

Увод в програмирането

5: Инструкции
доц. Атанас Семерджиев

Съдържание

- Инструкция-израз (expression statement)
- Съставна инструкция (compound statement)
- Инструкции за избор (selection statements)
 - if-then-else, switch
- Инструкции за цикъл (iteration statements)
 - while, do...while, for
- Инструкции за преход (jump statements)
 - continue, break, goto

Инструкция (statement)

- Инструкциите в C++ определят:
 - Как се манипулират обектите.
 - В какъв ред се манипулират обектите.

3

Инструкция-израз (Expression Statement)

- Общ вид:
 - `<expression-statement> ::= [<expression>] ;`
- В този тип изрази не се прехвърля контрола, нито се извършва итерация.
- Всички изрази в инструкцията-израз се оценяват и техните странични ефекти се прилагат, преди да се премине към следващата инструкция.

4

Инструкция-израз (Expression Statement)

```
int Value;  
  
cout << "Enter an integer: ";  
  
cin >> Value;  
  
1 + 2;  
  
Value = Value * 100;  
  
; // Празната инструкция (null statement)
```

5

Съставна инструкция (Compound Statement) / Блок (Block)

- Състои се от нула или повече инструкции, оградени във фигурни скоби.

```
if(Value > 0)  
{  
    cout << Value << " is positive.\n";  
    int Result = Value * 100;  
}
```

6

Инструкции за избор (selection statements)

7

if-then-else

Общ вид:

```
if ( <условие> )  
    <if-инструкция>  
[ else  
    <else-инструкция> ]opt
```

Семантика:

1. Оценява се <условие>
 - A. Ако то е истина, изпълнява се <if-инструкция>
 - B. В противен случай, ако имаме else израз изпълняваме <else-инструкция>

8

```
int Value = 0;

cout << "Enter an integer: ";
cin >> Value;

if(Value > 0)
{
    cout << Value << " is positive.\n";
}

// Когато имаме само един statement в тялото на if,
// фигурните скоби могат да се пропуснат:
if(Value > 0)
    cout << Value << " is positive.\n";
```

9

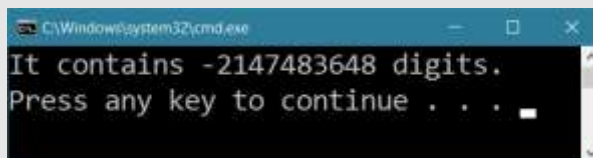
```
// Тъй като тялото на if трябва да се състои от точно
// един statement, ако искаме да се изпълнят няколко
// неща, трябва да ги групираме в блок
// (или казано по-просто, тогава фигурните скоби са
// задължителни):
if(Value > 0)
{
    cout << Value << " is positive.\n";
    cout << "It contains "
        << (int)log10((double)Value) + 1)
        << " digits.\n";
}
```

10

```
int Value = 0;
```

```
// Пропускането на скобите е грешка!  
// Този код ще работи, но не както искаме.
```

```
if (Value > 0)  
    cout << Value << " is positive.\n";  
    cout << "It contains "  
        << (int)(log10((double)Value) + 1)  
        << " digits.\n";
```



11

```
int Value = 0;
```

```
cout << "Enter an integer: ";  
cin >> Value;
```

```
if (Value > 0)  
{  
    cout << Value << " is positive.\n";  
}  
else  
{  
    cout << Value << " is NOT positive!\n";  
}
```

12

```
// if може да се вложи в друг if

if (Value > 0)
{
    cout << Value << " is positive.\n";
}
else
{
    if (Value == 0)
        cout << Value << " is equal to zero.\n";
    else
        cout << Value << " is negative.\n";
}
```

13

```
// По-компактен запис

if (Value > 0)
    cout << Value << " is positive.\n";

else if (Value == 0)
    cout << Value << " is equal to zero.\n";

else
    cout << Value << " is negative.\n";
```

14

Видимост на променливите

```
int Variable = 100;

if (true)
{
    int Local = 100;
}

cout << "Variable = " << Variable << endl; // ОК
cout << "Local = " << Local << endl;       // Грешка!
```

