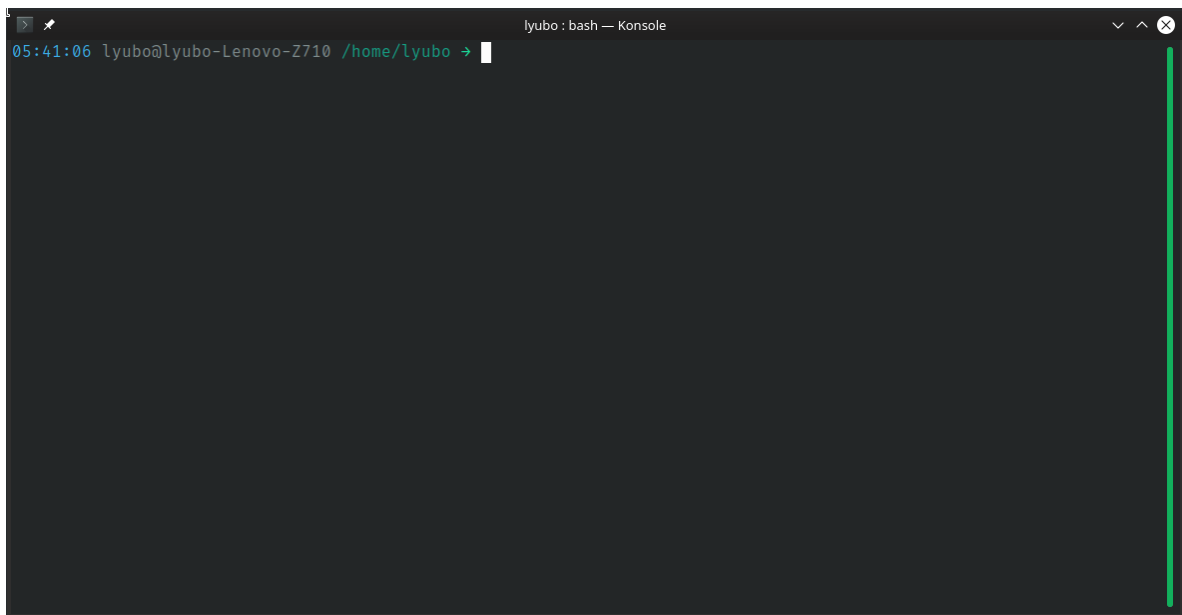


Въведение в C++. Основни елементи на езика

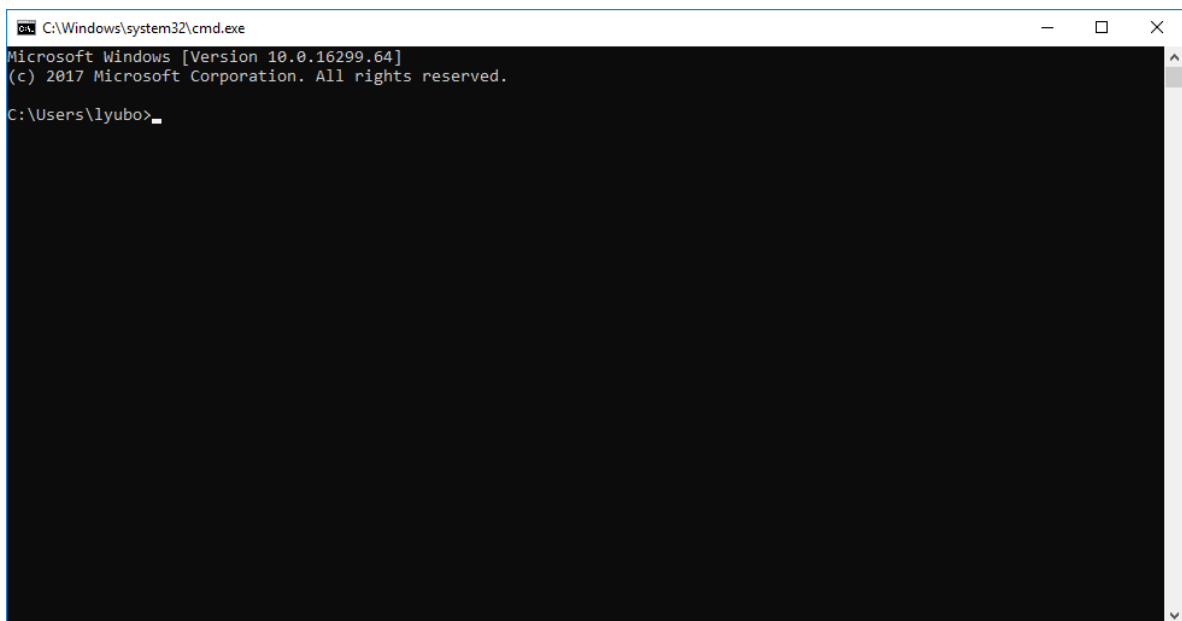
Конзолен интерфейс

Програмите на C++, които ще срещнете в този курс ще използват конзолата като графичен интерфейс.

Основната разлика с графичните интерфейси е, че на конзолния интерфейс могат да бъдат изобразявани само символи.



