

# «ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԻՄՈՒՆՔՆԵՐ» դասընթաց

այլ ոլորտներից դեպի տեխնոլոգիական ոլորտ  
սկսնականների համար



edu2020.am

## ԴԱՍ #3



ՀԱՅ-ՌՈՒՄԱԿԱՆ  
ՀԱՄԱԼՍԱՐԱՆ



ՀԱՅԱՍՏԱՆԻ ՀԱՆՐԱՊԵՏՈՒԹՅԱՆ  
ԲԱՐՁՐ ՏԵԽՆՈԼՈԳԻԱԿԱՆ  
ԱՐԴՅՈՒՆԱԲԵՐՈՒԹՅԱՆ ՆԱԽԱՐԱՐՈՒԹՅՈՒՆ

# Առանց նշանի թվեր (unsigned)

C++-ում գոյություն ունեցող ամբողջ տիպերը կարելի է օգտագործել առանց նշանի թվեր (*0-ից մեծ հավասար*) պահելու համար, դիմացից **unsigned** ավելացնելով՝

1. **unsigned short** համեմատաբար փոքր թվերի համար
2. **unsigned int** միջին մեծության թվերի համար
3. **unsigned long** և **unsigned long long** շատ մեծ թվերի համար

Օրինակ՝ **int** տիպը կարող է որոշ դեպքերում պահել թվեր  $[-32768, 32767]$  միջակայքից, իսկ հաճախ պահում է  $[-2147483648, 2147483647]$  միջակայքից:

Իսկ **unsigned int** տիպը կարող է պահել համապատասխանաբար  $[0, 65535]$ ,  $[0, 4294967295]$

Եթե տրված  $X$  տիպը (**short**, **int**, **long**, **long long**) պահում է արժեք  $[-A, B]$  ինտերվալում, ապա **unsigned X** (**short**, **int**, **long**, **long long**) պահում է արժեքներ  $[0, A + B]$



# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    unsigned short x = 1;  
    unsigned int y = 2;  
    unsigned long z = 4;  
    unsigned long long k = 3;  
    std::cout << "x = " << x << ", y = " << y << ", z = " << z << ", k = " << k;  
}
```



**x = 1, y = 2, z = 4, k = 3**

<https://repl.it/@SevakRAU/UnsignedInts>





# Առանց նշանի ամբողջ տիպեր, օրինակ

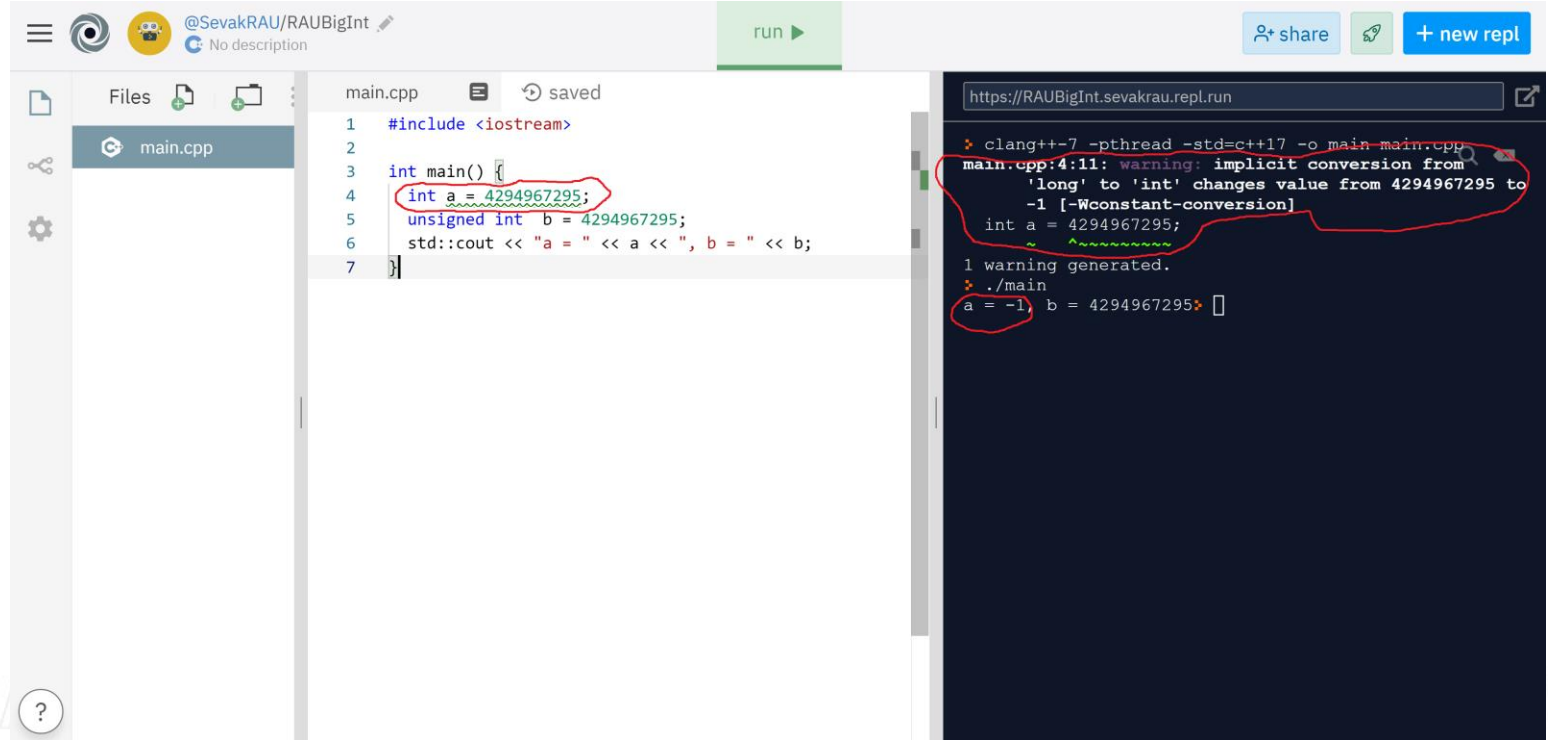
```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b;  
}
```



**a = -1, b = 4294967295**

<https://repl.it/@SevakRAU/RAUBigInt>



