

Оператори за цикли

Ако искаме да повторим даден код няколко пъти, можем да използваме оператори за цикли

В C++ съществуват три оператора за цикли: `while`, `do while` и `for`. Те се разделят на индуктивни (с предварително ясен брой повторения) и итеративни цикли (такива с не предварително ясен брой повторения)

Оператор `while`

```
while(условие)
{
    //код
}
```

While цикълът работи по следният начин: проверява дали условието е вярно. Ако условието е вярно, се изпълнява тялото на цикъла.

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

int main()
{
    int counter = 0;

    while(counter < 5)
    {
        cout << "Counter is " << counter << endl;
        counter++;
    }
    return 0;
}
```

```
cpp_playground : bash — Konsole
Counter is 0
Counter is 1
Counter is 2
Counter is 3
Counter is 4
04:21:08 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

В началот променливата е със стойност 0, и условието `counter < 5` е изпълнено, следователно се изпълнява тялото на цикъла. `counter` получава стойност 1, условието `counter < 5` е изпълнено отново, и се изпълнява тялото на цикъла. Когато `counter` стане равно на 5, условието `counter < 5` не е изпълнено, и тялото на цикъла не се изпълнява. Така последният път, в който тялото на цикъла е изпълнено е при `counter = 4` и затова последния принтиран ред е `Counter is 4`

Цикълът `while` първо проверява условието, затова се нарича също и цикъл с пред-условие.

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

int main()
{
    int counter = 10;

    while(counter < 5)
    {
        cout << "Counter is " << counter << endl;
        counter++;
    }

    cout << "While loop has ended" << endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
While loop has ended
04:33:33 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Тук при първата проверка, условието `counter < 5` не е вярно (10 не е по-малко от 5), следователно тялото на цикъла не се изпълнява.

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

int main()
{
    int counter = 5;

    while(counter <= 5)
    {
        cout << "Counter is " << counter << endl;
        counter++;
    }

    cout << "While loop has ended" << endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
Counter is 5
While loop has ended
04:41:27 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

В този случай, условието `counter <= 5` е изпълнено точно веднъж, затова тялото на цикъла се изпълнява точно веднъж.

Оператор do/while

Друга вариация на цикъла `while` е `do while`. Разликата с `while`, е че първо се изпълнява тялото на цикъла, а после се проверява условието.

```
do
{
    //код
}
while(условие);
```

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

int main()
{
    int counter = 0;

    do
    {
        cout << "Counter is " << counter << endl;
        counter++;
    }
    while(counter < 5);

    cout << "Do/while loop has ended" << endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
Counter is 0
Counter is 1
Counter is 2
Counter is 3
Counter is 4
Do/while loop has ended
04:47:29 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Първоначално `counter` има стойност 0. Изпълнява се тялото на цикъла, след това се извършва проверката `counter < 5`. Важното е да отбележим, че при първата проверка `counter = 1`. Като `counter` стигне стойност 4, отново първо се изпълнява тялото на цикъла (и се принтира `Counter is 4`), а след това се увеличава `counter`, и условието `counter < 5` не е изпълнено.

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

int main()
{
    int counter = 10;

    do
    {
        cout << "Counter is " << counter << endl;
        counter++;
    }
    while(counter < 5);

    cout << "Do/while loop has ended" << endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
Counter is 10
Do/while loop has ended
04:55:51 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

В този случай, първо се изпълнява тялото на цикъла, след това се оценява, че `counter < 5` има стойност false и се получава напред.

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

int main()
{
    int counter = 5;

    do
    {
        cout << "Counter is " << counter << endl;
        counter++;
    }
    while(counter <= 5);

    cout << "Do/while loop has ended" << endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
Counter is 5
Do/while loop has ended
04:58:12 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

В този случай, тялото на цикъла се изпълнява, `counter` се увеличава с едно, и след това се проверява условието `counter <= 5` (което в този случай е същото като `6 <= 5`) и се продължава напред.

