

«ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԻՄՈՒՆՔՆԵՐ» դասընթաց

այլ ոլորտներից դեպի տեխնոլոգիական ոլորտ
սկսնականների համար



edu2020.am

ԴԱՍ #10



ՀԱՅ-ՌՈՒՄԱԿԱՆ
ՀԱՄԱԼՍԱՐԱՆ



ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ
ԲԱՐՁՐ ՏԵԽՆՈԼՈԳԻԱԿԱՆ
ԱՐԴՅՈՒՆԱԲԵՐՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

Բազմաչափ զանգվածներ

C++ լեզվում կարելի է հայտարարել բազմաչափ զանգվածներ

```
int main()  
{  
    int arr2[10][5]; // Էլեմենտների քանակ 10*5  
  
    int arr3[10][5][2]; // Էլեմենտների քանակ 10*5*2  
}
```



Երկչափ զանգվածներ

```
int arr[4][3];
```

arr[0]	arr[0][0]	arr[0][1]	arr[0][2]
arr[1]	arr[1][0]	arr[1][1]	arr[1][2]
arr[2]	arr[2][0]	arr[2][1]	arr[2][2]
arr[3]	arr[3][0]	arr[3][1]	arr[3][2]

Մեքենայի հիշողությունը գծային է, և
տվյալներն այնտեղ պահվում են հաջորդաբար



arr[0][0]	arr[0][1]	arr[0][2]	arr[1][0]	arr[1][1]	arr[1][2]	arr[2][0]	arr[2][1]	arr[2][2]	arr[3][0]	...
-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----------	-----



Երկչափ զանգվածներ

```
#include <iostream>
```

```
int main()
{
    int arr[4][3];
    for (int i = 0; i < 4; i++) {
        for (int j = 0; j < 3; j++) {
            std::cout << &arr[i][j] << std::endl;
        }
    }
}
```

<https://repl.it/@SevakRAU/2DArrEx1>

Հասցեները հաջորդական են



Երկչափ զանգվածի սկզբնաբեքավորում

```
#include <iostream>
```

```
int main()
{
    const int rows = 3;
    const int cols = 2;
    int arr[rows][cols]; // not initialized
    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            std::cin >> arr[i][j];
        }
    }
}
```

<https://repl.it/@VahagVardanyan/MatrixInit>

i = 0:
j = 0: a[0][0]
j = 1: a[0][1]
i = 1:
j = 0: a[1][0]
j = 1: a[1][1]
i = 2:
j = 0: a[2][0]
j = 1: a[2][1]



Երկչափ զանգվածի սկզբնաբաժանում

```
#include <iostream>
int main()
{
    const int rows = 3;
    const int cols = 2;
    int a[rows][cols] = {
        {0,1}, // a[0] = {0,1}
        {2,3}, // a[1] = {2,3}
        {4,5}  // a[2] = {4,5}
    };

    for (int i = 0; i < rows; i++) {
        for (int j = 0; j < cols; j++) {
            std::cout << a[i][j] << " ";
        }
        std::cout << std::endl;
    }
}
```



<https://repl.it/@VahagVardanyan/MatrixInit2>

Հասցեի միջոցով էլեմենտներին դիմում

```
#include <iostream>
```

```
int main()
```

```
{
```

```
    int arr[4][3] = {{0, 1, 2},    // arr[0] = {0, 1, 2}  
                    {3, 4, 5},    // arr[1] = {3, 4, 5}  
                    {6, 7, 8},    // arr[2] = {6, 7, 8}  
                    {9, 10, 11}}; // arr[3] = {9, 10, 11}
```

```
    for (int i = 0; i < 4 * 3; i++) {  
        std::cout << *(*arr + i) << std::endl;
```

```
    }
```

```
}
```

arr ցուցիչ է **arr[0]**-ի վրա, և զանգվածի ամենաառաջին (**arr[0][0]**) էլեմենտի հասցեն ստանալու համար նախ պետք է ստանանք **arr[0]**-ի հասցեն (***arr**-ն **a[0]**-ի հասցեն է): Իսկ **arr[0][0]**-ի հասցեն դա ****arr**-ն է:



<https://repl.it/@SevakRAU/2DArrEx2>

Խնդիրներ

1. Մուտքագրել երկու 5×5 մատրիցներ և հաշվել դրանց գումարը/տարբերությունը:
2. $N \times N$ մատրիցում էլեմենտները սիմետրիկ տեղափոխել գլխավոր անկյունագծի նկատմամբ ($a[i][j] \leftrightarrow a[j][i]$):
3. Տպել երկչափ զանգվածի ամեն տողի մեծագույն էլեմենտների գումարը



