

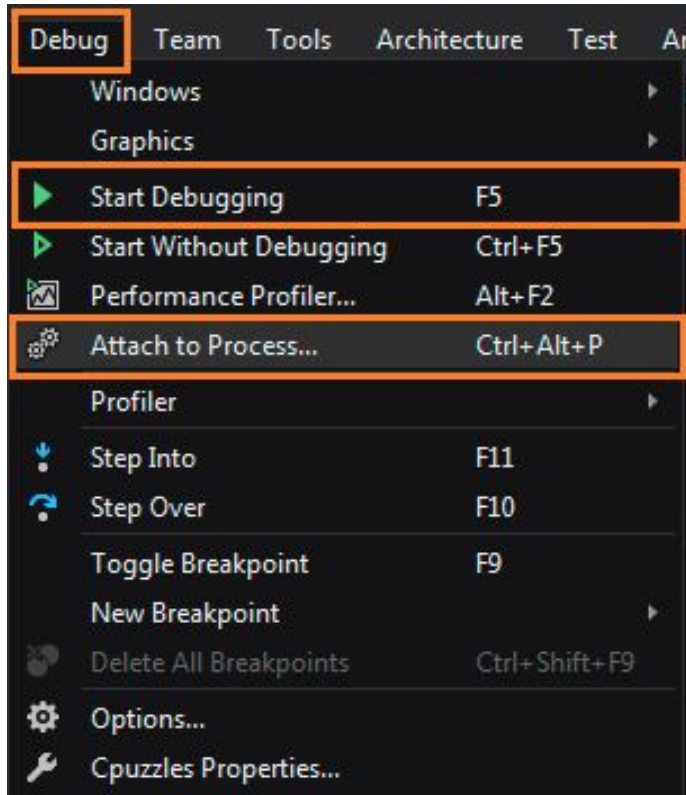
# Отладка в Visual Studio

# Отладка

Существуют две технологии отладки:

- Использование отладчиков — программ, которые включают в себя пользовательский интерфейс для пошагового выполнения программы: оператор за оператором, функция за функцией, с остановками на некоторых строках исходного кода или при достижении определённого условия.
- Вывод текущего состояния программы с помощью расположенных в критических точках программы операторов вывода — на экран или в файл (создание логов).

# Как начать



Нажать F5.  
Отладка начнется  
если стоят точки  
останова (**breakpoints**)

# Точки останова (Breakpoints)

```
1  #include <stdio.h>
2
3  void main()
4  {
5      char *names[] = { "John", "Lisa", "Albert", "Lana"};
6
7      for(int i = 0; i < 4; i++)
8          puts(names[i]);
9  }
```



- Точки останова используются чтобы показать , где отладчику необходимо остановиться.
- Точка ставится кликом на сайдбар слева от исходного кода, либо нажатием на **F9**.
- Точки останова обычно ставятся там, где есть сомнения в корректности кода.

# Отладка с использованием точек останова (Debugging with Breakpoints)

- **Перешагнуть (Step Over) F10** – автоматически выполняет блок кода под курсором.
- **Зайти (Step Into) F11** – заходит в блок кода под курсором.
- **Выйти (Step Out) Shift + F11** - выходит из текущего блока.
- **Продолжить (Continue) F5** - переходит к следующей точке останова.

```
1  #include <stdio.h>
2  void function()
3  {
4      puts("Break Point in function()");
5  }
6
7  void main()
8  {
9      int num = 0;
10     function();
11     puts("We are in main()");
12 }
```

Нажать F10

```
1  #include <stdio.h>
2  void function()
3  {
4      puts("Break Point in function()");
5  }
6
7  void main()
8  {
9      int num = 0;
10     function();
11     puts("We are in main()");
12 }
```

Нажать F11

# Условные остановки (Conditional Breakpoint)



The screenshot shows a code editor with a C program. The code is as follows:

```
1 #include <stdio.h>
2
3 void main()
4 {
5     char *names[] = { "John", "Lisa", "Albert", "Lana"};
6
7     for(int i = 0; i < 4; i++)
8     {
9         printf("%s\n", names[i]);
10    }
```

A breakpoint is set on line 8. A context menu is open over the breakpoint, with the following options:

- Delete Breakpoint
- Disable Breakpoint Ctrl+F9
- Conditions... (highlighted with an orange border)
- Actions...
- Edit labels...
- Export...

- В циклах может обрабатываться большое количество данных.
- Условная остановка нужна чтобы остановить выполнение кода в нужном месте

# УСЛОВНЫЕ ОСТАНОВКИ (Conditional Breakpoint)

Location: Source1.cpp, Line: 8, Must match source

Conditions

Conditional Expression    Is true    **i==2** × Saved

[Add condition](#)

Location: Source1.cpp, Line: 8, Must match source

Conditions

Conditional Expression    Is true    **strcmp(names[i], "Albert") == 0** × Saved

[Add condition](#)

```
{  
    char *names[] = { "John",  
                    "Lisa", "Albert", "John" };  
    for(int i = 0; i < 4; i++)  
        puts(names[i]);  
}
```

```
{  
    char *names[] = { "John", "Lisa", "Albert", "John" };  
    for(int i = 0; i < 4; i++)  
        puts(names[i]);  
}
```

▶ names[i]    0x00f76b58 "Albert"

# Количество остановок (Breakpoint Hit Count)

The screenshot displays the Visual Studio IDE with a breakpoint set at line 8 of Source1.cpp. The code is as follows:

```
4 {  
5     char *names[] = { "John", "Lisa", "Albert", "Lana"};  
6  
7     for(int i = 0; i < 4; i++)  
8         puts(names[i]);
```

