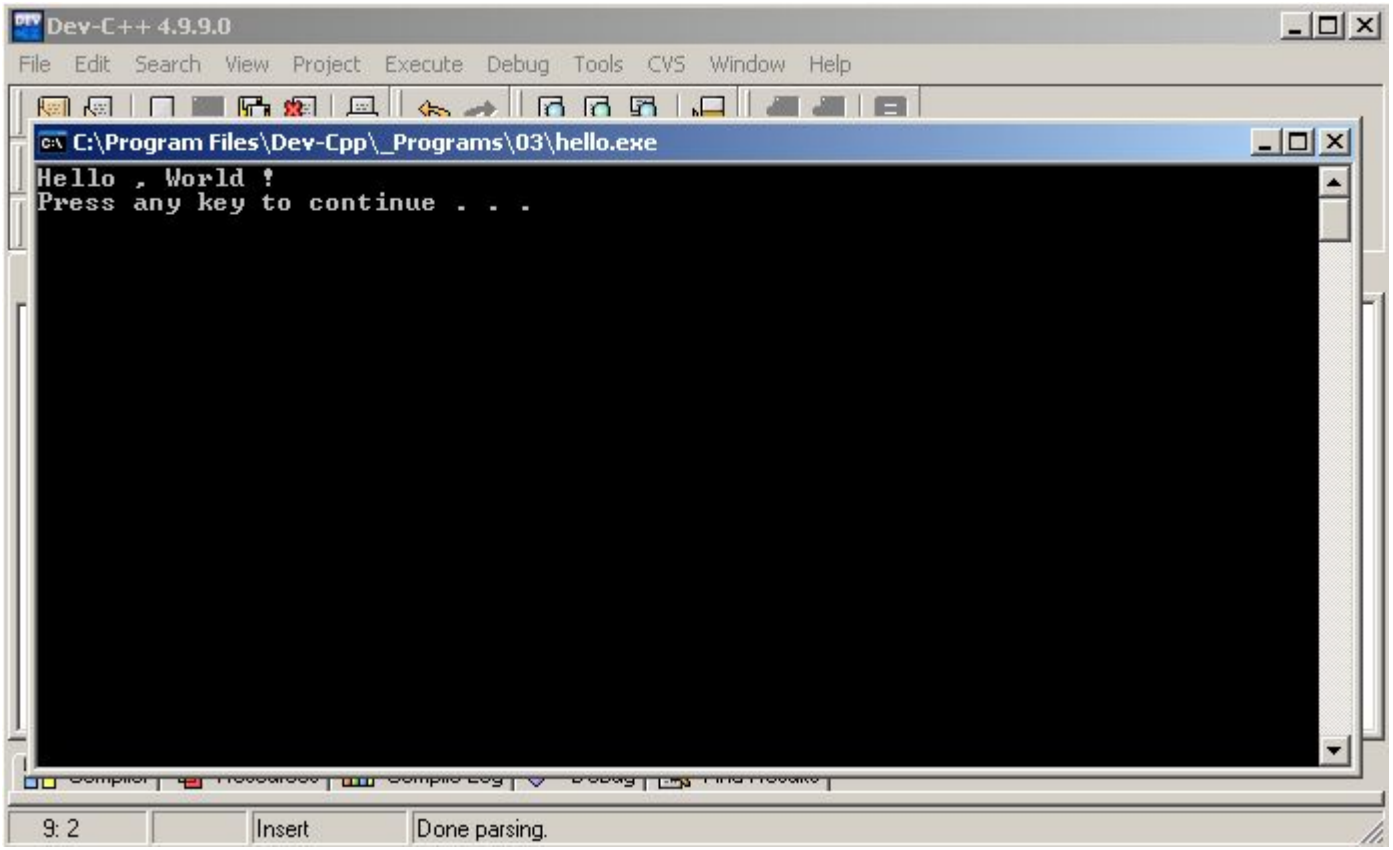
```
1.    #include <iostream >
2.    using namespace std;
3.
4.    int main()
5.    {
6.        cout << "Hello , World !" << endl ;
7.        system ("pause");
8.        return 0;
9.    }
```

Saglabājam failu ar nosaukumu *hello.cpp*

Kompilējam un palaižam (shortcut F9)

