Учреждение образования «Полоцкий государственный университет имени Евфросинии Полоцкой»

Кафедра технологий программирования

Отчёт по курсам «Введение в программирование (С++)» и «Программирование на языке (С++)» на stepik.org

Дисциплина «Ознакомительная практика»

(практическая часть курса)

Выполнил: гр. 22-ИТзд

Шастовская М.С.

Проверил: ст. преподаватель

Виноградова А. Д.,

**Полоцк, 2023**

**Раздел 1.2 Знакомство со средой разработки**

**Задача 1.**

Напишите программу, выводящую «Hello, World!». Можно скопировать ее из конспекта.

**Код:**

#include <iostream>

using namespace std;

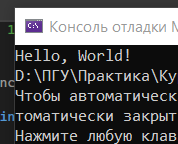
int main() {

cout << "Hello, World!" ;

return 0;

}

**Результат:**



**Задача 2.**

А теперь нужно поздороваться с нашей тестирующей системой. Исправьте предыдущую программу и выведите "Hello, Stepik!".

**Код:**

#include <iostream>

using namespace std;

int main() {

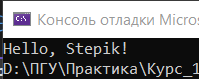
//cout << "Hello, World!";

cout << "Hello, Stepik!";

return 0;

}

**Результат:**



**Раздел 1.3 Целые числа**

**Задача 1.**

N белочек нашли K орешков и решили разделить их поровну. Определите, сколько орешков достанется каждой белочке.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n, k;

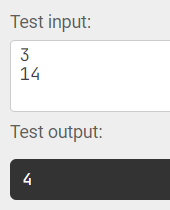
cin >> n >> k;

cout << k / n;

return 0;

}

**Результат:**



**Задача 2.**

N белочек нашли K орешков и решили разделить их поровну. Определите, сколько орешков останется после того, как все белочки возьмут себе равное количество орешков.

**Код:**

#include <iostream>

using namespace std;

int main() {

int k, n, c;

cin >> k >> n;

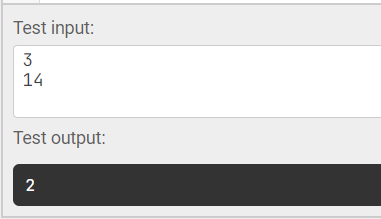
c = n % k;

cout << c;

return 0;

}

**Результат:**



**Задача 3.**

**Последняя цифра**

Дано натуральное число, выведите его последнюю цифру.

**Формат входных данных**

На вход дается натуральное число N, не превосходящее 10000.

**Формат выходных данных**

Выведите одно целое число - ответ на задачу.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n;

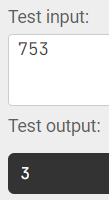
cin >> n;

cout << n % 10;

return 0;

}

**Результат:**



**Задача 4.**

**Первая цифра двузначного числа**

Дано двузначное число. Выведите его первую цифру (число десятков)

**Формат входных данных**

На вход дается натуральное двузначное число N.

Формат выходных данных

**Выведите одно целое число - ответ на задачу.**

**Код:**

#include <iostream>

using namespace std;

int main() {

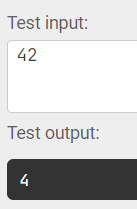
int a;

cin>>a;

cout<<a/10;

}

**Результат:**



**Задача 5.**

**Число десятков**

Дано целое неотрицательное число N, определите число десятков в нем (предпоследнюю цифру числа). Если предпоследней цифры нет, то можно считать, что число десятков равно нулю.

**Формат входных данных**

На вход дается целое положительное число N (0 ≤ N ≤ 1000000).

**Формат выходных данных**

Выведите одно целое число - ответ на задачу.

**Код:**

#include <iostream>

int main() {

int n, a;

std:: cin >> n;

a = n;

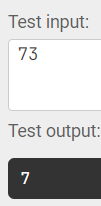
a = a / 100 \* 100;

std:: cout << (n - a)/10;

return 0;

}

**Результат:**



**Задача 6.**

**Сумма цифр трехзначного числа**

Дано целое трехзначное число. Найдите сумму его цифр.

**Формат входных данных**

На вход дается число от 100 до 999.

**Формат выходных данных**

Выведите одно целое число - ответ на задачу.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n;

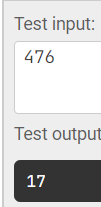
cin >> n;

cout << n / 100 + n % 100 / 10 + n % 10;

return 0;

}

**Результат:**



**Задача 7.**

**Следующее четное**

На вход дается натуральное число N. Выведите следующее за ним четное число

**Формат входных данных**

На вход дается целое положительное число N, не превышающее 10000.

**Формат выходных данных**

Выведите одно целое число - ответ на задачу.

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int a, b;

cin >> a;

b = a % 2;

if (b == 0)

{

a = a + 2;

cout << a << endl;

}

else

{

a = a + 1;

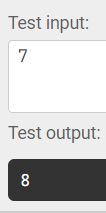
cout << a << endl;

}

return 0;

}}

**Результат:**



**Задача 8.**

**Парты**

В некоторой школе решили набрать три новых математических класса и оборудовать кабинеты для них новыми партами. За каждой партой может сидеть два учащихся. Известно количество учащихся в каждом из трех классов. Выведите наименьшее число парт, которое нужно приобрести для них. Каждый класс сидит в своем кабинете.

**Формат входных данных**

Программа получает на вход три целых неотрицательных числа: количество учащихся в каждом из трех классов (числа не превышают 1000).

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main () {

int a, b, c;

cin >> a >> b >> c;

if ( a % 2 != 0 ) {

a++;

}

if ( b % 2 != 0 ) {

b++;

}

if ( c % 2 != 0 ) {

c++;

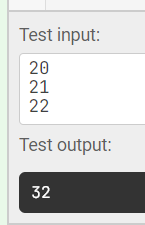
}

cout << a / 2 + b / 2 +c / 2;

return 0;

}

**Результат:**



**Задача 9.**

**Покупка пирожков**

Пирожок в столовой стоит A рублей и B копеек. Определите, сколько рублей и копеек нужно заплатить за N пирожков.

**Формат входных данных**

Программа получает на вход три числа: A, B, N - целые, положительные, не превышают 10000.

**Формат выходных данных**

Программа должна вывести два числа через пробел: стоимость покупки в рублях и копейках.

**Код:**

#include <iostream>

using namespace std;

int main() {

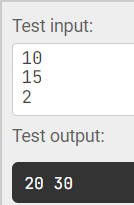
int rubles, kopeika, kolvo;

cin >> rubles >> kopeika >> kolvo;

cout << (rubles \* kolvo) + (kopeika\*kolvo) / 100 << " " << (kopeika \* kolvo) % 100;

return 0;

}**Результат:**



**Задача 10.**

**Электронные часы**

Электронные часы показывают время в формате h:mm:ss (от 0:00:00 до 23:59:59), то есть сначала записывается количество часов, потом обязательно двузначное количество минут, затем обязательно двузначное количество секунд. Количество минут и секунд при необходимости дополняются до двузначного числа нулями.

С начала суток прошло N секунд. Выведите, что покажут часы.

**Код:**

#include <iostream>

using namespace std;

int main()

{

int N;

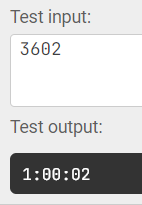
string y;

cin >> N;

cout << (N / 3600 - (N / 3600) / 24 \* 24) << ":" << (((N - (N / 3600) \* 3600) / 60) / 10 ? "" : "0")<< (N - (N / 3600) \* 3600) / 60 << ":" << (((N - (N / 3600) \* 3600 - ((N - (N / 3600) \* 3600) / 60) \* 60) / 10) ? "" : "0") << N - (N / 3600) \* 3600 - ((N - (N / 3600) \* 3600) / 60) \*60;

return 0;

}**Результат:**



**Задача 11.**

**Разность времен**

Даны значения двух моментов времени, принадлежащих одним и тем же суткам: часы, минуты и секунды для каждого из моментов времени. Известно, что второй момент времени наступил не раньше первого. Определите, сколько секунд прошло между двумя моментами времени.

**Формат входных данных**

Программа на вход получает три целых числа: часы, минуты, секунды, задающие первый момент времени и три целых числа, задающих второй момент времени.

**Формат выходных данных**

Выведите число секунд между этими моментами времени.

**Код:**

#include <iostream>

using namespace std;

int main()

{

int x, c, v, b, n, m;

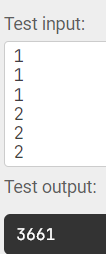
cin >> x >> c >> v >> b >> n >> m;

cout << (b \* 3600 + n \* 60 + m) - (x \* 3600 + c \* 60 + v);

return 0;

}

**Результат:**



**Задача 12.**

**МКАД**

Длина Московской кольцевой автомобильной дороги —109 километров. Байкер Вася стартует с нулевого километра МКАД и едет со скоростью V километров в час. На какой отметке он остановится через T часов?

**Формат входных данных**

Программа получает на вход целые числа V и T. Если V > 0, то Вася движется в положительном направлении по МКАД, если же значение V < 0, то в отрицательном. 0 ≤ T ≤ 1000, -1000 ≤ V ≤ 1000

**Формат выходных данных**

Программа должна вывести целое число от 0 до 108 — номер отметки, на которой остановится Вася.

**Код:**

#include <iostream>

using namespace std;

int main() {

int a, b, s;

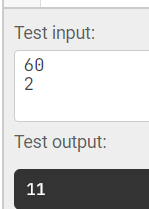
cin >> a >> b ;

s = 109 ;

cout << (s \* 1000 + a\*b) % s;

return 0;}

**Результат:**



**Задача 13.**

**Симметричное число**

Дано четырехзначное число. Определите, является ли его десятичная запись симметричной. Если число симметричное, то выведите 1, иначе выведите любое другое целое число. Число может иметь меньше четырех знаков, тогда нужно считать, что его десятичная запись дополняется слева незначащими нулями.

**Формат входных данных**

Вводится единственное число.

**Формат выходных данных**

Выведите одно целое число - ответ на задачу.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n;

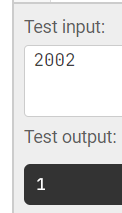
cin >> n;

cout << n/100 - n%100%10\*10 - n%100/10 + 1;

return 0;

}

**Результат:**



**Задача 14.**

**Улитка**

Улитка ползет по вертикальному шесту высотой H метров, поднимаясь за день на A метров, а за ночь спускаясь на B метров. На какой день улитка доползет до вершины шеста?

**Формат входных данных**

Программа получает на вход целые неотрицательные числа H, A, B, причем H > B. Числа не превосходят 100.

**Формат выходных данных**

Программа должна вывести одно натуральное число. Гарантируется, что A > B.

**Код:**

#include <iostream>

using namespace std;

int main() {

int h,a,b;

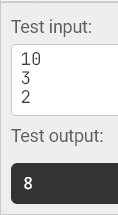
cin >> h >> a >> b;

cout << (h-a+a-b-1)/(a-b)+1;

return 0;

}

**Результат:**



**1.4. Условный оператор.**

**Задание 1.**

Даны два целых числа. Выведите значение наибольшего из них. Если числа равны, выведите любое из них.

**Код:**

#include <iostream>

using namespace std;

int main()

{

int a, b;

cin >> a >> b;

if (a>b)

{

cout << a;

}

else

{

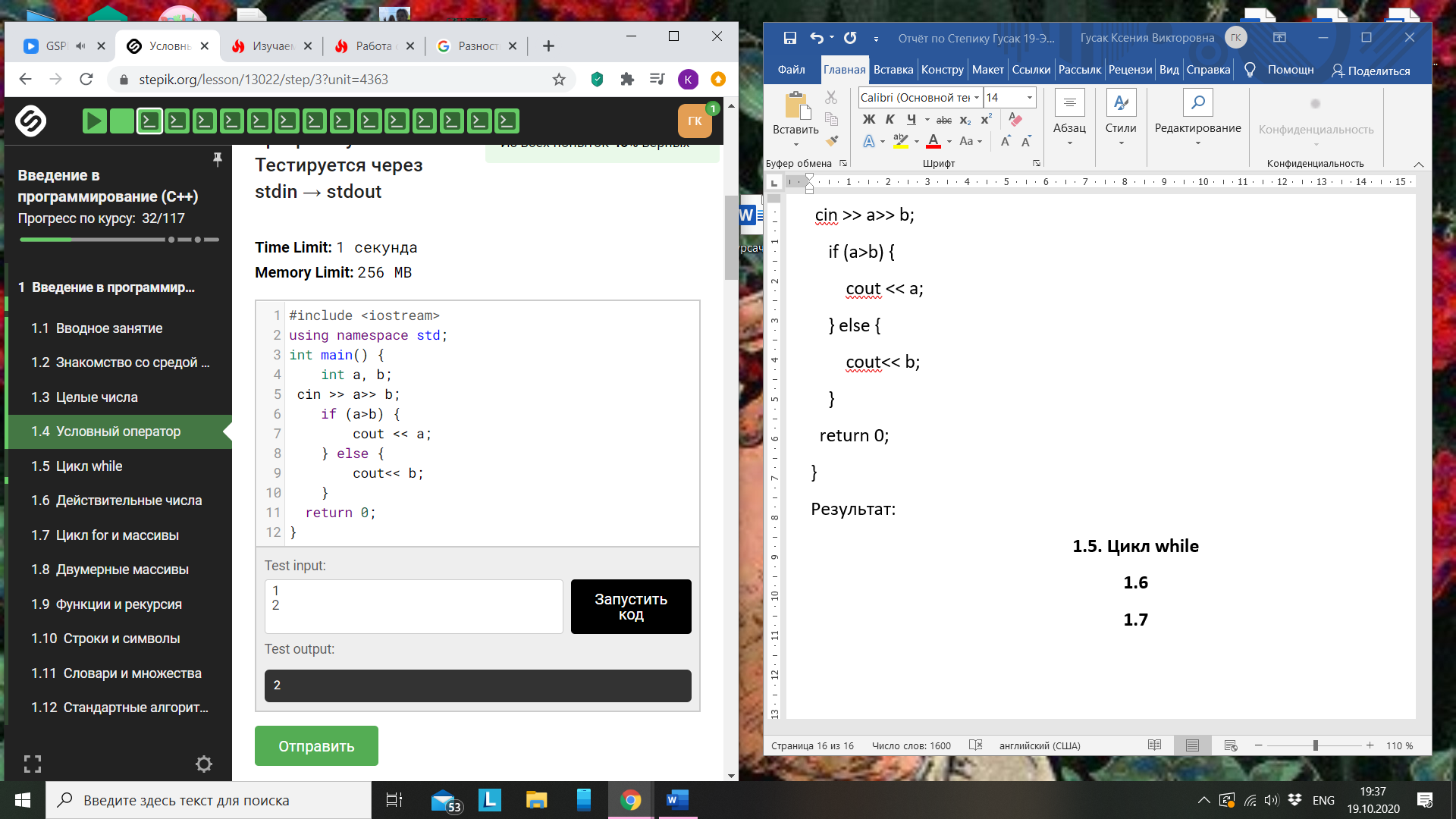
cout << b;

}

return 0;

}

**Результат:**



**Задание 2.**

Даны два целых числа. Выведите значение наибольшего из них. Если числа равны, выведите любое из них.

**Код:**

#include <iostream>

int main()

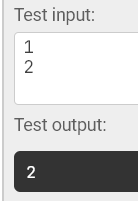
{

int a, b; std::cin >> a >> b;

std::cout << (a > b ? '1' : a == b ? '0' : '2');

}

**Результат:**



**Задание 3.**

Даны три целых числа. Найдите наибольшее из них (программа должна вывести ровно одно целое число). Под наибольшим в этой задаче понимается число, которое не меньше, чем любое другое.

**Код:**

#include <iostream>

using namespace std;

int main() {

int a,b,c;

cin>>a>>b>>c;

if(a>=b&&a>=c) {

cout<<a;

} else if(b>=a&&b>=c) {

cout<<b;

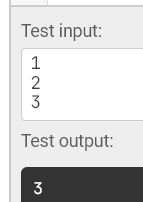
} else if(c>=a&&c>=b) {

cout<<c;

}

}

**Результат:**



**Задание 4.**

Даны три натуральных числа A, B, C. Определите, существует ли треугольник с такими сторонами. Если треугольник существует, выведите строку YES, иначе выведите строку NO.

Треугольник — это три точки, не лежащие на одной прямой.

**Код:**

#include <iostream>

using namespace std;

int main() {

int a, b, c;

cin >> a >> b >> c;

if (a<b+c && b<a+c && c<a+b) {

cout << "YES";

}

else {

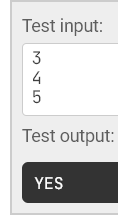
cout << "NO";

}

return 0;

}}

**Результат:**



**Задание 5.**

Даны три целых числа. Определите, сколько среди них совпадающих. Программа должна вывести одно из чисел: 3 (если все совпадают), 2 (если два совпадает) или 0 (если все числа различны).

**Код:**

#include <iostream>

using namespace std;

int main() {

int a, b, c;

cin >> a >> b >> c;

if(a == b && a == c && b == c) {

cout << 3;

}

else if (a != b && a != c && b != c){

cout << 0;

}

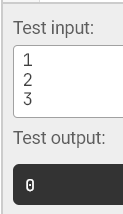
else {

cout << 2;

}

}

**Результат:**



**Задание 6.**

Шахматная ладья ходит по горизонтали или вертикали. Даны две различные клетки шахматной доски, определите, может ли ладья попасть с первой клетки на вторую одним ходом.

**Код:**

#include <iostream>

using namespace std;

int main() {

int a, b, c;

cin >> a >> b >> c;

if(a == b && a == c && b == c) {

cout << 3;

}

else if (a != b && a != c && b != c){

cout << 0;

}

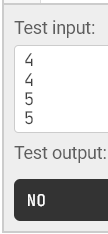
else {

cout << 2;

}

}

**Результат:**



**Задание 7.**

Шахматный король ходит по горизонтали, вертикали и диагонали, но только на 1 клетку. Даны две различные клетки шахматной доски, определите, может ли король попасть с первой клетки на вторую одним ходом.**Код:**

**Код:**

#include <iostream>

using namespace std;

int main()

{

int a, b, a1, b1;

cin >> a >> b >> a1 >> b1;

if (((a == a1) || (a == a1 - 1) || (a == a1 + 1)) && ((b == b1) || (b == b1 + 1) || (b == b1 - 1)))

{

cout << "YES";

}

else

{

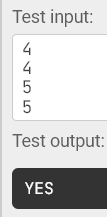
cout << "NO";

}

return 0;

}

**Результат:**



**Задание 8.**

Шахматный слон ходит по диагонали. Даны две различные клетки шахматной доски, определите, может ли слон попасть с первой клетки на вторую одним ходом.