Դիմամիկ հիշողություն

- Մինչ այս գրված ծրագրերում օգտագործվել են ստատիկ զանգվածներ
- Ստատիկ զանգվածի չափը որոշվում է ծրագրի թարգմանության ժամանակ (**compile time**)
- Անհրաժեշտություն կա ստեղծել զանգվածներ, որոնց չափը որոշվում է ծրագրի կատարման ընթացքում (**runtime**)
 - Օրինակ, զանգվածի չափը ներմուծվում է օգտագործողի կողմից



Դինամիկ հիշողություն

- Դինամիկ հիշողություն կարելի է ստեղծել օգտագործելով **new** օպերատորը:
- **new** օպերատորը ստեղծում է նոր հիշողություն նշված չափով և վերադարձնում է ցուցիչ այդ հիշողության սկզբի վրա

```
int* dyn_arr = new int [5];
```

```
int* int_p = new int(4);
```

- Ստեղծել 20 բայթանոց ($5 * \text{sizeof(int)}$) հիշողություն
- Վերադարձնել ցուցիչ այդ հիշողության սկզբի վրա



Դինամիկ հիշողություն, փոփոխական

```
#include <iostream>
```

```
int main()  
{  
    int* int_p = new int;  
    *int_p = 10;  
    std::cout << *int_p << std::endl;  
}
```

<https://repl.it/@VahagVardanyan/DynInt>



Դինամիկ հիշողություն, զանգված

```
#include <iostream>
int main()
{
    unsigned size;
    std::cin >> size;
    int* dynArr = new int[size];
    std::cout << "Input Array" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> dynArr[i];
    }
    // delete []dynArr;
}
```

<https://repl.it/@VahagVardanyan/DynamicArray>



Դինամիկ երկչափ զանգվածի ստեղծում

```
#include <iostream>
int main() {
    int rows; int cols;
    std::cin >> rows >> cols;
    int** dynMatrix = new int*[rows];
    for (int i = 0; i < rows; ++i)
        dynMatrix[i] = new int[cols];
    std::cout << "Input Matrix" << std::endl;
    for (int i = 0; i < rows; i++) {
        std::cout << "Input " << i << " row" << std::endl;
        for (int j = 0; j < cols; j++) {
            std::cin >> dynMatrix[i][j];
        }
    }
    for (int i = 0; i < rows; i++) {
        std::cout << std::endl;
        for (int j = 0; j < cols; j++) {
            std::cout << dynMatrix[i][j] << " ";
        }
    }
}
```



Հիշողությունը անսահմանափակ չէ

```
#include <iostream>
#include <limits.h>
int main()
{
    int size = INT_MAX;
    for (long i = 0; i < 999999999; i++) {
        int * dynArr = new int[size];

    }
}
```

Ցիկլի մեջ անընդհատ
պահանջել նոր հիշողություն

<https://repl.it/@VahagVardanyan/NoMemeory>



Դինամիկ հիշողություն

- Սովորական փոփոխականները ստեծվում և ջնջվում է ավտոմատ
- Դինամիկ փոփոխականները ջնջելը ծրագրավորողի պատասխանատվությունն է
- Դինամիկ ստեղծված հիշողությունը ջնջելու համար օգտագործվում է `delete (delete [])` հրամանը



Դինամիկ հիշողության ազատում

```
#include <iostream>
int main()
{
    int* p = new int;
    *p = 100;
    std::cout << *p << std::endl;
    delete p;
}
```

<https://repl.it/@VahagVardanyan/SimpleDelete>



Դինամիկ հիշողության ազատում

```
#include <iostream>
int main()
{
    int size;
    std::cin >> size;
    int* dynArr = new int[size];
    std::cout << "Input Array" << std::endl;
    for (int i = 0; i < size; i++) {
        std::cin >> dynArr[i];
    }
    delete[] dynArr;
}
```

<https://repl.it/@VahagVardanyan/DynamicArray>



Դինամիկ հիշողության ազատում

```
#include <iostream>
int main()
{
    int rows;  int cols;
    std::cin >> rows >> cols;
    int** dynMatrix = new int*[rows];
    for (int i = 0; i < rows; ++i)
        dynMatrix[i] = new int[cols];
    std::cout << "Input Matrix" << std::endl;
    for (int i = 0; i < rows; i++) {
        std::cout << "Input " << i << " row" << std::endl;
        for (int j = 0; j < cols; j++) {
            std::cin >> dynMatrix[i][j];
        }
    }
    for (int i = 0; i < rows; ++i)
        delete [] dynMatrix[i];
    delete [] dynMatrix;
}
```