Цикъл for

Цикълът for е цикъл, при който броя на повторенията са известни предварително.
Синтаксис:

```
for(инициализация; условие; корекция)
```

Където `инициализация` е инициализирането на една или повече променливи, `условие` е израз, който трябва да е изпълнен, за да се изпълни тялото на цикъла, а `корекция` е оператор, който се изпълнява при изпълнено условие

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

int main()
{
    for(int i = 0; i < 5; i++)
    {
        cout << "Counter is " << i << endl;
    }
    cout << "For loop has ended" << endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
Counter is 0
Counter is 1
Counter is 2
Counter is 3
Counter is 4
For loop has ended
05:13:13 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

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

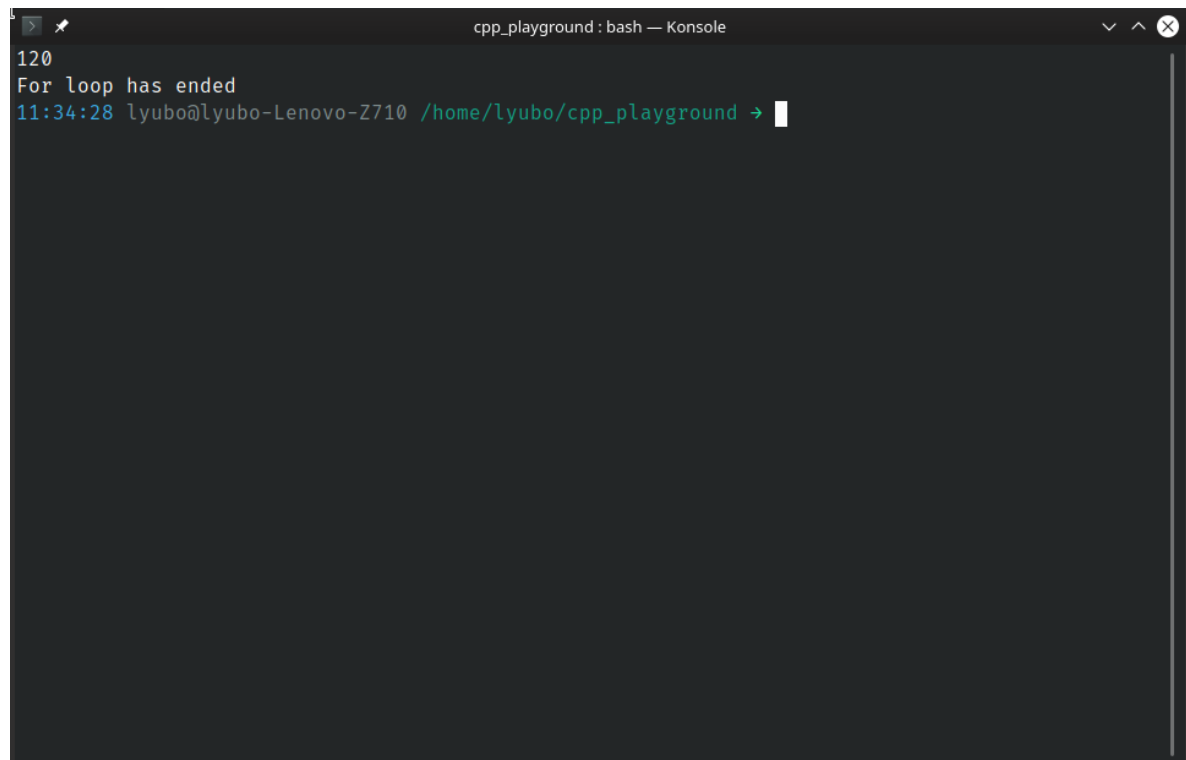
int main()
{
    for(int i = 10; i < 5; i++)
    {
        cout << "Counter is " << i << endl;
    }
    cout << "For loop has ended" << endl;
    return 0;
}
```

```
cpp_playground : bash — Konsole
For loop has ended
07:47:40 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```


Може да променяме повече от една променлива в цикъла

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

int main()
{
    int fact;
    int i;
    int n = 5;
    for(fact = 1, i = 1; i <= n; i++)
    {
        fact = fact * i;
    }
    cout << fact << endl;
    cout << "For loop has ended" << endl;
    return 0;
}
```



The screenshot shows a terminal window titled 'cpp_playground : bash — Konsole'. The output of the program is displayed as follows:

```
120
For loop has ended
11:34:28 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Безкрайни цикли

Съществуват няколко начина на направим безкраен цикъл:

```
for (int i = 10; i > 5; i++)
{
    //Безкраен цикъл
}
```

```
while(true)
{
    //Безкраен цикъл
}
```

Прието е безкрайните цикли да се правят по втория вариант.

Оператор break

Срещнахме оператора `break`, когато говорехме за `switch`. Сега ще го разгледаме в контекста на цикли.