**Код:**

#include <iostream>

using namespace std;

int main() {

int a, b, c, d, z, s;

cin >> a >> b >> c >> d;

z = c - a;

s = d - b;

if (z == s || z == (-1)\*s || (-1)\*z ==s)

cout << "YES" << endl;

else {

cout << "NO" << endl;

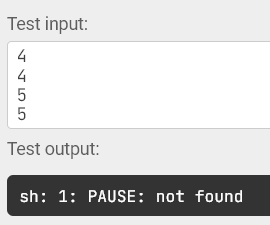
}

system("PAUSE");

return 0;

}

**Результат:**



**Задание 9.**

Шахматный ферзь ходит по диагонали, горизонтали или вертикали. Даны две различные клетки шахматной доски, определите, может ли ферзь попасть с первой клетки на вторую одним ходом.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {

int x1, y1, x2, y2;

cin>>x1>>y1>>x2>>y2;

if (x1 == x2 || y1 == y2 || abs(x1 - x2) == abs(y1 - y2)) {

cout<<"YES";

} else {

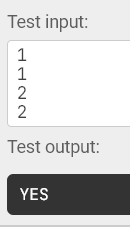
cout << "NO";

}

return 0;

}

**Результат:**



**Задание 10.**

Шахматный конь ходит буквой “Г” — на две клетки по вертикали в любом направлении и на одну клетку по горизонтали, или наоборот. Даны две различные клетки шахматной доски, определите, может ли конь попасть с первой клетки на вторую одним ходом.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {ios\_base::sync\_with\_stdio(NULL);cin.tie(nullptr);

int x1, y1, x2, y2;

cin>>x1>>y1>>x2>>y2;

int z = abs(x1 - x2);

int y = abs(y1 - y2);

if ((z == 1 && y == 2) || (z == 2 && y == 1)) {

cout<<"YES";

} else {

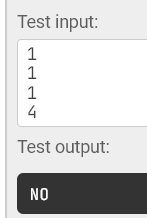
cout<<"NO";

}

return 0;

}

**Результат:**



**Задание 11.**

Шоколадка имеет вид прямоугольника, разделенного на N×M долек. Шоколадку можно один раз разломить по прямой на две части. Определите, можно ли таким образом отломить от шоколадки ровно K долек.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {

int n, m, k;

cin>>n>>m>>k;

if ((k % n == 0 && k / n < m) || (k % m == 0 && k / m < n)) {

cout << "YES";

} else {

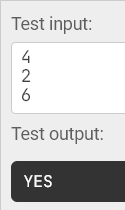
cout << "NO";

}

return 0;

}

**Результат:**



**Задание 12.**

Яша плавал в бассейне размером N×M метров и устал. В этот момент он обнаружил, что находится на расстоянии X метров от одного из длинных бортиков (не обязательно от ближайшего) и Y метров от одного из коротких бортиков. Какое минимальное расстояние должен проплыть Яша, чтобы выбраться из бассейна на бортик?

**Код:**

#include <iostream>

using namespace std;

int main() {

int n,m,x,y;

cin>>n>>m>>x>>y;

if(m<n)

swap(m,n);

int d=min(x,y);

int v=min(n-x,m-y);

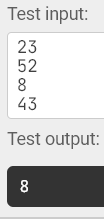
int g=min(d,v);

cout<<g;

return 0;

}

**Результат:**



**Задание 13.**

Дано три числа. Упорядочите их в порядке неубывания.

**Код:**

#include <iostream>

using namespace std;

int main() {

int a, b, c, min, max, sr, sum;

cin >> a >> b >> c;

if (a <= b && a <= c) {

min = a;

}

if (b <= a && b <= c) {

min = b;

}

if (c <= a && c <= b) {

min = c;

}

if (a >= b && a >= c) {

max = a;

}

if (b >= a && b >= c) {

max = b;

}

if (c >= a && c >= b) {

max = c;

}

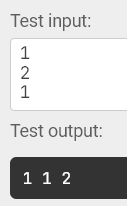
sr = (a + b + c) - (min + max);

cout << min << " " << sr << " " << max;

return 0;

}

**Результат:**



**Задание 14.**

Есть две коробки, первая размером A1×B1×C1, вторая размером A2×B2×C2. Определите, можно ли разместить одну из этих коробок внутри другой, при условии, что поворачивать коробки можно только на 90 градусов вокруг ребер.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {ios\_base::sync\_with\_stdio(NULL);cin.tie(nullptr);

int A1, B1, C1, A2, B2, C2;

cin >> A1 >> B1 >> C1 >> A2 >> B2 >> C2;

int s[] = {A1, B1, C1};

int z[] = {A2, B2, C2};

sort(s, s + 3);

sort(z, z + 3);

if (s[0] == z[0] && s[1] == z[1] && s[2] == z[2]) {

cout << "Boxes are equal";

} else if (s[0] >= z[0] && s[1] >= z[1] && s[2] >= z[2]) {

cout << "The first box is larger than the second one";

} else if (s[0] <= z[0] && s[1] <= z[1] && s[2] <= z[2]) {

cout << "The first box is smaller than the second one";

} else {

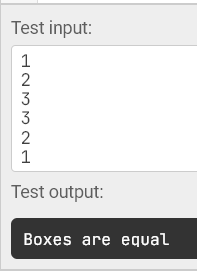
cout << "Boxes are incomparable";

}

return 0;

}}

**Результат:**



**1.5. Цикл while.**

**Задание 1.**

По данному целому числу N распечатайте все квадраты натуральных чисел, не превосходящие N, в порядке возрастания.

**Код:**

#include<bits/stdc++.h>

using namespace std;

int main(){ios\_base::sync\_with\_stdio(NULL);cin.tie(nullptr);

int n;

int a=1;

int z=a \* a;

cin>>n;

while(z<=n) {

cout<<z<<' ';

++a;

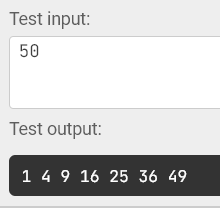
z=a\*a;

}

return 0;

}

**Результат:**



**Задание 2.**

Дано целое число, не меньшее 2. Выведите его наименьший натуральный делитель, отличный от 1.

**Код:**

#include<bits/stdc++.h>

using namespace std;

int main(){ios\_base::sync\_with\_stdio(NULL);cin.tie(nullptr);

int n,b=2;

cin>>n;

while(n>=b) {

if(n%b==0) {

break;

}

b=b+1;

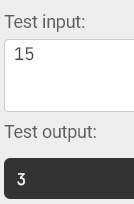
}

cout<<b;

return 0;

}

**Результат:**



**Задание 3.**

По данному числу N распечатайте все целые степени двойки, не превосходящие N, в порядке возрастания.

**Код:**

#include <iostream>

using namespace std;

int main()

{

int n, i=1;

cin >> n;

while (i<=n) {

cout << i << " ";

i=i\*2;

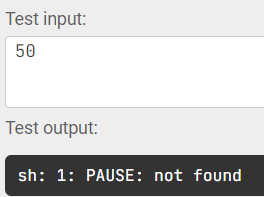
}

system ("PAUSE");

return EXIT\_SUCCESS;

}

**Результат:**



**Задание 4.**

Дано натуральное число N. Выведите слово YES, если число N является точной степенью двойки, или слово NO в противном случае.

**Код:**

#include<bits/stdc++.h>

using namespace std;

int main(){ios\_base::sync\_with\_stdio(NULL);cin.tie(nullptr);

int i, n;

cin >> n ;

i = 1;

while (n > i) {

i = i \* 2;

}

if (i == n) {

cout << "YES";

}

else {

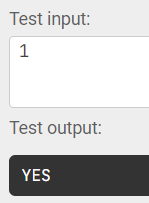
cout << "NO";

}

return 0;

}

**Результат:**



**Задание 5.**

Программа получает на вход последовательность целых неотрицательных чисел, каждое число записано в отдельной строке. Последовательность завершается числом 0, при считывании которого программа должна закончить свою работу и вывести количество членов последовательности (не считая завершающего числа 0).

Числа, следующие за числом 0, считывать не нужно.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main(){ios\_base::sync\_with\_stdio(NULL);cin.tie(nullptr);

int n;

int c=0;

while(true) {

cin>>n;

if (n==0) {

break;

}

c++;

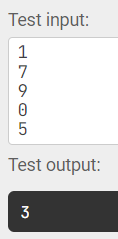
}

cout<<c<<endl;

return 0;

}

**Результат:**



**Задание 6.**

Определите сумму всех элементов последовательности, завершающейся числом 0.

**Код:**

#include <iostream>

using namespace std;

int main() {

int x = 1, y = 0;

while(x != 0){

cin >> x;

y = y + x;

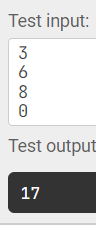
}

cout << y;

return 0;

}

**Результат:**



**Задание 7.**

Последовательность состоит из натуральных чисел и завершается числом 0. Определите значение наибольшего элемента последовательности.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n, max = 0;

cin >> n;

while(n != 0){

if(max <= n){

max=n;

}

cin >> n;

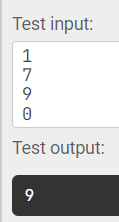
}

cout << max;

return 0;

}

**Результат:**



**Задание 8.**

Последовательность состоит из натуральных чисел и завершается числом 0. Определите, какое количество элементов этой последовательности, равны ее наибольшему элементу

**Код:**

#include <iostream>

using namespace std;

int main() {

int x = 1,z = 0,i = 1;

while ( x != 0) {

cin >> x;

if ( x == z ) {

i++;

}

if ( x > z ) {

z = x;

i = 1;

}

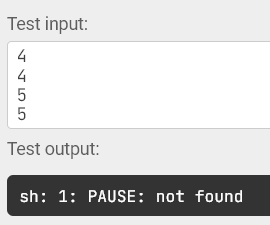
}

cout << i;

return 0;

}

**Результат:**



**Задание 9.**

Последовательность состоит из натуральных чисел и завершается числом 0. Определите значение второго по величине элемента в этой последовательности, то есть элемента, который будет наибольшим, если из последовательности удалить наибольший элемент.

**Код:**

#include <iostream>

using namespace std;

int main() {

int i, n, m = 1;

cin >> n;

i = n;

while (n != 0) {

cin >> n;

if (n >= i){

m = i;

i = n;

}

else if (n >= m)

{

m = n;

}

else

{

continue;

}

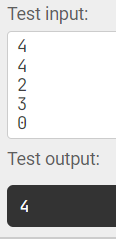
}

cout << m;

return 0;

}

**Результат:**



**Задание 10.**

Последовательность Фибоначчи определяется так:

F(0) = 0, F(1) = 1, …, F(n) = F(n−1) + F(n−2).

По данному числу N определите N-е число Фибоначчи F(N).

**Код:**

#include <iostream>

using namespace std;

int main() {

int fubNum,num=1,first=0,second=1,need;

cin>>fubNum;

if(fubNum==1){cout<<"1";return 0;}

while(num<fubNum){

need=first + second;

first=second;

second=need;

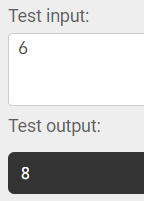
num++;

}

cout<<need;

}

**Результат:**



**Задание 11.**

Последовательность Фибоначчи определяется так:

F(0) = 0, F(1) = 1, …, F(n) = F(n−1) + F(n−2).

Дано натуральное число A. Определите, каким по счету числом Фибоначчи оно является, то есть выведите такое число N, что F(N) = A. Если А не является числом Фибоначчи, выведите число -1.

**Код:**

#include <iostream>

using namespace std;

int main() {

int N, f2 = 2, f1 = 1, i = 1, d = 0;

cin >> N;

while (i<=N) {

if (N == i) {

cout << f2;

return 0;

}

f2++;

d = i;

i+= f1;

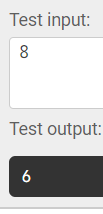
f1 = d;

}

cout << -1;

}

**Результат:**



**Задание 12.**

Дана последовательность натуральных чисел, завершающаяся числом 0. Определите, какое наибольшее число подряд идущих элементов этой последовательности равны друг другу. Если не нашлось ни одной пары, тройки и т.д. элементов, равных друг другу, то программа должна вывести число 1.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main(){ios\_base::sync\_with\_stdio(NULL);cin.tie(nullptr);

int n;

int z = -1;

int c = 1;

int s = 1;

while (true) {

std::cin >> n;

if (n == 0) {

break;

}

if (n == z) {

s++;

} else {

z = n;

s = 1;

}

if (s > c) {

c = s;

}

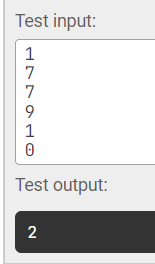
}

std::cout << c << std::endl;

return 0;

}

**Результат:**



**Задание 13.**

Элемент последовательности называется локальным максимумом, если он строго больше предыдущего и последующего элемента последовательности. Первый и последний элемент последовательности не являются локальными максимумами.

Дана последовательность натуральных чисел, завершающаяся числом 0. Определите количество строгих локальных максимумов в этой последовательности.

**Код:**

#include <iostream>

using namespace std;

int main() {

int prev = -1;

int cur = -1;

bool prev\_gr = false;

int cnt = 0;

if (cin >> prev && prev != 0) {

if (cin >> cur && cur != 0) {

int next;

if (cin >> next) {

while (next != 0) {

if (prev < cur && cur > next) {

++cnt;

}

prev = cur;

cur = next;

cin >> next;

}

}

}

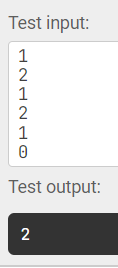
}

cout << cnt;

return 0;

}

**Результат:**



**Задание 14.**

Определите наименьшее расстояние между двумя локальными максимумами последовательности натуральных чисел, завершающейся числом 0. Если в последовательности нет двух локальных максимумов, выведите число 0.

Начальное и конечное значение при этом локальными максимумами не считаются.

Расстоянием считается количество пробелов между элементами. В качестве примера смотрите первый тест.

**Код:**

#include<iostream>

using namespace std;

int main() {

int prv, cur, nxt, pos = 0, prv\_max = 0, min\_len = 0, cur\_len;

int count = 0;

if (cin>>prv && prv)

{

if (cin >> cur && cur)

{

if (cin >> nxt)

{

pos = 2;

while (nxt)

{

if ((prv < cur) && (cur > nxt))

{

if (prv\_max != 0 )

{

cur\_len = pos - prv\_max;

if (min\_len == 0)

min\_len = cur\_len;

else

min\_len = min(min\_len,cur\_len);

}

prv\_max = pos;

}

prv = cur;

cur = nxt;

cin >> nxt;

pos++;

}

}

}

}

cout<<min\_len;

return 0;

}

**Результат:**



**1.5. Действительные числа**

**Задание 1.**

Дано положительное действительное число X. Выведите его дробную часть.

**Код:**

#include <iostream>

using namespace std;

int main() {

int a;

double b,c;

cin >> b;

c=b;

a=b;

cout << c-a;

return 0;

}

**Результат:**



**Задание 2.**

Дано положительное действительное число X. Выведите его первую цифру после десятичной точки.

**Код:**

#include <iostream>

using namespace std;

int main() {

double x;

cin >> x;

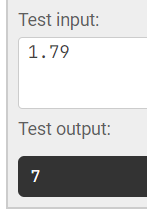
cout <<int (x\*10)%10;

// put your code here

return 0;

}

**Результат:**



**Задание 3.**

Даны длины сторон треугольника. Вычислите площадь треугольника.

**Код:**

#include <iostream>

#include <math.h>

using namespace std;

int main() {

// put your code here

//S = sqrt(p\*(p-a)\*(p-b)\*(p-c)), где sqrt - это корень, а p = (a + b + c) / 2.

//int a,b,c;

double a,b,c,p,s;

cin>>a;

cin>>b;

cin>>c;

p = (a+b+c)/2.0;

s = sqrt(p\*(p-a)\*(p-b)\*(p-c));

cout<<s;

return 0;

}

**Результат:**



**Задание 4.**

Процентная ставка по вкладу составляет P процентов годовых, которые прибавляются к сумме вклада в конце года. Вклад составляет X рублей Y копеек. Определите размер вклада через год.

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int p,x,y;

double i;

cin >> p>>x>>y;

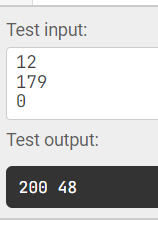
i=(x\*100+y)+(x\*100+y)\*p/100;

cout<<trunc(i/100)<<" "<< round((i/100-trunc(i/100))\*100);

return 0;

}

**Результат:**



**Задание 5.**

Процентная ставка по вкладу составляет P процентов годовых, которые прибавляются к сумме вклада через год. Вклад составляет X рублей Y копеек. Определите размер вклада через K лет.

**Код:**

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

int main() {

int k,a,b,c,d;

double p;

cin >> p >> b >> c >> k;

a=1;

d=b\*100+c;

while (a<=k){

a++;

d=d+trunc(d\*p/100);

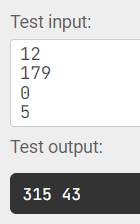
}

cout << d/100 << " " << d%100;

return 0;

}

**Результат:**



**Задание 6.**

Определите сумму всех элементов последовательности, завершающейся числом 0.

**Код:**

#include <iostream>

#include <iomanip>

#include <cmath>

using namespace std;

int main() {

int a, b = 0, i =0;

do

{

cin >> a;

b += a;

i++;

} while (a>0);

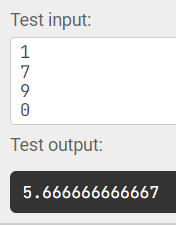
cout << fixed << setprecision(12);

cout << (double)b / (i-1);

return 0;

}

**Результат:**



**Задание 7.**

Дана последовательность натуральных чисел x1, x2, ..., xn. Стандартным отклонением называется величина

- среднее значение последовательности.

Определите стандартное отклонение для данной последовательности натуральных чисел, завершающейся числом 0.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {

long long n,b=0,x=0,m=0;

cin>>n;

m++;

while(n!=0){

b=b+n\*n;

x=x+n;

m++;

cin>>n;}

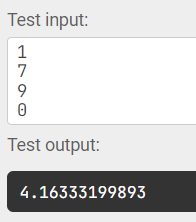
m--;

double j=b-floor(x\*x)/floor(m);

cout<<fixed<<setprecision(11)<<sqrt(j/(m-1));

return 0;

}**Результат:**



**Задание 8.**

Дан многочлен P(x)=anxn + an−1xn−1+ … + a1x + a0 и число x. Вычислите значение этого многочлена, воспользовавшись схемой Горнера:

P(x)=(…(((anx + an−1)x + an−2)x + an−3) … )x+ a0

**Код:**

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

int main() {

// put your code here

int n = 0;

double x = 0, a = 0, s = 0;

cin >> n >> x;

cin >> a;

do {

s \*= x;

s += a;

cin >> a;

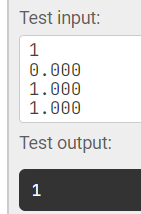
} while (n--);

cout << s;

return 0;

}

**Результат:**



**Задание 9.**

Даны действительные коэффициенты a, b, c, при этом a ≠ 0 . Решите квадратное уравнение ax2 + bx + c = 0 и выведите все его корни.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {

double a,b,c;

cin>>a>>b>>c;

double d=b\*b-4\*a\*c;

if(d>0){

double x1=(-b+sqrt(d))/(a\*2);

double x2=(-b-sqrt(d))/(a\*2);

cout<<fixed<<setprecision(10)<<min(x1,x2)<<" "<<max(x1,x2);}

else if(d==0){

cout<<(-b/(2\*a));}

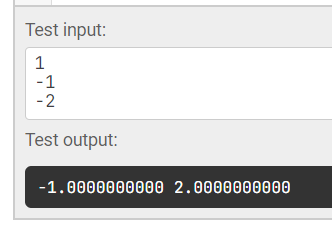
else{

cout<<" ";}

return 0;

}

**Результат:**



**Задание 10.**

Даны действительные коэффициенты a, b, c. Решите уравнение ax2 + bx + c = 0 и выведите все его корни.

**Код:**

#include <iostream>

#include <cmath>

int main() {

double a,b,c;

std::cin >> a>> b>> c;

std::cout << std::fixed << std::showpoint;

std::cout.precision(6);

if(a){

double d=b\*b-4\*a\*c;

if(d>0){

if (a>0)

std::cout<< 2 <<" "<< (-b-std::sqrt(d))/(2\*a) << " "<< (-b+std::sqrt(d))/(2\*a);

else

std::cout<< 2 <<" " << (-b+std::sqrt(d))/(2\*a) << " "<< (-b-std::sqrt(d))/(2\*a);

}else if(d==0)

std::cout<< 1 <<" " << (-b)/(2\*a);

else

std::cout << 0;

}else if(b)

std::cout<< 1 <<" " << -c/b;

else if(c)

std::cout<< 0;

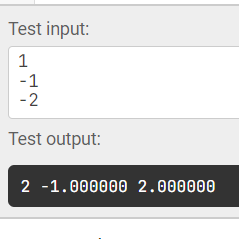
else

std::cout<< 3;

return 0;

}

**Результат:**



**Задание 11.**

Даны вещественные числа a, b, c, d, e, f. Известно, что система линейных уравнений имеет ровно одно решение. Выведите два числа x и y, являющиеся решением этой системы.