```
main.cpp  
1 #include <iostream>  
2  
3 int main() {  
4     int a = 4294967295;  
5     unsigned int b = 4294967295;  
6     std::cout << "a = " << a << ", b = " << b;  
7 }
```

```
clang++-7 -pthread -std=c++17 -o main-main.cpp  
main.cpp:4:11: warning: implicit conversion from  
      'long' to 'int' changes value from 4294967295 to  
      -1 [-Wconstant-conversion]  
    int a = 4294967295;  
           ^~~~~~  
1 warning generated.  
./main  
a = -1, b = 4294967295
```



# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b;  
}
```

Եթե ավելի մեծ թիվ է վերագրվում քան  
կարող է պահվել իրականում այլ  
արժեք կստացվի: Թե ինչպես  
կհասկանանք ավելի ուշ

**a = -1, b = 4294967295**

<https://repl.it/@SevakRAU/RAUBigInt>

The screenshot shows a C++ REPL interface with the following code in main.cpp:

```
1 #include <iostream>  
2  
3 int main() {  
4     int a = 4294967295;  
5     unsigned int b = 4294967295;  
6     std::cout << "a = " << a << ", b = " << b;  
7 }
```

The output shows a warning: "warning: implicit conversion from 'long' to 'int' changes value from 4294967295 to -1 [-Wconstant-conversion]". The final output is "a = -1, b = 4294967295".



# Առանց նշանի ամբողջ տիպերում պահվող թվերի միջակայք

Բերված օրինակը տպում է մեզ հետաքրքիր տիպերի չափերը՝

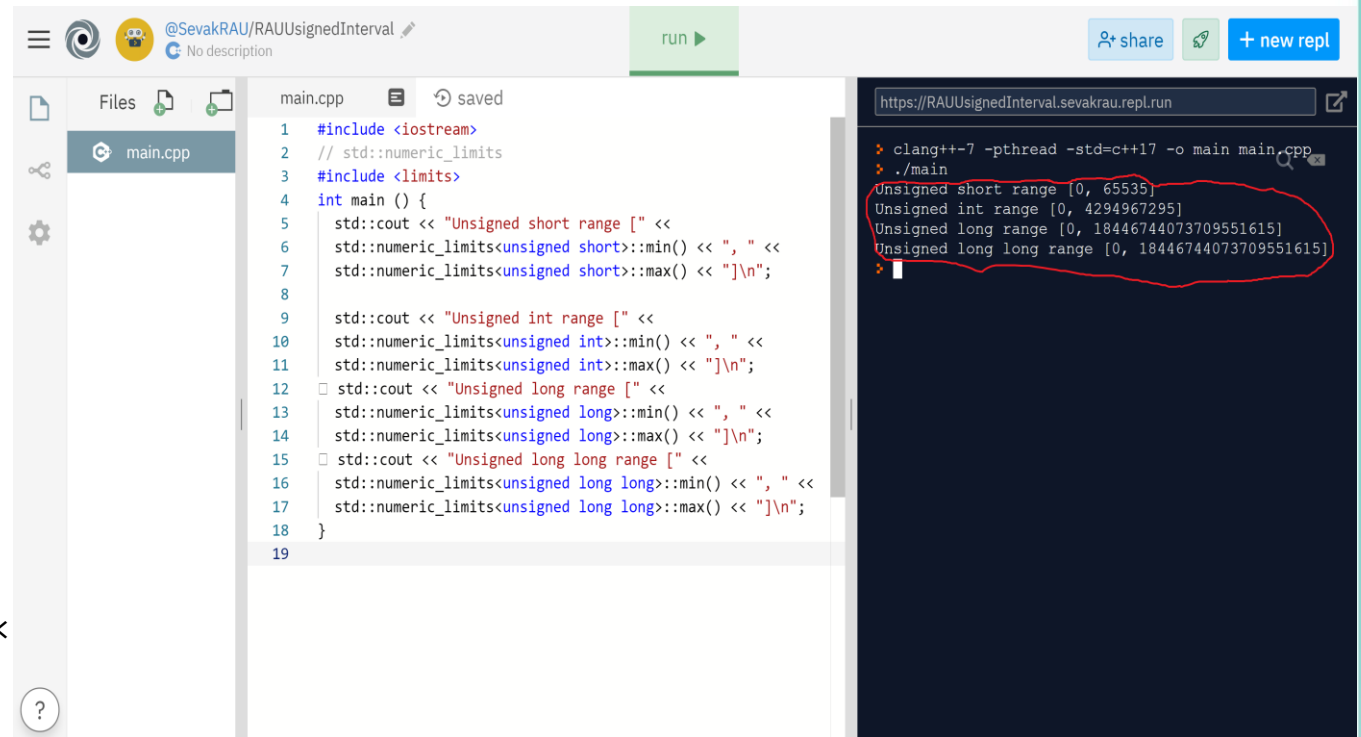
```
#include <iostream>
// std::numeric_limits
#include <limits>
int main () {
    std::cout << "Unsigned short range [" <<
    std::numeric_limits<unsigned short>::min() << ", " <<
    std::numeric_limits<unsigned short>::max() << "]\n";

    std::cout << "Unsigned int range [" <<
    std::numeric_limits<unsigned int>::min() << ", " <<
    std::numeric_limits<unsigned int>::max() << "]\n";

    std::cout << "Unsigned long range [" <<
    std::numeric_limits<unsigned long>::min() << ", " <<
    std::numeric_limits<unsigned long>::max() << "]\n";

    std::cout << "Unsigned long long range [" <<
    std::numeric_limits<unsigned long long>::min() << ", " <<
    std::numeric_limits<unsigned long long>::max() << "]\n";
}
```