15

Оператор switch

Общ вид:

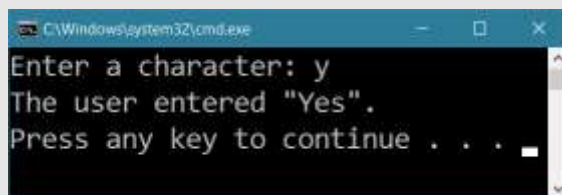
```
switch ( <израз> )
{
    case <константен-израз-1>:
        <case-израз-1>
    ...
    case <константен-израз-N>:
        <case-израз-N>
    [ default:
        <default-израз> ]opt
}
```

Семантика:

1. Оценява се израз
2. Оценката се сравнява с константните изрази
 - A. Ако е равна на някой от тях, влиза се в първия case-израза за първия такъв;
 - B. Ако не е равна на никой от тях, влиза се в default-израза, ако има такъв
 - C. В противен случай не се прави нищо.

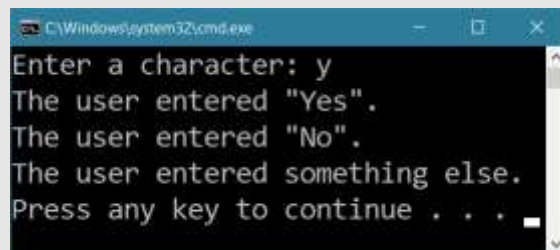
16


```
cout << "Enter a character: ";  
char Input;  
cin >> Input;  
  
switch (Input)  
{  
case 'y':  
    cout << "The user entered \"Yes\".\n";  
    break;  
  
case 'n':  
    cout << "The user entered \"No\".\n";  
    break;  
  
default:  
    cout << "The user entered something else.\n";  
}
```



17

```
cout << "Enter a character: ";  
char Input;  
cin >> Input;  
  
switch (Input)  
{  
case 'y':  
    cout << "The user entered \"Yes\".\n";  
  
case 'n':  
    cout << "The user entered \"No\".\n";  
  
default:  
    cout << "The user entered something else.\n";  
}
```



18

```
cout << "Enter a character: ";
char Input;
cin >> Input;

switch (Input)
{
case 'y':
case 'Y':
    cout << "The user entered \"Yes\".\n";
    break;

case 'n':
case 'N':
    cout << "The user entered \"No\".\n";
    break;

default:
    cout << "The user entered something else.\n";
}
```

19

```
cout << "Enter a character: ";
char Input;
cin >> Input;

if (Input == 'y' || Input == 'Y')
    cout << "The user entered \"Yes\".\n";

else if (Input == 'n' || Input == 'N')
    cout << "The user entered \"No\".\n";

else
    cout << "The user entered something else.\n";
```

20

Итеративни инструкции (Iterative Statements) / Цикли (Loops)

21

Цикъл while

Общ вид

```
while( <условие> )  
    <тяло-на-цикъла>
```

Семантика

1. Оценява се <условие>
 - A. Ако то е истина се изпълнява <тяло на цикъла> и след това преминаваме обратно на 1.
 - B. Ако то не е истина, изпълнението на цикъла се прекратява.

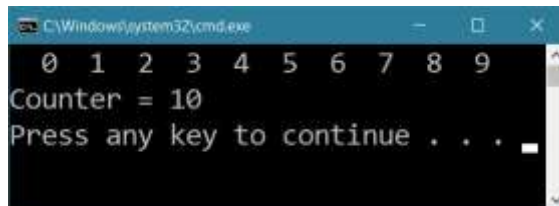
22

Цикъл while

```
int Counter = 0;

while (Counter < 10)
{
    cout << setw(3) << Counter;
    Counter++;
}

cout << endl;
cout << "Counter = " << Counter << endl;
```



23

Локални променливи



```
int Counter = 0;

while (Counter < 10)
{
    int Local = Counter * 10; // Променливата е локална за цикъла!
    cout << setw(3) << Local;
    Counter++;
}

cout << "\nCounter = " << Counter
    << "\nLocal = " << Local; // Грешка!
```

24

Безкраен цикъл



```
int Counter = 0;  
  
while (Counter < 10)  
    cout << Counter << endl;
```

25

Цикъл do...while

Общ вид

```
do  
    <тяло на цикъла>  
while( <условие> );
```

Семантика

1. Изпълнява се <тяло на цикъла>
2. Оценява се <условие>
 - А. Ако то е истина преминаваме обратно на 1.
 - В. Ако то не е истина, изпълнението на цикъла се прекратява.