**Код:**

#include <iostream>

#include <cmath>

#include <iomanip>

using namespace std;

int main() {

double a, b, c, d, e, f;

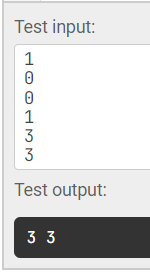
cin >> a >> b >> c >> d >> e >> f;

cout << ((e \* d - b \* f) / (a \* d - b \* c)) << " " << ((a \* f - c \* e) / (a \* d - b \* c));

return 0;

}

**Результат:**



**Задание 12.**

Даны вещественные числа a, b, c, d, e, f. Решите систему линейных уравнений

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

double eps = 0.000001;

double norm(double x) {

return fabs(x) <= eps ? 0 : x;

}

int main() {

double a, b, c, d, e, f, x, y, n , k;

cin >> a >> b >> c >> d >> e >> f;

double D, Dx, Dy;

D = a \* d - b \* c;

Dx = e \* d - b \* f;

Dy = a \* f - e \* c;

if (D != 0) {

cout << 2 << ' ' << norm(Dx / D) << ' ' << norm(Dy / D) << endl;

} else {

if (Dx == 0 && Dy == 0) {

if (a == 0 && b == 0 && c == 0 && d == 0) {

if (e != 0 || f != 0) {

cout << 0 << endl;

} else {

cout << 5 << endl;

}

} else {

if (a == 0 && c == 0) {

if (b != 0) {

y = e / b;

cout << 4 << ' ' << y << endl;

} else {

y = f / d;

cout << 4 << ' ' << y << endl;

}

} else {

if (b == 0 && d == 0) {

if (a != 0) {

x = e / a;

cout << 3 << ' ' << x << endl;

} else {

x = f / c;

cout << 3 << ' ' << x << endl;

}

} else {

if (b != 0) {

n = e / b;

k = -a / b;

cout << 1 << ' ' << k << ' ' << n << endl;

} else {

n = f / d;

k = -c / d;

cout << 1 << ' ' << k << ' ' << n << endl;

}

}

}

}

} else {

cout << 0 << endl;

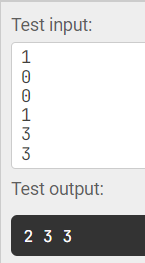
}

}

return 0;

}

**Результат:**



**1.7. Цикл for и массивы.**

**Задание 1.**

Выведите все элементы массива с четными индексами (то есть A[0], A[2], A[4], ...).

**Код:**

#include <iostream>

#include <vector>

using namespace std;

int main()

{

int n;

cin>>n;

vector <int> a(n);

for (int i=0;i<n;i++ ){

cin>> a[i];

}

for (int i = 0; i < n; i+=2) {

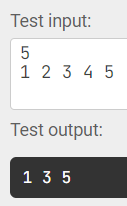
cout << a[i] << " ";

}

return 0;

}

**Результат:**



**Задание 2.**

Выведите все четные элементы массива.

**Код:**

#include <iostream>

using namespace std;

int main() {

int i = 0, n = 1,x = 0;

cin >> n;

while (i < n){

cin>>x;

if(x % 2 == 0)

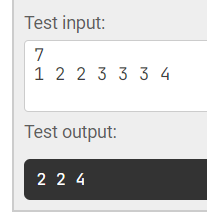
cout << x << ' ';

i++;

}

return 0;

}**Результат:**



**Задание 3.**

Найдите количество положительных элементов в данном массиве.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n,k=0;

cin>>n;

int a[n];

for(int i=0;i<n;i++)

cin>>a[i];

for(int i=0;i<n;i++){

if(a[i]>0)

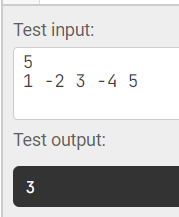
k++;}

cout<<k;

return 0;

}

**Результат:**



**Задание 4.**

Дан массив чисел. Выведите все элементы массива, которые больше предыдущего элемента .

**Код:**

#include <iostream>

using namespace std;

int main() {

int n;

cin>>n;

int a[n];

for(int i=0;i<n;i++){

cin>>a[i];}

for(int i=0;i<n;i++){

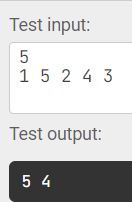
if(a[i]>a[i-1] && i>0)

cout<<a[i]<<" ";}

return 0;

}

**Результат:**



**Задание 5.**

Дан массив целых чисел. Если в нем есть два соседних элемента одного знака, выведите эти числа. Если соседних элементов одного знака нет - не выводите ничего. Если таких пар соседей несколько - выведите первую пару.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main ()

{

int a,cnt=0,c=0;

cin>>a;

int n[a];

for(int i=0;i<a;i++)

{

cin>>n[i];

}

for(int i=1;i<a;i++)

{

if(n[i]>0 and n[i-1]>0 or n[i]<0 and n[i-1]<0)cout<<n[i]<< " "<<n[i-1]<< " ";

}

}

**Результат:**



**Задание 6.**

Выведите значение наименьшего из всех положительных элементов в массиве. Известно, что в массиве есть хотя бы один положительный элемент.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {

int a,b,cnt=0;

cin>>a;

vector<int> v(a);

for (int i=0;i<a;i++) {

cin>>v[i];

}

for(int i=1;i<a;i++) {

if ((v[i] > 0) == (v[i-1] > 0)) {

cout << min(v[i-1], v[i]) << " " << max(v[i - 1], v[i]);

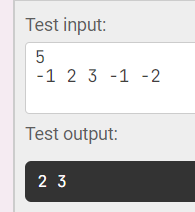
return 0;

}

}

}

**Результат:**



**Задание 7.**

Выведите значение наименьшего нечетного элемента списка, а если в списке нет нечетных элементов - выведите число 0.

**Код:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int n, min = 0;

cin >> n;

vector <int> a(n);

for (int i = 0; i < n; i++) {

cin >> a[i];

}

for (int i = 0; i < n; i++) {

if (a[i] % 2 != 0 && a[i] != 0) {

if (a[i] < min) {

min = a[i];

}

}

else if (a[i] % 2 == 0){

cout << "0";

}

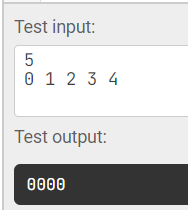
}

cout << min;

return 0;

}

**Результат:**



**Задание 8.**

Дан список, упорядоченный по неубыванию элементов в нем. Определите, сколько в нем различных элементов.

**Код:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int n = 0, uniq = 0;

cin >> n;

vector<int> a(n);

for (int i = 0; i < n; i++)

cin >> a[i];

for (int i = 0; i < n; i++) {

//cout << a[i] << " " << a[i+1] << endl;

if (a[i] != a[i+1]) uniq++; // обращаемся к элементу вне массива (n)

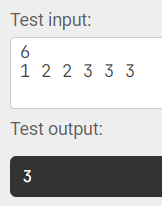
}

cout << uniq << endl;

return 0;

}

**Результат:**



**Задание 9.**

Переставьте соседние элементы массива (A[0] c A[1], A[2] c A[3] и т.д.). Если элементов нечетное число, то последний элемент остается на своем месте.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n;

cin >> n;

int arr[n];

for(int i = 0; i < n; i++) {

cin >> arr[i];

}

for(int i = 0; i < n - 1; i += 2) {

int temp = arr[i];

arr[i] = arr[i + 1];

arr[i + 1] = temp;

}

for(int i = 0; i < n; i++) {

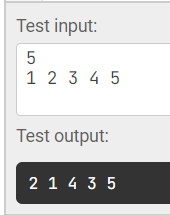
cout << arr[i] << " ";

}

return 0;

}

**Результат:**



**Задание 10.**

Циклически сдвиньте элементы списка вправо (A[0] переходит на место A[1], A[1] на место A[2], ..., последний элемент переходит на место A[0]).

**Код:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int n;

cin >> n;

vector<int> arr(n);

for(int i=0; i<n; i++) {

cin >> arr[i];

}

int temp = arr[n-1];

for(int i=n-1; i>0; i--) {

arr[i] = arr[i-1];

}

arr[0] = temp;

for(int i=0; i<n; i++) {

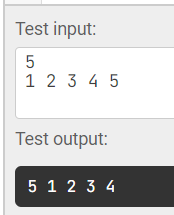
cout << arr[i] << " ";

}

return 0;

}

**Результат:**



**Задание 11.**

Дан массив чисел. Посчитайте, сколько в нем пар элементов, равных друг другу. Считается, что любые два элемента, равные друг другу образуют одну пару, которую необходимо посчитать.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n; cin>>n;

int a[n];

for (int g=0; g<n; g++)

{

cin>>a[g];

}

int k=0;

for (int i=0; i<n; i++)

{

for (int j=i+1;j<n;j++)

{

if (a[i]==a[j])

{

k++;

}

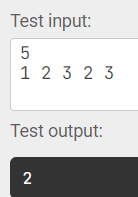
}

}

cout<<k;

}

**Результат:**



**Задание 12.**

Дан массив. Выведите те его элементы, которые встречаются в массиве только один раз. Элементы нужно выводить в том порядке, в котором они встречаются в списке.

**Код:**

#include <iostream>

#include <unordered\_map>

#include <vector>

using namespace std;

int main() {

int n;

cin >> n;

vector<int> a(n);

for (int i = 0; i < n; i++) {

cin >> a[i];

}

unordered\_map<int, int> count;

for (int i = 0; i < n; i++) {

count[a[i]]++;

}

for (int i = 0; i < n; i++) {

if (count[a[i]] == 1) {

cout << a[i] << " ";

}

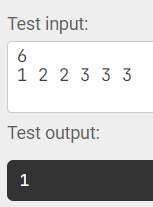
}

cout << endl;

return 0;

}

**Результат:**



**Задание 13.**

Известно, что на доске 8×8 можно расставить 8 ферзей так, чтобы они не били друг друга. Вам дана расстановка 8 ферзей на доске, определите, есть ли среди них пара бьющих друг друга.

**Код:**

#include <iostream>

using namespace std;

int main()

{

int x[8], y[8], x1, y1;

bool f = false;

for (int i=0; i < 8; i++)

{

cin >> x1 >> y1;

x[i] = x1; y[i] = y1;

}

for (int i = 0; i < 7; i++)

{

for (int j = i + 1; j < 8; j++)

{

if (x[i] == x[j] or y[i] == y[j] or abs(x[i] - x[j]) == abs(y[i] - y[j]))

{

f = true;

break;

}

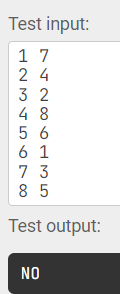
}

}

f ? cout << "YES": cout << "NO";

}

**Результат:**



**Задание 14.**

N кеглей выставили в один ряд, занумеровав их слева направо числами от 1 до N. Затем по этому ряду бросили K шаров, при этом i-й шар сбил все кегли с номерами от li до ri включительно. Определите, какие кегли остались стоять на месте.

**Код:**

#include<bits/stdc++.h>

using namespace std;

int main(){

int n, k;

cin >> n >> k;

vector <string> xe(n);

for (int i = 0; i < n; i++){

xe[i] = "I";

}

vector <int> x(k);

vector <int> y(k);

for( int j = 0; j < k; j++){

cin >> x[j];

cin >> y[j];

}

for(int i = 0; i < k; i++){

for (int j = x[i] - 1; j <=y[i] - 1; j++)

xe[j] = ".";

}

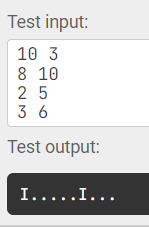
for (int i = 0; i < n; i++) {

cout << xe[i];

}

}

**Результат:**



**1.8. Двумерные массивы.**

**Задание 1.**

Найдите индексы первого вхождения максимального элемента.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n, m;

cin >> n >> m;

int a[100][100];

// чтение

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

cin >> a[i][j];

}

}

int max = a[0][0], max\_i = 0, max\_j = 0;

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

if (a[i][j] > max) {

max = a[i][j];

max\_i = i;

max\_j = j;

}

}

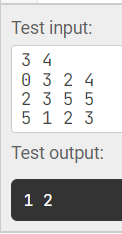
}

cout << max\_i << " " << max\_j;

return 0;

}

**Результат:**



**Задание 2.**

Дано нечетное число n, не превосходящее 15. Создайте двумерный массив из n×n элементов, заполнив его символами "." (каждый элемент массива является строкой из одного символа). Затем заполните символами "\*" среднюю строку массива, средний столбец массива, главную диагональ и побочную диагональ. В результате "\*" в массиве должны образовывать изображение звездочки. Выведите полученный массив на экран, разделяя элементы массива пробелами.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n;

cin >> n;

char a[15][15];

// обработка

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

if (i == j || i == n - 1 - j || i == n / 2 || j == n / 2) a[i][j] = '\*';

else a[i][j] = '.';

}

}

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cout << a[i][j] << " ";

}

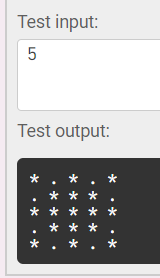
cout << endl;

}

return 0;

}

**Результат:**



**Задание 3.**

Дано число n, не превышающее 100. Создайте массив размером n×n и заполните его по следующему правилу. На главной диагонали должны быть записаны числа 0. На двух диагоналях, прилегающих к главной, числа 1. На следующих двух диагоналях числа 2, и т.д. Выведите полученный массив на экран, разделяя элементы массива пробелами.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {

int n;

cin >> n;

int a[101][101];

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

a[i][j] = abs(i - j);

cout << a[i][j]<<" ";

}

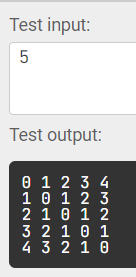
cout << endl;

}

return 0;

}

**Результат:**



**Задание 4.**

Дан двумерный массив и два числа: i и j. Поменяйте в массиве столбцы с номерами i и j

**Код:**

#include <iostream>

#include <vector>

using namespace std;

int main() {

int n,m,x = 0, y = 0 ;

cin >> n >> m;

int a[100][100];

// ввод

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

cin >> a[i][j];

}

}

cin >> x >> y;

//обработка

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

if (j == x) {

int temp = 0;

temp = a[i][j];

a[i][j] = a[i][y];

a[i][y] = temp;

}

}

}

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

cout << a[i][j] << " ";

}

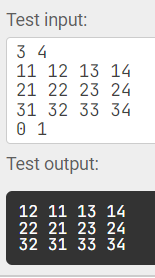
cout << endl;

}

return 0;

}

**Результат:**



**Задание 5.**

Дано число n, не превосходящее 10, и массив размером n × n. Проверьте, является ли этот массив симметричным относительно главной диагонали. Выведите слово “YES”, если массив симметричный, и слово “NO” в противном случае.

**Код:**

#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

int main()

{

int n,l=0;

cin>>n;

int mass[n][n];

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

cin>>mass[i][j];

}

}

for(int i=0;i<n;i++)

{

for(int j=0;j<n;j++)

{

if(mass[i][j]==mass[j][i])l++;

}

}

if(l==n\*n)cout<<"YES";

else cout<<"NO";

}

**Результат:**



**Задание 6.**

Дан квадратный двумерный массив размером n × n и число k. Выведите элементы k-й по счету диагонали ниже главной диагонали (т.е. если k = 1, то нужно вывести элементы первой диагонали, лежащей ниже главной, если k = 2, то второй диагонали и т.д.).

Значение k может быть отрицательным, например, если k = −1, то нужно вывести значение первой диагонали лежащей выше главной. Если k = 0, то нужно вывести элементы главной диагонали.

Программа получает на вход число n, не превосходящие 10, затем массив размером n × n, затем число k.

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

int main() {

int n, m, x, y, temp;

cin >> n ; int a[n][n];

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

cin >> a[i][j];

} }

cin >> m;

for (int i = 0; i < n; i++) {

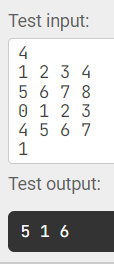
for (int j = 0; j < n; j++) {

if ((i-j)==m){cout << a[i][j]<< " ";}

}}

return 0; }

**Результат:**



**Задание 7.**

Дан двумерный массив размером n×m (n и m не превосходят 1000). Симметричный ему относительно главной диагонали массив называется транспонированным к данному. Он имеет размеры m×n: строки исходного массива становятся столбцами транспонированного, столбцы исходного массива становятся строками транспонированного.

Для данного массива постройте транспонированный массив и выведите его на экран.

**Код:**

#include <iostream>

using namespace std;

int main()

{

int n, m;

cin >> n >> m;

int a[n][m], b[m][n];

for(int i = 0; i < n; i++)

for(int j = 0; j < m; j++)

{

cin >> a[i][j];

b[j][i] = a[i][j];

}

for(int i = 0; i < m; i++)

{

for(int j = 0; j < n; j++)

cout << b[i][j] << ' ';

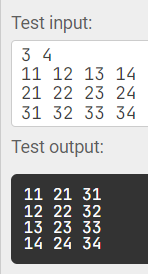
cout << endl;

}

return 0;

}

**Результат:**



**Задание 8.**

В кинотеатре n рядов по m мест в каждом (n и m не превосходят 20). В двумерном массиве хранится информация о проданных билетах, число 1 означает, что билет на данное место уже продан, число 0 означает, что место свободно. Поступил запрос на продажу k билетов на соседние места в одном ряду. Определите, можно ли выполнить такой запрос.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n, m, y=0, k;

cin >> n >> m;

int a[20][20];

for(int i=0; i<n; i++)

{

for(int i1=0; i1<m; i1++)

{

cin >> a[i][i1];

}

}

cin >> k;

int i1, c;

bool flag;

for(int i=0; i<n; i++)

{

flag = false;

for(i1=0; i1<m; i1++)

{

if(a[i][i1]==0 && flag == false)

{

c=0;

flag = true;

}

if(a[i][i1]==1 && flag == true)

{

if(c>=k && y==0)

{

y = i+1;

}

flag = false;

}

if(a[i][i1]==0 && flag == true)

{

c++;

}

}

if(flag == true)

{

if(c>=k && y==0)

{

y = i+1;

}

}

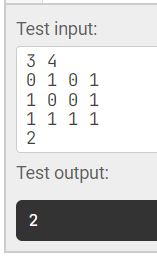
}

cout << y << endl;

return 0;

}

**Результат:**



**Задание 9.**

Дан прямоугольный массив размером n×m. Поверните его на 90 градусов по часовой стрелке, записав результат в новый массив размером m×n.

**Код:**

#include <iostream>

using namespace std;

int main() {

int n,m;

cin>>n>>m;

int a[100][100];

for (int i=0;i<n;i++){

for (int j=0;j<m;j++){

cin>>a[i][j];}

}

for (int i=0;i<m;i++){

for (int j=0;j<n;j++){

}

}

for (int i=0;i<m;i++){

for (int j=n-1;j>=0;j--){

int tmp=a[j][i];

cout<<tmp<<' ';

tmp-=1;}

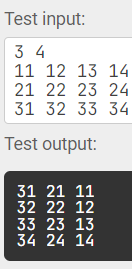
cout<<endl;

}

return 0;

}

**Результат:**



**Задание 10.**

По данным числам n и m заполните двумерный массив размером n×m числами от 1 до n×m “змейкой”, как показано в примере.

**Код:**

#include <iostream>

#include <iomanip>

using namespace std;

int main() {

int n, m;

cin >> n >> m;

int a[n][m];

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

a[i][j] = i \* m + (j + 1) \* (1 - i % 2) + (m - j) \* (i % 2);

}

}

for (int i = 0; i < n; ++i) {

for (int j = 0; j < m; ++j) {

cout << setw(4) << a[i][j];

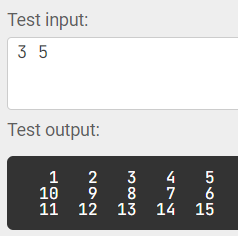
}

cout << endl;

}

}

**Результат:**



**Задание 11.**

По данным числам n и m заполните двумерный массив размером n×m числами от 1 до n×m “диагоналями”, как показано в примере.