<https://repl.it/@VahagVardanyan/RowdyAgreeableArchives>

Ինչու՞ է հարկավոր ազատել դինամիկ ստեղծված հիշողությունը

- Հիշողության չափը սահմանափակ է
- **new** օպերատորով ստեղծված հիշողությունը կմնա զբաղված մինչև ծրագրի ավարտ, եթե ծրագրավորորդը այն չազատի **delete** հրամանով
- Ազատ հիշողությունը շատ արագ կավարտվի, եթե չօգտարժեք **delete** օպերատորը.



Խնդիրներ

1. Մուտքագրել N , հայտարարել $N \times N$ չափի դինամիկ երկչափ զանգված, սկզբնարժեքավորել այն հետևյալ կերպ

$$a[i][j] = N * i + 2^j \quad (2\text{-ի } j \text{ աստիճան})$$



Ի՞նչ կլինի, եթե նույն հիշողությունը ազատվի մեկից ավել անգամ

- Հիշողության կրկնակի (բազմակի ազատումը) կարող է ձեր ծրագիրը դարձնել խոցելի
- Ազատված հիշողության օգտագործումը կարող է ձեր ծրագիրը դարձնել խոցելի



Ի՞նչ կլինի, եթե նույն հիշողությունը ազատվի մեկից ավել անգամ

```
#include <iostream>
int main()
{
    int* p = new int;
    *p = 100;
    std::cout << *p << std::endl;
    delete p;
    // This code can cause vulnerability
    // In some cases
    delete p;
}
```

Մեծ ծրագրերում հիշողության
կրկնակի ազատումը կարող է բերել
անկանխատեսելի հետևանքների

Ի՞նչ կլինի ազատված հիշողությանը դիմելու դեպքում

```
#include <iostream>
int main()
{
    int* p = new int;
    *p = 100;
    std::cout << *p << std::endl;
    delete p;
    // This code can cause vulnerability
    // In some cases
    std::cout << *p << std::endl;
}
```

Մեծ ծրագրերում ազատված
հիշողության օգտագործումը
կարող է բերել անկանխատեսելի
հետևանքների

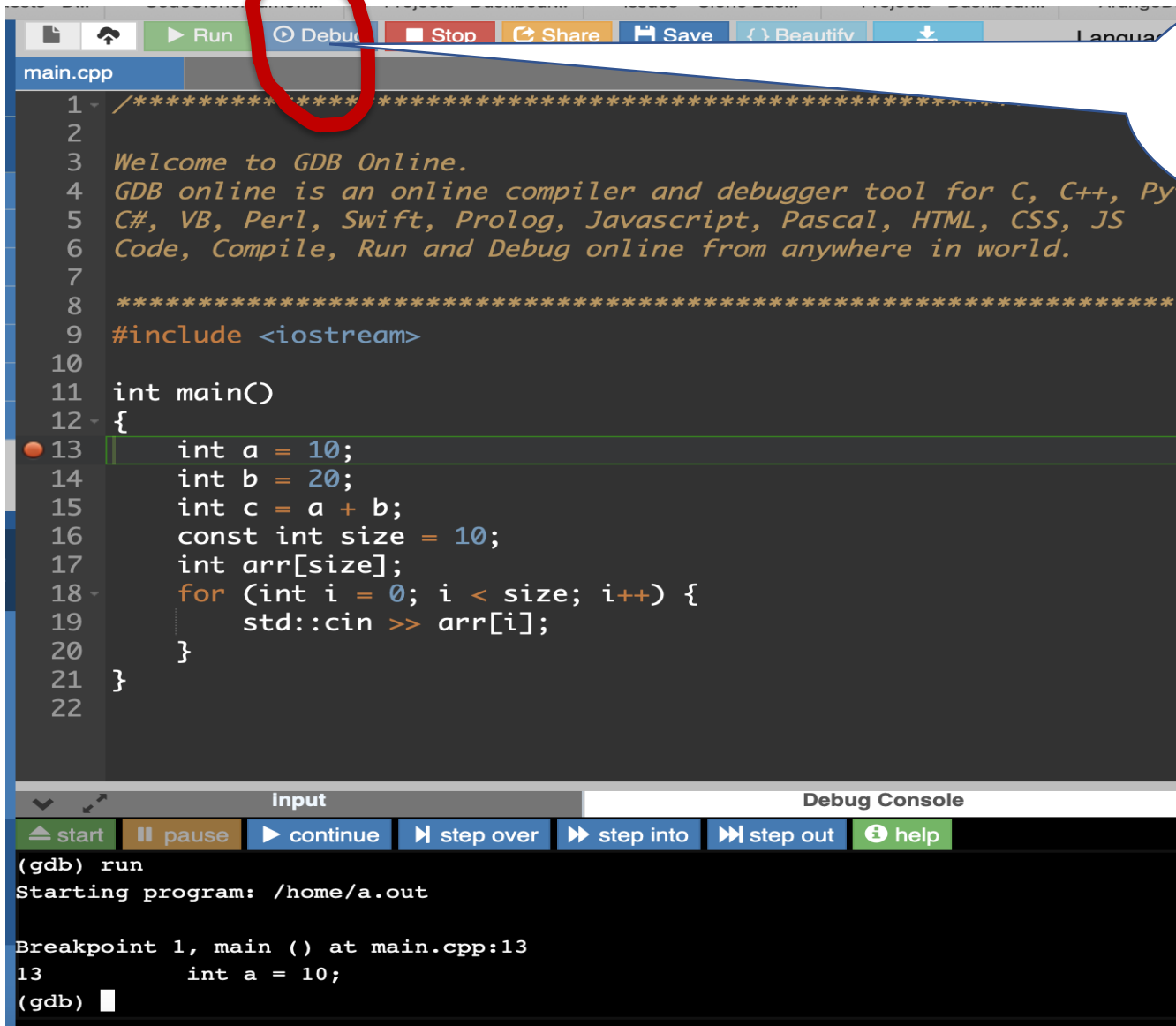
Դեբագեր (debug, debugger)

