

Увод в програмирането

6: Функции

доц. Атанас Семерджиев

Съдържание

- Функции
- Декларации
- Област на видимост
- Предаване на параметри по стойност
- Предефиниране на функции
- Параметри по подразбиране

```
// Намиране на НОД на две неотрицателни цели числа

int a, b;

// ... задаваме стойности на двете променливи ...

while (a != b)
{
    if (a > b)
        a = a - b;
    else
        b = b - a;
}

cout << "GCD: " << a << endl;
```

3

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

int main()
{
    int a;
    int b;

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

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



4



```
while (a != b)
{
    if (a > b)
        a = a - b;
    else
        b = b - a;
}

cout << "GCD: " << a << endl;

return 0;
} // край на main()
```

5

```
long gcd(long a, long b)
{
    while (a != b)
    {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }

    return a;
}
```

6

```
int main()
{
    int a;
    int b;

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

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

    long Result = gcd(a, b);

    cout << "GCD: " << Result << endl;

    return 0;
}
```

7

```
int main()
{
    int a;
    int b;

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

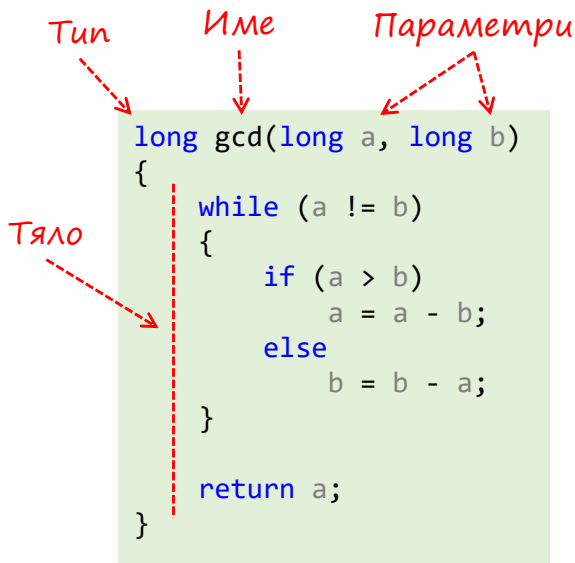
    cout << "Enter an integer: ";
    cin >> b;

    cout << "GCD: " << gcd(a, b) << endl;

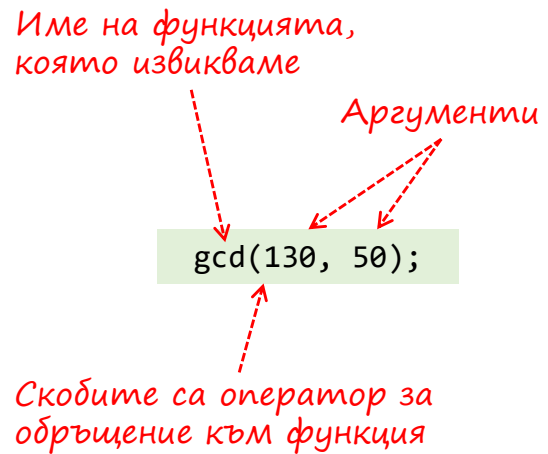
    return 0;
}
```

8

Дефиниция на функция



Обръщение към функция (извикване)



9

Разлика между параметър и аргумент

Ако бъркате понятията параметър и аргумент, може да ви е полезна аналогията използвана в MSD:

*"You can think of the **P**arameter as a **P**arking space and the **A**rgument as an **A**utomobile."*

"Just as different automobiles can park in a parking space at different times, the calling code can pass a different argument to the same parameter every time that it calls the procedure."

Източник: <https://msdn.microsoft.com/en-us/library/9kewt1b3.aspx>

10

Функции без параметри

```
long PromptForLong(void)
{
    long Input;

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

    return Input;
}
```

11

Функции без параметри

```
long PromptForLong() // Пропускаме void
{
    long Input;

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

    return Input;
}
```

12

ФУНКЦИИ

```
int main()
{
    int a = PromptForLong();
    int b = PromptForLong();

    cout << "GCD: " << gcd(a, b) << endl;

    return 0;
}
```

13

```
// Една функция може и да не връща нищо
void PrintLine(unsigned int Size)
{
    for(int i=0; i < Size; i++)
    {
        cout << '-';
    }
}

int main()
{
    PrintLine(10); // ОК
    int x = PrintLine(20); // Грешка!
}
```