**Код:**

#include <iostream>

using namespace std;

int main() {

// put your code here

int n,m,b=1;

cin>>n>>m;

int a[n][m];

for(int i=0;i<m;i++) { int k=0,j=i;

while(k<n && j>=0) {a[k][j]=b;b++;j--;k++;}}

for(int i=1;i<n;i++) { int k=m-1,j=i;

while(k>=0 && j<n) {a[j][k]=b;b++;k--;j++;}}

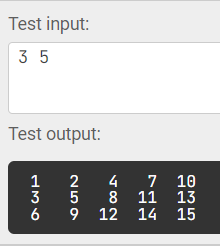
for(int i=0;i<n;i++){for(int j=0;j<m;j++){if(a[i][j]<10){cout<<" "<<a[i][j]<<" ";}

else{cout<<a[i][j]<<" ";}}cout<<endl;}

return 0;

}

**Результат:**



**Задание 12.**

Даны числа n и m. Заполните массив размером n × m в шахматном порядке: клетки одного цвета заполнены нулями, а другого цвета - заполнены числами натурального ряда сверху вниз, слева направо. В левом верхнем углу записано число 1.

**Код:**

#include <iostream>

#include <vector>

#include <iomanip>

using namespace std;

int main() {

int n, m, x = 1, y = 1, i = 0, j = 0, z = 0;

cin >> n >> m;

int a[100][100]{};

// ввод

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

a[i][j] = 0;

}

}

//обработка

if ((m % 2) != 0) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

z++;

if (z % 2 != 0) {

a[i][j] = x;

x++;

}

}

}

}

else if ((m % 2) == 0) {

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

if (((i + j) == 0)) {

a[i][j] = x;

x++;

}else if ((i + j) % 2 == 0) {

a[i][j] = x;

x++;

}

}

}

}

for (int i = 0; i < n; i++) {

for (int j = 0; j < m; j++) {

cout << setw(4) << a[i][j];

}

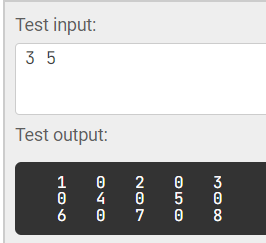
cout << endl;

}

return 0;

}

**Результат:**



**Задание 13.**

По данным числам n и m заполните двумерный массив размером n×m числами от 1 до n×m по спирали, выходящей из левого верхнего угла и закрученной по часовой стрелке, как показано в примере.

**Код:**

#include <iostream>

using namespace std;

int main()

{

int n, m;

cin >> n >> m;

int a[100][100] = {0};

int c = 1, k = 0, x\_s = 1, y\_s = 0, x = 0, y = 0;

for (int c = 1; c <= n\*m; c++)

{

a[y][x] = c;

x += x\_s;

y += y\_s;

if (x < 0 || y<0 || x>m-1 || y>n-1 || a[y][x] != 0)

{

x -= x\_s;

y -= y\_s;

k++;

if (k == 4)

{

k = 0;

}

if (k == 0)

{

x\_s = 1;

y\_s = 0;

}

if (k == 1)

{

x\_s = 0;

y\_s = 1;

}

if (k == 2)

{

x\_s = -1;

y\_s = 0;

}

if (k == 3)

{

x\_s = 0;

y\_s = -1;

}

x += x\_s;

y += y\_s;

}

}

cout<<" ";

for(int i = 1; i <= n; i++){

for(int j = 1; j <= m; j++){

std::cout.width(4); std::cout << a[i-1][j-1];

}

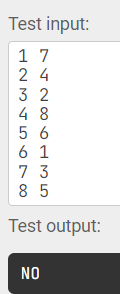
cout<<endl;

}

return 0;

}

**Результат:**



**1.9. Функции и рекурсия.**

**Задание 1.**

Напишите функцию min(a, b), вычисляющую минимум двух чисел. Затем напишите функцию min4(a, b, c, d), вычисляющую минимум 4 чисел с помощью функции min. Считайте четыре целых числа и выведите их минимум.

**Код:**

#include <iostream>

using namespace std;

/\* Напишите функцию min(a, b), вычисляющую минимум двух чисел. \*/

int min(int a, int b) {

return a < b ? a : b;

}

/\* Затем напишите функцию min4(a, b, c, d), вычисляющую минимум 4 чисел с помощью функции min. \*/

int min4(int a, int b, int c, int d) {

return min(min(a, b), min(c, d));

}

/\* Считайте четыре целых числа и выведите их минимум. \*/

int main() {

int a = 0, b = 0, c = 0, d = 0;

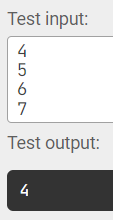
cin >> a >> b >> c >> d;

cout << min4(a, b, c, d) << endl;

return 0;

}

**Результат:**



**Задание 2.**

Даны четыре действительных числа: x1, y1, x2, y2. Напишите функцию distance(x1, y1, x2, y2), вычисляющую расстояние между точкой (x1. y1) и (x2, y2). Считайте четыре действительных числа и выведите результат работы этой функции.

**Код:**

#include <iostream>

#include <math.h>

using namespace std;

int d(int a,int b,int c,int d){

return a;}

int main() {

float a,b,c,d;

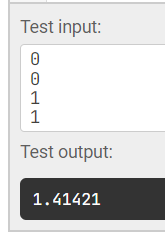
cin>>a>>b>>c>>d;

cout<<sqrt((d-b)\*(d-b)+(a-c)\*(a-c));

return 0;

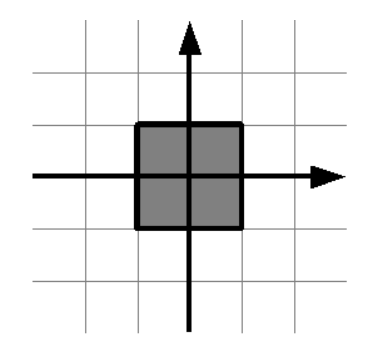
}

**Результат:**



**Задание 3.**

Даны два действительных числа x и y. Проверьте, принадлежит ли точка с координатами (x, y) заштрихованному квадрату (включая его границу). Если точка принадлежит квадрату, выведите слово YES, иначе выведите слово NO. На рисунке сетка проведена с шагом 1.



Решение должно содержать функцию IsPointInSquare(x, y), возвращающую true, если точка принадлежит квадрату и false, если не принадлежит. Основная программа должна считать координаты точки, вызвать функцию IsPointInSquare и в зависимости от возвращенного значения вывести на экран необходимое сообщение.

Функция IsPointInSquare не должна содержать инструкцию if.

**Код:**

#include <iostream>

using namespace std;

double IsPointInSquare( double x,double y){

(x<=1 && x>=-1) && ((y<=1 && y>=-1))?cout<<"YES" : cout<<"NO"; ///Тернарный оператор;

}

int main() {

double x , y;

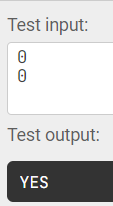
cin>>x>>y;

IsPointInSquare(x,y);

return 0;

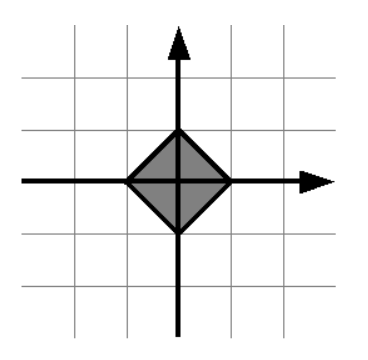
}

**Результат:**



**Задание 4.**

Даны два действительных числа x и y. Проверьте, принадлежит ли точка с координатами (x, y) заштрихованному квадрату (включая его границу). Если точка принадлежит квадрату, выведите слово YES, иначе выведите слово NO. На рисунке сетка проведена с шагом 1.



Решение должно содержать функцию IsPointInSquare(x, y), возвращающую True, если точка принадлежит квадрату и False, если не принадлежит. Основная программа должна считать координаты точки, вызвать функцию IsPointInSquare и в зависимости от возвращенного значения вывести на экран необходимое сообщение.

Функция IsPointInSquare не должна содержать инструкцию if.

**Код:**

#include <iostream>

#include <bits/stdc++.h>

using namespace std;

bool IsPointInSquare(float x, float y) {

bool a;

a=((y>=abs(x)-1)&&(y<=(-abs(x))+1));

return a;

}

int main() {

float x,y;

cin >> x >> y;

bool a = IsPointInSquare(x,y);

if (a) {cout << "YES";}

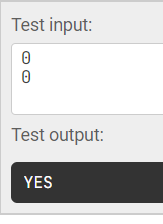
else {cout << "NO";}

// put your code here

return 0;

}

**Результат:**



**Задание 5.**

Даны пять действительных чисел: x, y, xc, yc, r. Проверьте, принадлежит ли точка (x, y) кругу с центром (xc, yc) и радиусом r. Если точка принадлежит кругу, выведите слово YES, иначе выведите слово NO.

Решение должно содержать функцию IsPointInCircle(x, y, xc, yc, r), возвращающую True, если точка принадлежит кругу и False, если не принадлежит. Основная программа должна считать координаты точки, вызвать функцию IsPointInCircle и в зависимости от возвращенного значения вывести на экран необходимое сообщение.

Функция IsPointInCircle не должна содержать инструкцию if.

**Код:**

#include <iostream>

#include <math.h>

using namespace std;

bool round(float a, float b, float x, float y, float r){

if(((x \* x) - (a \* a)) + ((y \* y) - (b \* b)) <= r \* r)

return true;

else

return false;

}

int main() {

float a, b, x, y, r;

cin >> a >> b >> x >> y >> r;

if(round(a, b, x, y, r) == 1)

cout << "YES";

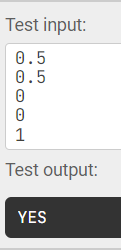
else

cout << "NO";

return 0;

}

**Результат:**



**Задание 6.**

Проверьте, принадлежит ли точка данной закрашенной области:



Если точка принадлежит области (область включает границы), выведите слово YES, иначе выведите слово NO.

Решение должно содержать функцию IsPointInArea(x, y), возвращающую True, если точка принадлежит области и False, если не принадлежит. Основная программа должна считать координаты точки, вызвать функцию IsPointInArea и в зависимости от возвращенного значения вывести на экран необходимое сообщение.

Функция IsPointInArea не должна содержать инструкцию if.

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

bool IsPointInArea(double x, double y)

{

return (pow(x+1,2)+pow(y-1,2)<=4 && y>=2\*x/1+2 && y>=-x) || (pow(x+1,2)+pow(y-1,2)>=4 && y<=2\*x/1+2 && y<=-x);

}

int main()

{

double x,y;

cin>>x>>y;

if( IsPointInArea(x, y))

{

cout<<"YES";

}

else

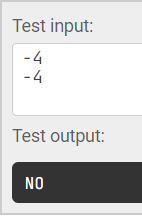
{

cout<<"NO";

}

}

**Результат:**



**Задание 7.**

Дано действительное положительное число a и целоe число n.

Вычислите an. Решение оформите в виде рекурсивной функции power(a, n).

**Код:**

#include <iostream>

double power(double a, int n) {

if (n == 0) {

return 1;

} else if (n > 0) {

return a \* power(a, n - 1);

} else {

return 1 / (a \* power(a, -n - 1));

}

}

int main() {

double a;

int n;

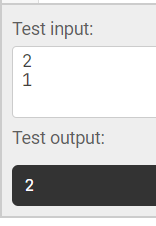
std::cin >> a >> n;

std::cout << power(a, n) << std::endl;

return 0;

}

**Результат:**



**Задание 8.**

Дано натуральное число n > 1. Выведите его наименьший делитель, отличный от 1.

Решение оформите в виде функции MinDivisor(n). Количество операций в программе должно быть пропорционально корню из n.

Указание. Если у числа n нет делителя, меньшего n , то число n — простое и ответом будет само число n.

**Код:**

#include <iostream>

#include <cmath>

#include <cstdlib>

#include <vector>

using namespace std;

int MinDivisor(int n) {

int p = 2, k = n;

while (p \* p <= n) {

if (n % p == 0) {

k = p;

break;

}

p = p + 1;

}

return k;

}

int main() {

int n;

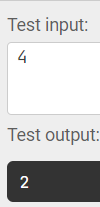
cin >> n;

cout << MinDivisor(n);

return 0;

}

**Результат:**



**Задание 9.**

Дано натуральное число n > 1. Проверьте, является ли оно простым. Программа должна вывести слово YES, если число простое и NO, если число составное.

Решение оформите в виде функции IsPrime(n), которая возвращает True для простых чисел и False для составных чисел. Количество операций в программе должно быть пропорционально корню из n.

**Код:**

#include <iostream>

bool IsPrime(int n) {

if (n <= 1) {

return false;

}

for (int i = 2; i \* i <= n; i++) {

if (n % i == 0) {

return false;

}

}

return true;

}

int main() {

int n;

std::cin >> n;

if (IsPrime(n)) {

std::cout << "YES";

} else {

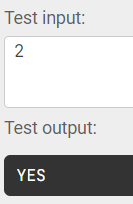
std::cout << "NO";

}

return 0;

}

**Результат:**



**Задание 10.**

Возводить в степень можно гораздо быстрее, чем за n умножений! Для этого нужно воспользоваться следующими рекуррентными соотношениями:

an = (a2)n/2 при четном n,

an = a × an−1 при нечетном n.

Реализуйте алгоритм быстрого возведения в степень с помощью рекурсивной функции.

**Код:**

#include <iostream>

double power(double a, int n) {

if (n == 0) {

return 1;

} else if (n % 2 == 0) {

double temp = power(a, n/2);

return temp \* temp;

} else {

return a \* power(a, n - 1);

}

}

int main() {

double a;

int n;

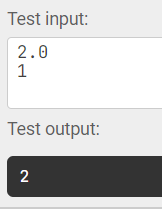
std::cin >> a >> n;

std::cout << power(a, n);

return 0;

}

**Результат:**



**Задание 11.**

Дана последовательность чисел, завершающаяся числом 0. Найдите сумму всех этих чисел, не используя цикл.

**Код:**

#include <iostream>

using namespace std;

double rec(int a,int sum)

{

if(a==0) return 1;

sum=sum+a;

cin>>a;

return rec(a,sum);

}

int main() {

// put your code here

int a;

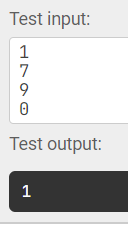
cin>>a;

cout<<rec(a,0);

return 0;

}

**Результат:**



**Задание 12.**

Напишите функцию fib(n), которая по данному целому положительному n возвращает n-e число Фибоначчи. В этой задаче нельзя использовать циклы - используйте рекурсию.

Первое и второе числа Фибоначчи равны 1, а каждое следующее равно сумме двух предыдущих.

**Код:**

#include <iostream>

using namespace std;

int fib (int n, int ans, int x, int y)

{

if (n == 1 || n == 2) {

return ans;

} else

{

ans = x + y;

x = y;

y = ans;

return fib(n - 1, ans, x, y);

}

}

int main()

{

int n, ans = 1, x = 1, y = 1;;

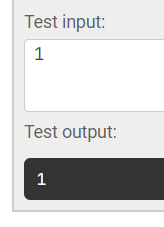
cin >> n;

cout << fib(n, ans, x, y);

return 0;

}

**Результат:**



**Задание 13.**

В небоскребе n этажей. Известно, что если уронить стеклянный шарик с этажа номер p, и шарик разобьется, то если уронить шарик с этажа номер p+1, то он тоже разобьется. Также известно, что при броске с последнего этажа шарик всегда разбивается.Вы хотите определить минимальный номер этажа, при падении с которого шарик разбивается. Для проведения экспериментов у вас есть два шарика. Вы можете разбить их все, но в результате вы должны абсолютно точно определить этот номер.

Определите, какого числа бросков достаточно, чтобы заведомо решить эту задачу.

**Код:**

#include <iostream>

int min\_broken\_floor(int n) {

int current\_floor = 1;

int drop\_step = 1;

while (current\_floor + drop\_step < n) {

current\_floor += drop\_step;

drop\_step++;

}

return drop\_step;

}

int main() {

int n;

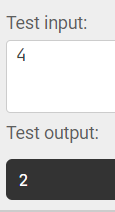
std::cin >> n;

std::cout << min\_broken\_floor(n) << std::endl;

return 0;

}

**Результат:**



**Задание 14.**

Дано число N. Определите, сколькими способами можно расставить на доске N×N N ферзей, не бьющих друг друга.

**Код:**

#include <iostream>

#include <vector>

using namespace std;

int n, cnt = 0;

vector<vector<int>> board(n, vector<int> (n));

void set\_queen(int i, int j) {

for (int x = 0; x < n; x++) {

board[x][j] += 1;

board[i][x] += 1;

if (i + j - x < n && i + j - x >= 0) {

board[i + j - x][x] += 1;

} if (i - j + x < n && i - j + x >= 0) {

board[i + j - x][x] += 1;

}

}

board[i][j] = -1;

}

void remove\_queen(int i, int j) {

for (int x = 0; x < n; x++) {

board[x][j] -= 1;

board[i][x] -= 1;

if (i + j - x < n && i + j - x >= 0) {

board[i + j - x][x] -= 1;

} if (i - j + x < n && i - j + x >= 0) {

board[i + j - x][x] -= 1;

}

}

board[i][j] = 0;

}

void solve(int i) {

for (int j = 0; j < n; j++) {

if (board[i][j] == 0) {

set\_queen(i, j);

}

if (i == n - 1) {

cnt++;

} else if (i < n - 1) {

solve(i + 1);

remove\_queen(i, j);

}

}

}

int main() {

cin >> n;

board.resize(n, vector<int>(n));

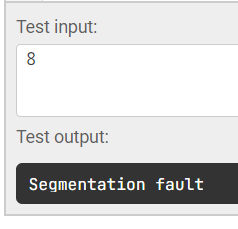
solve(0);

cout << cnt;

return 0;

}

**Результат:**



**1.8. Строки и символы.**

**Задание 1.**

По введенному символу определите, является ли он цифрой

**Код:**

#include <iostream>

#include <string>

using namespace std;

int main() {

string a;

string b;

getline(cin, a);

for (auto i : a) {

b += a[i];

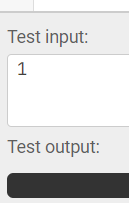
}

cout << b;

return 0;

}

**Результат:**



**Задание 2.**

Переведите символ в верхний регистр.

**Код:**

#include <iostream>

#include <cctype>

int main() {

char c;

std::cin >> c;

if (std::islower(c)) {

c = std::toupper(c);

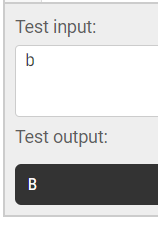
}

std::cout << c << std::endl;

return 0;

}

**Результат:**



**Задание 3.**

Вводится один символ, измените его регистр. То есть, если была введена строчная буква - сделайте ее заглавной и наоборот. Символы, не являющиеся латинской буквой, нужно выводить без изменений.

**Код:**

#include <iostream>

#include <cctype>

int main() {

char ch;

std::cin >> ch;

if (std::islower(ch)) {

ch = std::toupper(ch);

} else if (std::isupper(ch)) {

ch = std::tolower(ch);

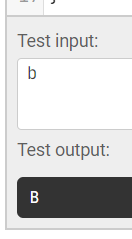
}

std::cout << ch << std::endl;

return 0;

}

**Результат:**



**Задание 4.**

Дана строка, содержащая пробелы. Найдите, сколько в ней слов (слово – это последовательность непробельных символов, слова разделены одним пробелом, первый и последний символ строки – не пробел).

**Код:**

#include <iostream>

#include <string>

using namespace std;

int main()

{

string s;

getline(cin, s);

char p = ' ';

char d = s[0];

int h = 1;

for (auto d : s)

if (d == p)

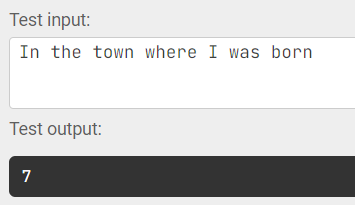
h++;

cout << h;

return 0;

}

**Результат:**



**Задание 5.**

По данной строке определите, является ли она палиндромом (то есть, читается одинаково как слева-направо, так и справа-налево).