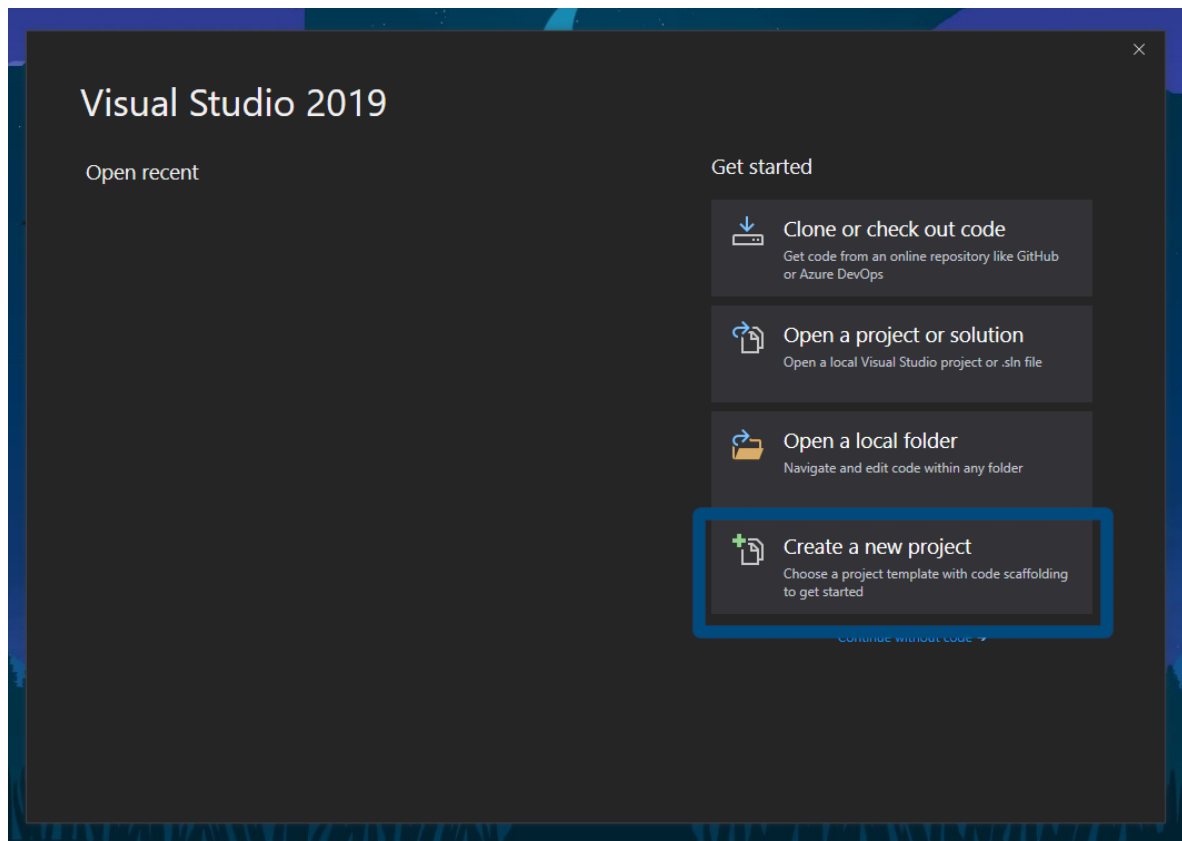
(Linux terminal)



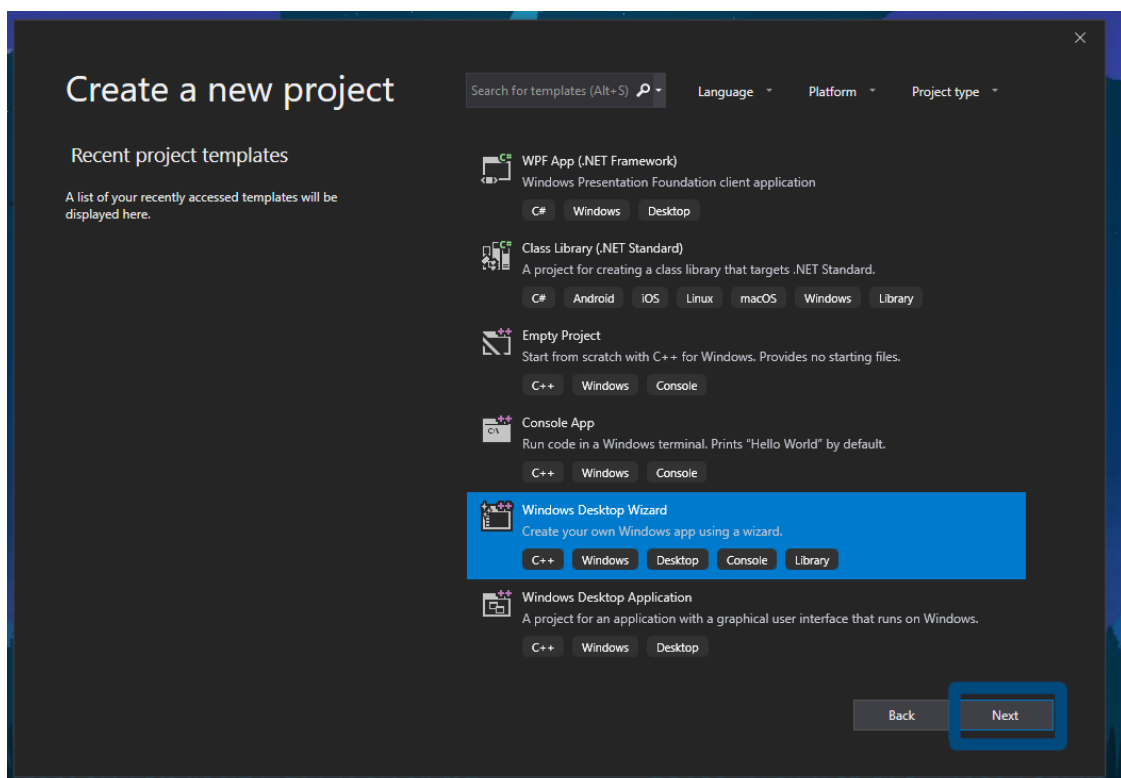
(Windows command prompt)