- Հատուկ ծրագիր է, որը թույլ է տալիս ձեր գրած ծրագիրը կատարել քայլ առ քայլ, և ամեն քայլում տեսնել ծրագրի վիճակը (փոփոխականների անուններ, արժեքներ և այլն)
- Օգտագործվում է ծրագիր գրելիս թույլ տված սխալները արագ հայտնաբերելու համար

https://www.onlinegdb.com/online_c++_debugger



Օրինակներ



```
1  /*****  
2  
3  Welcome to GDB Online.  
4  GDB online is an online compiler and debugger tool for C, C++, Pyt  
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6  Code, Compile, Run and Debug online from anywhere in world.  
7  
8  *****/  
9  #include <iostream>  
10  
11  int main()  
12  {  
13  int a = 10;  
14  int b = 20;  
15  int c = a + b;  
16  const int size = 10;  
17  int arr[size];  
18  for (int i = 0; i < size; i++) {  
19  std::cin >> arr[i];  
20  }  
21  }  
22
```

input

Debug Console

(gdb) run
Starting program: /home/a.out

Breakpoint 1, main () at main.cpp:13
13 int a = 10;
(gdb)

Մկսել ծրագիրը «debug» ռեժիմով

a	
b	
c	4196445
size	0
arr	{124, 0, 0, 0, 4196368, 0, 4195936, 0, -4912, 32767}

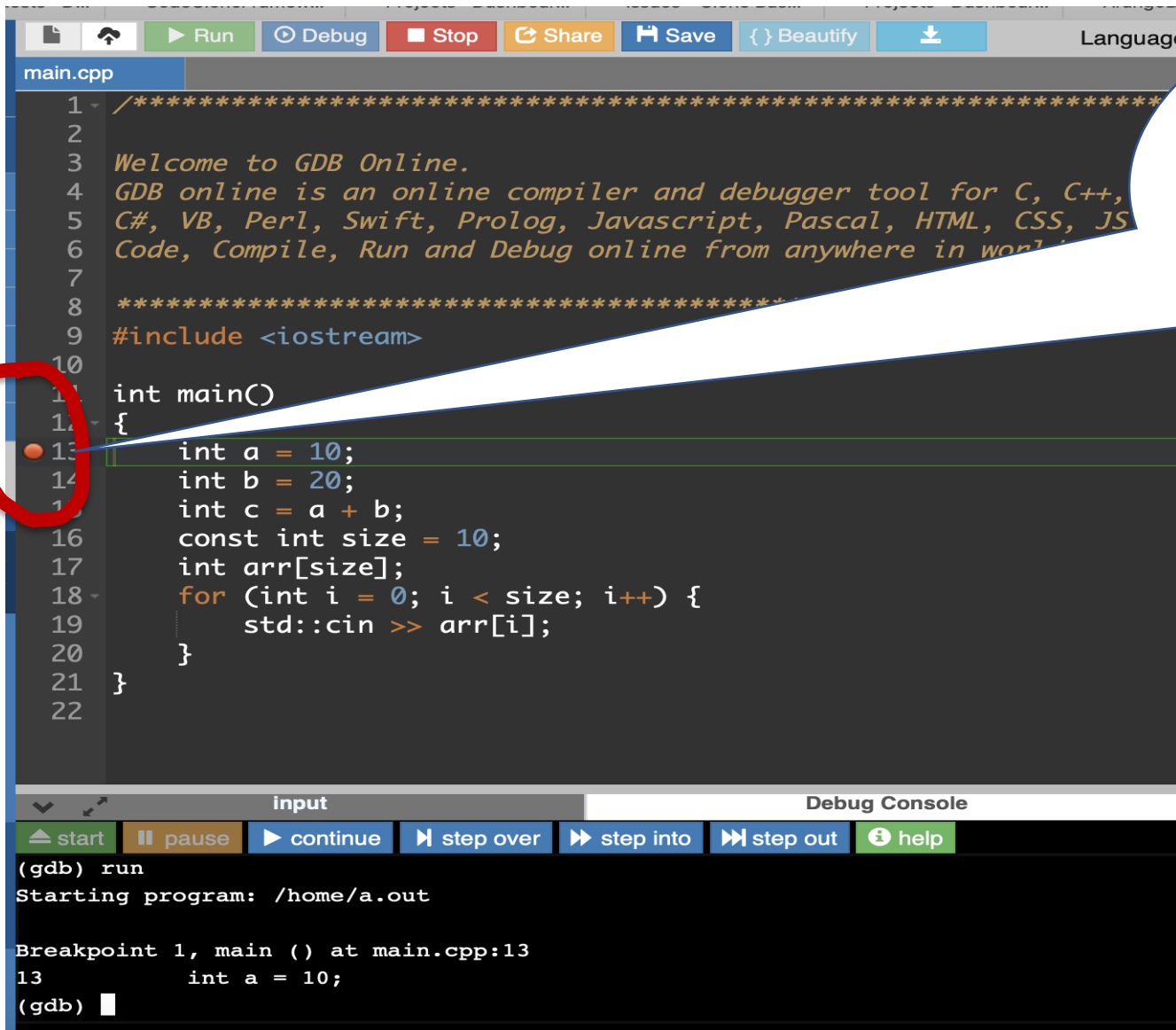
Display Expressions

Expression	Value
Enter expression to watch	

Breakpoints and Watchpoints

	#	Description	
<input checked="" type="checkbox"/>	1	in main() at main.cpp:13	✕

Օրինակներ



```
1  /*****
2
3  Welcome to GDB Online.
4  GDB online is an online compiler and debugger tool for C, C++,
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
6  Code, Compile, Run and Debug online from anywhere in world
7
8  *****/
9  #include <iostream>
10
11 int main()
12 {
13     int a = 10;
14     int b = 20;
15     int c = a + b;
16     const int size = 10;
17     int arr[size];
18     for (int i = 0; i < size; i++) {
19         std::cin >> arr[i];
20     }
21 }
22
```