**Код:**

#include <iostream>

using namespace std;

int main() {

string s;

int cnt = 1;

getline(cin, s);

for (int i=0; i<s.size(); i++){

if (s[i] != s[s.size() - i - 1]) {

cout << "no";

return 0;

}

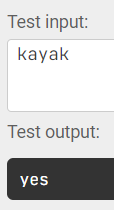
}

cout << "yes";

return 0;

}

**Результат:**



**Задание 6.**

Найдите в данной строке самое длинное слово и выведите его.

**Код:**

#include <iostream>

using namespace std;

int main(){

int g = 1000;

while(g != 0)

cout<<"Hi I'm learning C++!!!!!!!!";

g--;

}

**Задание 7.**

В сети интернет каждому компьютеру присваивается четырехбайтовый код, который принято записывать в виде четырех чисел, каждое из которых может принимать значения от 0 до 255, разделенных точками. Вот примеры правильных IP-адресов:

#include <iostream>

using namespace std;

int main() {

char n, j = 0;

while(cin >> n){

if(n <=255){

j++;

}

else if(n == '.'){

continue;

}

}

if (j >= 4){

cout << "YES";

}

else {

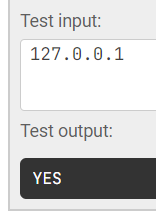
cout << "NO";

}

return 0;

}

**Результат:**



**1.8. Словари и множества.**

**Задание 1.**

Дан список целых чисел, который может содержать до 100000 чисел. Определите, сколько в нем встречается различных чисел.

**Код:**

#include <iostream>

#include <set>

int main() {

int N;

std::cin >> N;

std::set<int> numbers;

for (int i = 0; i < N; ++i) {

int num;

std::cin >> num;

numbers.insert(num);

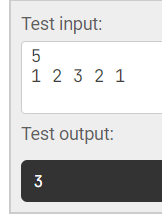
}

std::cout << numbers.size() << std::endl;

return 0;

}

**Результат:**



**Задание 2.**

Во входной строке записана последовательность чисел через пробел. Для каждого числа выведите слово YES (в отдельной строке), если это число ранее встречалось в последовательности или NO, если не встречалось.

**Код:**

#include <iostream>

#include <set>

using namespace std;

int main() {

set <int> s;

int n,k;

cin >> n;

for (int i = 0; i < n; i++) {

cin >> k;

int cnt = 0;

for (auto now = s.lower\_bound(k); now != s.upper\_bound(k); now++) {

cnt++;

}

cnt != 0 ? cout << "YES" : cout << "NO";

cout << endl;

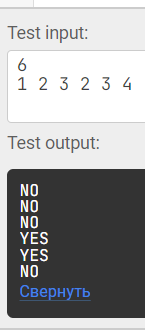
s.insert(k);

}

return 0;

}

**Результат:**



**Задание 3.**

Даны два списка чисел, которые могут содержать до 100000 чисел каждый. Посчитайте, сколько чисел содержится одновременно как в первом списке, так и во втором.

**Код:**

#include<iostream>

#include <set>

using namespace std;

int main()

{int a;

cin>>a;

set<int> ss;

int k[100000];

for(int i=0;i<a;i++)

{

cin>>k[i];

}

int b,z=0;

cin>>b;

set<int> s;

int l[100000];

for(int i=0;i<a;i++)

{

cin>>l[i];

}

for(int i=0;i<b;i++)

{

for(int j=0;j<a;j++)

{

if(k[i]==l[j])

{

z++;

break;

}

}

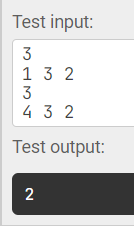
}

cout<<z;

return 0;

}

**Результат:**



**Задание 4.**

Даны два списка чисел, которые могут содержать до 100000 чисел каждый. Выведите все числа, которые входят как в первый, так и во второй список в порядке возрастания.

**Код:**

#include <iostream>

#include <set>

using namespace std;

int main()

{

set <int> s1, s2, s3;

int n, m;

cin >> n;

for(int i = 0; i < n; i++)

{

int a;

cin >> a;

s1.insert(a);

}

int k = 0;

cin >> m;

for(int i = 0; i < m; i++)

{

int a;

cin >> a;

s2.insert(a);

if(s1.find(a) != s1.end())

s3.insert(a);

}

set <int>::iterator j;

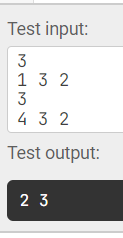
for(j = s3.begin(); j != s3.end(); j++)

cout << \*j << ' ';

return 0;

}

**Результат:**



**Задание 5.**

Вам дан словарь, состоящий из пар слов. Каждое слово является синонимом к парному ему слову. Все слова в словаре различны. Для одного данного слова определите его синоним.

**Код:**

#include <iostream>

#include <map>

#include <string>

using namespace std;

int main()

{

map<string, string> s1, s2;

map<string, string>::iterator j;

int n;

cin >> n;

for(int i = 0; i < n; i++)

{

string temp1, temp2;

cin >> temp1 >> temp2;

s1.insert(pair<string, string>(temp1, temp2));

s2.insert(pair<string, string>(temp2, temp1));

}

string key;

cin >> key;

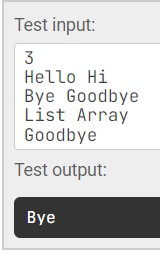
j = s1.find(key) != s1.end() ? s1.find(key) : s2.find(key);

cout << j->second;

return 0;

}

**Результат:**



**Задание 6.**

Однажды, разбирая старые книги на чердаке, школьник Вася нашёл англо-латинский словарь. Английский он к тому времени знал в совершенстве, и его мечтой было изучить латынь. Поэтому попавшийся словарь был как раз кстати.

К сожалению, для полноценного изучения языка недостаточно только одного словаря: кроме англо-латинского необходим латинско-английский. За неимением лучшего он решил сделать второй словарь из первого.

Как известно, словарь состоит из переводимых слов, к каждому из которых приводится несколько слов-переводов. Для каждого латинского слова, встречающегося где-либо в словаре, Вася предлагает найти все его переводы (то есть все английские слова, для которых наше латинское встречалось в его списке переводов), и считать их и только их переводами этого латинского слова.

Помогите Васе выполнить работу по созданию латинско-английского словаря из англо-латинского.

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main() {

char ch;

string a,b;

map <string, vector <string>>mp;

set <string>st;

long long n;

cin>>n;

for(int i=1;i<=n;i++){

cin>>a>>ch>>b;

while(b[b.size()-1]==','){

b.erase(b.size()-1,1);

st.insert(b);

mp[b].push\_back(a);

cin>>b;

}

st.insert(b);

mp[b].push\_back(a);

}

cout<<st.size()<<endl;

for(auto g:st){

cout<<g<<" - ";

int k=0;

for(auto h:mp[g]){

k++;

cout<<h;

if(k!=mp[g].size())

cout<<", ";

}

cout<<endl;

}

}

**Результат:**



**1.8. Стандартные алгоритмы STL**

**Задание 1.**

Отсортируйте массив.

**Код:**

#include <iostream>

int main() {

int size;

std::cin >> size;

int \*arr = (int\*)malloc(size \* sizeof(int));

for (int i = 0; i < size; i++) std::cin >> arr[i];

for (int i=1;i < size;i++) for(int j=i;j>0 && arr[j-1]>arr[j];j--) std::swap(arr[j-1],arr[j]);

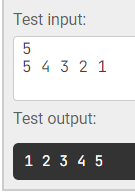
for (int i = 0; i < size; i++) std::cout << arr[i] << " ";

free(arr);

return 0;

}

**Результат:**



**Задание 2.**

В обувном магазине продается обувь разного размера. Известно, что одну пару обуви можно надеть на другую, если она хотя бы на три размера больше. В магазин пришел покупатель. Требуется определить, какое наибольшее количество пар обуви сможет предложить ему продавец так, чтобы он смог надеть их все одновременно.

**Код:**

#include <iostream>

#include <vector>

#include <algorithm>

using namespace std;

int main() {

int s,n,k = 0,tmp,h = 0;

cin >> s >> n;

tmp = s;

vector<int>v(n);

for(int i = 0; i < n; i++){

cin >> v[i];

}

sort(v.begin(), v.end());

for(auto c : v){

if(c == v[0] && c > tmp){

k++;

s = c;

}

if(c == tmp || c >= s+3){

if(c >= s+3){

k++;

s = c;

}

if(c == tmp){

if(h < 1){

k++;

h++;

}

k = k;

}

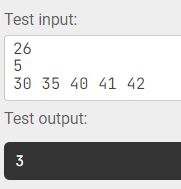
}

}

cout << k;

}

**Результат:**



**Задание 3.**

Во время проведения олимпиады каждый из участников получил свой идентификационный номер – натуральное число. Необходимо отсортировать список участников олимпиады по количеству набранных ими баллов.

**Код:**

#include <bits/stdc++.h>

typedef std::pair<int, int> pair;

struct cmp {

bool operator()(const pair &a, const pair &b) const {

if (a.second != b.second)

return a.second > b.second;

return a.first < b.first;

}

};

int main() {

std::multiset<pair, cmp> mset;

int N, first, second;

std::cin >> N;

while (N) {

std::cin >> first >> second;

mset.emplace(first, second);

N--;

}

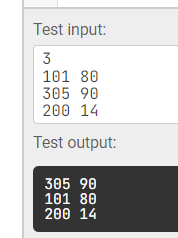
for (const pair &p: mset)

std::cout << p.first << " " << p.second << std::endl;

return 0;

}

**Результат:**



**Задание 4.**

Выведите все исходные точки в порядке возрастания их расстояний от начала координат.

**Код:**

#include <iostream>

#include <vector>

#include <utility>

#include <algorithm>

using namespace std;

int main()

{

int n; cin >> n;

if (n<=0) return 0; \\ на случай, если кол-во точек 0 или меньше

vector <pair <int, int>> a(n);

for (int i=0; i<n; i++)

{

int f, s;

cin >> f; a[i].first = f;

cin >> s; a[i].second = s;

}

sort(a.begin(), a.end());

for (int i = 0; i < n; i++)

{

cout << a[i].first << " " << a[i].second << endl;

}

return 0;

}

**Результат:**



**Задание 5.**

Выведите фамилии и имена учащихся в порядке убывания их среднего балла.

**Код:**

#include <string>

#include <vector>

#include <iostream>

#include <algorithm>

using namespace std;

struct Student {

string name, surname;

int math, phys, comp\_science;

};

bool cmp (Student a, Student b) {

return a.math + a.phys + a.comp\_science > b.math + b.phys + b.comp\_science;

}

int main (void) {

int n;

cin >> n;

if (n < 1) return 1;

vector <Student> data(n);

for (int i = 0; i < n; ++i) {

cin >> data[i].name >> data[i].surname;

cin >> data[i].math >> data[i].phys >> data[i].comp\_science;

}

stable\_sort(data.begin(), data.end(), cmp);

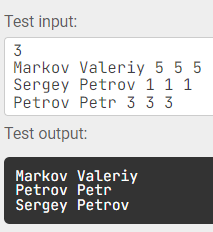
for (auto s : data)

cout << s.name << " " << s.surname << endl;

return 0;

}

**Результат:**



**Раздел 1.5 Структура кода на С++**

**Задача 1.**

В этом простом упражнении вам нужно окружить объявление функции foo стражами включения. В качестве имени стража используйте FOO\_HPP и не используйте pragma once.

**Код:**

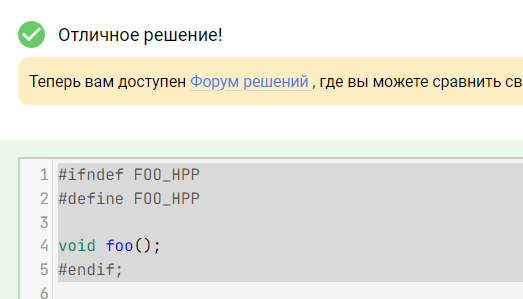
#ifndef FOO\_HPP

#define FOO\_HPP

void foo();

#endif;

**Результат:**



**Раздел 1.6 Как компилируется программа на C++**

**Задача 1.**

Пришло время скомпилировать вашу первую программу. По традиции первой программой является "Hello, world!". В данном степе мы предлагаем вам взять программу из шаблона, сохранить её в текстовом файле с расширением .cpp, скомпилировать её самостоятельно и запустить. Это позволит вам проверить, что ваш компилятор установлен и работает.

**Код:**

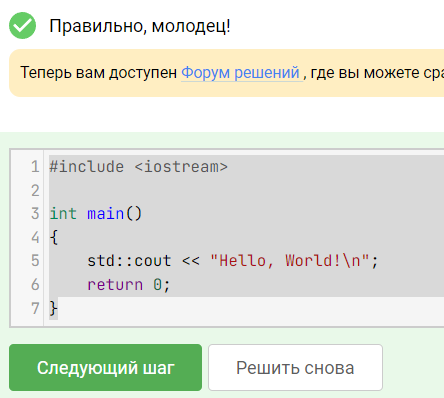
#include <iostream>

int main()

{ std::cout << "Hello, World!\n";

return 0;}

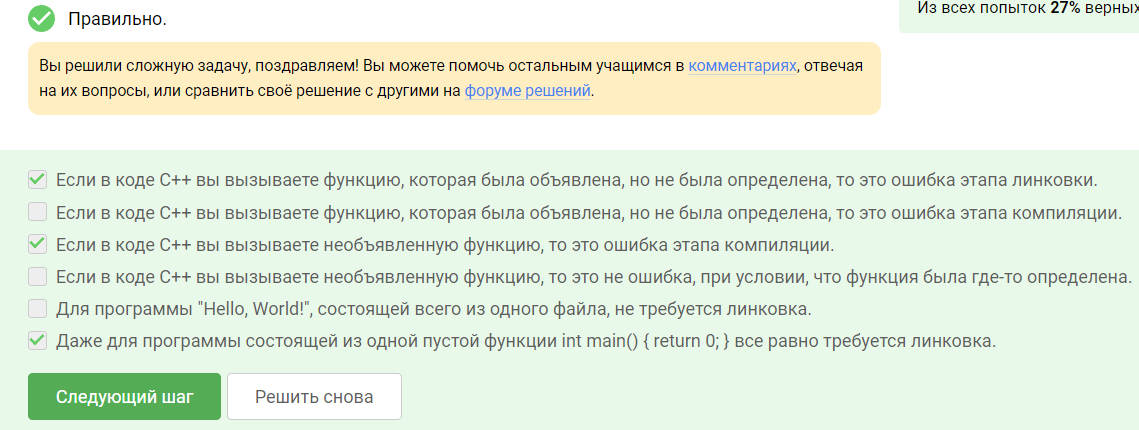
**Результат:**



**Задача 2.**

Выберите все верные утверждения из списка.

**Результат:**



**Раздел 1.8 Введение в синтаксис C++, часть 2**

**Задача 1.**

Напишите функцию power, реализующую возведение целого числа в неотрицательную целую степень. Функция power должна принимать на вход два целых числа и возвращать целое число (смотрите шаблон кода). При выполнении задания учтите, что функция обязательно должна называться power, функция ничего не должна читать со входа или выводить.

**Код:**

int power(int x, unsigned int p) {

int answer = 1;

while (p) {

if (p & 1) {

answer \*= x;

}

p >>= 1;

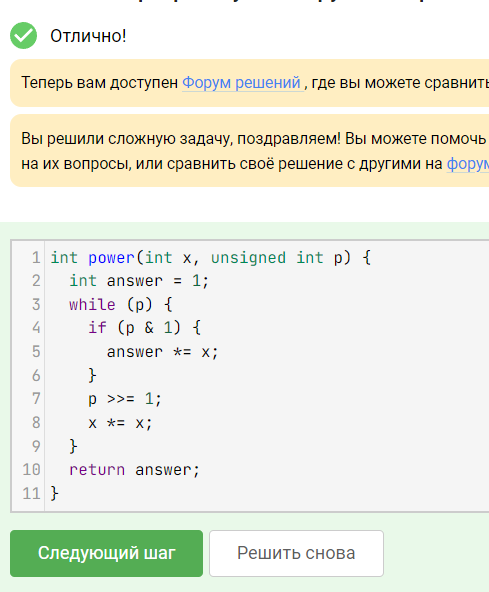
x \*= x;

}

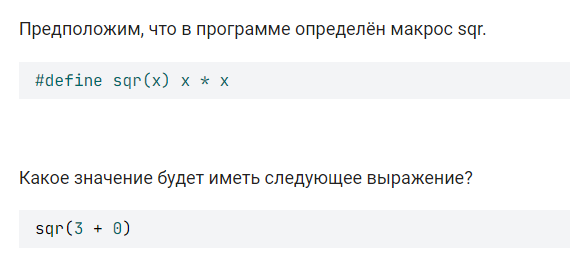
return answer;

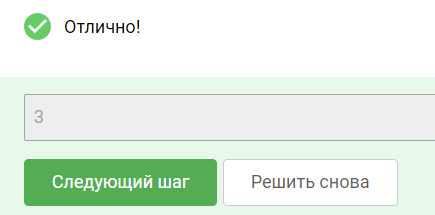
}

**Результат:**



**Задача 2.**

**Результат:**



**Задача 3.**

Реализуйте макрос MAX от трёх параметров, который присваивает целочисленной (int) переменной, переданной в качестве третьего аргумента, наибольшее из значений, переданных в первых двух аргументах.

**Код:**

#define MAX(a, b, c) do { \

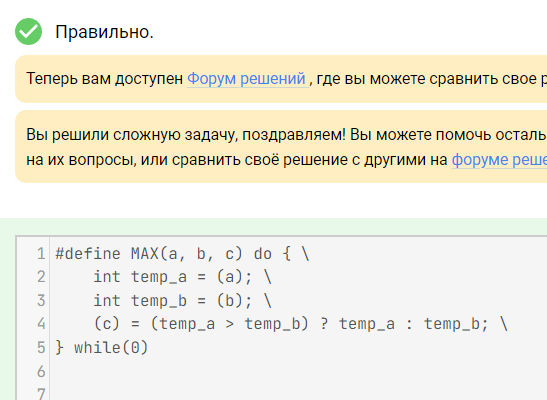
int temp\_a = (a); \

int temp\_b = (b); \

(c) = (temp\_a > temp\_b) ? temp\_a : temp\_b; \

} while(0)

**Результат:**



**Задача 4.**

**Код:**

#include <bits/stdc++.h>

using namespace std;

int main(){

int n;

cin >> n;

while (n--)

{

int a,b;

cin >> a >> b;

cout << a + b;

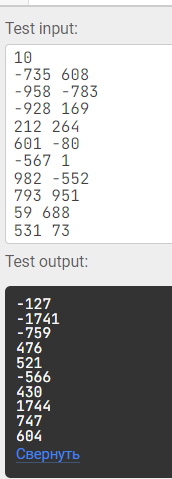
cout << endl;

a=0;b=0;

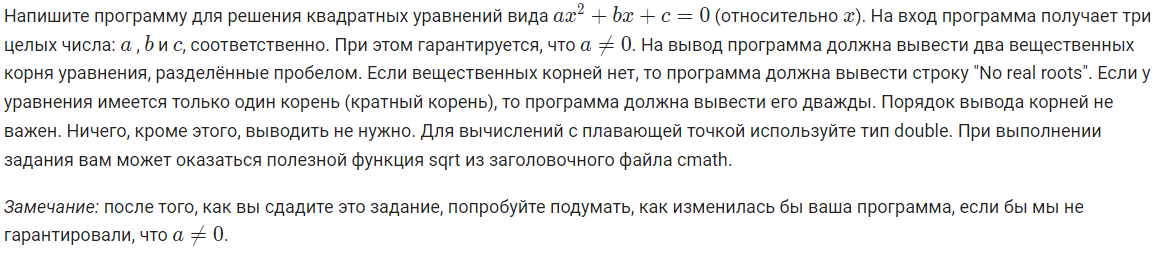
}

}

**Результат:**



**Задача 5.**

**Код:**

#include <iostream>

#include <cmath>

using namespace std;

int main()

{

int a, b, c;

cin >> a >> b >> c;

double x1, x2, d;

d = (b \* b) - (4 \* a \* c);

if (d >= 0) {

x1 = (-b + sqrt(d)) / (2 \* a);

x2 = (-b - sqrt(d)) / (2 \* a);

cout << x1 << " " << x2;

} else {

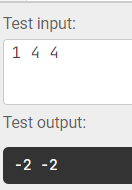
cout << "No real roots";

}

return 0;

}

**Результат:**



**Задача 6.**

Напишите программу, которая посимвольно читает из std::cin, пока не достигнет конца потока ввода, и заменяет несколько подряд идущих пробелов одним и выводит полученный результат в std::cout. Никаких других символов, кроме пробелов удалять не нужно. При выполнении задания вам не разрешается пользоваться дополнительной памятью, а именно: массивами, стандартными контейнерами и строками, даже если вы уже с ними знакомы. Вы можете определять любые вспомогательные функции, если они вам нужны.