14

```
int MyFunction(int a)
{
    // Ако функцията има тип (в случая – int), то
    // всички пътища на изпълнение трябва да връщат
    // стойност от този тип!
    if (a > 0)
    {
        cout << "a is positive\n";
        return a;
    }
    else
    {
        cout << "a is negative\n";
    }
}
```

15

```
long gcd(long a, long b)
{
    if (a <= 0 || b <= 0)
        return 0;

    while (a != b)
    {
        if (a > b)
            a = a - b;
        else
            b = b - a;
    }

    return a;
}
```

16


```
// Предполагаме, че ReadPositiveIntFromFile()
// е функция, която прочита положително число от
// някакъв файл и връща -1 ако не успее.

void MyFunction()
{
    int Data = ReadPositiveIntFromFile();

    if (Data == -1)
    {
        cout << "ERROR: Cannot read from file!\n";
        return;
    }

    // Тук използваме прочетените стойности ...
}
```

17

```
int f()                // Дефиниция на f()
{
    return 10;
}

int main()
{
    int a = f();        // ОК - f() е дефинирана по-горе
    int b = g(5, 10);  // Грешка! Не е ясно какво е g?
}

int g(int a, int b)    // Дефиниция на g()
{
    return a + b * 100;
}
```

18

```
int f()                // Дефиниция на f()
{
    return 10;
}

int g(int a, int b)    // Дефиниция на g()
{
    return a + b * 100;
}

int main()
{
    int a = f();        // ОК - f() е дефинирана по-горе
    int b = g(5, 10);   // ОК - g() е дефинирана по-горе
}
```

19

```
int f();               // Декларация на f()
int g(int a, int b);   // Декларация на g()

int main()
{
    int a = f();        // ОК - функцията f() е декларирана по-горе
    int b = g(5, 10);   // ОК - функцията g() е декларирана по-горе
}

int f()                // Дефиниция на f()
{
    return 10;
}

int g(int a, int b)    // Дефиниция на g()
{
    return a + b * 100;
}
```

20

```
int f();  
int g(int, int); // В дефиницията можем да пропуснем  
                // имената на параметрите  
  
int main()  
{  
    int a = f();  
    int b = g(5, 10);  
}  
  
int f()  
{  
    return 10;  
}  
  
int g(int a, int b)  
{  
    return a + b * 100;  
}
```

21

Важна разлика между C и C++

Дадената по-долу декларация означава различни неща в C и C++:

```
int MyFunction();
```

C: Функция с неопределен, но фиксиран, брой параметри

C++: Функция без параметри

22

// Област на видимост

```
void f()
{
    int b = 1000;
    a = 2000; // Грешка!
}
```

Област на
видимост на *b*

```
int main()
{
    int a = 1;
    f();
    b = 2; // Грешка!
    return 0;
}
```

Област на
видимост на *a*

23

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

```
int c;

void f()
{
    c = 200;
}

int main()
{
    c = 100;
    f();
    cout << c << endl;
    return 0;
}
```

Област на
видимост на *c*

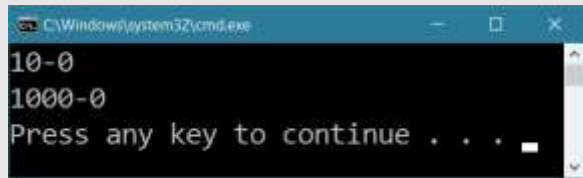
24

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

int a = 0;

void f()
{
    int a = 10;
    cout << a << "-" << ::a << endl; // Извежда 10-0
}

int main()
{
    int a = 1000;
    f();
    cout << a << "-" << ::a << endl; // Извежда 1000-0
}
```

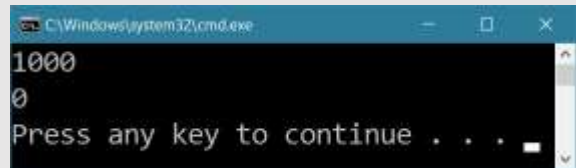


```
C:\Windows\system32\cmd.exe
10-0
1000-0
Press any key to continue . . .
```

25