input

Debug Console

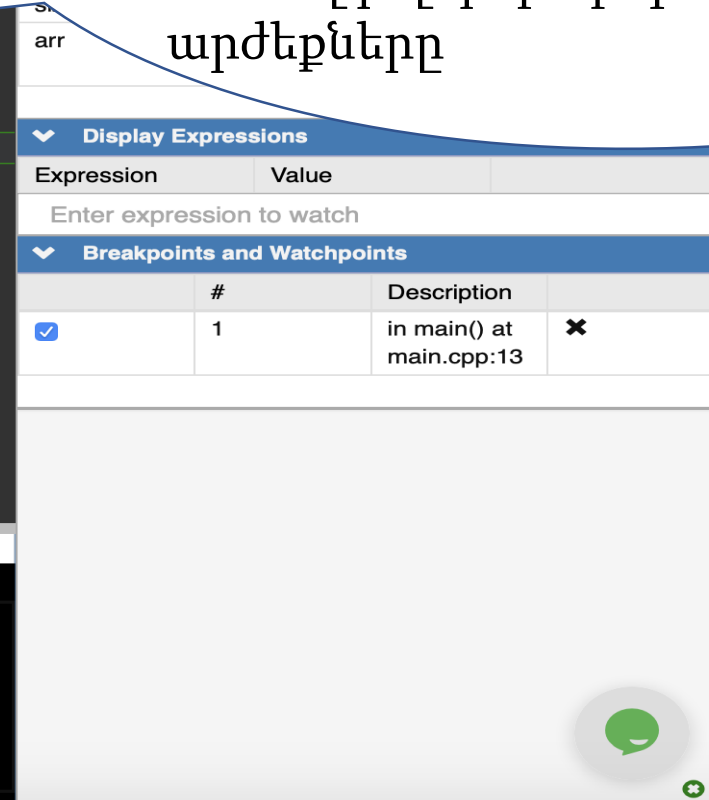
(gdb) run
Starting program: /home/a.out

Breakpoint 1, main () at main.cpp:13
13 int a = 10;
(gdb)

Breakpoint.

Ծրագիրը կանգ կառնի այն տողերի վրա, որոնց վրա դրված է «breakpoint»

Breakpoint-ի կետերում կարելի տեսնել բոլոր փոփոխականների արժեքները



arr

▼ Display Expressions

Expression	Value
Enter expression to watch	

▼ Breakpoints and Watchpoints

	#	Description	
<input checked="" type="checkbox"/>	1	in main() at main.cpp:13	✕

Օրինակներ

The screenshot displays the GDB Online IDE interface. The main code editor shows a C++ program with a breakpoint set at line 13. The program includes a welcome message and a loop that reads input into an array. The right sidebar contains several panels: 'Call Stack' showing the current function 'main' at line 13; 'Local Variables' showing variables 'a' (2), 'b' (0), 'c' (4196445), 'size' (0), and 'arr' (an array of 10 elements); 'Display Expressions' with an input field for watching expressions; and 'Breakpoints and Watchpoints' showing a single breakpoint at line 13. The bottom panel is the 'Debug Console', which shows the command '(gdb) run' and the output 'Starting program: /home/a.out'. The 'start' button in the debug console is circled in red. The 'input' field is also visible above the debug console.

```
1  /*****  
2  
3  Welcome to GDB Online.  
4  GDB online is an online compiler and debugger tool for C, C++, Pyt  
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6  Code, Compile, Run and Debug online from anywhere in world.  
7  
8  *****/  
9  #include <iostream>  
10  
11 int main()  
12 {  
13     int a = 10;  
14     int b = 20;  
15     int c = a + b;  
16     const int size = 10;  
17     int arr[size];  
18     for (int i = 0; i < size; i++) {  
19         std::cin >> arr[i];  
20     }  
21 }  
22
```

Call Stack

#	Function	File:Line
0	main	main.cpp:13

Local Variables

Variable	Value
a	2
b	0
c	4196445
size	0
arr	{124, 0, 0, 0, 4196368, 0, 4195936, 0, -4912, 32767}

Display Expressions

Expression	Value
Enter expression to watch	

Breakpoints and Watchpoints

	#	Description	
<input checked="" type="checkbox"/>	1	in main() at main.cpp:13	✕

input

Debug Console

(gdb) run
Starting program: /home/a.out

Breakpoint 1, main () at main.cpp:13
13 int a = 10;
(gdb)

Օրինակներ

The screenshot displays the GDB Online IDE interface. The main editor shows a C++ program with a breakpoint set at line 13. The program's output and GDB console are visible at the bottom.

```
1  /*****  
2  
3  Welcome to GDB Online.  
4  GDB online is an online compiler and debugger tool for C, C++, Pyt  
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6  Code, Compile, Run and Debug online from anywhere in world.  
7  
8  *****/  
9  #include <iostream>  
10  
11 int main()  
12 {  
13     int a = 10;  
14     int b = 20;  
15     int c = a + b;  
16     const int size = 10;  
17     int arr[size];  
18     for (int i = 0; i < size; i++) {  
19         std::cin >> arr[i];  
20     }  
21 }  
22
```

Call Stack

#	Function	File:Line
0	main	main.cpp:13

Local Variables

Variable	Value
a	2
b	0
c	4196445
size	0
arr	{124, 0, 0, 0, 4196368, 0, 4195936, 0, -4912, 32767}

Display Expression

Expression

Enter expres...