Създаване на проект с Visual Studio 2019

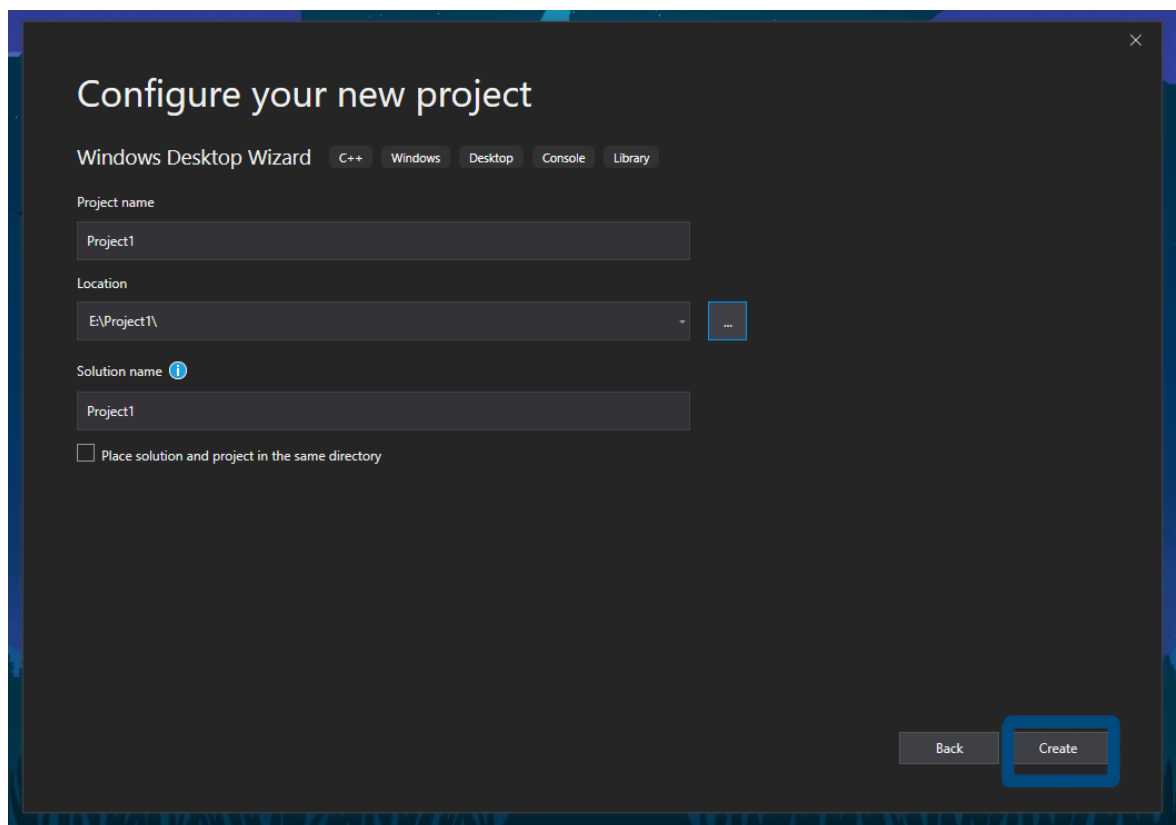
1. Отваряме Visual Studio 2019 и избираме "Create a new project"



2. От отворилото се меню намираме "Windows Desktop Wizard", натискаме го, и натискаме бутона "Next"