26

Цикъл do...while

- За разлика от while, do...while гарантира поне едно преминаване през тялото на цикъла:

```
int Input = 0;

while(Input > 0)
{
    cout << "*";
    Input--;
}
```

```
int Input = 0;

do
{
    cout << "*";
    Input--;
} while(Input > 0);
```

27

```
int Value = 0;

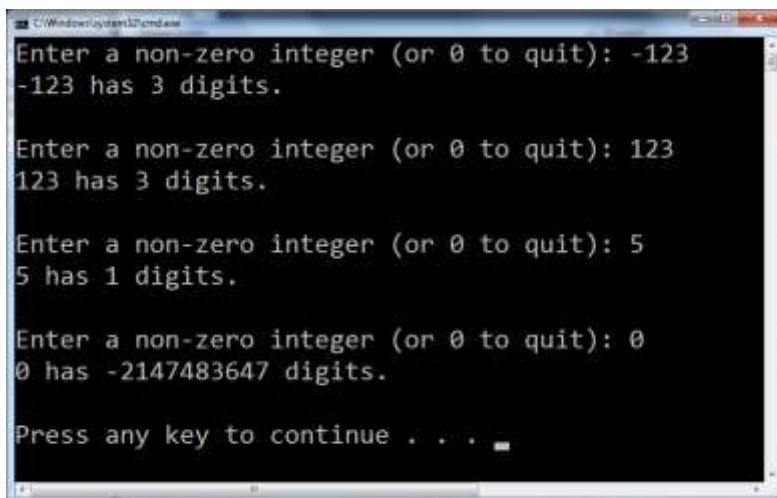
do
{
    cout << "Enter a non-zero integer (0 to end): ";
    cin >> Value;

    cout << Value << " has "
         << (int)(log10((double)abs(Value)) + 1)
         << " digits.\n\n";

} while (Value != 0);
```

28

Цикъл do...while



```
Enter a non-zero integer (or 0 to quit): -123
-123 has 3 digits.

Enter a non-zero integer (or 0 to quit): 123
123 has 3 digits.

Enter a non-zero integer (or 0 to quit): 5
5 has 1 digits.

Enter a non-zero integer (or 0 to quit): 0
0 has -2147483647 digits.

Press any key to continue . . .
```

29

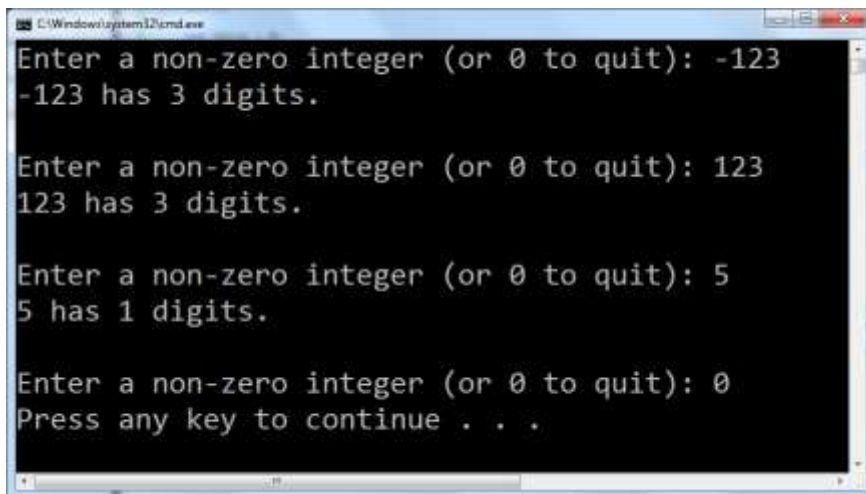
```
int Value = 0;

do
{
    cout << "Enter a non-zero integer (0 to end): ";
    cin >> Value;

    if (Value != 0)
    {
        cout << Value << " has "
              << (int)(log10((double)abs(Value)) + 1)
              << " digits.\n\n";
    }
} while (Value != 0);
```