Breakpoints

Description	
in main() at main.cpp:13	✕

Input

start pause continue step over

(gdb) run

Starting program: /home/a.out

Breakpoint 1, main () at main.cpp:13

13 int a = 10;

(gdb) █

Փոփոխականների արժեքները 13-րդ տողում

Օրինակներ

The screenshot displays the GDB Online IDE interface. The main editor shows a C++ program with a breakpoint set at line 16. The program's output is visible in the top left, and the GDB console at the bottom shows the execution state.

main.cpp

```
1  /*****
2
3  Welcome to GDB Online.
4  GDB online is an online compiler and debugger tool for C, C++, Pyt
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
6  Code, Compile, Run and Debug online from anywhere in world.
7
8  *****/
9  #include <iostream>
10
11  int main()
12  {
13      int a = 10;
14      int b = 20;
15      int c = a + b;
16      const int size = 10;
17      int arr[size];
18      for (int i = 0; i < size; i++) {
19          std::cin >> arr[i];
20      }
21  }
22
```

Call Stack

#	Function	File:Line
0	main	main.cpp:16

Local Variables

Variable	Value
a	10
b	20
c	30
size	0
arr	{124, 0, 0, 0, 4196368, 0, 4195936, 0, -4912, 32767}

Display Expression

Expression	Value
Enter expression to display	

Breakpoints

Description	✕
in main() at main.cpp:19	

input

(gdb) continue
Continuing.

Breakpoint 2, main () at main.cpp:16
16 const int size = 10;

Փոփոխականների արժեքները 16-րդ տողում

Օրինակներ

The screenshot displays the GDB Online IDE interface. The main editor shows a C++ program with a breakpoint set at line 19. The program's logic is as follows:

```
1  /*****  
2  
3  Welcome to GDB Online.  
4  GDB online is an online compiler and debugger tool for C, C++, Pyt  
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS  
6  Code, Compile, Run and Debug online from anywhere in world.  
7  
8  *****/  
9  #include <iostream>  
10  
11 int main()  
12 {  
13     int a = 10;  
14     int b = 20;  
15     int c = a + b;  
16     const int size = 10;  
17     int arr[size];  
18     for (int i = 0; i < size; i++) {  
19         std::cin >> arr[i];  
20     }  
21 }  
22
```

The right sidebar contains the following panels:

- Call Stack:**

#	Function	File:Line
0	main	main.cpp:19
- Local Variables:**

Variable	Value
i	0
a	10
b	20
c	30
size	10
arr	{124, 0, 0, 0, 4196368, 0, 4195936, 0, -4912, 32767}
- Display Expressions:**

Expression	Value
Enter expression to watch	
- Breakpoints:**

Description	
in main() at main.cpp:19	✕
	✕

The bottom panel shows the GDB console with the following output:

```
(gdb) continue  
Continuing.  
  
Breakpoint 1, main () at main.cpp:19  
19         std::cin >> arr[i];  
(gdb) 0D
```

Փոփոխականների արժեքները
19-րդ տողում

Օրինակներ

The screenshot shows the GDB Online IDE interface. The main editor displays a C++ program with several breakpoints (orange dots) set at lines 13, 16, and 19. The program includes a welcome message and a loop that reads input into an array. The right sidebar shows the 'Call Stack' and 'Local Variables' panels. The 'Call Stack' panel shows the current function 'main' at line 19. The 'Local Variables' panel shows variables 'i' (0), 'a' (10), 'b', and 'c'. The bottom panel shows the 'input' field and the 'Debug Console' with the command '(gdb) continue' and the output 'Continuing.'. A red circle highlights the 'continue' button in the 'input' field.

main.cpp

```
1  /*****
2
3  Welcome to GDB Online.
4  GDB online is an online compiler and debugger tool for C, C++, Pyt
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
6  Code, Compile, Run and Debug online from anywhere in world.
7
8  *****/
9  #include <iostream>
10
11 int main()
12 {
13     int a = 10;
14     int b = 20;
15     int c = a + b;
16     const int size = 10;
17     int arr[size];
18     for (int i = 0; i < size; i++) {
19         std::cin >> arr[i];
20     }
21 }
22
```

Call Stack

#	Function	File:Line
0	main	main.cpp:19

Local Variables

Variable	Value
i	0
a	10
b	
c	

Breakpoints and Watchpoints

	#	Description	
<input checked="" type="checkbox"/>	1	in main() at main.cpp:19	✕
<input checked="" type="checkbox"/>	2	in main() at main.cpp:16	✕
<input checked="" type="checkbox"/>	3	in main() at main.cpp:13	✕

input

start pause continue step over step into step out help

(gdb) continue
Continuing.