<https://repl.it/@SevakRAU/RAUUnsignedInterval>



```
main.cpp
1 #include <iostream>
2 // std::numeric_limits
3 #include <limits>
4 int main () {
5     std::cout << "Unsigned short range [" <<
6     std::numeric_limits<unsigned short>::min() << ", " <<
7     std::numeric_limits<unsigned short>::max() << "]\n";
8
9     std::cout << "Unsigned int range [" <<
10    std::numeric_limits<unsigned int>::min() << ", " <<
11    std::numeric_limits<unsigned int>::max() << "]\n";
12
13    std::cout << "Unsigned long range [" <<
14    std::numeric_limits<unsigned long>::min() << ", " <<
15    std::numeric_limits<unsigned long>::max() << "]\n";
16
17    std::cout << "Unsigned long long range [" <<
18    std::numeric_limits<unsigned long long>::min() << ", " <<
19    std::numeric_limits<unsigned long long>::max() << "]\n";
20 }
```

```
clang++-7 -pthread -std=c++17 -o main main.cpp
./main
Unsigned short range [0, 65535]
Unsigned int range [0, 4294967295]
Unsigned long range [0, 18446744073709551615]
Unsigned long long range [0, 18446744073709551615]
```



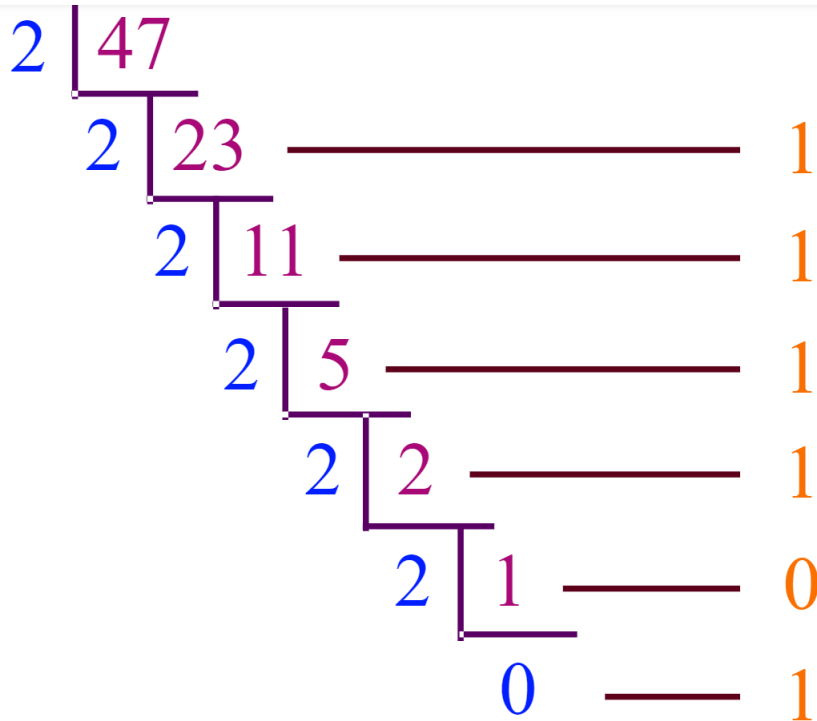
# Տվյալների ներկայացում

Համակարգչում բոլոր տվյալները ներկայացվում են 0 և 1 հաջորդականությամբ:  
Այդպես հարմար է քանի որ համակարգչում 0 և 1 կոդավորվում են որպես  
էլեկտրական հոսանքի առկայություն կամ բացակայություն:

**0 և 1 տվյալների ներկայացումը կոչվում է տվյալների երկուական ներկայացում:**



# Անցում 10-ականից 2-ական համակարգ



$$47 \% 2 = 23 \text{ մնացորդ } 1$$

$$23 \% 2 = 11 \text{ մնացորդ } 1$$

$$11 \% 2 = 5 \text{ մնացորդ } 1$$

$$5 \% 2 = 2 \text{ մնացորդ } 1$$

$$2 \% 2 = 1 \text{ մնացորդ } 0$$

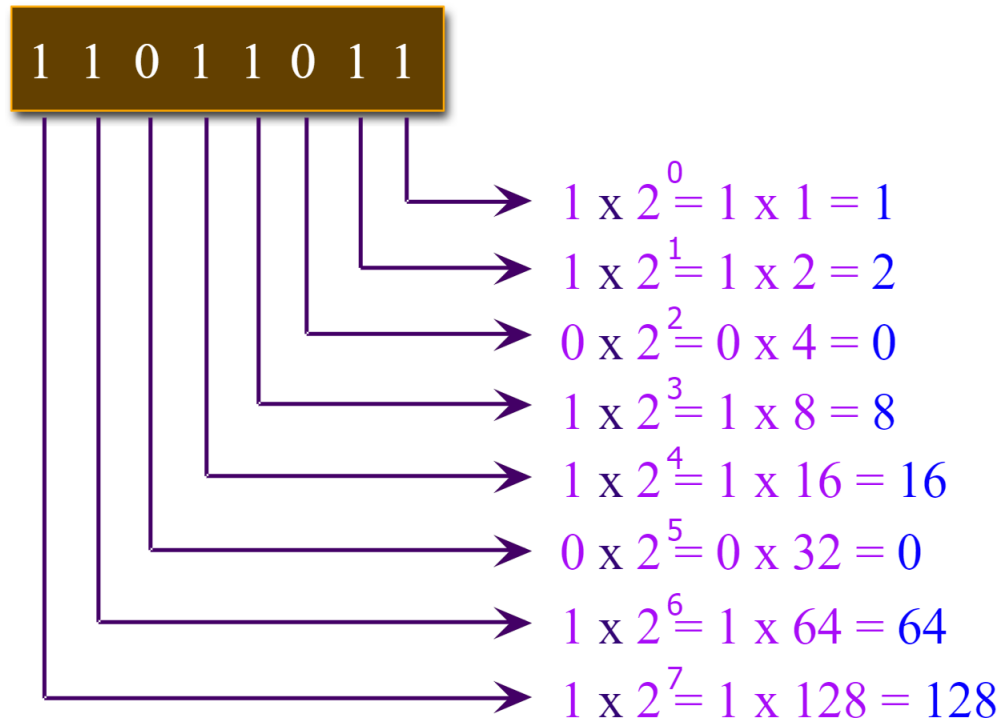
$$1 \% 2 = 0 \text{ մնացորդ } 1$$

$$(47)_{10} = (101111)_2$$





# Անցում 2-ականից 10-ական համակարգ

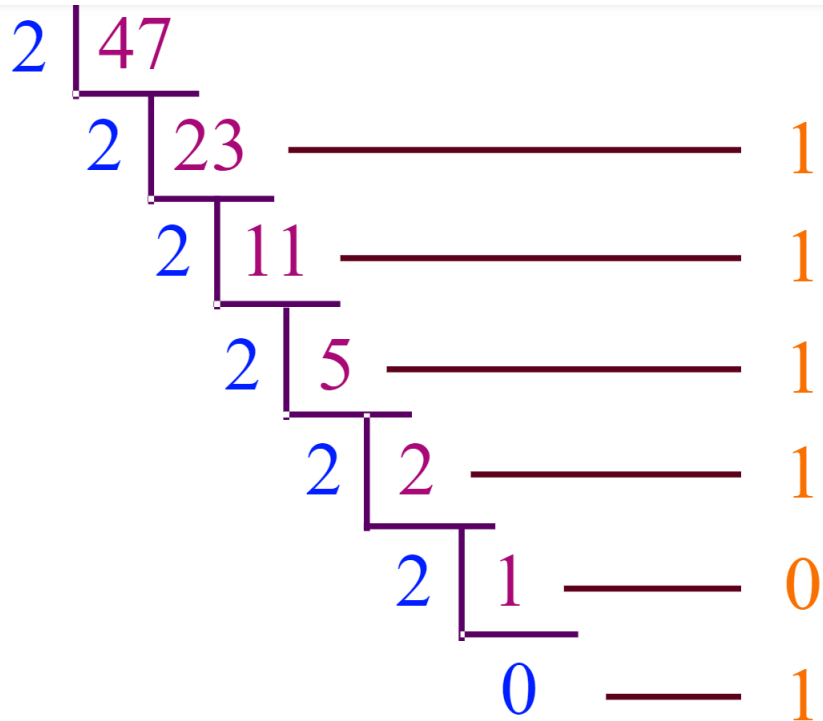


$$1 + 2 + 8 + 16 + 64 + 128 = 219$$

$$(11011011)_2 = (219)_{10}$$



# Անցում 10-ականից 2-ական և հակառակ



Միարժեք է

$$(47)_{10} = (101111)_2 \iff 1 \cdot 2^5 + 0 \cdot 2^4 + 1 \cdot 2^3 + 1 \cdot 2^2 + 1 \cdot 2^1 + 1 \cdot 2^0 = 47$$

# ԹՎԱԲԱՆԱԿԱՆ ԳՈՐԾՈՂՈՒԹՅՈՒՆՆԵՐ

## ԳՈԼՄԱՐՈԼՄ

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

1 + 1 = 0, 1-նիշի փոխանցում հաջորդին, կատացվի 10

$$\begin{array}{rcccccc} & 1_0 & 1_1 & 1_1 & 1_0 & 1 \\ + & 1 & 0 & 1 & 1 & 1 \\ \hline = & 1 & 0 & 0 & 1 & 0 & 0 \end{array}$$

## Հանդիմ

$$0 - 0 = 0$$

0 - 1 = 1, նախորդից 1 պարտք

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$\begin{array}{r} -11 \ 20 \ 1 \ 1 \ 1 \\ - \quad 0 \ 1 \ 1 \ 0 \ 1 \\ \hline = \quad 0 \ 1 \ 0 \ 1 \ 0 \end{array}$$