30

Цикъл do...while



```
C:\Windows\system32\cmd.exe
Enter a non-zero integer (or 0 to quit): -123
-123 has 3 digits.

Enter a non-zero integer (or 0 to quit): 123
123 has 3 digits.

Enter a non-zero integer (or 0 to quit): 5
5 has 1 digits.

Enter a non-zero integer (or 0 to quit): 0
Press any key to continue . . .
```

31

Цикъл for

Общ вид

```
for(<инициализация>; <условие>; <стъпка>)
    <тяло а цикъла>
```

Семантика

1. Изпълнява се <инициализация>
2. Оценява се <условие>
 - A. Ако то е лъжа, прекратяваме изпълнението на цикъла.
 - B. Ако то е истина, преминаваме на 3.
3. Изпълнява се <тяло на цикъла>
4. Изпълнява се <стъпка>

32

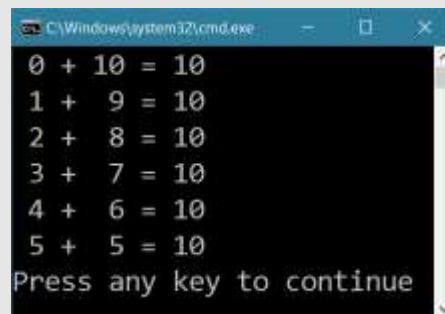

```
// Извежда числата от 0 до 9 на екрана
for (int Counter = 0; Counter < 10; Counter++)
{
    cout << Counter << endl;
}

// Еквивалентно решение с while
int Counter = 0;
while (Counter < 10)
{
    cout << Counter << endl;
    Counter++;
}
```

33

// Възможно е в клаузите да се включат няколко израза
// и/или да се създадат няколко променливи.

```
for (int Start = 0, End = 10;
     Start <= End;
     Start++, End--)
{
    cout << setw(2) << Start
         << " + "
         << setw(2) << End
         << " = "
         << (Start + End)
         << endl;
}
```



```
C:\Windows\system32\cmd.exe
0 + 10 = 10
1 + 9 = 10
2 + 8 = 10
3 + 7 = 10
4 + 6 = 10
Press any key to continue
```

34

Няколко бележки относно for

- Няма нещо, което да можете да направите с for и да не можете да направите с while и обратното.
- Обикновено избираме между двете възможности така, че кодът да бъде оформен по-добре.
- Когато използвате for, обикновено има различни начини да напишете един и същ итеративен процес, като пренасяте код между клаузите на цикъла и/или тялото му. На следващия слайд е даден пример за това.

35

```
for (int i = 0; i <= 10; i++)
    cout << i << endl;
//-----
int i = 0;
for (; i <= 10; i++)
    cout << i << endl;
//-----
int i = 0;
for (; i <= 10;)
    cout << i++ << endl;
//-----
int i = 0;
for (;;)
{
    if (i > 10) break;
    cout << i++ << endl;
}
//-----
for (int i = 0; i <= 10; cout << i << endl, i++)
    ;
```

36

Безкраен цикъл for

```
// Ако оставим и трите клаузи празни и в тялото  
// няма логика, която да го прекрати,  
// цикълът ще се изпълнява до безкрай  
int Counter = 0;  
for (;;)   
    cout << Counter << endl;
```

37

Jump statements (инструкции за преход)