Pirmā programma uz C++

Palaišanas rezultāts

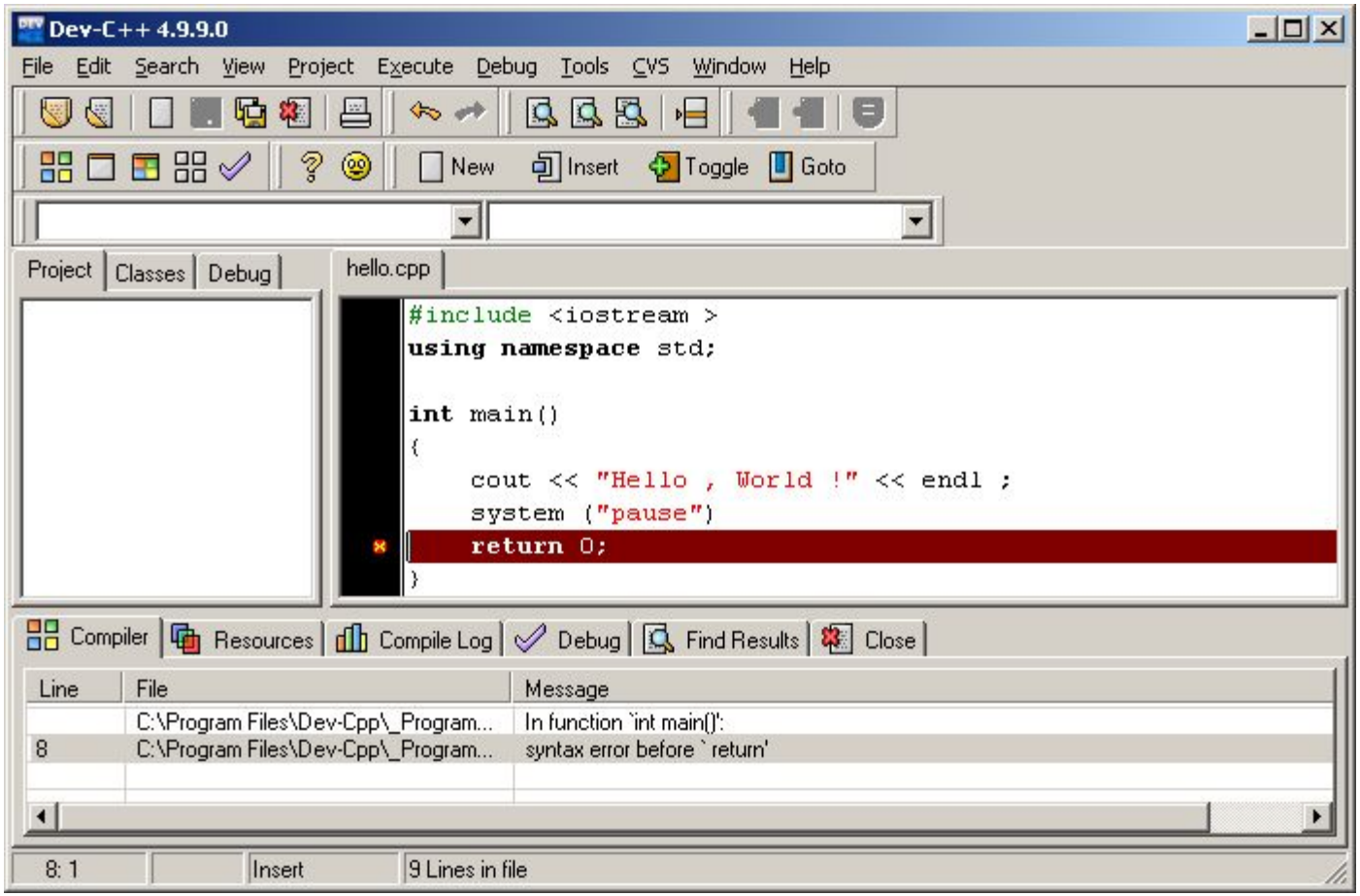


Ja viss bija pareizi jūs redzēsiet “Hello World!”

Lai iziet no programmas nospiediet jeb kuru taustiņu.

Pirmā programma uz C++

Ja kompilācijas laikā notiks kļūda, jūs par to dabūsiet atbilstošo ziņojumu.



Programmas analīze

```
1.    #include <iostream>
2.    using namespace std;
3.
4.    int main()
5.    {
6.        cout << "Hello , World !" << endl ;
7.        system ("pause");
8.        return 0;
9.    }
```

1. Ievades/izvades (Input/Output) plūsmu (streams) atbalsts
2. Izmantot nosaukumus no C++ standarta bibliotēkas
3. Tukšas rindiņas ignorējas, tas lieto lai uzlabot koda lasamību
4. main() — jeb kādai programmai uz C++ jābūt funkcijai ar šādu nosaukumu; atslēgvārds int nozīmē ka programma atgriezīs operētājsistēmai veselo skaitli
5. un 9. – jeb kādas funkcijas ķermeni jāiekļauj figūriekavās