```
// Предаване на параметри по стойност
// Функцията работи върху копие на оригиналните данни!
void f(int a)
{
    a = 1000;
    cout << a << endl;
}

int main()
{
    int a = 0;
    f(a);
    cout << a << endl;
    return 0;
}
```



```
C:\Windows\system32\cmd.exe
1000
0
Press any key to continue . . .
```

26

```
// Предефиниране на функция (overload)

void MyFunction(int a)
{
    cout << "MyFunction (int)\n";
}

void MyFunction(double a)
{
    cout << "MyFunction (double)\n";
}

double MyFunction(int a, double b)
{
    cout << "MyFunction (int, double)\n";
    return (double)a * 10.0 + b;
}
```

27

```
int a = 5;
double b = 10;

MyFunction(a);           // MyFunction(int)
MyFunction(b);           // MyFunction(double)
MyFunction(a, b);        // MyFunction(int, double)

MyFunction(10);          // MyFunction(int)

MyFunction(10.0);        // MyFunction(double)
MyFunction((double)10);  // MyFunction(double)

MyFunction(10, 10.0);    // MyFunction(int, double)
MyFunction(10.0, 10);    // MyFunction(double), WARNING!
```

28

```
// Функция НЕ МОЖЕ да се предефинира само по нейния тип.  
// Трябва да има и разлика в броя или вида на аргументите.
```

```
int MyFunction(int a)  
{  
    cout << "MyFunction(int)\n";  
}
```



```
double MyFunction(int a)  
{  
    cout << "MyFunction(double)\n";  
}
```

```
int main()  
{  
    MyFunction(10); // Един пример защо това не е позволено.  
                   // Коя версия бихме извикали тук?  
}
```

29

Параметри по подразбиране

```
int MyFunction(int a, int b = 0)  
{  
    return a + b;  
}  
  
int main()  
{  
    int x = MyFunction(10, 20); // x = 30;  
    int y = MyFunction(10);     // y = 10;  
    int z = MyFunction();       // Грешка!  
}
```

30



// Подразбиращите се параметри трябва винаги
// да бъдат в края!

```
int MyFunction(int a = 0, int b) // Грешка!  
{  
    return a + b;  
}
```

31



```
int MyFunction(int a = 10, int b = 10)  
{  
    return a + b;  
}  
  
int MyFunction()  
{  
    return 0;  
}  
  
int main()  
{  
    int x = MyFunction(); // Грешка!  
}
```

32


```
// Редът на оценяване на аргументите на една функция
// е неопределен! (unspecified behavior)
int PrintAndReturn(int value)
{
    cout << value << endl;
    return value;
}

int Sum(int a, int b)
{
    return a + b;
}

int main()
{
    // Какво ще се изведе?
    Sum(PrintAndReturn(10), PrintAndReturn(20));
}
```

33

```
// ВАЖНО: Внимавайте, когато използвате изрази със страничен
// ефект като аргументи на функция, защото можете да предизвикате
// undefined behavior! Това е показано долу, когато два пъти
// променяме една и съща променлива и то с постфиксно
// инкрементиране.
```

```
void PrintTwo(int a, int b)
{
    cout << a << " " << b << endl;
}

int main()
{
    int i = 5;
    PrintTwo(i++, i++); // Undefined behavior!
                        // Например може да инкрементира i два пъти или
                        // пък и двата изрази да работят върху i==5
}
```

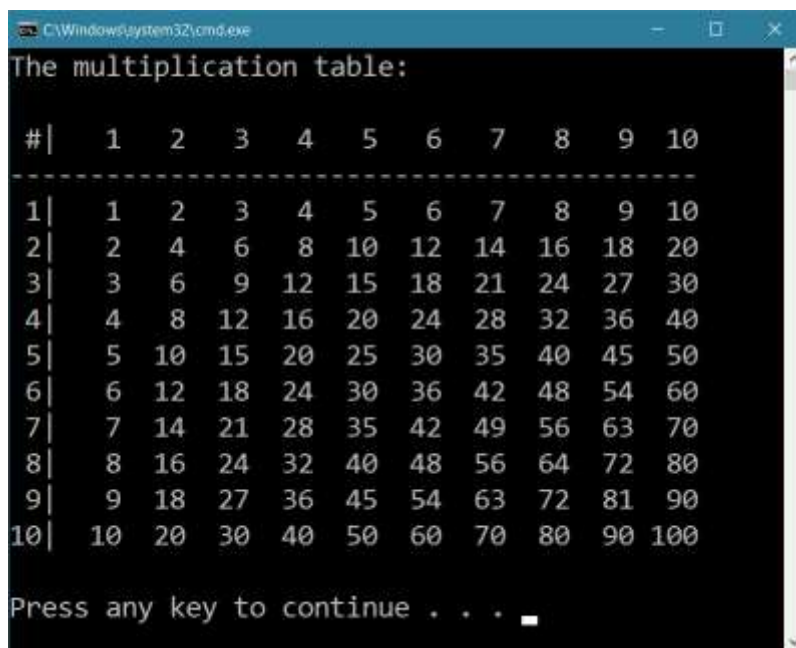
34

Пример: подобряване на таблицата за умножение

35

- В този пример ще подобрим приложението генериращо таблица за умножение.
- Ще разпределим кода във функции, като всяка ще отговаря за различна задача.

36



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The text inside the window is as follows:

```
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 . . .
```

37

Кодът от предишния път

38

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

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

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

std::cout << std::endl;

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

std::cout << std::endl;
```



39

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

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

    std::cout << std::endl;
}

std::cout << std::endl;
```



40

Подобряване на цикъла for

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

```
// Извеждаме линията под числата  
int LineLength = 3 + 10 * 4;  
  
for (int count = 1; count <= LineLength; count++)  
    std::cout << '-';
```

Тъй като изразът е константен, почти е сигурно, че компилаторът ще го оптимизира. Въпреки това, подходът по-долу ще ни свърши работа по-късно.

41

Премахване на „магически“ числа

42


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

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

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

std::cout << std::endl;
```

Какво е това 10?



43

```
// Въвеждаме константа за размер на таблицата
const int TABLE_SIZE = 10;

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

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

for (int number = 1; number <= TABLE_SIZE; number++)
    std::cout << std::setw(4) << number;

std::cout << std::endl;
```



44



// Извеждаме линията под числата

```
int LineLength = 3 + TABLE_SIZE * 4;
```

```
for (int count = 1; count <= LineLength; count++)  
    std::cout << '-';
```

```
std::cout << std::endl;
```



45



// Извеждаме таблицата

```
for (int lhs = 1; lhs <= TABLE_SIZE; lhs++)
```

```
{
```

```
    std::cout << std::setw(2) << lhs << '|';
```

```
    for (int rhs = 1; rhs <= TABLE_SIZE; rhs++)
```

```
        std::cout << std::setw(4) << (lhs * rhs);
```

```
    std::cout << std::endl;
```

```
}
```

46

```
const int TABLE_SIZE = 10;
```

```
// Въвеждаме константа за размер на клетка в таблицата
```

```
const int FIELD_SIZE = 4;
```

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

```
// Извеждаме числата в заглавния ред
```

```
std::cout << " #|";
```

```
for (int number = 1; number <= TABLE_SIZE; number++)
```

```
    std::cout << std::setw(FIELD_SIZE) << number;
```

```
std::cout << std::endl;
```



47



```
// Извеждаме линията под числата
```

```
int LineLength = 3 + TABLE_SIZE * FIELD_SIZE;
```

```
for (int count = 1; count <= LineLength; count++)
```

```
    std::cout << '-';
```

```
std::cout << std::endl;
```



48



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

    for (int rhs = 1; rhs <= TABLE_SIZE; rhs++)
        std::cout << std::setw(FIELD_SIZE) << (lhs * rhs);