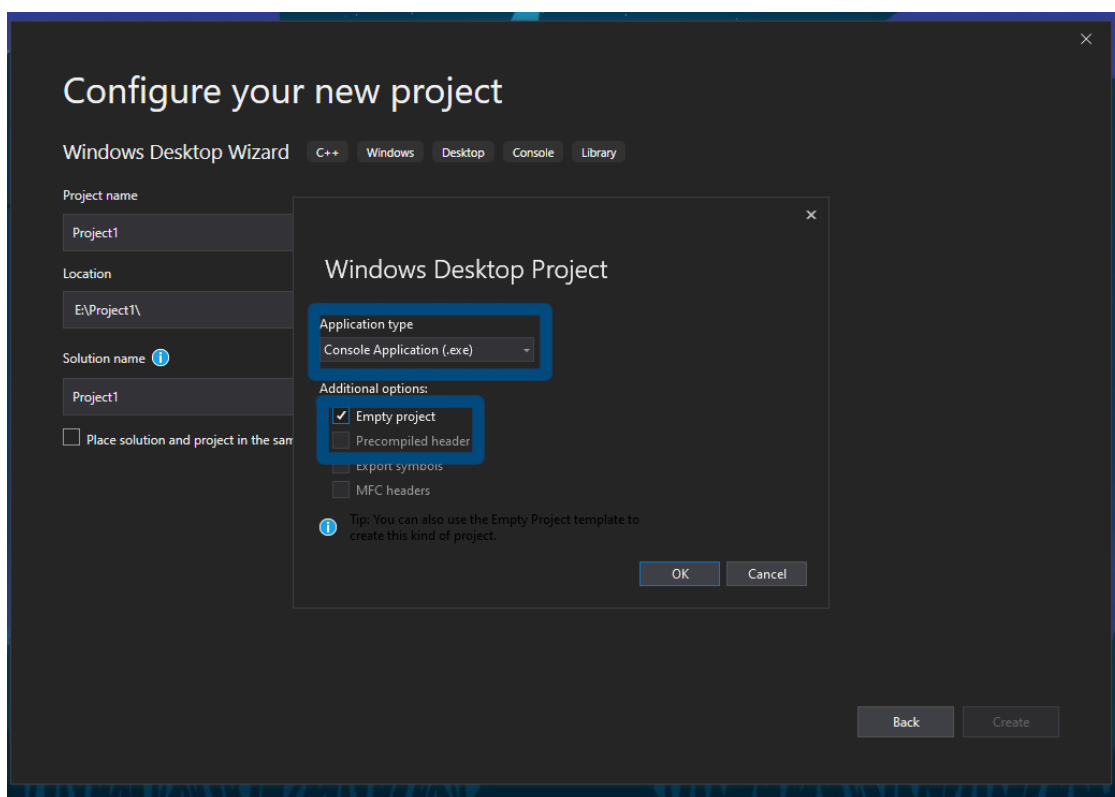


3. Избираме как да се казва нашия проект и къде да го съхраним, след това натискаме "Create"

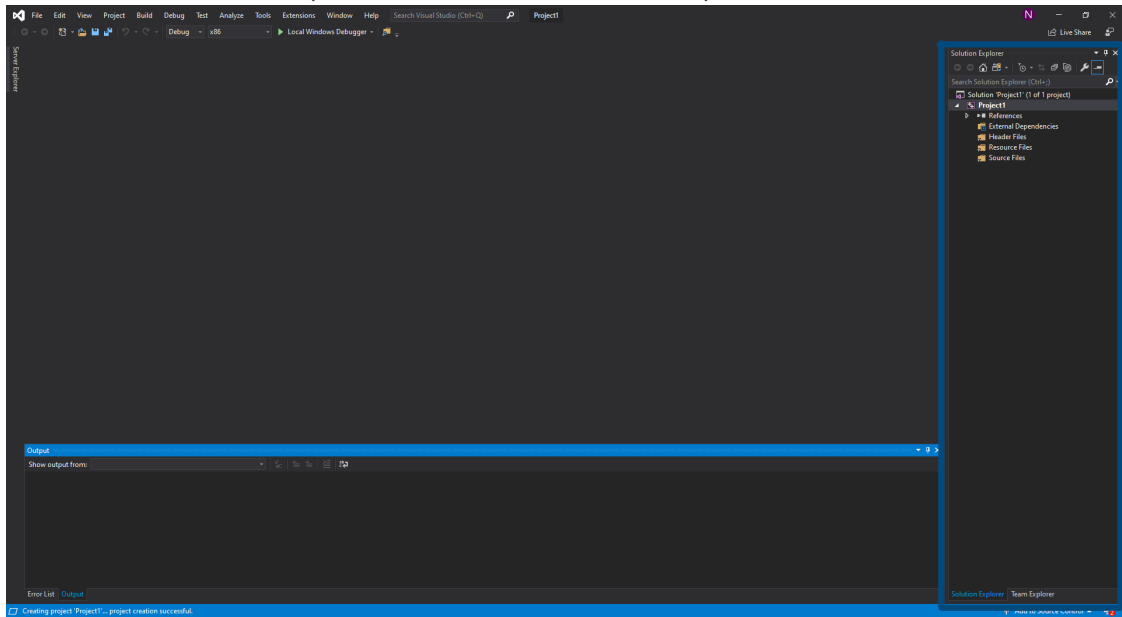


4. От следващия диалогов прозорец задаваме следните настройки:

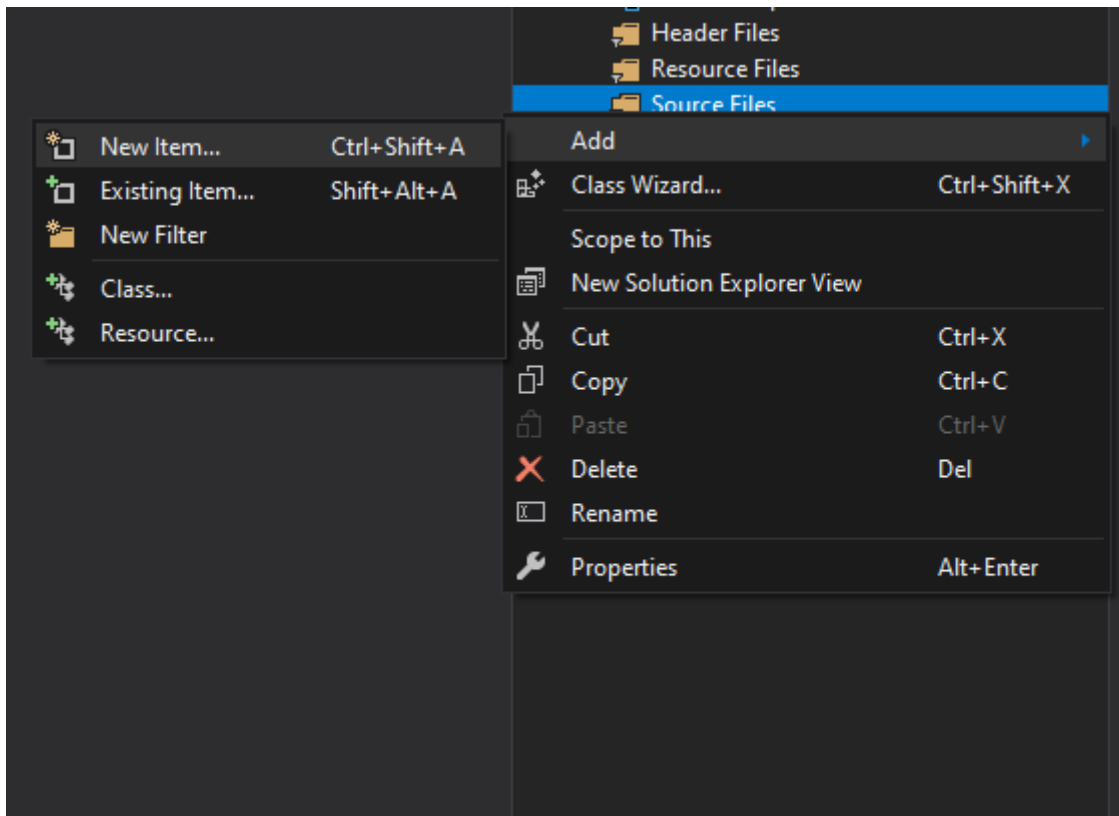
1. Application type: Console application (.exe)
2. Additional options: Empty project е избрано
3. Additional options: Precompiled headers не е избрано