**Код:**

#include <iostream>

int main() {

char c;

bool is\_ok = true;

while(std::cin.get(c))

if(c != ' '){

std::cout << c;

is\_ok = true;

}else if(is\_ok) {

std::cout << c;

is\_ok = false;

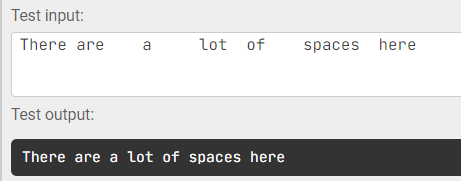
}

//system("pause");

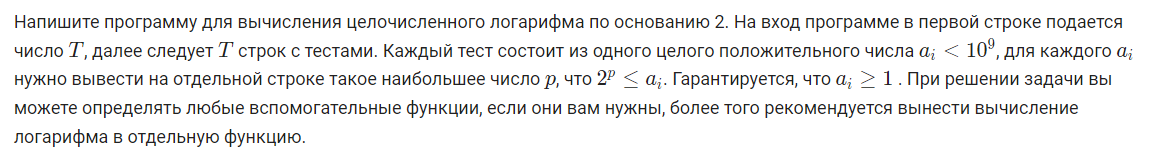
return 0;

}

**Результат:**



**Задача 7.**

**Код:**

#include <iostream>

using namespace std;

int log(int n) {

int p = 0, i = 1;

while (i <= n) {

i \*= 2;

p++;

}

return p - 1;

}

int main() {

int T;

cin >> T;

for (int i = 0; i < T; i++) {

int t;

cin >> t;

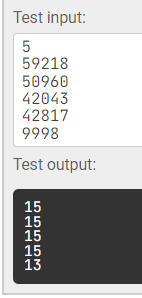
cout << log(t) << endl;

}

return 0;

}

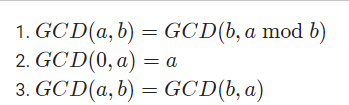
**Результат:**



**Раздел 2.2 Стек вызовов**

**Задача 1.**

Напишите рекурсивную функцию вычисления наибольшего общего делителя двух положительных целых чисел (Greatest Common Divisor, GCD). Для этого воспользуйтесь следующими свойствами:



**Код:**

unsigned gcd(unsigned a, unsigned b)

{

if (b == 0)

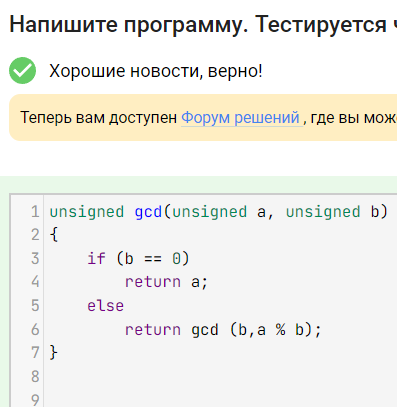
return a;

else

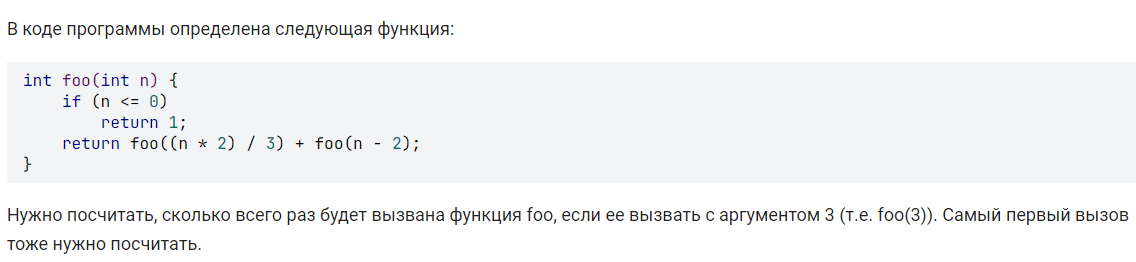
return gcd (b,a % b);

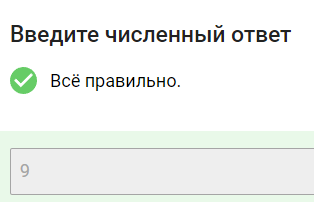
}

**Результат:**



**Задача 2.**

**Результат:**



**Задача 3.**

Вам требуется написать программу, которая "переворачивает" последовательность положительных целых чисел. На вход подается последовательность разделенных пробелами положительных целых чисел. Последовательность заканчивается нулем. Требуется вывести эту последовательность в обратном порядке.

На выводе числа нужно так же разделить пробелами. Завершающий ноль — это просто индикатор конца последовательности, он не является ее частью, т.е. выводить его не нужно.

**Код:**

#include <iostream>

using namespace std;

void rec()

{

int a=0;

cin >> a;

if ( a ==0 ) return;

rec();

cout << a << " ";

}

int main()

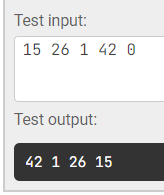
{

rec();

return 0;

}

**Результат:**



**Раздел 2.3 Указатели и массивы**

**Задача 1.**

В этой задаче вам нужно реализовать функцию, которая сдвигает содержимое массива влево на заданное число позиций (циклический сдвиг).

На вход функция принимает массив, его размер и величину сдвига. Например, если на вход функции подан массив: int a[] = { 1, 2, 3, 4, 5 }; и требуется циклически сдвинуть его влево на 2 позиции, то на выходе мы получим числа в таком порядке: 3, 4, 5, 1, 2.

Обратите внимание, что величина сдвига может быть нулевой, а может быть и больше размера массива, все эти случаи нужно учесть.

**Код:**

void rotate(int a[], unsigned size, int shift)

{

int b[size];

for (int i = 0; i < size; i++) {

b[i] = a[i];

}

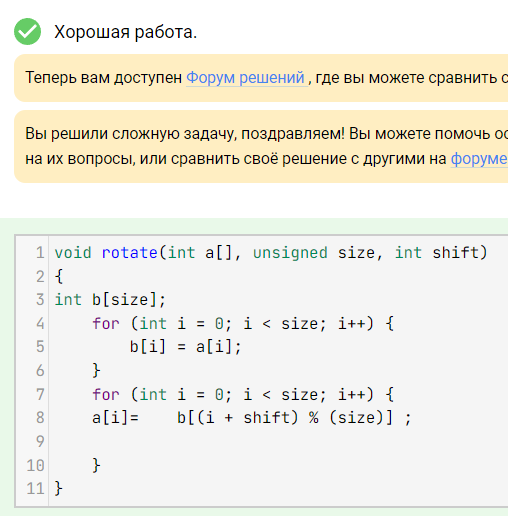
for (int i = 0; i < size; i++) {

a[i]= b[(i + shift) % (size)] ;

}

}

**Результат:**



**Раздел 2.4 Использование указателей**

**Задача 1.**

Очень часто для работы со строками нам нужно сначала вычислить длину строки. Для C-style строк длина нигде явно не хранится, но её можно вычислить. Напишите функцию, которая вычисляет длину C-style строки. Учтите, что завершающий нулевой символ считать не нужно.

**Код:**

unsigned strlen(const char \*str)

{

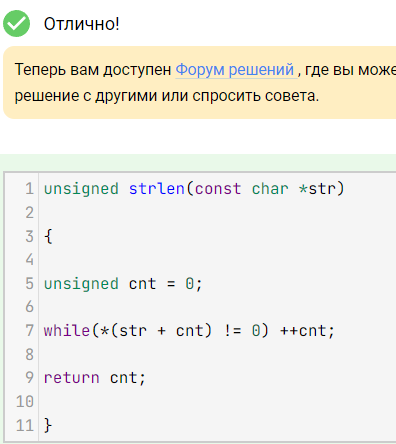
unsigned cnt = 0;

while(\*(str + cnt) != 0) ++cnt;

return cnt;

}

**Результат:**



**Задача 2.**

Вам требуется реализовать функцию конкатенации (склейки) двух C-style строк. Функция конкатенации принимает на вход две C-style строки и дописывает вторую в конец первой так, чтобы первая строка представляла из себя одну C-style строку равную конкатенации двух исходных.

Не забудьте, что в результирующей строке должен быть только один нулевой символ — тот, что является маркером конца строки.

Гарантируется, что в первой строке достаточно памяти (т.е. она располагается в массиве достаточной длины), чтобы разместить конкатенацию обеих строк, но не больше.

**Код:**

void strcat(char \*to, const char \*from)

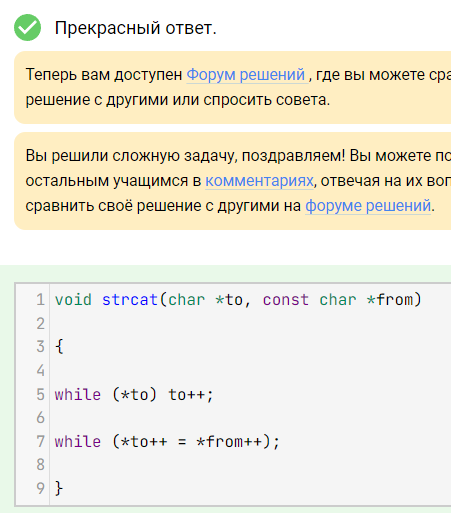
{

while (\*to) to++;

while (\*to++ = \*from++);

}

**Результат:**



**Задача 3.**

Напишите функцию поиска первого вхождения шаблона в текст. В качестве первого параметра функция принимает текст (C-style строка), в которой нужно искать шаблон. В качестве второго параметра строку-шаблон (C-style строка), которую нужно найти. Функция возвращает позицию первого вхождения строки-шаблона, если он присутствует в строке (помните, что в C++ принято считать с 0), и -1, если шаблона в тексте нет.

Учтите, что пустой шаблон (строка длины 0) можно найти в любом месте текста.

**Код:**

int strstr(const char \*text, const char \*pattern)

{

int i = 0;

int j = 0;

int result=0;

if (text == pattern || \*pattern == '\0') return 0;

while (pattern[i] != '\0') {

if(text[j]=='\0'){

result = -1;

break;

}

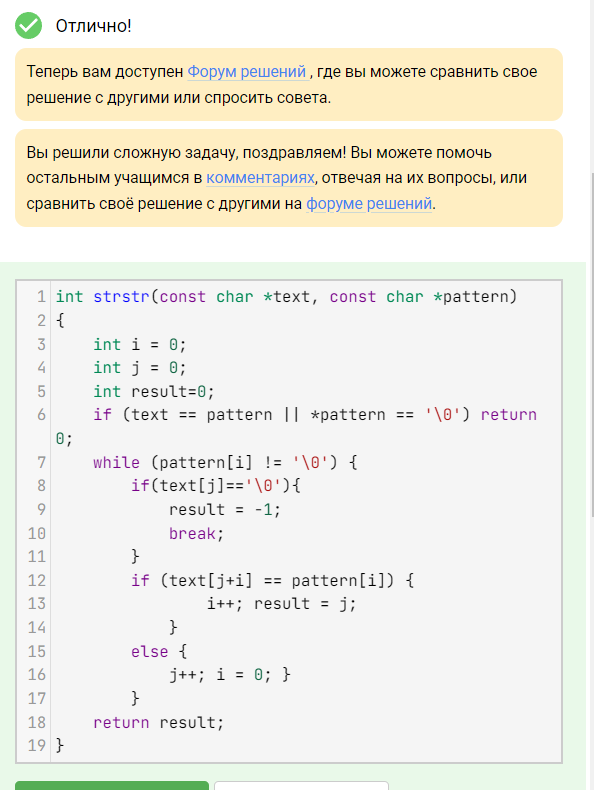
if (text[j+i] == pattern[i]) {

i++; result = j;

}

else {

j++; i = 0; }



**Раздел 2.6 Динамическая память**

**Задача 1.**

В этой задаче вам нужно реализовать функцию resize. Функция resize принимает на вход три аргумента: указатель на область памяти, выделенную с помощью оператора new[], размер области (size) и новый размер (new\_size). Функция должна выделить память размера new\_size, скопировать в нее данные из переданной области памяти, освободить старую область памяти и вернуть выделенную область памяти нового размера со скопированными данными.

Обратите внимание, что память, возвращенная из функции, будет освобождена с помощью оператора delete[]. Несмотря на то, что входной тип данных — это const char \*, не стоит рассчитывать, что там хранится C-style строка. При выполнении задания также учтите, что new\_size может быть меньше size.

**Код:**

char \*resize(const char \*str, unsigned size, unsigned new\_size)

{

char \* new\_str = new char[new\_size];

for(int i = 0; i < size && i < new\_size; ++i)

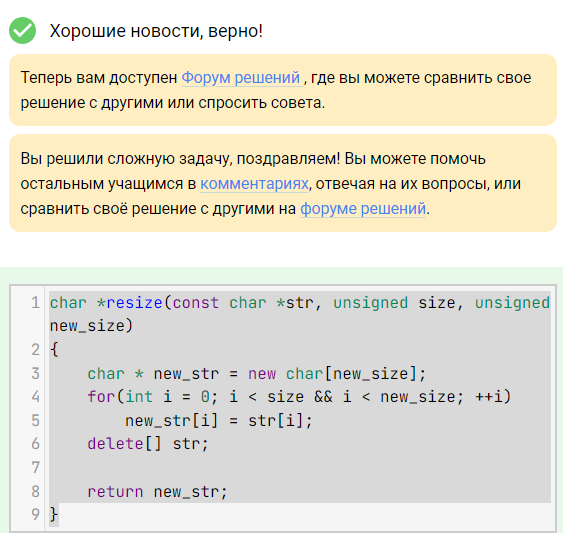
new\_str[i] = str[i];

delete[] str;

return new\_str;

}

**Результат:**



**Задача 2.**

Реализуйте функцию getline, которая считывает поток ввода посимвольно, пока не достигнет конца потока или не встретит символ переноса строки ('\n'), и возвращает C-style строку с прочитанными символами.

Обратите внимание, что так как размер ввода заранее неизвестен, то вам нужно будет перевыделять память в процессе чтения, если в потоке ввода оказалось больше символов, чем вы ожидали.

Память, возвращенная из функции будет освобождена оператором delete[]. Символ переноса строки ('\n') добавлять в строку не нужно, но не забудьте, что в конце C-style строки должен быть завершающий нулевой символ.

**Код:**

#include <cstring>

#include <iostream>

#include <string>

#ifdef string

#undef string

char \*

getline()

{

std::string result;

std::getline(std::cin, result);

char \* arr = new char[result.size()];

strcpy(arr, result.c\_str());

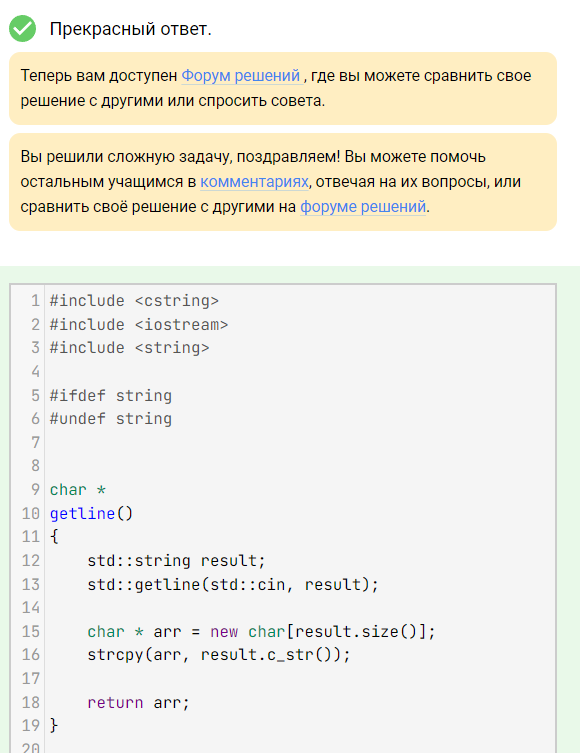
return arr;

}

#endif

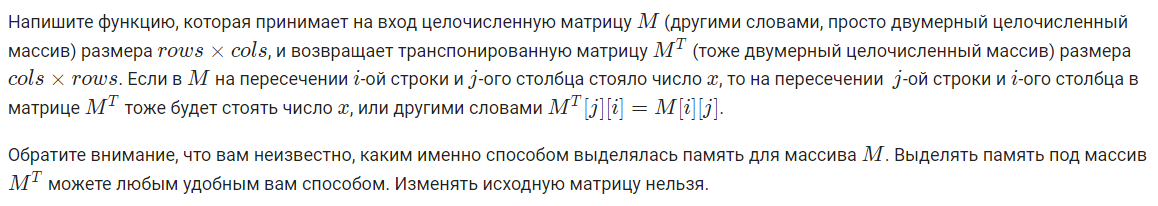
#define string do\_not\_use\_stl

**Результат:**



**Раздел 2.7 Многомерные массивы**

**Задача 1.**

**Код:**

int\*\* transpose(const int\* const\* m, unsigned rows, unsigned cols)

{

int\*\* new\_m = new int\* [cols];

for (int i = 0; i != cols; ++i)

new\_m[i] = new int[rows];

for (int i = 0; i < cols; i++) {

for (int j = 0; j < rows; j++) {

new\_m[i][j] = m[j][i];

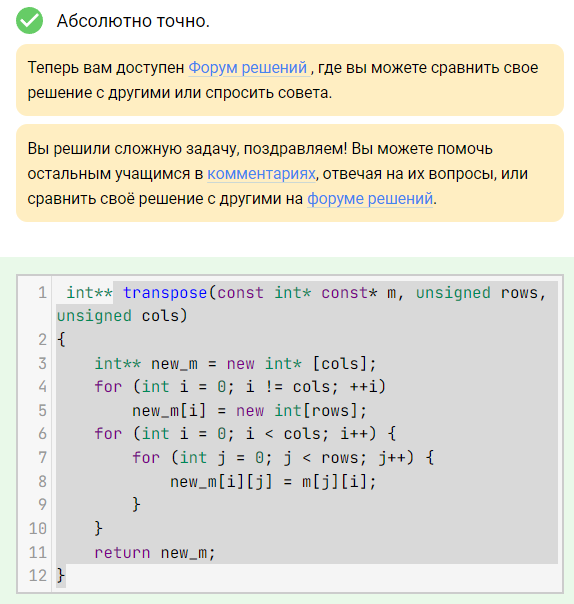
}

}

return new\_m;

}

**Результат:**



**Задача 2.**

Реализуйте функцию swap\_min, которая принимает на вход двумерный массив целых чисел, ищет в этом массиве строку, содержащую наименьшее среди всех элементов массива значение, и меняет эту строку местами с первой строкой массива.

Подумайте, как обменять строки массива, не обменивая элементы строк по-отдельности.