Programmas analīze

```
1.    #include <iostream>
2.    using namespace std;
3.
4.    int main()
5.    {
6.        cout << "Hello , World !" << endl ;
7.        system ("pause");
8.        return 0;
9.    }
```

6. *cout* ir objekta vārds kas nodrošina informācijas izvadi uz ekrānu, tas kas ir jāizvada atdala ar “<<” simboliem, *endl* pārvieto kursoru uz jauno rindiņu.
7. *system ("pause")* konstrukcija gaida kamēr būs nospiesta kāda no taustiņam un nedod programmas logam aizvērties.
8. Punkts kurā notiek izeja no programmas, operētājsistēma saņem vērtību “0”

Piezīmes

- Kompilators ignorē tukšas rindiņas
- Pārejas uz nākamo rindiņu un atkāpes arī neietekmē uz kompilatora darbu.
- Sēkojoša programma ir pilnīgi ekvivalenta tikko apskatītai:

```
#include <iostream> using namespace std;int main (){ cout <<
"Hello , World !"<< endl ; system ("pause"); return 0;}
```

- Lai uzlabot koda lasāmību kodā jāliek atkāpes un tukšas rindiņas.
- Gandrīz visas komandas beidzās ar “;”
- Konstrukcija *#include <iostream>* nozīmē: “iekļaut programmas tekstā failu *iostream*”

Praktiskais darbs

- Izmainiet programmas tekstu tā lai tiktu izvadīta vēl viena rindiņa “I love C++”
- Cik reizes bija jāizmanto *cout* objektu?
- Mēģiniet izmainīt programmas tekstu tā lai izmantot to tikai vienu reizi.

Komentāri C++ valodā

Komentāru lietošana dod iespēju komentēt sarežģītas koda vietas, vai nestandartus risinājumus.

Komentārs ir simbolu secība kuru kompilators uztver ka vienu atstarpi, citiem vārdiem sakot – ignorē.

```
#include <iostream>
using namespace std;

int main() //galvena programmas funkcija
{
    cout << "Hello , World !" << endl; //izvade uz ekranu
    system ("pause");
    return 0;
} //programmas beigas
```

Viss teksts kas ir ievietots aiz "//" simboliem un iet līdz rindiņas beigām skaitās par komentāru.

Komentāri C++ valodā

```
#include <iostream>
using namespace std;

int main() //galvena programmas funkcija
{
    cout << "Hello , World !" << endl; //izvade uz ekranu
    /* Seit ir loti gars komentars kas
    aiznem vairakas rindinas
    cout << ”I love C++”;
    gara komentara beigas */
    system ("pause");
    return 0;
} //programmas beigas
```

Viss teksts kas sākās aiz “/*” simboliem skaitās par komentāru, tāda veida komentāru var pabeigt ar “*/” simbolu secību.

Komentārus var izmantot arī programmas atkļūdošanas procesā, ieliekot komentāros aizdomīgas koda gabalus.

Mainīgie

a + x = b vienādojuma risināšana

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  int main()
5.  {
6.      double a, b;
7.
8.      cout << "Input a: ";
9.      cin >> a;
10.
11.     cout << "Input b: ";
12.     cin >> b;
13.
14.     double x = b-a;
15.     cout << "Root of equation a+x=b is " << x << endl;
16.
17.     system ("pause");
18.     return 0;
19. }
```

Mainīgie

- *cin* ir objekta vārds kas nodrošina informācijas ievadi no klaviatūras, tas kas ir jāievada atdala ar “>>” simboliem
- `double a, b;` mainīgo definēšana
- `double x = b-a;` mainīgo definēšana un sākotnējas vērtības piešķiršana.

Mainīgie

- Lietotāja ievades datiem jābūt kaut kur saglabātiem
- Tam kalpo mainīgie
- Visiem mainīgiem C++ valodā pirms to izmantošanas jābūt nodefinētiem
- C++ valodā mainīgo var nodefinēt jeb kurā vietā, bet
  - nedrīkst vairākkārtīgi definēt mainīgus ar vienu un to pašu nosaukumu vienā un tā pašā blokā (bloks turpinās no atverošas figūriekavas līdz aizverošas figūriekavas)
  - mainīgo var izmantot uzreiz pēc definēšanas un līdz tekošā bloka aizvēršanas (līdz tuvākai aizverošai figūriekavai)

Namespaces

Programmu, kas parāda namespace nepieciešamību

```
1.  int main()
2.  {
3.      int value;
4.      value = 0;
5.      double value; // Kļūda
6.      value = 0.0;
7.  }
```

```
Compiler Error: 'value' has a previous declaration as 'int value'
```

Katrā blokā nosaukums var eksistēt tikai vienā eksemplārā.

Namespaces

Izmantojot namespaces, mēs varam izveidot divus mainīgus ar vienādiem nosaukumiem.