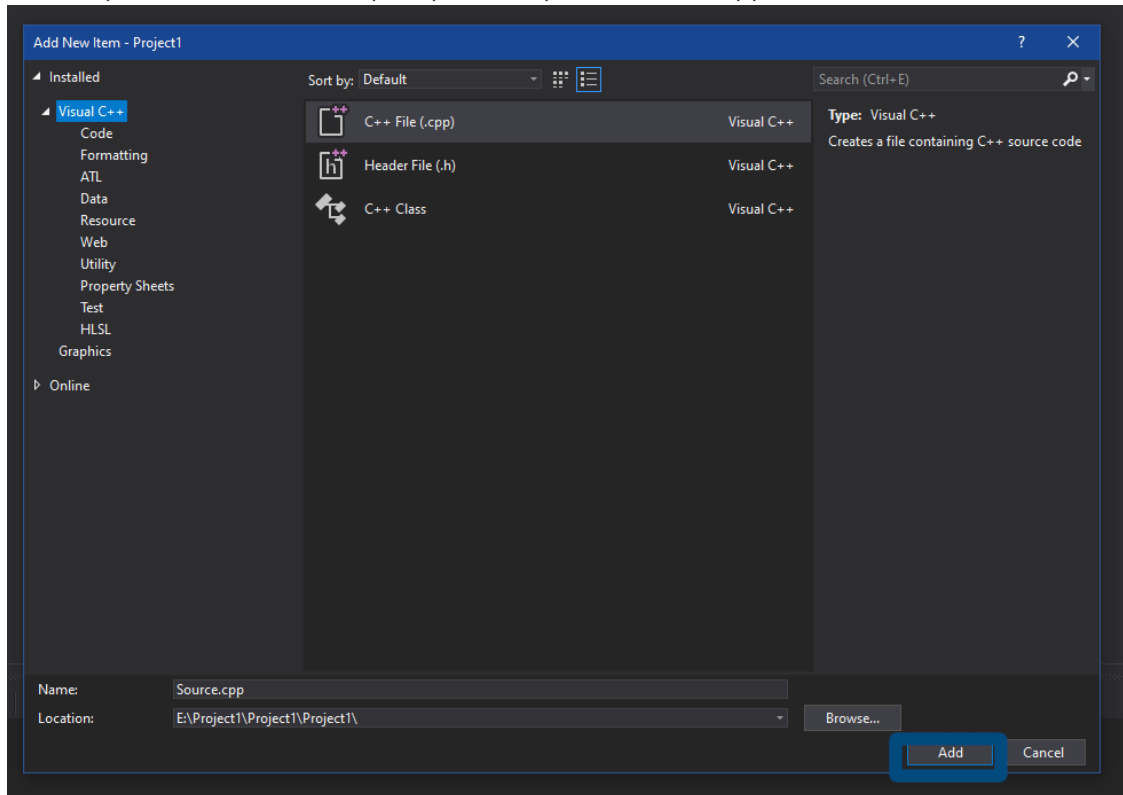
5. След като се създаде проекта, отворете Solution Explorer, ако вече не е отворен (ако не го виждате, той се намира в менюто View -> Solution Explorer)



6. Натиснете десен бутон върху папката Source Files, след това от менюто Add, изберете New item



7. От отворили я се диалогов прозорец изберете C++ File (.cpp), и след това натиснете Add