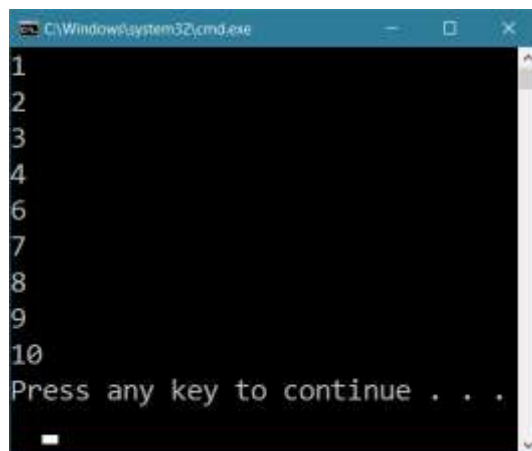
38

Оператор continue

```
int i = 0;

while (++i <= 10)
{
    if (i == 5)
        continue;

    cout << i << endl;
}
```



39

Оператор continue

Ако срещнем continue в **цикъл while**:

1. Оценява се <условие>
2. Ако <условие> е истина, изпълняваме следващата итерация на цикъла.

Ако срещнем continue в **цикъл for**:

1. Изпълнява се <стъпка>
2. Оценява се <условие>
3. Ако <условие> е истина, изпълняваме следващата итерация на цикъла.

40

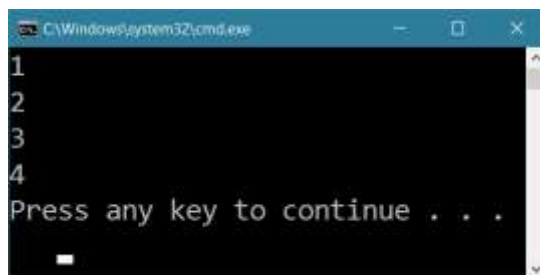
```
// Когато използвате continue, помислете дали не  
// предизвиквате безкраен цикъл, както е показано на  
// този пример
```

```
int i = 0;  
  
while (i <= 10)  
{  
    if (i == 5)  
        continue;  
  
    cout << i++ << endl;  
}
```

41

Оператор break

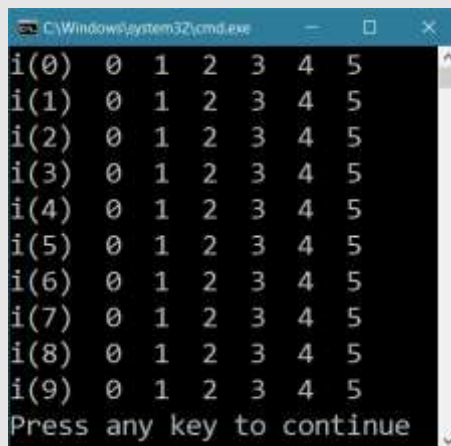
```
int i = 0;  
  
while (++i <= 10)  
{  
    if (i == 5)  
        break;  
  
    cout << i << endl;  
}
```



42

// При вложени цикли, break и continue влияят само
// върху цикъла, в който се намират

```
for (int i = 0; i < 10; i++)  
{  
    cout << "i(" << i << ")";  
  
    for (int j = 0; j < 10; j++)  
    {  
        if (j > 5)  
            break;  
        cout << setw(3) << j);  
    }  
  
    cout << endl;  
}
```



```
C:\Windows\system32\cmd.exe  
i(0) 0 1 2 3 4 5  
i(1) 0 1 2 3 4 5  
i(2) 0 1 2 3 4 5  
i(3) 0 1 2 3 4 5  
i(4) 0 1 2 3 4 5  
i(5) 0 1 2 3 4 5  
i(6) 0 1 2 3 4 5  
i(7) 0 1 2 3 4 5  
i(8) 0 1 2 3 4 5  
i(9) 0 1 2 3 4 5  
Press any key to continue
```