# Թվաբանական գործողություններ

## Գումարում

$$0 + 0 = 0$$

$$0 + 1 = 1$$

$$1 + 0 = 1$$

$$1 + 1 = 0, 1\text{-նիշի փոխանցում հաջորդին, կստացվի } 10$$

$$\begin{array}{r} \phantom{+} 1011101 \rightarrow 13 \\ + \phantom{00} 10111 \rightarrow 23 \\ \hline = 100100 \rightarrow 36 \end{array}$$

## Հանում

$$0 - 0 = 0$$

$$0 - 1 = 1, \text{ նախորդից } 1 \text{ պարտք}$$

$$1 - 0 = 1$$

$$1 - 1 = 0$$

$$\begin{array}{r} \phantom{-} -1120111 \rightarrow 23 \\ - \phantom{00} 01101 \rightarrow 13 \\ \hline = \phantom{00} 01010 \rightarrow 10 \end{array}$$





# Համակարգչի հիշողություն

- **bit** (**b**inary **d**igit), 0 կամ 1 գրելու համար անհրաժեշտ հիշողություն
- byte – 8 bit (b)
- kilobyte – 1024 byte (Kb)
- megabyte – 1024 kilobyte (Mb)
- gigabyte – 1024 megabyte (Gb)



# Համակարգչի հիշողություն, օրինակ

47-ի երկուական ներկայացումը (101111) կտեղավորվի 1 byte հիշողության մեջ

0	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

Ավելացած մասում  
0-ներ է գրվում

256 -> երկուական ներկայացումը (100000000) չի տեղավորվի 1 byte հիշողության մեջ (9 նիշ է պարունակում)



# Տվյալների կոդավորում

Ուղիղ կոդը դա երկուական ներկայացումն է: Օրինակ, 47-ի երկուական ներկայացումը (101111)՝

0	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

Հակադարձ կոդի դեպքում բոլոր բիթերի արժեքները ժխտվում են՝

1	1	0	1	0	0	0	0
---	---	---	---	---	---	---	---

Լրացուցիչ կոդի դեպքում բոլոր բիթերի արժեքները ժխտվում են և գումարվում է 1՝

1	1	0	1	0	0	0	0
				+			
0	0	0	0	0	0	0	1
				↓			
1	1	0	1	0	0	0	1



# Ամբողջ թվերի ներկայացում

1. Դրական թվերը ներկայացվում են ուղիղ կոդով
2. Բացասական թվերը ներկայացվում են լրացուցիչ կոդով

0	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

Նշանի բիթ, եթե 0 է թիվը  
համարվում է դրական, հակառակ  
դեպքում բացասական



# Ամբողջ թվերի ներկայացում, օրինակ

1. Դրական թվերը ներկայացվում են ուղիղ կոդով
2. Բացասական թվերը ներկայացվում են լրացուցիչ կոդով

0	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

Այս դեպքում գրված թիվը համարվում է դրական և այն ստանալու համար պետք ուղակի անցում կատարել 2-ական համակարգից դեպի 10-ական համակարգ՝

$$1*2^5 + 0*2^4 + 1*2^3 + 1*2^2 + 1*2^1 + 1*2^0 = 47$$



# Ամբողջ թվերի ներկայացում, օրինակ

1. Դրական թվերը ներկայացվում են ուղիղ կոդով
2. Բացասական թվերը ներկայացվում են լրացուցիչ կոդով

1	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

Այս դեպքում համարվում է, որ գրված է բացասական թվի լրացուցիչ կոդը:  
Սկզբից պետք է լրացուցիչ կոդից ստանալ ուղիղ կոդը,  
ապա անցում կատարել 2-ական համակարգից դեպի 10-ական համակարգ՝



# Ամբողջ թվերի ներկայացում, օրինակ

1. Դրական թվերը ներկայացվում են ուղիղ կոդով
2. Բացասական թվերը ներկայացվում են լրացուցիչ կոդով

1	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

Այս դեպքում համարվում է, որ գրված է բացասական թվի լրացուցիչ կոդը:  
Սկզբից պետք է լրացուցիչ կոդից ստանալ ուղիղ կոդը,  
ապա անցում կատարել 2-ական համակարգից դեպի 10-ական համակարգ՝

1	0	1	0	1	1	1	1
---	---	---	---	---	---	---	---

-

0	0	0	0	0	0	0	1
---	---	---	---	---	---	---	---



1	0	1	0	1	1	1	0
---	---	---	---	---	---	---	---

Ժխտում

0	1	0	1	0	0	0	1
---	---	---	---	---	---	---	---



$$1 \cdot 2^6 + 0 \cdot 2^5 + 1 \cdot 2^4 + 0 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 81 \rightarrow -81$$



# Ամբողջ տիպերի չափերը

1. **short** համեմատաբար փոքր թվերի համար, *ամենաքիչը 2 byte (16 bit), ավելիքը հենց 2 byte*
2. **int** միջին մեծության թվերի համար, *ամենաքիչը 2 byte (16 bit), ավելիքը 4 byte*
3. **long**, *ամենաքիչը 4 byte (32 bit)*
4. **long long**, *ամենաքիչը 8 byte (64 bit)*





# Ամբողջ տիպերի չափերը

1. **short** համեմատաբար փոքր թվերի համար, *ամենաքիչը 2 byte (16 bit), ավերաբար հենց 2 byte*
2. **int** միջին մեծության թվերի համար, *ամենաքիչը 2 byte (16 bit), ավերաբար 4 byte*
3. **long**, *ամենաքիչը 4 byte (32 bit)*
4. **long long**, *ամենաքիչը 8 byte (64 bit)*