Първа C++ програма

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

int main()
{
    return 0;
}
```

Тази програма не прави нищо съществено. Поставена е тук с цел да бъде като основа за програмите, които ще пишем. Обяснения ред по ред:

`#include <iostream>` - `#include` е инструкция към "предпроцесора" (мислите си го като част от компилатора за момента) да вкара в нашия код библиотеката `iostream`, в която се съдържат по-голямата част от необходимите ни неща

`using namespace std;` - указваме на компилатора да използва `std` частта от `iostream` - по-дълбоко обяснение в курса по ООП

`int main()` - главната функция - от тук започва всяка програма на C++

`return 0;` - функцията `main()` връща 0 (приемете го за даденост)

По-късно в курса ще се разгледат тези редове в по-големи детайли.

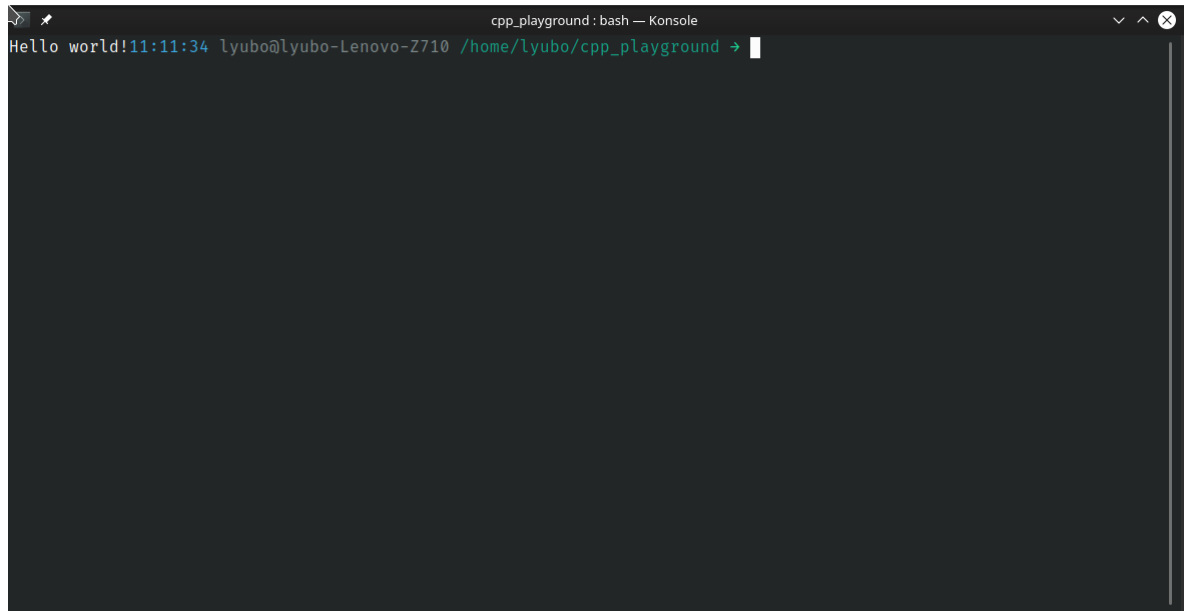
Оператори за вход и изход

Оператор за изход към конзолата (cout)

Когато искаме да напишем нещо към конзолата, използваме оператора `cout <<` и след това текста в кавичка. Напомням, че всяка инструкция завършва с `;`

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

int main()
{
    cout << "Hello world!";
    return 0;
}
```

A screenshot of a terminal window titled 'cpp_playground: bash — Konsole'. The terminal shows the output 'Hello world!' followed by a timestamp '11:11:34' and the user's login information 'lyubo@lyubo-Lenovo-Z710'. The current directory is '/home/lyubo/cpp_playground'. The prompt is a green arrow pointing to a white cursor.

```
cpp_playground: bash — Konsole
Hello world!11:11:34 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

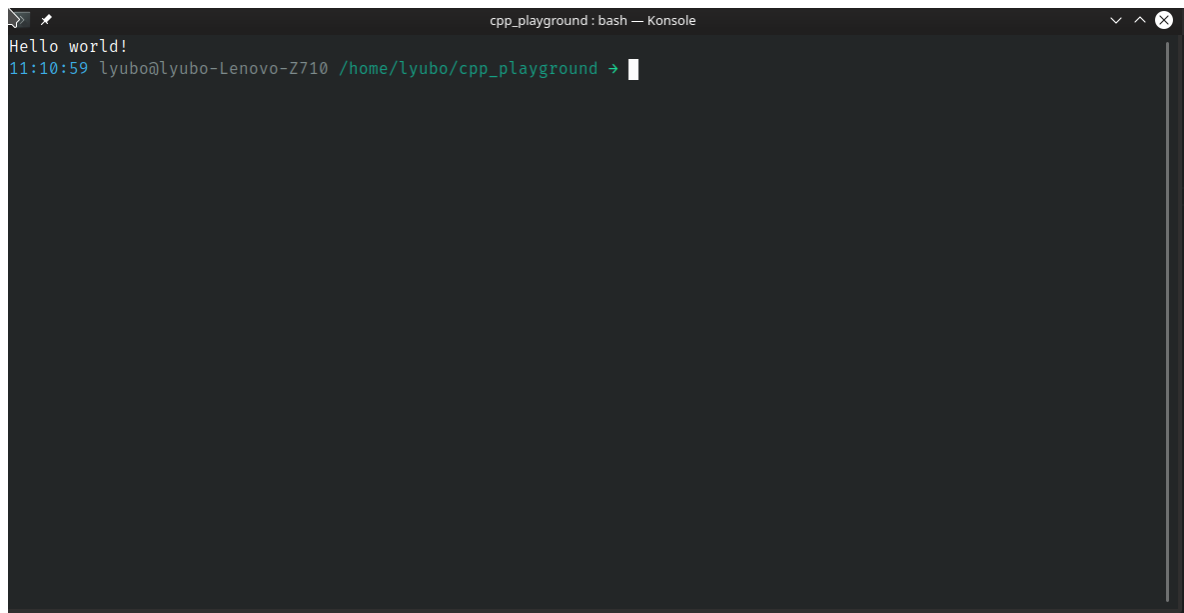
Ако искаме да изведем нов ред, това става по един от следните начини:

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

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

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

int main()
{
    cout << "Hello world!\n";
    return 0;
}
```