**Код:**

#include <iostream>

using namespace std;

void swap\_min(int \*m[], unsigned rows, unsigned cols)

{

int min = m[0][0];

int k = 0;

for(int i = 0; i < rows; i++)

{

for(int j = 0; j< cols; j++)

{

if(m[i][j] < min)

{

min = m[i][j];

k = i;

}

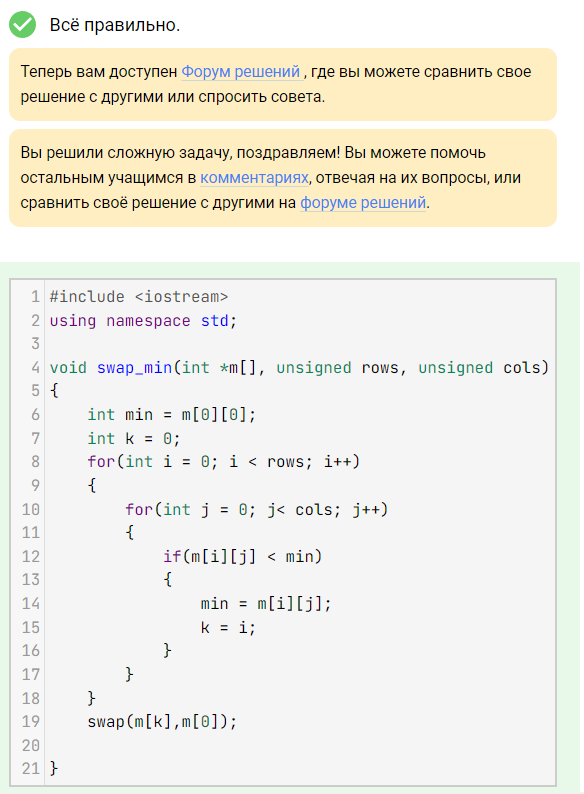
}

}

swap(m[k],m[0]);

}

**Результат:**



**Раздел 2.6 Динамическая память**

**Задача 1.**

В этой задаче вам нужно реализовать функцию resize. Функция resize принимает на вход три аргумента: указатель на область памяти, выделенную с помощью оператора new[], размер области (size) и новый размер (new\_size). Функция должна выделить память размера new\_size, скопировать в нее данные из переданной области памяти, освободить старую область памяти и вернуть выделенную область памяти нового размера со скопированными данными.

Обратите внимание, что память, возвращенная из функции, будет освобождена с помощью оператора delete[]. Несмотря на то, что входной тип данных — это const char \*, не стоит рассчитывать, что там хранится C-style строка. При выполнении задания также учтите, что new\_size может быть меньше size.

**Код:**

char \*resize(const char \*str, unsigned size, unsigned new\_size)

{

char \* new\_str = new char[new\_size];

for(int i = 0; i < size && i < new\_size; ++i)

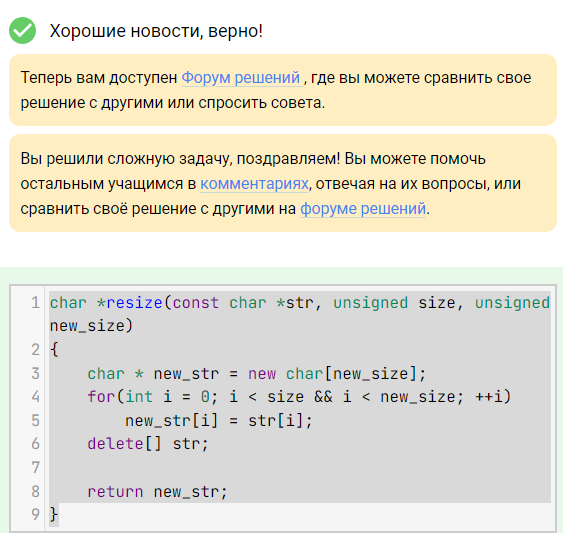
new\_str[i] = str[i];

delete[] str;

return new\_str;

}

**Результат:**



**Задача 2.**

Реализуйте функцию getline, которая считывает поток ввода посимвольно, пока не достигнет конца потока или не встретит символ переноса строки ('\n'), и возвращает C-style строку с прочитанными символами.

Обратите внимание, что так как размер ввода заранее неизвестен, то вам нужно будет перевыделять память в процессе чтения, если в потоке ввода оказалось больше символов, чем вы ожидали.

Память, возвращенная из функции будет освобождена оператором delete[]. Символ переноса строки ('\n') добавлять в строку не нужно, но не забудьте, что в конце C-style строки должен быть завершающий нулевой символ.

**Код:**

#include <cstring>

#include <iostream>

#include <string>

#ifdef string

#undef string

char \*

getline()

{

std::string result;

std::getline(std::cin, result);

char \* arr = new char[result.size()];

strcpy(arr, result.c\_str());

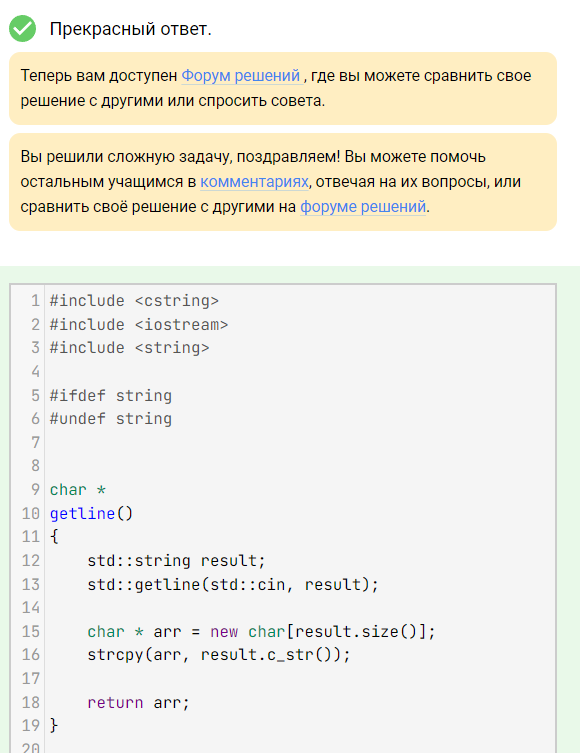
return arr;

}

#endif

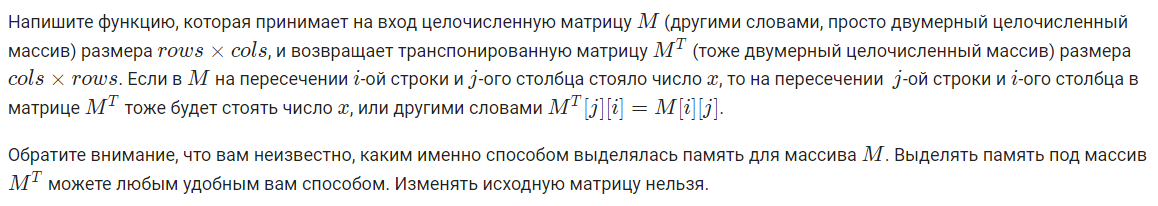
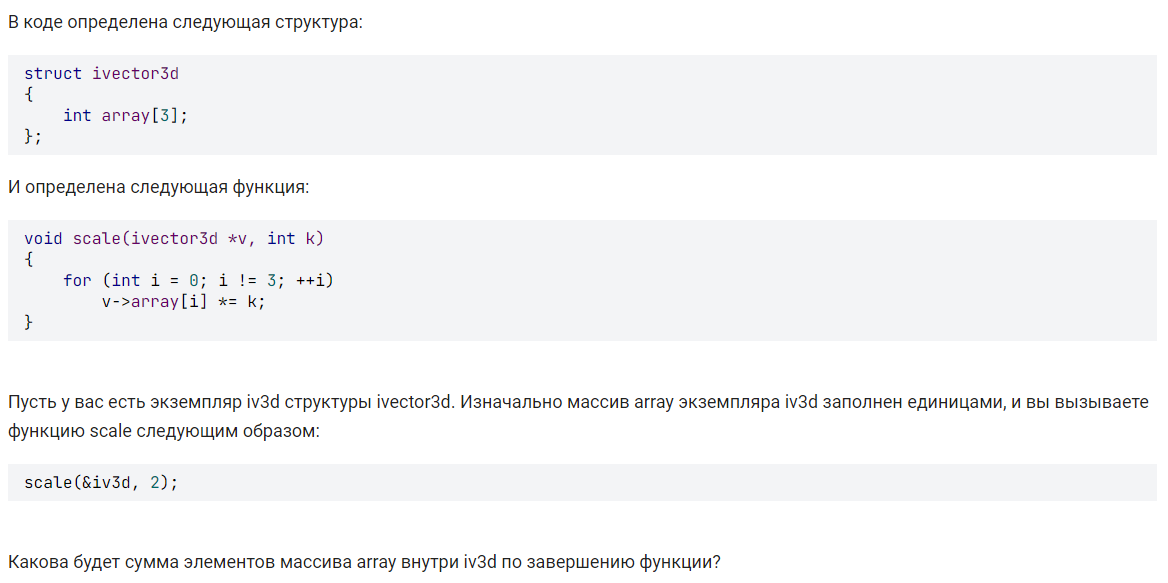
#define string do\_not\_use\_stl

**Результат:**

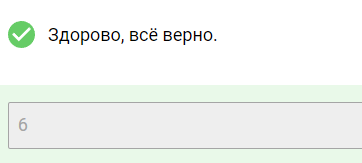


**Раздел 3.1 Структуры**

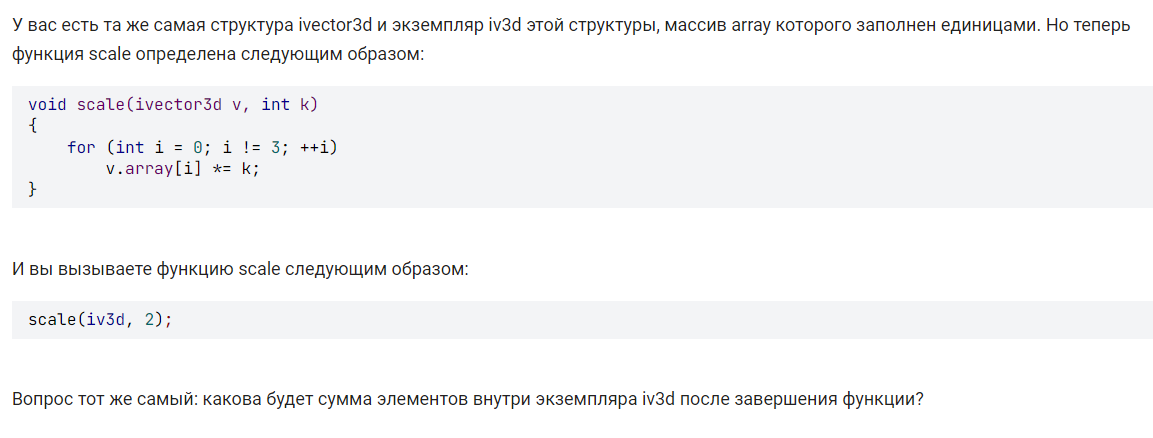
**Задача 1.**



**Результат:**



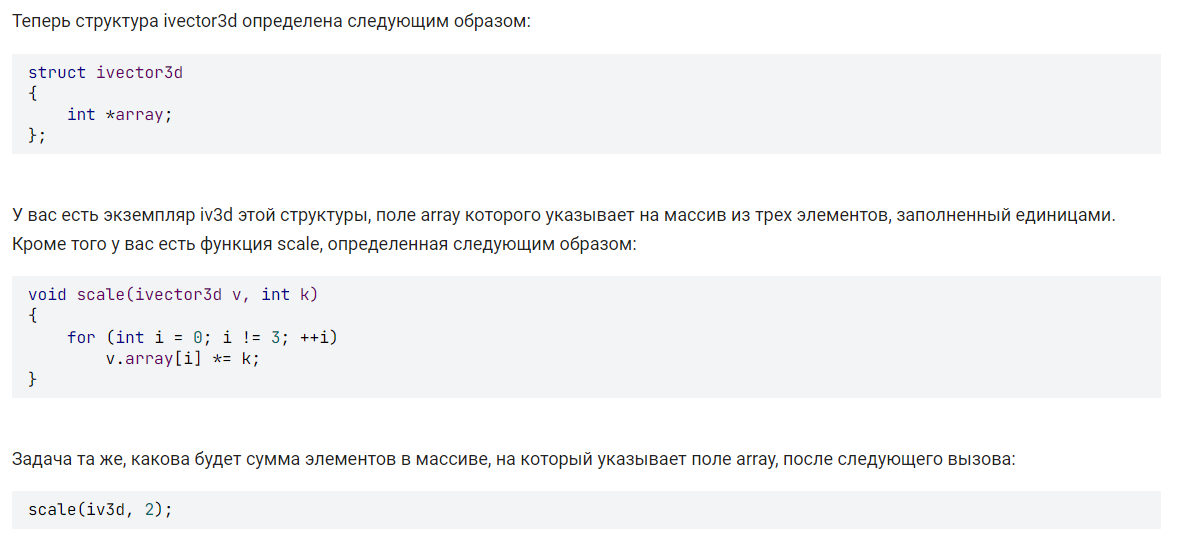
**Задача 2.**



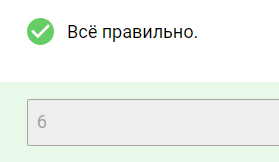
**Результат:**



**Задача 2.**



**Результат:**



**Раздел 3.3 Конструкторы и деструкторы**

**Задача 1.**

В этой и последующих задачах мы создадим свой простой аналог стандартного класса string для удобной работы со строками.

Начнем мы с написания конструктора. В этой задаче вам требуется реализовать конструктор, который принимает на вход C-style строку, вычисляет ее размер (без учета завершающего 0 символа) и сохраняет его в поле size, кроме того, конструктор должен аллоцировать память достаточную для хранения копии переданной строки (вместе с завершающим 0 символом), копирует переданную строку в выделенную память и сохраняет указатель на начало этой области памяти в поле str. Т.е. в результате работы конструктора в поле str должен оказаться указатель на C-строку c копией исходной строки, а в поле size — длина строки без учета завершающего нулевого символа.

**Код:**

#include <cstddef> // size\_t

#include <cstring> // strlen, strcpy

struct String {

String(const char \* str = "");

size\_t size;

char \*str;

};

String::String(const char \* str) {

size = strlen(str);

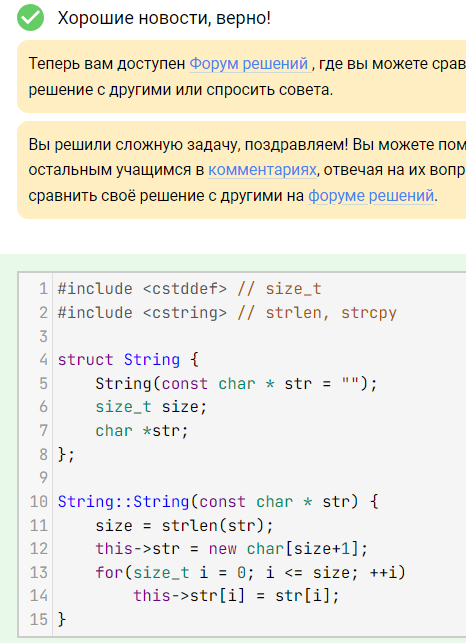
this->str = new char[size+1];

for(size\_t i = 0; i <= size; ++i)

this->str[i] = str[i];

}

**Результат:**



**Задача 2.**

Конструкторов у структуры может быть несколько. Для строки может оказаться полезным заполняющий конструктор (например, чтобы создать строку пробелов). Заполняющий конструктор принимает число и символ, и создает строку с заданным количеством повторений переданного символа. Условия налагаемые на реализацию конструктора те же самые (в поле size размер без учета завершающего 0 символа, в поле str C-style строка, т.е. с завершающим 0 символом). Кроме конструктора в этой задаче вам нужно также реализовать и деструктор, который освободит выделенную память.

**Код:**

#include <cstddef> // size\_t

#include <cstring> // strlen, strcpy

struct String {

/\* Реализуйте этот конструктор \*/

String(size\_t n, char c){

char\* s = new char[n + 1];

for(int i = 0;i <= n; i++){

s[i] = c;

}

s[n]= '\0';

size = n;

this->str = s;

}

/\* и деструктор \*/

~String(){

delete [] str;

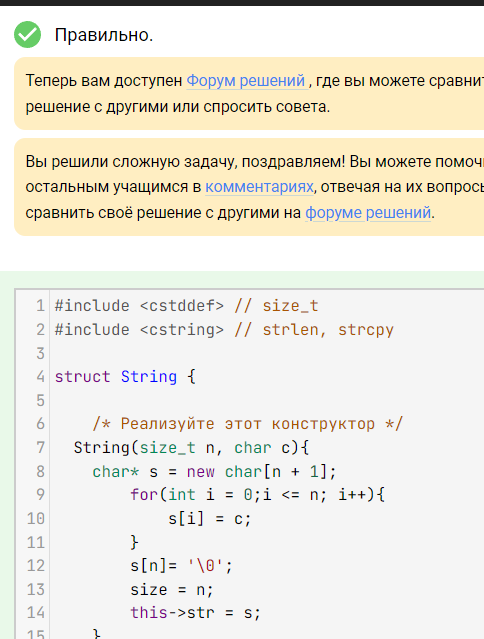
}

size\_t size;

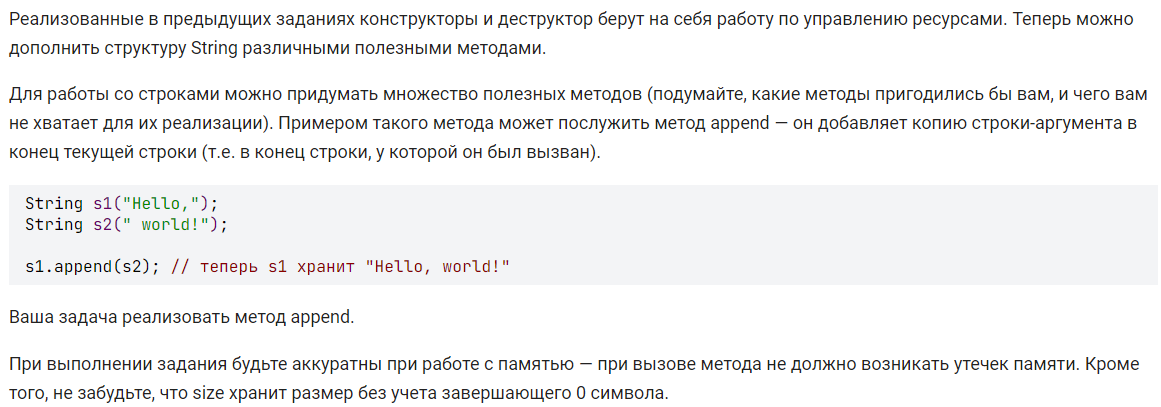
char \*str;

};

**Результат:**



**Задача 3.**

**Код:**

#include <cstddef> // size\_t

#include <cstring> // strlen, strcpy

struct String {

String(const char \*str = "");

String(size\_t n, char c);

~String();

/\* Реализуйте этот метод. \*/

void append(String &other){

size\_t new\_size = size + other.size;

char\* new\_ptr = new char [new\_size + 1];

strcpy(new\_ptr, str);

strcat(new\_ptr + size, other.str);

new\_ptr[new\_size] = '\0';

delete[] str;

str = new\_ptr;

size = strlen(str);

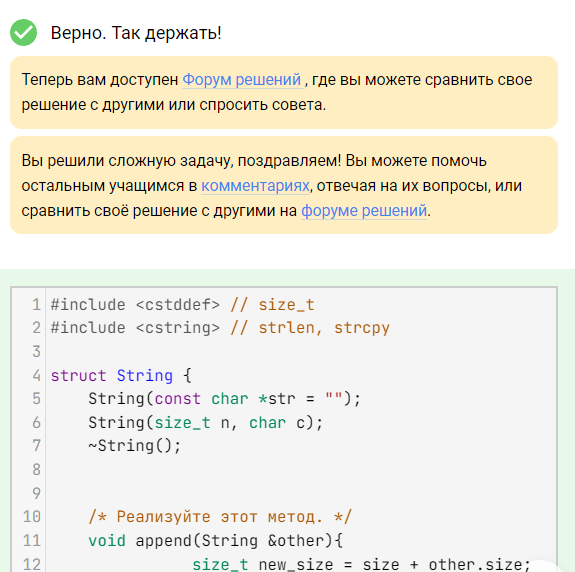
}

size\_t size;

char \*str;