Եթե **int** տիպին հատկացվե է *2 byte (16 bit)* այդ դեպքում այն կարող է պահել թվեր *[-32768, 32767]* միջակայքից, քանի-որ՝

*-32768 - 10000000000000000* լրացուցիչ կոդով գրված ամենափոքր թիվն է, որ տեղավորվում է 2 բայթում  
*32767- 0111111111111111* ուղիղ կոդով գրված ամենամեծ թիվն է, որ տեղավորվում է 2 բայթում



# Ամբողջ տիպերի չափերը

1. **short** համեմատաբար փոքր թվերի համար, *ամենաքիչը 2 byte (16 bit), ավերաբար հենց 2 byte*
2. **int** միջին մեծության թվերի համար, *ամենաքիչը 2 byte (16 bit), ավերաբար 4 byte*
3. **long**, *ամենաքիչը 4 byte (32 bit)*
4. **long long**, *ամենաքիչը 8 byte (64 bit)*

Եթե **int** տիպին հատկացվե է *2 byte (16 bit)* այդ դեպքում այն կարող է պահել թվեր *[-32768, 32767]* միջակայքից, քանի-որ՝

*-32768 - 10000000000000000 լրացուցիչ կոդով գրված ամենափոքր թիվն է, որ տեղավորվում է 2 բայթում*  
*32767- 0111111111111111 ուղիղ կոդով գրված ամենամեծ թիվն է, որ տեղավորվում է 2 բայթում*

Եթե **unsigned int** տիպին հատկացվե է *2 byte (16 bit)* այդ դեպքում այն կարող է պահել թվեր *[0, 65535]* միջակայքից, քանի-որ՝

*0 - 0000000000000000 ուղիղ կոդով գրված ամենափոքր թիվն է, որ տեղավորվում է 2 բայթում*  
*65535 - 1111111111111111 ուղիղ կոդով գրված ամենամեծ թիվն է, որ տեղավորվում է 2 բայթում*



# *sizeof* օպերատոր

*sizeof* օպերատորը տրված տիպի կամ փոփոխականի համար վերադարձնում է դրա չափը բայթերով

```
#include <iostream>
int main() {
    short a;
    int b;
    std::cout << "Length of short in bytes is " << sizeof(short) << std::endl;
    std::cout << "Length of short variable in bytes is " << sizeof(a) << std::endl;
    std::cout << "Length of int in bytes is " << sizeof(int) << std::endl;
    std::cout << "Length of int variable in bytes is " << sizeof(b) << std::endl;
    std::cout << "Length of long in bytes is " << sizeof(long) << std::endl;
    std::cout << "Length of long long variable in bytes is " << sizeof(long long) << std::endl;
}
```

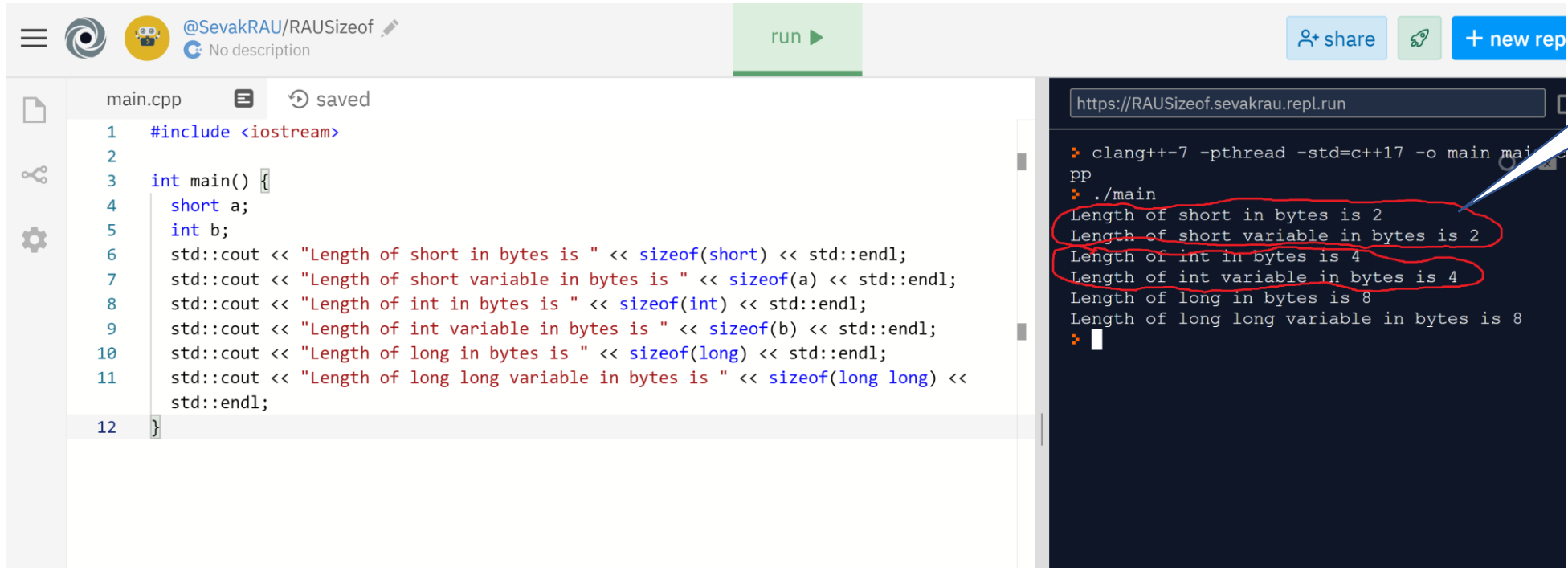
Տիպ կամ այդ տիպի  
փոփոխական

<https://repl.it/@SevakRAU/RAUSizeof>



# *sizeof* օպերատոր

*sizeof* օպերատորը տրված տիպի կամ փոփոխականի համար վերադարձնում է դրա չափը բայթերով