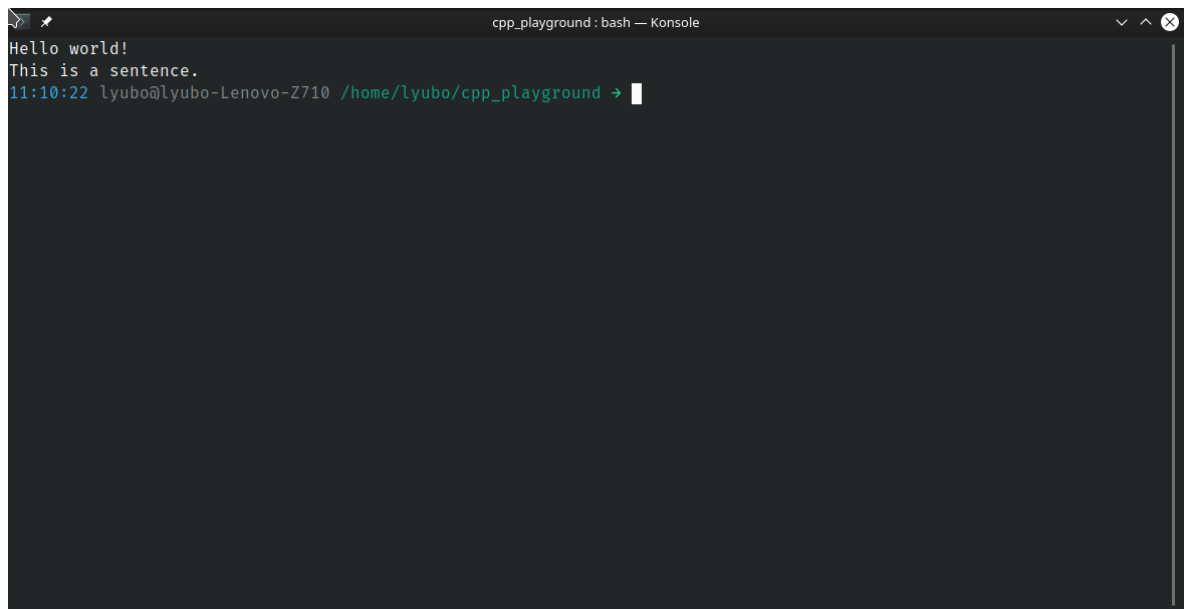
A terminal window titled 'cpp_playground : bash — Konsole'. The prompt is 'lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →'. The output is 'Hello world!' followed by a new line.

```
cpp_playground : bash — Konsole
Hello world!
11:10:59 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Ако искаме да изведем текст на няколко реда, или просто няколко последователни изречения, можем да ги разделяме с `<<endl`:

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

int main()
{
    cout << "Hello world!" << endl;
    cout << "This " << "is " << "a " << "sentence." << endl;
    return 0;
}
```

A terminal window titled 'cpp_playground : bash — Konsole'. The prompt is 'lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →'. The output is 'Hello world!' followed by 'This is a sentence.' on a new line.

```
cpp_playground : bash — Konsole
Hello world!
This is a sentence.
11:10:22 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Оператор за вход към конзолата (cin)

Ако искаме да въведем нещо към конзолата от клавиатурата, използваме оператора `cin`

```
>>
```

Тук изниква въпроса - къде ще се пази това, което потребителя ще въвежда от клавиатурата ?

Променливи

Променливите са места в паметта, в които можем да съхраняваме различни видове информация. Тези различни видове информация наричаме "типове данни". Примери за типове данни са: целочислен тип (int), числа с плаваща запетая (float), числа с двойна прецизност (double), символ (char) и булев (bool). Съществуват и други типове, които ще разгледаме по-късно в курса.

Всеки тип променлива е различен размер в паметта, т.е. побира различен размер информация.

Тип	Размер в байтове	Минимална и максимална стойност
bool	1бит (1/8 байта)	0 (false) или 1 (true)
char	1 байт	от -127 до 127 (възможно е и да е от 0 до 255)
int	4 байта	от -2147483648 до 2147483647
unsigned int	4 байта	от 0 до 4294967295
float	4 байта	+/- 3.4e +/- 38 (~7 цифри)
double	8 байта	+/- 1.7e +/- 308 (~15 цифри)

Променливи се дефинират по следният начин:

<тип_на_променливата> име_на_променливата;

Ако искаме да направим програма, в която потребителя въвежда число от клавиатурата и го изкарва обратно на екрана, тя би изглеждала по следният начин:

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

int main()
{
    cout << "Please enter a number" << endl;

    int user_number = 0;
    //Декларираме променлива на име user_number от тип int, която има
    първоначална стойност 0

    cin >> user_number;
    //Потребителя въвежда число от клавиатурата, което се запазва в променливата

    cout << "Your number is: " << user_number << endl;
    //Извеждаме съобщение и числото на екрана
    return 0;
}
```



```
cpp_playground : bash — Konsole
Please enter a number
20
Your number is: 20
11:50:57 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Забелязваме наличието на `//` в кода - това са т.нар. коментари - линии от кода, които не се компилират, а са с цел да помогнат на програмистите с обяснения на кода и други

Константи

Когато искаме да зададем стойност на някоя променлива, която да не можем да променим после, използваме запазената дума `const`