```
1.  #include <iostream>
2.  using namespace std;
3.
4.  namespace first
5.  {
6.      int val = 500; // Mainīgais izveidots iekš namespace
7.  }
8.
9.  int main()
10. {
11.     int val = 200; // Lokālais mainīgais
12.
13.     // namespace mainīgajiem var piekļūt
14.     // ārpus namespace, izmantojot operatoru ::
15.     cout << first::val << endl;
16.
17.     return 0;
18. }
```

Output:

```
500
```

Mainīgo tipi valodā C++

Pamattipi

Keyword	Low	High	Dig. of Prec.	Bytes
char	-128	127	n/a	1
short	-32’768	32’767	n/a	2
int	-2’147’483’648	2’147’483’647	n/a	4
long	-2’147’483’648	2’147’483’647	n/a	4
float	$3.4*10^{-38}$	$3.4*10^{38}$	7	4
double	$1.7*10^{-308}$	$1.7*10^{308}$	15	8
long double	$3.4*10^{-4932}$	$3.4*10^{4932}$	19	16
bool	false	true	n/a	1

Unsigned Integer Types

Keyword	Low	High	Bytes
unsigned char	0	255	1
unsigned short	0	65’535	2
unsigned int	0	4’294’967’295	4
unsigned long	0	4’294’967’295	4

Mainīgo nosaukumi

Mainīgo nosaukumiem drīkst izmantot

- latīņu alfabēta burtus a–z, A–Z
- pasvītrošanas simbolu _
- ciparus 0–9

Mainīgo un funkciju nosaukumi

- nedrīkst sākties ar ciparu
- nerekomendējas sākt un beigt ar pasvītrošanas simbolu
- Nedrīkst sakrīt ar rezervētiem atslēgvārdiem (int, return, . . .)

Piemēri:

- pieļaujamie nosaukumi: `a, b, x1, y_42, day_of_week, num_of_students, route_66`
- nepieļaujamie nosaukumi: `3rd_law, double`
- nerekomendējamie nosaukumi: `_temp_var, new_var_`

Aritmētiskie operatori

- **“=“** piešķiršanas operators
- **“+”** saskaitīšanas operators
- **“–”** atņēmšanas operators
- **“*”** reizināšanas operators
- **“/”** dalīšanas operators (6/4 → 1, 6.0/4 → 1.25, 6/4.0 → 1.25)
- **“%”** atlikuma saņemšanas operators no veselo skaitļu dalīšanas (6%4 → 2, 13%6 → 1)

Praktiskais darbs

Uzrakstiet programmu kas satur sekojošo mainīgo definēšanas.

```
1.  int a = 13/5;
2.  int b = 13%5;
3.  int c = 13.0/5;
4.  double d = 13/5;
5.  double e = 13%5;
6.  double f = 13.0/5;
7.  double g = 13/5 + 2/5;
8.  double h = 13.0/5 + 2.0/5;
9.  int i = 13.0/5 + 2.0/5;
```

Izvadiet tos uz ekrānu, paskaidrojiet kāpēc sanāca tieši tāds rezultāts.

Praktiskais darbs

1. Doti divi veselie mainīgie ar vērtībām. Jāuzraksta programmu kas nomainītu mainīgo vērtības vietām (a=2, b=3 -> a=3, b=2).
2. Dots skaitlis x. Aprēķināt x^4.
3. Dots skaitlis x. Aprēķināt x^7 izmantojot 4 reizināšanas operācijas.
4. Dots skaitlis x. Aprēķināt x^{13} izmantojot 5 reizināšanas operācijas.
5. Dots naturāls skaitlis. Izvadiet tā pēdējo ciparu.

Praktiskais darbs

6. Dots divzīmju skaitlis. Atradiet tā desmitnieku skaitu.
7. Programma pieprasa ātrumu **v** km/h un pārveido to m/s.
8. Programma pieprasa atzīmes 3 kontroldarbos un aprēķina vidējo balli.
9. Zinot kuba malu **a**, aprēķināt kuba tilpumu, tā virsmas laukumu un diagonāles garumu.

Praktiskais darbs

Uzdevumi ar paaugstināto grūtības pakāpi

1. Doti divi veselie mainīgie a un b. Jāuzraksta programmu kura nomainītu vietām mainīgo vērtības, neizmantojot papildus mainīgo.
2. Doti trīs veselie skaitļi h, m, s. Atradiet leņķi (grādos) starp stundu bultiņu pusnaktī un stundu bultiņu laika momentā h:m:s.