The screenshot shows a C++ REPL interface. On the left, the source code in `main.cpp` is displayed, showing the use of `sizeof` to determine the size of various data types and variables. On the right, the output of the program is shown, with the results of the `sizeof` operations circled in red. A speech bubble points to the output, indicating the result is in bytes.

```
main.cpp saved
1  #include <iostream>
2
3  int main() {
4      short a;
5      int b;
6      std::cout << "Length of short in bytes is " << sizeof(short) << std::endl;
7      std::cout << "Length of short variable in bytes is " << sizeof(a) << std::endl;
8      std::cout << "Length of int in bytes is " << sizeof(int) << std::endl;
9      std::cout << "Length of int variable in bytes is " << sizeof(b) << std::endl;
10     std::cout << "Length of long in bytes is " << sizeof(long) << std::endl;
11     std::cout << "Length of long long variable in bytes is " << sizeof(long long) <<
12     std::endl;
13 }
```

```
https://RAUSizeof.sevakrau.repl.run
> clang++-7 -pthread -std=c++17 -o main main.cpp
> ./main
Length of short in bytes is 2
Length of short variable in bytes is 2
Length of int in bytes is 4
Length of int variable in bytes is 4
Length of long in bytes is 8
Length of long long variable in bytes is 8
>
```

Չափեր

<https://repl.it/@SevakRAU/RAUSizeof>





# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b;  
}
```



Ինչու -1 ?

**a = -1, b = 4294967295**

<https://repl.it/@SevakRAU/RAUBigInt>



# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b;  
}
```

Չափը 4 բայթ է, 32 բիթ



**a = -1, b = 4294967295**

<https://repl.it/@SevakRAU/RAUBigInt>



# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b;  
}
```

11111111111111111111111111111111  
(32 հայտ 1)



**a = -1, b = 4294967295**

<https://repl.it/@SevakRAU/RAUBigInt>





## 29

## 30



# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b  
}
```



**a = -1, b = 4294967295**

11111111111111111111111111111111  
(32 հատ 1)

Նշանի բիթը 1 է, նշանակում է թիվը  
բացասական թիվ է լրացուցիչ կոդով

<https://repl.it/@SevakRAU/RAUBigInt>

Ուղիղ կոդը ստանալու համար պետք է հանել 1 և ժխտել

1. 11111111111111111111111111111111
2. 11111111111111111111111111111110 (-1)



# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b  
}
```



**a = -1, b = 4294967295**

11111111111111111111111111111111  
(32 հատ 1)

Նշանի բիթը 1 է, նշանակում է թիվը  
բացասական թիվ է լրացուցիչ կոդով

<https://repl.it/@SevakRAU/RAUBigInt>

Ուղիղ կոդը ստանալու համար պետք է հանել 1 և ժխտել

1. 11111111111111111111111111111111

2. 11111111111111111111111111111110 (-1)

3. 00000000000000000000000000000001 (ժխտում)



# Առանց նշանի ամբողջ տիպեր, օրինակ

```
#include <iostream>
```

```
int main() {  
    int a = 4294967295;  
    unsigned int b = 4294967295;  
    std::cout << "a = " << a << ", b = " << b  
}
```



**a = -1, b = 4294967295**

<https://repl.it/@SevakRAU/RAUBigInt>

11111111111111111111111111111111  
(32 հատ 1)

Նշանի բիթը 1 է, նշանակում է թիվը  
բացասական թիվ է լրացուցիչ կոդով

Ուղիղ կոդը ստանալու համար պետք է հանել 1 և ժխտել

1. 11111111111111111111111111111111

2. 11111111111111111111111111111110 (-1)

3. 00000000000000000000000000000001 (ժխտում)

1

-1



# Միավորային (char) տիպ

*char* տիպը օգտագործվում է մեկ սիմվոլ պահելու համար

```
#include <iostream>
int main() {
    char oneSymbol = 'X';
    std::cout << "Symbol is " << oneSymbol << std::endl;
}
```



Symbol is X

<https://repl.it/@SevakRAU/RAUChar>



# ASCII ٧٨٨

Character	Decimal Value	Character	Decimal Value
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122



# ASCII կոդ, օրինակ

```
#include <iostream>

int main() {
    char symbolCode = 65;
    std::cout << "Symbol is " << symbolCode << std::endl;
}
```



Symbol is A

<https://repl.it/@SevakRAU/RAUASCII>

Character	Decimal Value	Character	Decimal Value
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122





# ASCII կոդ, օրինակ

```
#include <iostream>
```

```
int main() {  
    char symbolCode = 65;  
    std::cout << "Symbol is " << symbolCode << std::endl;  
}
```



Symbol is A

<https://repl.it/@SevakRAU/RAUASCII>

Character	Decimal Value	Character	Decimal Value
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122



# ASCII կոդ, օրինակ