```
const int PI = 3.14
```

Използва се в случаите, когато някоя стойност се повтаря няколко пъти в кода (математически константи, други примери ще се разгледат по-късно в курса)

Оператори за аритметика

C++ ни позволява да извършваме различни аритметични операции:

- Събиране (+)
- Изваждане (-)
- Умножение (*)
- Целочислено деление (/)
- Деление по модул (остатък) (%)
- Оператор за увеличаване на число с 1 (++)
- Оператор за намаляване на число с 1 (--)
- Оператор за равенство (==)
- Оператор за неравенство (!=)
- По-малко (<)
- По-голямо (>)
- По-малко или равно (<=)
- По-голямо или равно (>=)

Примери:

Събиране, изваждане, умножение, целочислено деление, деление по модул:

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

```

int main()
{
    int x = 5;
    int y = 3;

    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "x + y = " << x + y << endl;
    cout << "x - y = " << x - y << endl;
    cout << "x * y = " << x * y << endl;
    cout << "x / y = " << x / y << endl;
    cout << "x % y = " << x % y << endl;
    return 0;
}

```

```

cpp_playground : bash — Konsole
x = 5
y = 3
x + y = 8
x - y = 2
x * y = 15
x / y = 1
x % y = 2
10:36:51 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →

```

Оператори за сравнение

Забелязваме че тук трябва да оградим израза със скоби, когато имаме оператор за сравнение

```

#include <iostream>
using namespace std;

int main()
{
    int x = 5;
    int y = 3;

    cout << "x = " << x << endl;
    cout << "y = " << y << endl;
    cout << "x == y = " << (x == y) << endl;
    cout << "x != y = " << (x != y) << endl;
    cout << "x < y = " << (x < y) << endl;
    cout << "x > y = " << (x > y) << endl;
    cout << "x <= y = " << (x <= y) << endl;
    cout << "x >= y = " << (x >= y) << endl;
    return 0;
}

```

```
cpp_playground : bash — Konsole
x = 5
y = 3
x == y = 0
x != y = 1
x < y = 0
x > y = 1
x ≤ y = 0
x ≥ y = 1
10:42:18 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Оператор ++ и оператор --

Сега ще разгледаме действието на оператор ++ върху целочислени променливи в двата му варианта - префиксен и постфиксен

Префиксен оператор ++

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

int main()
{
    int x = 5;

    cout << "x = " << x << endl;
    cout << "++x = " << ++x << endl;
    cout << "x = " << x << endl;

    return 0;
}
```

```
cpp_playground : bash — Konsole
x = 5
++x = 6
x = 6
10:49:51 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

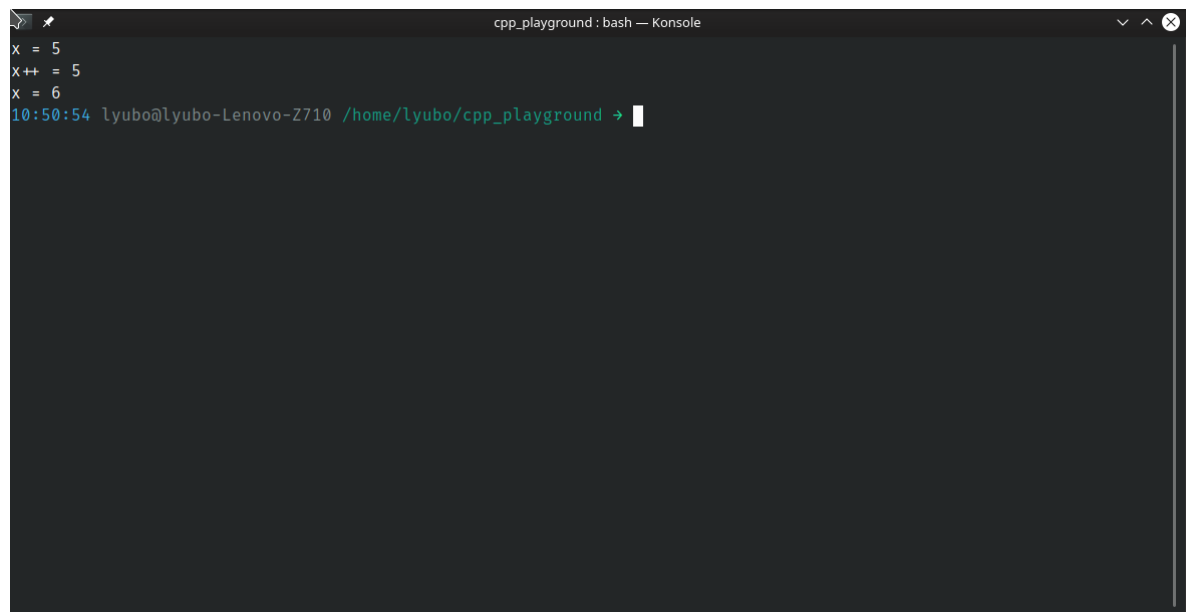
Постфиксен оператор ++

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

int main()
{
    int x = 5;

    cout << "x = " << x << endl;
    cout << "x++ = " << x++ << endl;
    cout << "x = " << x << endl;

    return 0;
}
```



```
cpp_playground: bash — Konsole
x = 5
x++ = 5
x = 6
10:50:54 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Забелязваме разлика между `++x` и `x++`

Когато използваме `++x` реда на операциите е "увеличи с едно и използвай стойността"

****Когато използваме `x++` реда на операциите е "използвай стойността и увеличи с едно"**