The Breakpoint Settings dialog shows the following configuration:

- Location: Source1.cpp, Line: 8, Must match source
- Conditions:  Hit Count = 3 (Current: 3 Reset) X Saved

The Autos window shows the current state of variables:

Name	Value	Type
i	2	int
names	0x002df8ec {0x00f76b30 "John", 0x00f76b50 "Lisa", 0x00f76b58 "Albert", 0x00f76b58 "Lana"}	char *[4]
names[i]	0x00f76b58 "Albert"	char *

The Breakpoints window shows the configured breakpoint:

Name	Labels	Condition	Hit Count
Source1.cpp, line 8	NAMES	(no condition)	when hit count is equal to 3 (currently 3)

Отслеживание сколько остановок отладчик сделает на конкретной точке  
останова



# Подсказки (Data Tip)

```
#include <stdio.h>

void main()
{
    char *names[] = { "John", "Lisa", "Albert", "Lana"};

    for(int i = 0; i < 10; i++)
        puts(names[i]); ≤ 2ms elapsed
}

i | 4

names[i] | 0xffffffff <Error reading characters of string.>
```

(names)[0]	↗	0x00366b30	"John"
(names)[1]	↗	0x00366b50	"Lisa"
(names)[2]	↗	0x00366b58	"Albert"
(names)[3]	↗	0x00366b60	"Lana"

- Можно через подсказки менять значения

# Окно просмотра данных (Watch Windows)

```
1  #include <stdio.h>
2
3  void main()
4  {
5      char names[5][10] = { "John", "Lisa", "Albert", "Lana"};
6
7      for(int i = 0; i < 4; i++)
8          puts(names[i]);
9  }
```

132 %

Locals

Name	Value	Type
i	0	int
names	0x0036fe58 {0x0036fe58 "John", 0x0036fe62 "Lisa", 0x0036fe6c "Albert", 0x0036fe76 "Lana", 0x0036fe80 ""}	char[5][10]
[0]	0x0036fe58 "John"	char[10]
[0]	74 'J'	char
[1]	111 'o'	char
[2]	104 'h'	char
[3]	110 'n'	char
[4]	0 '\0'	char
[5]	0 '\0'	char
[6]	0 '\0'	char
[7]	0 '\0'	char
[8]	0 '\0'	char
[9]	0 '\0'	char
[1]	0x0036fe62 "Lisa"	char[10]
[2]	0x0036fe6c "Albert"	char[10]
[3]	0x0036fe76 "Lana"	char[10]
[4]	0x0036fe80 ""	char[10]

Autos **Locals** Threads Modules

# Окно просмотра данных (Watch Windows)

```
#include <stdio.h>

void main()
{
    char names[5][10] = { "John", "Lisa", "Albert", "" };

    for(int i = 0; i < 4; i++)
        puts(names[i]);
}
```

Quick Actions and Refactorings... Ctrl+.  
Rename... Ctrl+R, Ctrl+R  
Generate Graph of Include Files  
Show Call Stack on Code Map Ctrl+Shift+'  
Surround With... Ctrl+K, Ctrl+S  
Peek Definition Alt+F12  
Go To Definition F12  
Go To Declaration Ctrl+Alt+F12  
Find All References  
View Call Hierarchy Ctrl+K, Ctrl+T  
Toggle Header / Code File Ctrl+K, Ctrl+O  
Breakpoint  
Add Watch  
Add Parallel Watch  
QuickWatch... Shift+F9  
Pin To Source  
Show Next Statement Alt+Num \*  
Step Into Specific  
Run To Cursor Ctrl+F10  
Run Flagged Threads To Cursor  
Set Next Statement Ctrl+Shift+F10  
Go To Disassembly  
Cut Ctrl+X  
Copy Ctrl+C  
Paste Ctrl+V  
Outlining

Value  
0x0036fe58 {0x0036fe58 "John"  
0x0036fe58 "John"  
4 'J'  
11 'o'  
04 'h'  
10 'n'  
'\0'  
'\0'  
'\0'  
'\0'  
'\0'  
0x0036fe62 "Lisa"  
0x0036fe6c "Albert"  
0x0036fe76 "Lana"  
0x0036fe80 ""

ads Modules

```
1 #include <stdio.h>
2
3 void main()
4 {
5     char names[5][10] = { "John", "Lisa", "Albert", "" };
6
7     for(int i = 0; i < 4; i++)
8         puts(names[i]);
9
10 }
```

132 %

Watch 1

Name	Value	Type
names[i]	0x0036fe58 "John"	char[10]
[0]	74 'J'	char
[1]	111 'o'	char
[2]	104 'h'	char
[3]	110 'n'	char
[4]	0 '\0'	char
[5]	0 '\0'	char
[6]	0 '\0'	char
[7]	0 '\0'	char
[8]	0 '\0'	char
[9]	0 '\0'	char

Autos Locals Watch 1 Threads Modules

# Оперативные изменения (Immediate Window)

```
1  #include <stdio.h>
2
3  void main()
4  {
5      char names[5][10] = { "John", "Lisa", "Albert", "Lana"};
6
7      for(int i = 0; i < 4; i++)
8          puts(names[i]);
9
10 }
```

132 %

Watch 1		
Name	Value	Type
names[i]	0x0031fc86 "Lana"	char[10]
i	3	int

Autos Locals Watch 1 Threads Modules

Immediate Window

i=5  
5  
i=6  
6  
i=3  
3

Immediate Window Call Stack Bre

- **Debug > Window > Immediate Window**
- Позволяет задавать значения выражений/переменных во время отладки