43

Оператор goto

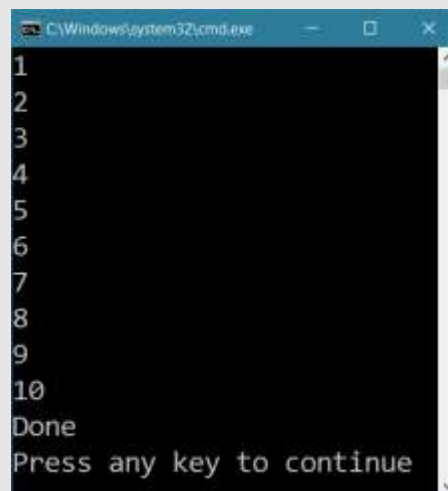
44

```
int main()
{
    int Counter = 1;

    LOOP_START:
        cout << Counter << endl;

        if (Counter < 10)
        {
            Counter++;
            goto LOOP_START;
        }

        cout << "Done\n";
}
```



45

Пример 1: Пресмятане на факториел

46

```
int Value = 0;

cout << "Enter an integer: ";
cin >> Value;

int Factoriel = 1;
int Counter = 1;

while (Counter <= Value)
{
    Factoriel *= Counter;
    Counter++;
}

cout << Value << "! = " << Factoriel << endl;
```

47

```
int Value = 0;

cout << "Enter an integer: ";
cin >> Value;

int Factoriel = 1;
int Counter = 1;

while (Counter <= Value)
{
    Factoriel *= Counter++;
}

cout << Value << "! = " << Factoriel << endl;
```

48


```
int Value = 0;

cout << "Enter an integer: ";
cin >> Value;

int Factoriel = 1;
int Counter = 0;

while (++Counter <= Value)
{
    Factoriel *= Counter;
}

cout << Value << "! = " << Factoriel << endl;
```

49

```
int Value = 0;

cout << "Enter an integer: ";
cin >> Value;

int Factoriel = 1;

for (int Counter = 1; Counter <= Value; Counter++)
{
    Factoriel *= Counter;
}

cout << Value << "! = " << Factoriel << endl;
```

50

```
int Value = 0;

cout << "Enter an integer: ";
cin >> Value;

int Factoriel; // Декларирана е извън цикъла, за да
               // може да се използва и след него

for (int Counter = 1, Factoriel = 1;
     Counter <= Value;
     Counter++, Factoriel *= Counter)
;

cout << Value << "! = " << Factoriel << endl;
```

51

```
int Value = 0;

cout << "Enter an integer: ";
cin >> Value;

int Factoriel = 1;
int Counter = 1;

LOOP:
Factoriel *= Counter;

if (++Counter <= Value)
    goto LOOP;

cout << Value << "! = " << Factoriel << endl;
```

52

Пример 2: генериране на таблицата за умножение

53

- Примерът илюстрира използването на няколко цикъла (вкл. вложени) за решаване на една задача.
- В кода има редица несъвършенства, които ще коригираме след лекцията за функции.

54

```
C:\Windows\system32\cmd.exe
The multiplication table:

#| 1 2 3 4 5 6 7 8 9 10
-----
1| 1 2 3 4 5 6 7 8 9 10
2| 2 4 6 8 10 12 14 16 18 20
3| 3 6 9 12 15 18 21 24 27 30
4| 4 8 12 16 20 24 28 32 36 40
5| 5 10 15 20 25 30 35 40 45 50
6| 6 12 18 24 30 36 42 48 54 60
7| 7 14 21 28 35 42 49 56 63 70
8| 8 16 24 32 40 48 56 64 72 80
9| 9 18 27 36 45 54 63 72 81 90
10| 10 20 30 40 50 60 70 80 90 100

Press any key to continue . . .
```

55

Стъпка 1: Генериране на таблицата

```
C:\Windows\system32\cmd.exe
The multiplication table:

1 2 3 4 5 6 7 8 9 10
2 4 6 8 10 12 14 16 18 20
3 6 9 12 15 18 21 24 27 30
4 8 12 16 20 24 28 32 36 40
5 10 15 20 25 30 35 40 45 50
6 12 18 24 30 36 42 48 54 60
7 14 21 28 35 42 49 56 63 70
8 16 24 32 40 48 56 64 72 80
9 18 27 36 45 54 63 72 81 90
10 20 30 40 50 60 70 80 90 100
```