    std::cout << std::endl;
}

std::cout << std::endl;
```

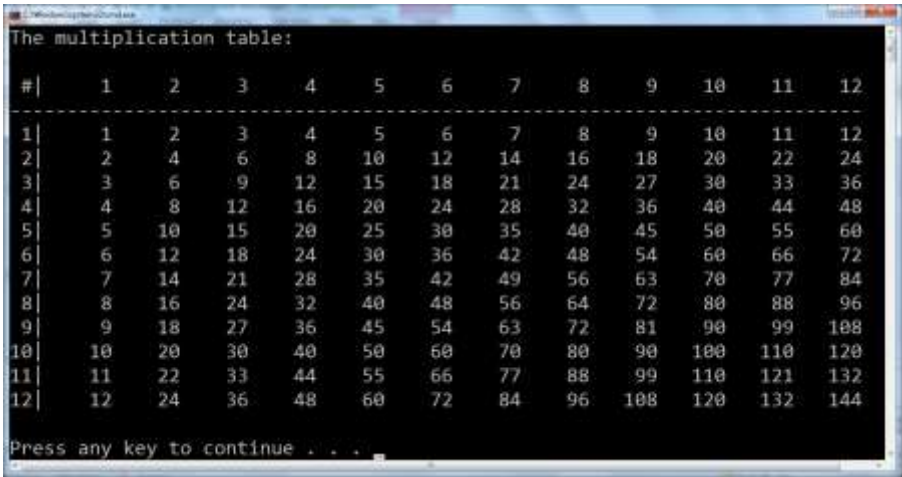
49

Въпрос

- Ако искаме да можем да генерираме таблица с произволен размер, ще ни върши ли работа фиксираният размер от 4 позиции за числата в таблицата?
- Защо фиксираме размерът на таблицата? Не е ли по-добре да оставим потребителя да избере такъв?

50

Размер 12x12 и увеличено поле



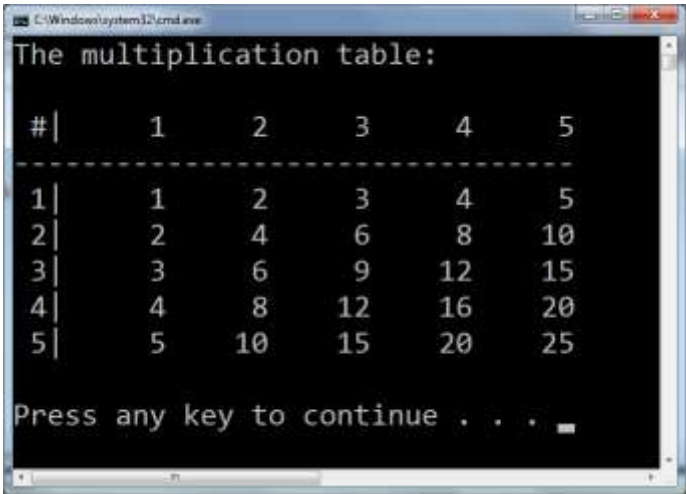
The multiplication table:

#	1	2	3	4	5	6	7	8	9	10	11	12
1	1	2	3	4	5	6	7	8	9	10	11	12
2	2	4	6	8	10	12	14	16	18	20	22	24
3	3	6	9	12	15	18	21	24	27	30	33	36
4	4	8	12	16	20	24	28	32	36	40	44	48
5	5	10	15	20	25	30	35	40	45	50	55	60
6	6	12	18	24	30	36	42	48	54	60	66	72
7	7	14	21	28	35	42	49	56	63	70	77	84
8	8	16	24	32	40	48	56	64	72	80	88	96
9	9	18	27	36	45	54	63	72	81	90	99	108
10	10	20	30	40	50	60	70	80	90	100	110	120
11	11	22	33	44	55	66	77	88	99	110	121	132
12	12	24	36	48	60	72	84	96	108	120	132	144

Press any key to continue . . .

51

Размер 5x5 и запазено поле



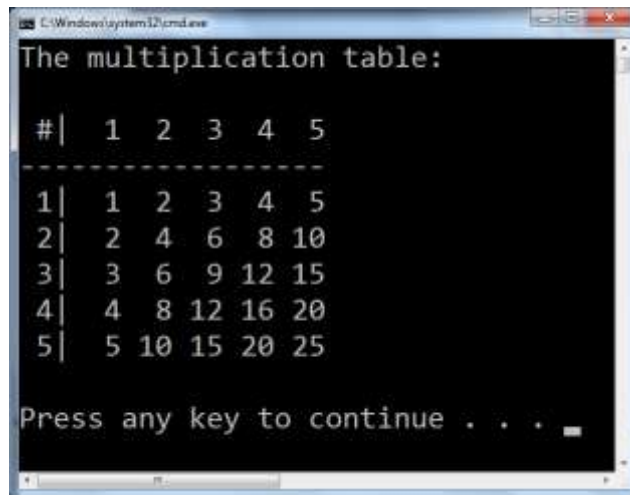
The multiplication table:

#	1	2	3	4	5
1	1	2	3	4	5
2	2	4	6	8	10
3	3	6	9	12	15
4	4	8	12	16	20
5	5	10	15	20	25

Press any key to continue . . .

52

Размер 5x5 и намалено поле



The screenshot shows a Windows command prompt window titled "C:\Windows\system32\cmd.exe". The text inside the window is as follows:

```
The multiplication table:  
  
#| 1 2 3 4 5  
-----  
1| 1 2 3 4 5  
2| 2 4 6 8 10  
3| 3 6 9 12 15  
4| 4 8 12 16 20  
5| 5 10 15 20 25  
  
Press any key to continue . . .
```

53

Контрол на входа

```
int TableSize = 0;

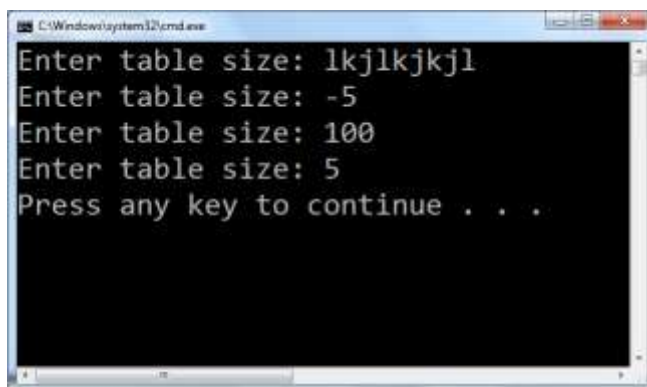
// Възможно е потребителят да въведе невалидна стойност
std::cout << "Enter table size: ";
std::cin >> TableSize;

// -----
int TableSize = 0;

// Цикълът по-долу опитва да реши този проблем,
// но не се държи коректно, ако въведем нещо различно от число
do
{
    std::cout << "Enter table size: ";
    std::cin >> TableSize;
} while (TableSize <= 1 || TableSize >= 12);
```

55

Какво бихме искали да прави
приложението



56

```
// Почистването на входния буфер НЕ Е тривиален проблем.  
// Даденото по-долу решение може да сработи,  
// но не е задължително. За повече информация:  
// https://www.daniweb.com/programming/software-development/threads/90228/flushing-the-input-stream  
do  
{  
    std::cin.clear();  
    std::cin.sync();  
    std::cout << "Enter table size: ";  
    std::cin >> TableSize;  
  
} while (TableSize <= 1 || TableSize >= 12);
```

57

Ширина на едно поле

- Формула:

Ширина = Максимален размер на число +
Отстъп между числата

```
int MaxNumberLength =  
    (int) log10((double)(TableSize * TableSize)) + 1;  
  
const int FIELD_MARGIN = 2;  
  
int FieldSize = MaxNumberLength + FIELD_MARGIN;
```

58

Функции

Брой цифри в запис на число

```
int NumberSize(int Number)
{
    int Absolute = (Number >= 0) ?
        Number :
        -Number;

    return (int)log10((double)Absolute) + 1;
}
```

60

```
// Често една и съща функционалност може
// да се имплементира по различни начини
int NumberSize(int Number)
{
    int Size = 1;
    int Current = (Number >= 0) ? Number : -Number;

    while (Current > 9)
    {
        Current = Current / 10;
        Size++;
    }

    return Size;
}
```

61

Реализиране на входа

```
// Въвеждане на число в даден интервал
int PromptForInteger(int LowerLimit, int UpperLimit)
{
    int Input = 0;

    do
    {
        std::cout << "Enter an integer between "
                    << LowerLimit << " and "
                    << UpperLimit << ": ";
        std::cin >> Input;

    } while (!std::cin.good() ||
             Input < LowerLimit ||
             Input > UpperLimit);

    return Input;
}
```

63

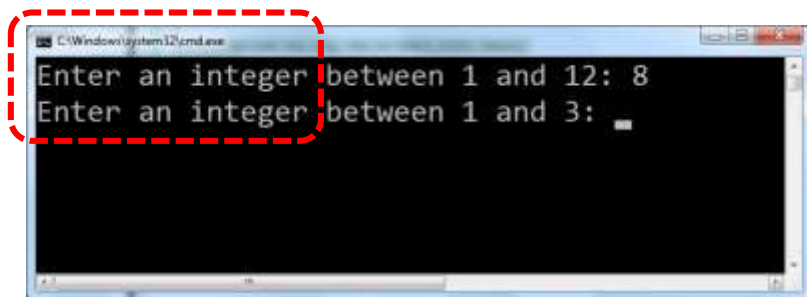
```
int main()
{
    int TableSize = PromptForInteger(1, 12);
    int FieldMargin = PromptForInteger(1, 3);