```
#include <iostream>
```

```
int main() {  
    char symb = 'c';  
    char symb1 = symb - 1;  
    char symb2 = symb + 3;  
    std::cout << "symb is " << symb << std::endl;  
    std::cout << "symb1 is " << symb1 << std::endl;  
    std::cout << "symb2 is " << symb2 << std::endl;  
}
```

↓  
symb is c  
symb1 is b  
symb2 is f

<https://repl.it/@SevakRAU/ASCIIArith>

Character	Decimal Value	Character	Decimal Value
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122



# ASCII կոդ, օրինակ

```
#include <iostream>
```

```
int main() {  
    char symb = 'c';  
    char symb1 = symb - 1;  
    char symb2 = symb + 3;  
    std::cout << "symb is " << symb << std::endl;  
    std::cout << "symb1 is " << symb1 << std::endl;  
    std::cout << "symb2 is " << symb2 << std::endl;  
}
```

↓  
symb is c  
symb1 is b  
symb2 is f

Character	Decimal Value	Character	Decimal Value
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122

<https://repl.it/@SevakRAU/ASCIIArith>



# ASCII կոդ, օրինակ

```
#include <iostream>

int main() {
    char symb = 'x';
    int code = symb;
    std::cout << "code of x is " << code << std::endl;
}
```



code of x is 120

<https://repl.it/@SevakRAU/ASCIIToInt>

Character	Decimal Value	Character	Decimal Value
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122





# ASCII կոդ, օրինակ

```
#include <iostream>

int main() {
    char symb = 'x';
    int code = symb;
    std::cout << "code of x is " << code << std::endl;
}
```

code of x is 120

Character	Decimal Value	Character	Decimal Value
A	65	a	97
B	66	b	98
C	67	c	99
D	68	d	100
E	69	e	101
F	70	f	102
G	71	g	103
H	72	h	104
I	73	i	105
J	74	j	106
K	75	k	107
L	76	l	108
M	77	m	109
N	78	n	110
O	79	o	111
P	80	p	112
Q	81	q	113
R	82	r	114
S	83	s	115
T	84	t	116
U	85	u	117
V	86	v	118
W	87	w	119
X	88	x	120
Y	89	y	121
Z	90	z	122

<https://repl.it/@SevakRAU/ASCIIToInt>



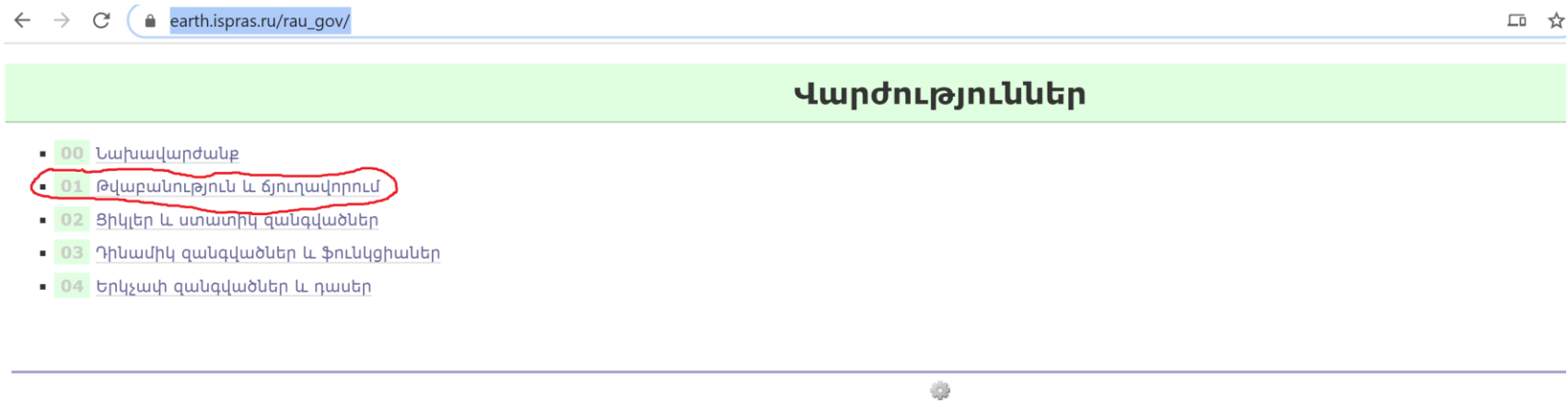
# Խնդիրներ

1. Մուտքագրված լատիներեն մեծատառի համար տպել համապատասխան փոքրատառը
2. Մուտքագրված լատիներեն փոքրատառի համար տպել համապատասխան մեծատառը
3. Մուտքագրված ASCII կոդի համար տպել համապատասխան սիմվոլը
4. Մուտքագրված սիմվոլի համար տպել համապատասխան ASCII կոդը





# Տնային աշխատանք




[https://earth.ispras.ru/rau\\_gov/](https://earth.ispras.ru/rau_gov/)



# Տնային աշխատանք

## 7-10 Խնդիրները տնային աշխատանք



Севак Саргсян [Մաս 1: Թվաբանություն և ճյուղավորում]: Сдать решение

НастройкиИнфоИтогПосылкиПоложение участниковОтправить вопросСообщенияВыйти

11:23:29 / RUNNING

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----	----	----	----	----

### Сдать решение задачи 10-Այբուբեն

Ограничение времени: 1 с  
Ограничение памяти: 64М  
Оставшиеся послылки: 16

### Задача 10: Այբուբեն

Գրել ծրագիր, որը կհաշվի և կարտածի անգլերեն լեզվի այբուբենի տառերի քանակը: (դաս 3)

Сдать решение

Язык: c++ - GNU C++ 4.4.3



# Շնորհակալություն. Հարցեր?