Breakpoint 1, main () at main.cpp:19
19 std::cin >> arr[i];
(gdb) 0D

Անցնել հաջորդ «breakpoint»-ին

Օրինակներ

The screenshot shows the GDB Online IDE interface. The main editor displays a C++ program named `main.cpp`. The program includes a welcome message and a `main` function. The `main` function initializes variables `a` and `b`, calculates `c = a + b`, and sets a constant `size = 10`. It then declares an array `arr` of size 10 and enters a `for` loop that reads input from `std::cin` into `arr[i]`. The current line of execution is line 19, `std::cin >> arr[i];`. The right sidebar shows the **Call Stack** with one entry: `main` at `main.cpp:19`. The **Local Variables** section shows variables `i` (0), `a` (10), `b` (20), `c` (30), `size` (10), and `arr` (array of 10 elements). The bottom panel shows the **Debug Console** with the `(gdb) continue` command and the output `Continuing.`. The `step over` button is highlighted with a red circle. A speech bubble points to the `step over` button with the text `Անցնել հաջորդ տողին`.

```
1  /*****
2
3  Welcome to GDB Online.
4  GDB online is an online compiler and debugger tool for C, C++, Pyt
5  C#, VB, Perl, Swift, Prolog, Javascript, Pascal, HTML, CSS, JS
6  Code, Compile, Run and Debug online from anywhere in world.
7
8  *****/
9  #include <iostream>
10
11 int main()
12 {
13     int a = 10;
14     int b = 20;
15     int c = a + b;
16     const int size = 10;
17     int arr[size];
18     for (int i = 0; i < size; i++) {
19         std::cin >> arr[i];
20     }
21 }
22
```

#	Function	File:Line
0	main	main.cpp:19

Variable	Value
i	0
a	10
b	20
c	30
size	10
arr	array of 10 elements

#	Description	
1	in main() at main.cpp:19	✗
2	in main() at main.cpp:16	✗
3	in main() at main.cpp:13	✗

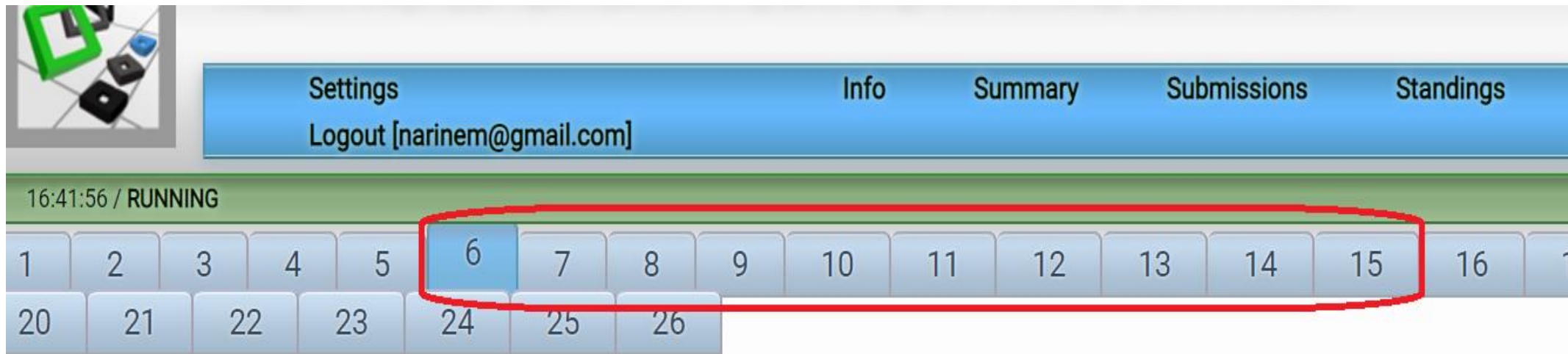
Debug Console

(gdb) continue
Continuing.

Breakpoint 1, main () at main.cpp:19
19 std::cin >> arr[i];
(gdb) 0D

Տնային աշխատանք

Բաժին 3: Դինամիկ զանգվածներ և ֆունկցիաներ



Խնդիրներ 6 - 15

Շնորհակալություն Հարցե՞ր