    // Това е най-голямото число, което ще се
    // появи в таблицата.
    int MaxNumber = TableSize * TableSize;

    int FieldSize = NumberSize(MaxNumber) +
                    FieldMargin;
}
```

64

Какво точно се въвежда???



65

```
int PromptForInteger(const char* Name, int LowerLimit, int UpperLimit)
{
    int Input;

    do
    {
        std::cout << "Enter " << Name
                  << " (between " << LowerLimit
                  << " and " << UpperLimit << "): ";

        std::cin >> Input;

    } while (!std::cin.good() ||
             Input < LowerLimit ||
             Input > UpperLimit);

    return Input;
}
```

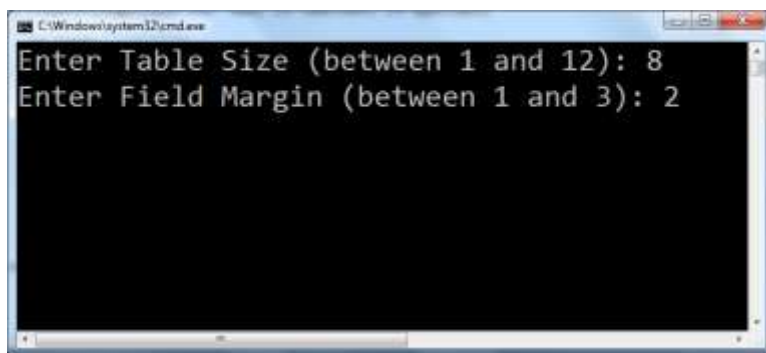
66

```
int main()
{
    int TableSize = PromptForInteger("Table Size", 1, 12);
    int FieldMargin = PromptForInteger("Field Margin", 1, 3);

    // Това е най-голямото число, което ще се появи в таблицата.
    int MaxNumber = TableSize * TableSize;

    int FieldSize = NumberSize(MaxNumber) +
        FieldMargin;
```

67



68

Други помощни функции

Извеждане на линия

```
void PrintLine(int Length)
{
    for (int i = 0; i < Length; i++)
        std::cout << '-';

    std::cout << std::endl;
}
```

70

Фрагмент от код извеждащ заглавния ред в таблицата

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



```
for (int number = 1; number <= 10; number++)  
    std::cout << std::setw(4) << (1 * number);
```

Фрагмент от код извеждащ един ред в таблицата

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

Извеждане на един ред

```
void PrintNumbers(int Start, int End,  
                 int Factor, int FieldSize)  
{  
    for (int i = Start; i <= End; i++)  
        std::cout << std::setw(FieldSize) << (i * Factor);  
  
    std::cout << std::endl;  
}
```

```
// Извеждаме заглавния ред
std::cout << " #|";
PrintNumbers(1, TableSize, 1, FieldSize);
PrintLine(3 + TableSize * FieldSize);

// Извеждаме таблицата
for (int row = 1; row <= TableSize; row++)
{
    std::cout << std::setw(2) << row << '|';
    PrintNumbers(1, TableSize, row, FieldSize);
}
std::cout << std::endl;
```

```
void PrintTable(int TableSize, int FieldMargin)
{
    int FieldSize = NumberSize(TableSize * TableSize) + FieldMargin;

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

    // Извеждаме заглавния ред
    std::cout << " #|";
    PrintNumbers(1, TableSize, 1, FieldSize);
    PrintLine(3 + TableSize * FieldSize);

    // Извеждаме таблицата
    for (int row = 1; row <= TableSize; row++)
    {
        std::cout << std::setw(2) << row << '|';
        PrintNumbers(1, TableSize, row, FieldSize);
    }

    std::cout << std::endl;
}
```

74

Окончателен вид на main()

```
int main()
{
    int TableSize = PromptForInteger("Table Size", 1, 12);

    int FieldMargin = PromptForInteger("Field Margin", 1, 3);

    PrintTable(TableSize, FieldMargin);

    return 0;
}
```

75