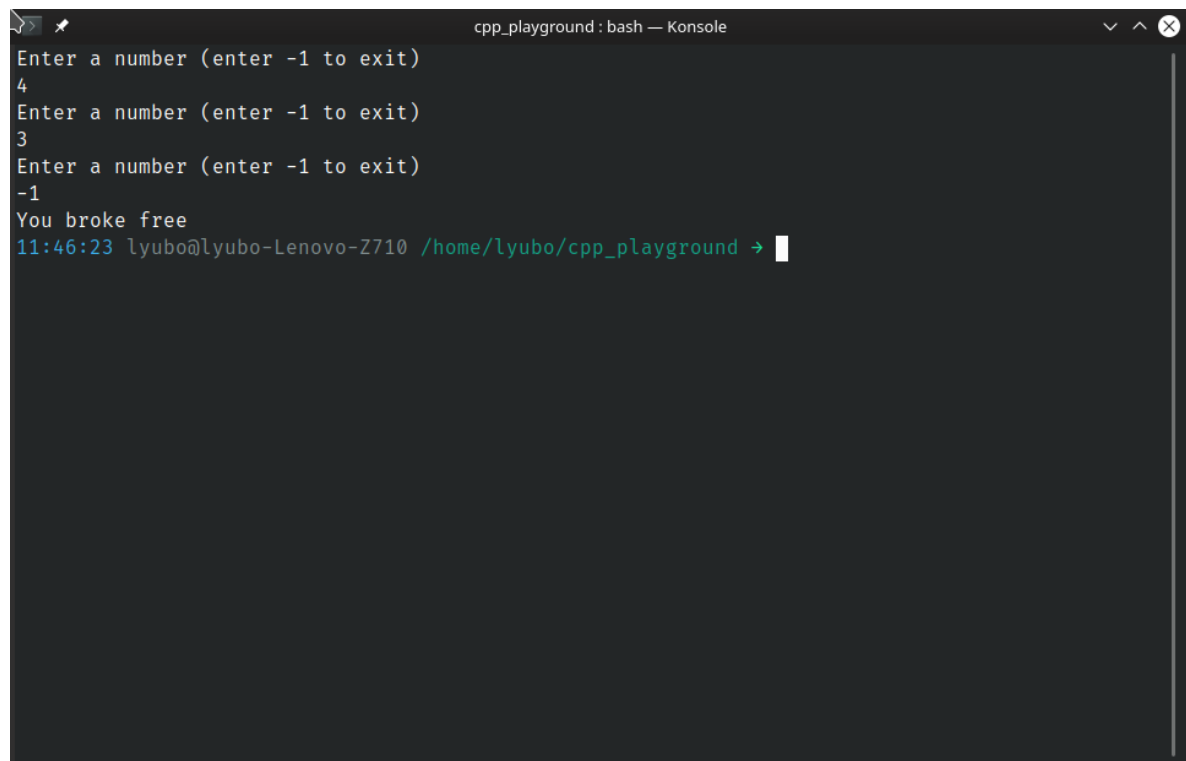
Оператор `break` прекъсва изпълнението на текущия блок.

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

int main()
{
    int user_input = 0;

    while(true)
    {
        cout << "Enter a number (enter -1 to exit)" << endl;
        cin >> user_input;

        if(user_input == -1)
        {
            break;
        }
    }
    cout << "You broke free" << endl;
    return 0;
}
```



```
cpp_playground : bash — Konsole
Enter a number (enter -1 to exit)
4
Enter a number (enter -1 to exit)
3
Enter a number (enter -1 to exit)
-1
You broke free
11:46:23 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