56

```
cout << "The multiplication table:\n\n";

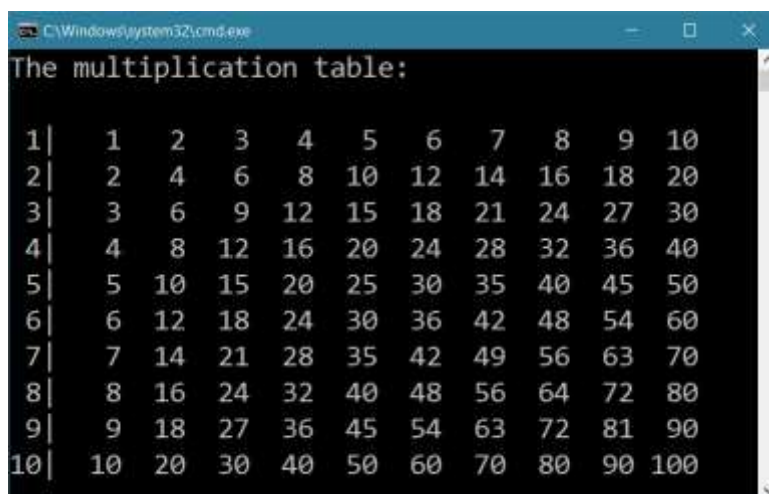
// Извеждаме таблицата
for (int lhs = 1; lhs <= 10; lhs++)
{
    for (int rhs = 1; rhs <= 10; rhs++)
        cout << setw(4) << (lhs * rhs);

    cout << endl;
}

cout << endl;
```

57

Стъпка 2: Добавяне на първата колона



1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30
4	4	8	12	16	20	24	28	32	36	40
5	5	10	15	20	25	30	35	40	45	50
6	6	12	18	24	30	36	42	48	54	60
7	7	14	21	28	35	42	49	56	63	70
8	8	16	24	32	40	48	56	64	72	80
9	9	18	27	36	45	54	63	72	81	90
10	10	20	30	40	50	60	70	80	90	100

58

```
cout << "The multiplication table:\n\n";

// Извеждаме таблицата
for (int lhs = 1; lhs <= 10; lhs++)
{
    cout << setw(2) << lhs << "|";

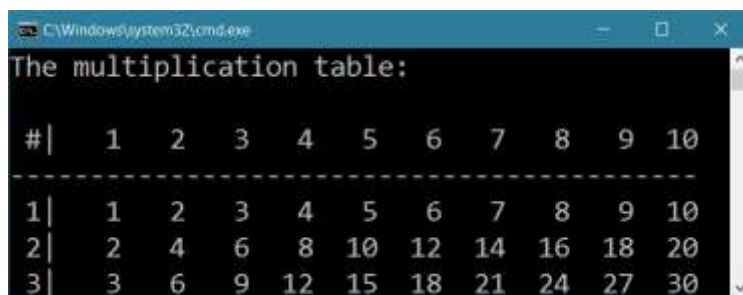
    for (int rhs = 1; rhs <= 10; rhs++)
        cout << setw(4) << (lhs * rhs);

    cout << endl;
}

cout << endl;
```

59

Стъпка 3: Генериране на заглавния ред



#	1	2	3	4	5	6	7	8	9	10
1	1	2	3	4	5	6	7	8	9	10
2	2	4	6	8	10	12	14	16	18	20
3	3	6	9	12	15	18	21	24	27	30

60

```
cout << "The multiplication table:\n\n";

// Извеждаме числата в заглавния ред
cout << " #|";

for (int number = 1; number <= 10; number++)
    cout << setw(4) << number;

cout << endl;

// Извеждаме линията под числата
for (int count = 1; count <= 3 + 10 * 4; count++)
    cout << "-";

cout << endl;
```



61

```
// Извеждаме таблицата
for (int lhs = 1; lhs <= 10; lhs++)
{
    cout << setw(2) << lhs << '|';

    for (int rhs = 1; rhs <= 10; rhs++)
        cout << setw(4) << (lhs * rhs);

    cout << endl;
}

cout << endl;
```



62