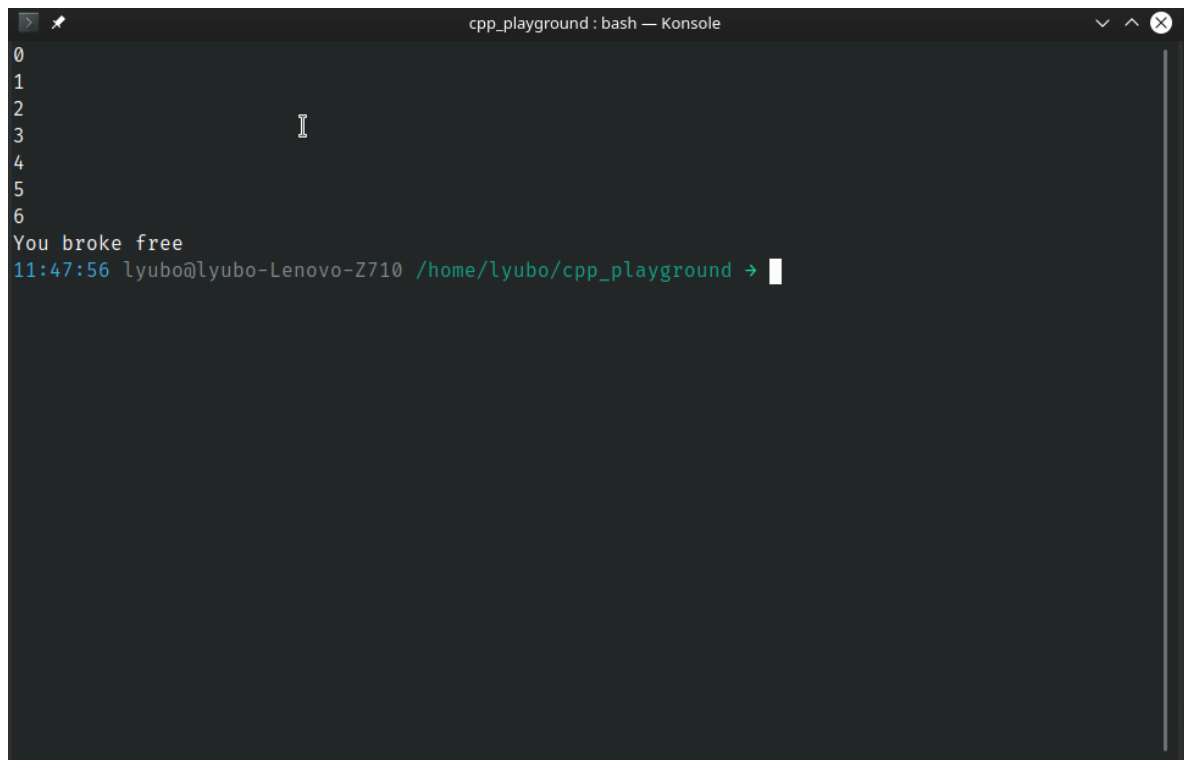
```
#include <iostream>
using namespace std;

int main()
```

```

{
    for(int i = 0; i < 10; i++)
    {
        if(i == 7)
        {
            break;
        }
        cout << i << endl;
    }
    cout << "You broke free" << endl;
    return 0;
}

```



```

cpp_playground : bash — Konsole
0
1
2
3
4
5
6
You broke free
11:47:56 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →

```

Оператор continue

Ако искаме да прескочим към следваща итерация на цикъл, без да довършваме изпълнението на текущия блок, може да използваме оператора `continue`

```

#include <iostream>
using namespace std;

int main()
{
    for(int i = 0; i < 10; i++)
    {
        if(i == 7)
        {
            continue;
        }
        cout << i << endl;
    }
    return 0;
}

```

```
cpp_playground : bash — Konsole
0
1
2
3
4
5
6
8
9
You broke free
06:52:40 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

Задачи:

1. Напишете програма, която извежда сумата на първите `n` естествени числа

```
cpp_playground : bash — Konsole
Enter N: 5
The sum of the first n numbers are: 15
01:48:36 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →

Size: 93 x 26
```

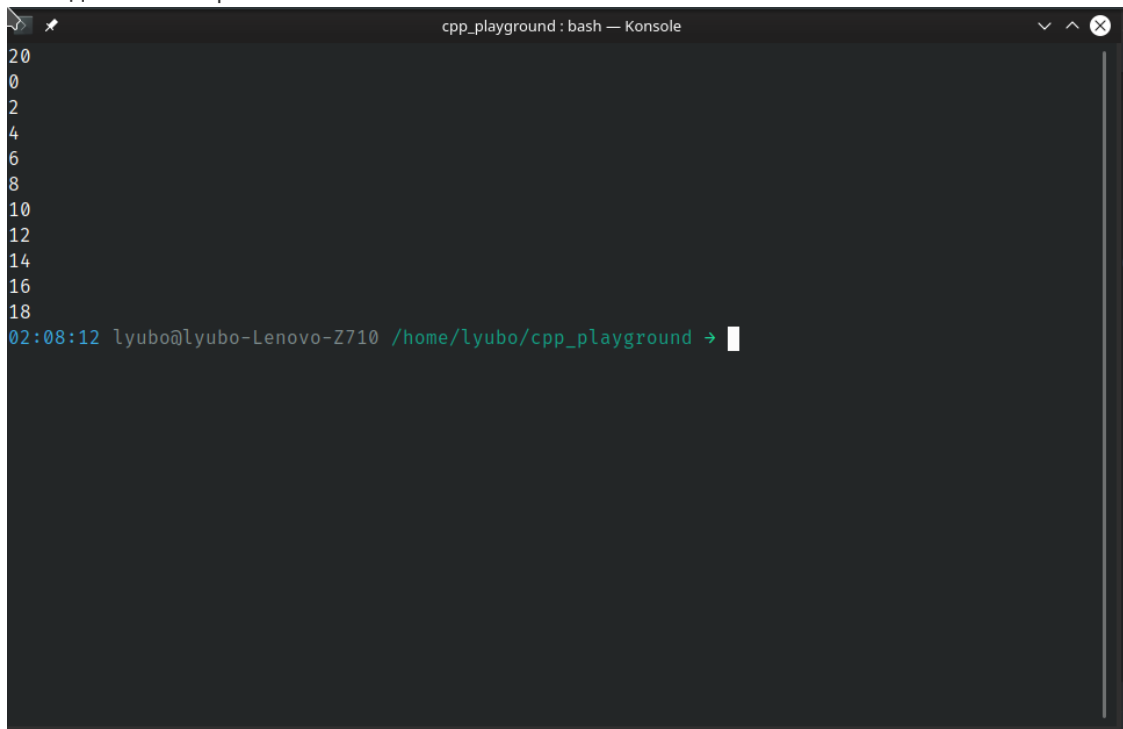
2. Напишете програма, която извежда на екрана квадрат съставен от символа `#`, с размер въведен от потребителя

```
cpp_playground : bash — Konsole
Enter N: 8
#####
#####
#####
#####
#####
#####
#####
#####
01:49:40 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

3. Напишете програма, в която потребителя въвежда естествени числа до въвеждане на -1. След приключване на въвеждането, да се изведат най-малкото, най-голямото и средната стойност на въведените числа.

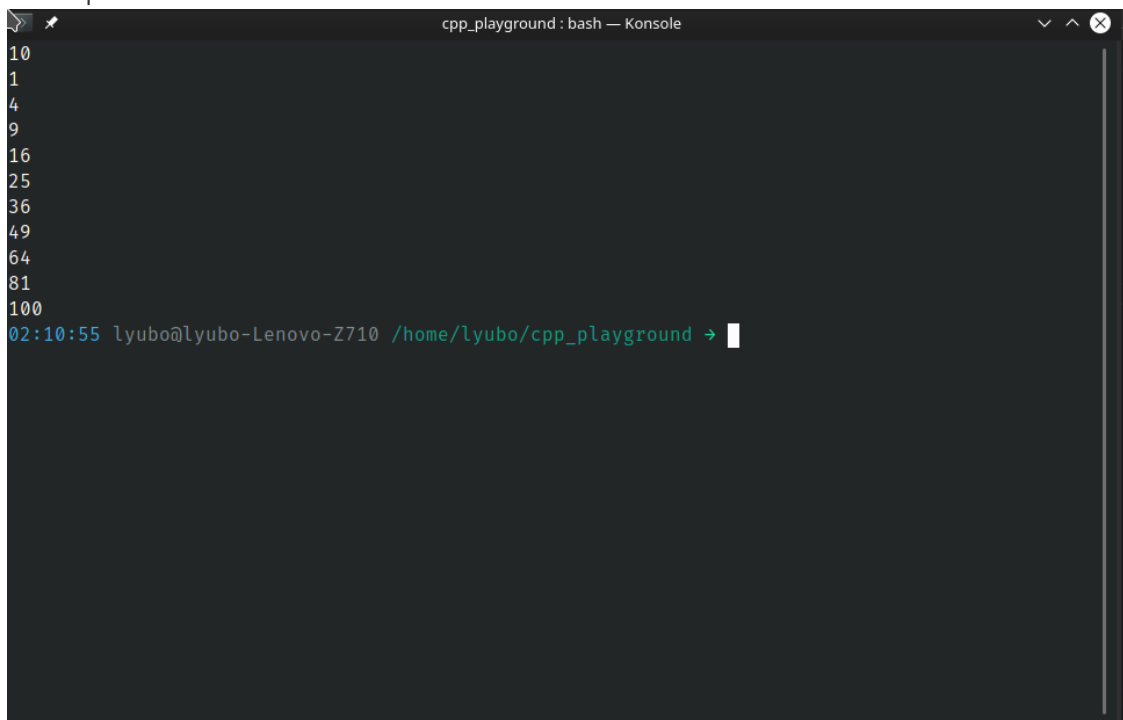
```
cpp_playground : bash — Konsole
Enter a number: 3
Enter a number: 4
Enter a number: 6
Enter a number: -1
Min: 3
Max: 6
Average: 4.33333
01:59:18 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

4. Напишете програма, която извежда всички четни числа от 1 до `n`, където `n` е въведено от потребителя



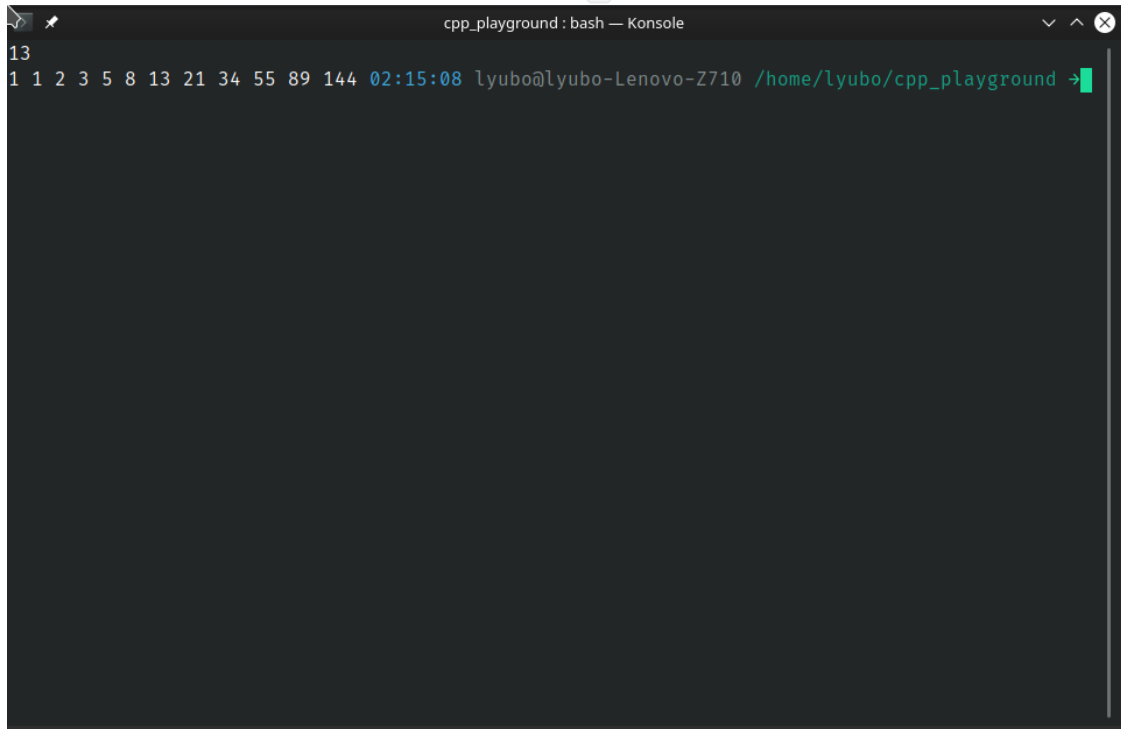
```
cpp_playground : bash — Konsole
20
0
2
4
6
8
10
12
14
16
18
02:08:12 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

5. Напишете програма, която извежда квадратите числа от 1 до `n`, където `n` е въведено от потребителя



```
cpp_playground : bash — Konsole
10
1
4
9
16
25
36
49
64
81
100
02:10:55 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

6. Напишете програма, която извежда първите n числа на Фибоначи



```
cpp_playground : bash — Konsole
13
1 1 2 3 5 8 13 21 34 55 89 144 02:15:08 lyubo@lyubo-Lenovo-Z710 /home/lyubo/cpp_playground →
```

7. Напишете примитивна версия на играта "Познай числото" - първия играч въвежда число от клавиатурата. Втория играч има 5 опита да познае числото на първия потребител. Ако числото на втория потребител е по-малко или по-голямо, да се извежда подходящо съобщение.