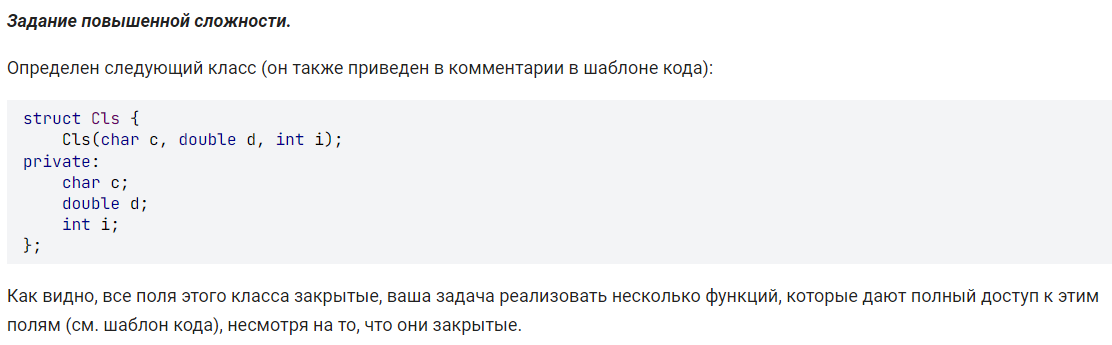
};

**Результат:**



**Раздел 3.5 Модификаторы доступа**

**Задача 1.**

**Код:**

char &get\_c(Cls &cls) {

return \*(char\*)(&cls);

}

double &get\_d(Cls &cls) {

return \*(double\*)((char\*)(&cls) + 8);

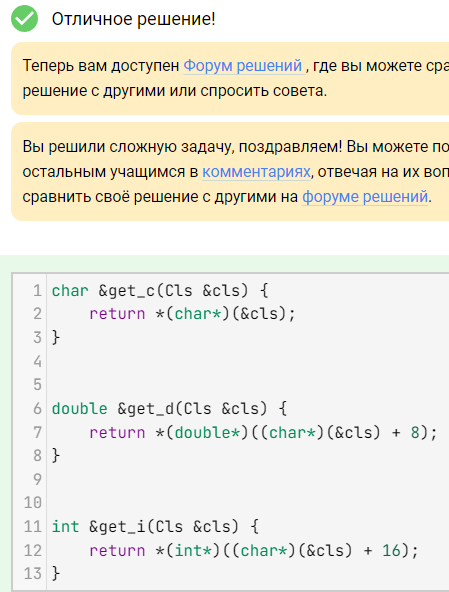
}

int &get\_i(Cls &cls) {

return \*(int\*)((char\*)(&cls) + 16);

}

**Результат:**

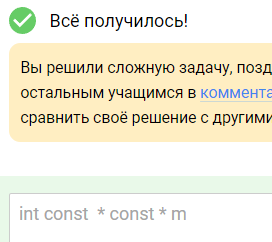


**Раздел 3.6 Константность**

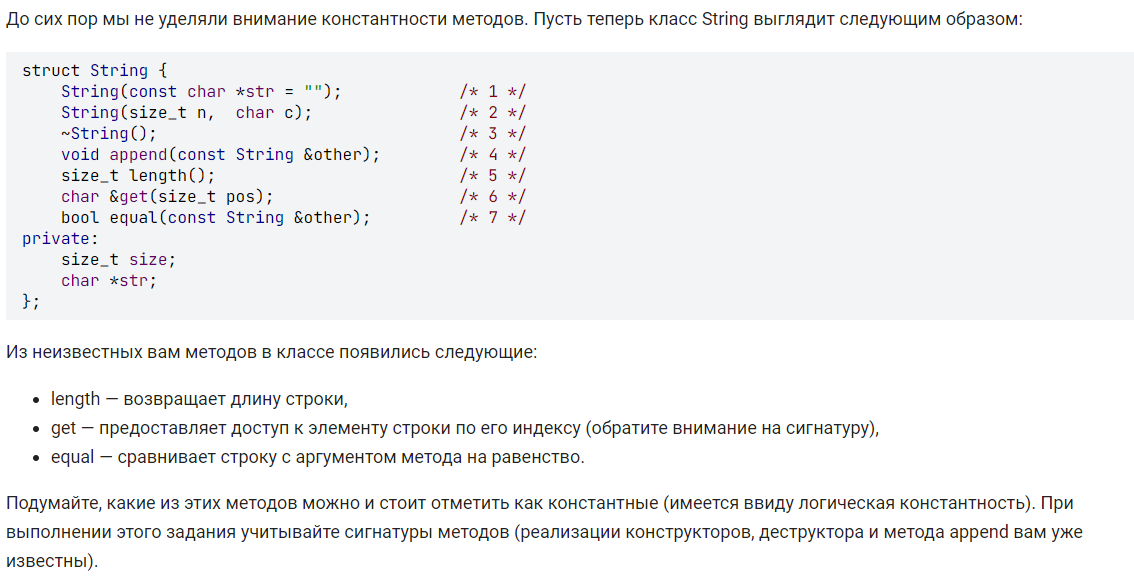
**Задача 1.**

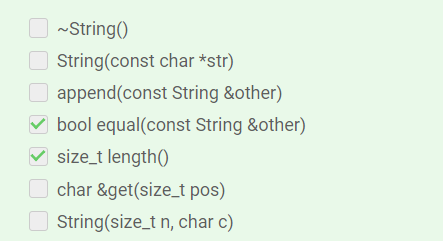
Конструкторов у структуры может быть несколько. Для строки может оказаться полезным заполняющий конструктор (например, чтобы создать строку пробелов). Заполняющий конструктор принимает число и символ, и создает строку с заданным количеством повторений переданного символа. Условия налагаемые на реализацию конструктора те же самые (в поле size размер без учета завершающего 0 символа, в поле str C-style строка, т.е. с завершающим 0 символом). Кроме конструктора в этой задаче вам нужно также реализовать и деструктор, который освободит выделенную память.

**Результат:**



**Задача 2.**

**Результат:**

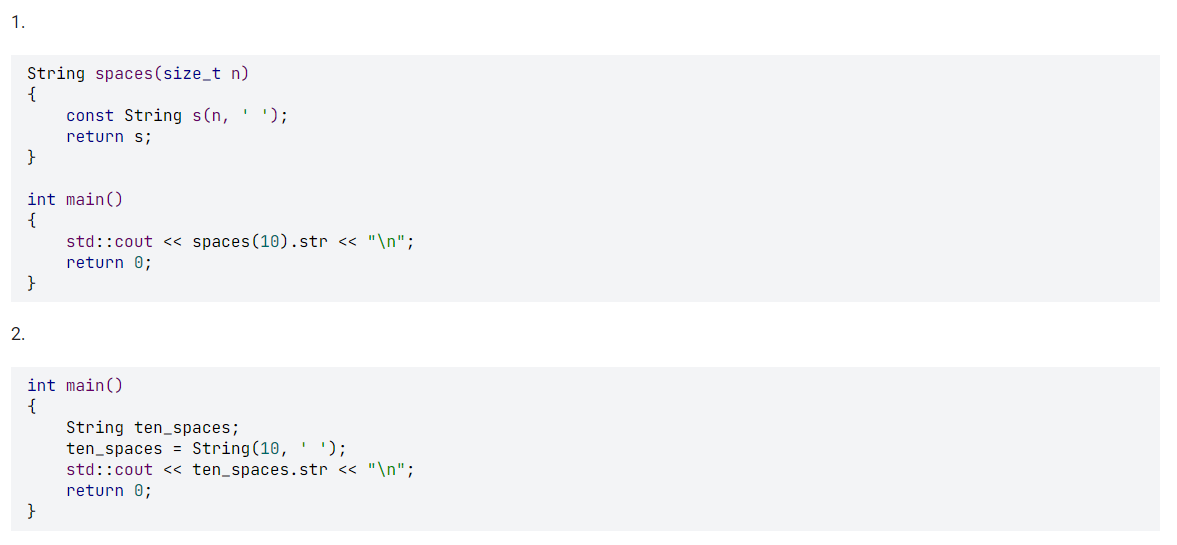


**Раздел 3.7 Конструктор копирования и оператор присваивания**

**Задача 1.**

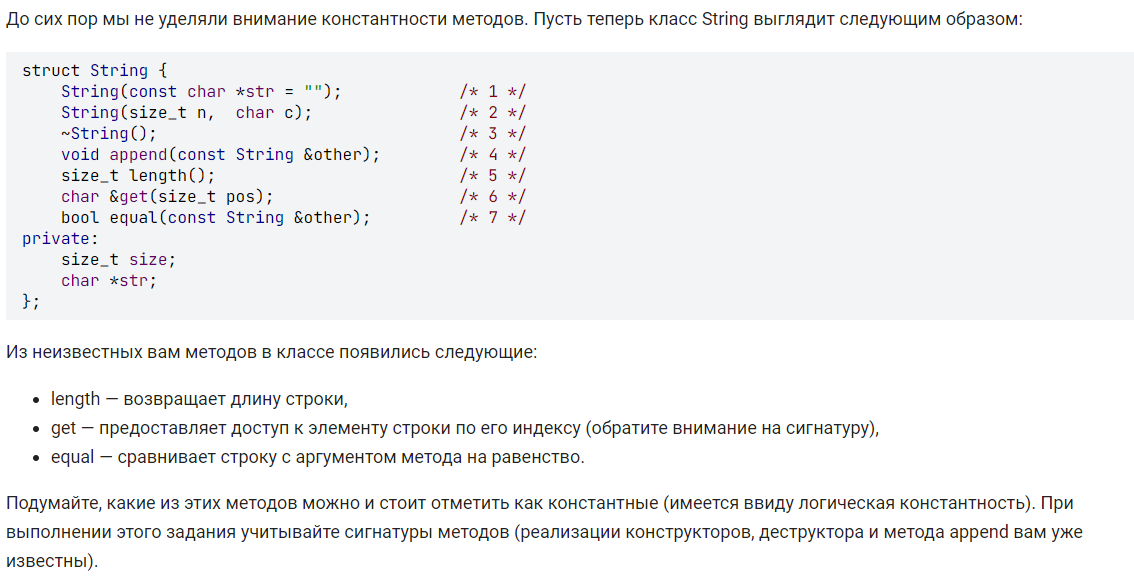
Компилятору C++ позволено выполнять оптимизации, связанные с удалением ненужных вызовов конструктора копирования (copy elision) при определенных условиях, даже если конструктор копирования содержит сайд эффекты (например, вывод в std::cout), поэтому на большинстве современных компиляторов конструктор копирования не будет вызван ни в одной из указанных ситуаций. Однако, компилятор не обязан выполнять эту оптимизацию.

В каких из приведенных ниже ситуаций может быть вызван конструктор копирования класса String:

**Результат:**

**Задача 2.**

**Результат:**



**Задача 2.**

Добавьте в класс String реализацию конструктора копирования. Инвариант класса остается тем же (в size хранится размер строки без завершающего 0 символа, str указывает на C-style строку, т. е. с завершающим нулевым символом).

Требования к реализации: вы можете заводить любые вспомогательные методы или функции, но не реализуйте заново методы из предыдущих заданий — они уже реализованы. При реализации не нужно вводить или выводить что-либо. Реализовывать main не нужно. Не используйте функции из cstdlib (malloc, calloc, realloc и free).

**Код:**

#include <cstddef> // size\_t

#include <cstring> // strlen, strcpy

struct String {

String(const char \*str = "");

String(size\_t n, char c);

~String();

String(const String &other)

: size(other.size) {

str = new char[other.size + 1];

strcpy(str, other.str);

}

void append(const String &other);

size\_t size;

char \*str;

};

**Результат:**



**Задача 3.**

Завершите класс String, добавив к нему оператор присваивания. Будьте аккуратны при работе с памятью. Инвариант класса остается тем же (в size хранится размер строки без завершающего 0 символа, str указывает на C-style строку, т. е. с завершающим нулевым символом).

**Код:**

#include <algorithm> // std::swap

#include <cstddef> // size\_t

#include <cstring> // strlen, strcpy

struct String {

String(const char \*str = "");

String(size\_t n, char c);

String(const String &other);

~String();

String &operator=(String other) {

if(other.size == size)

for(size\_t i = 0; i <= size; ++i)

this->str[i] = other.str[i];

else {

size = other.size;

char \* temp = new char[size + 1];

for(size\_t i = 0; i <= size; ++i)

temp[i] = other.str[i];

delete[] str;

str = temp;

}

return \*this;

}

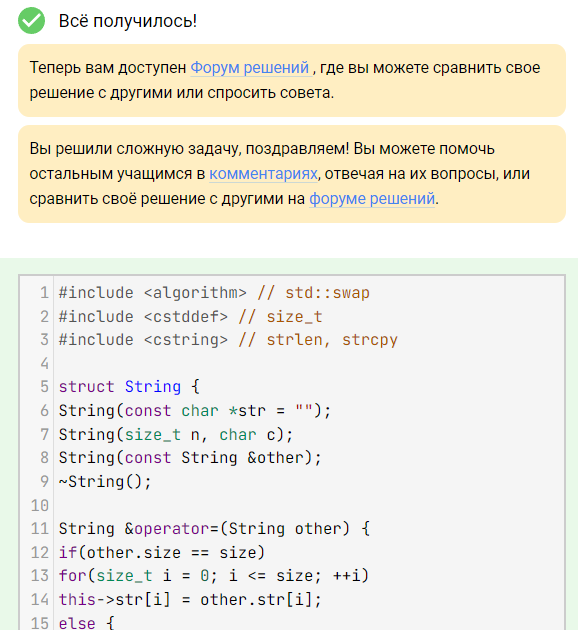
void append(const String &other);

size\_t size;

char \*str;

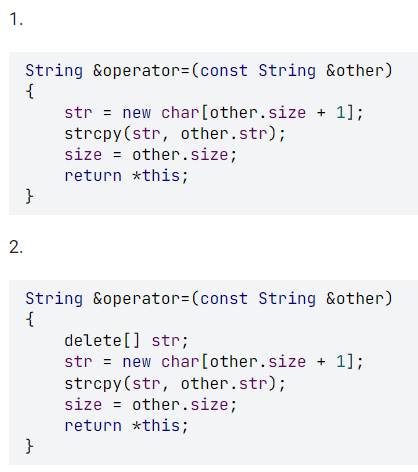
};

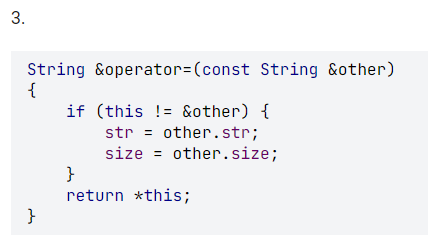
**Результат:**



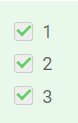
**Задача 4.**

Ниже даны несколько вариантов реализации оператора присваивания для класса String, выберите из них некорректные реализации, т.е. такие, которые неправильно работают с памятью (допускают утечки, обращаются к освобожденной памяти, используют неинициализированные указатели), нарушают семантику оператора присваивания или инвариант класса.





**Результат:**



**Раздел 4.1 Наследование**

**Задача 1.**

Свяжите следующие классы в иерархию наследования: Character (Персонаж), LongRange (Персонаж дальнего действия), Wizard (Маг), SwordsMan (Мечник), Archer (Лучник).Обратите внимание на методы, объявленные в классах — они приведены в качестве описания интерфейса, которым должны обладать классы. Не изменяйте объявлений методов — они даны как подсказка, просто добавьте наследование в определения классов.

**Код:**

#include <string> // std::string

struct Character

{

std::string const & name() const;

unsigned health() const;

};

struct LongRange : Character

{

std::string const & name() const;

unsigned health() const;

unsigned range() const;

};

struct SwordsMan : Character

{

std::string const & name() const;

unsigned health() const;

unsigned strength() const;

};

struct Wizard : LongRange

{

std::string const & name() const;

unsigned health() const;

unsigned range() const;

unsigned mana() const;

};

struct Archer : LongRange

{

std::string const & name() const;

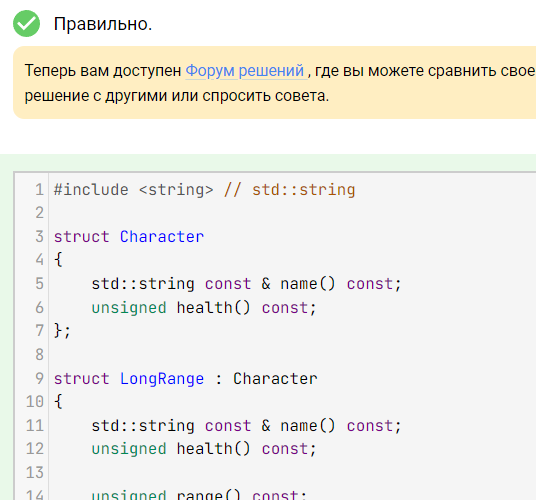
unsigned health() const;

unsigned range() const;

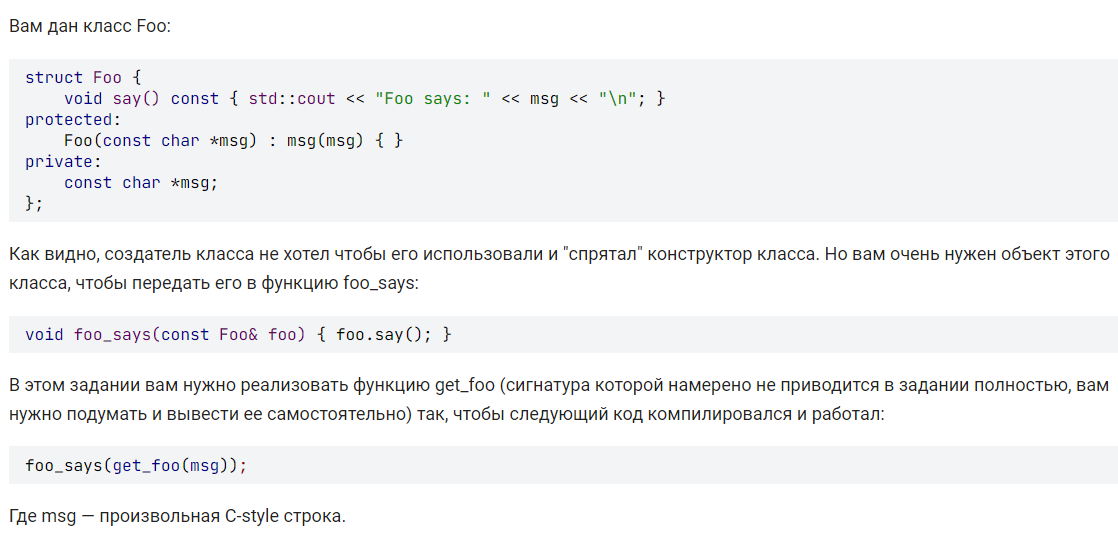
unsigned accuracy() const;

};

**Результат:**



**Задача 2.**

**Код:**

Foo get\_foo(const char \*msg) {

struct Foo1 : Foo {

public:

Foo1(const char \*msg) : Foo(msg) { }

};

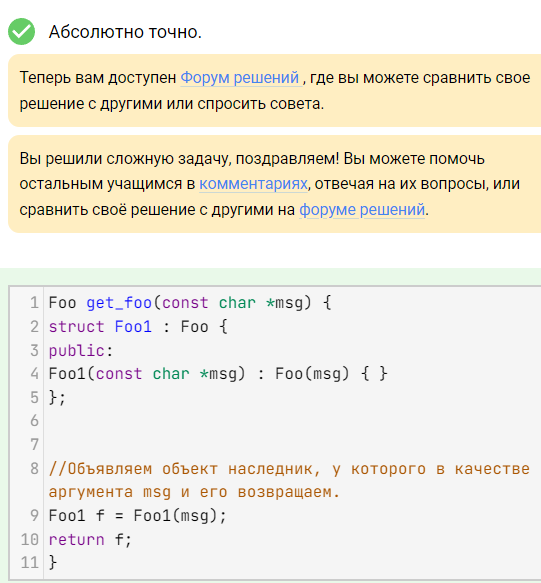
//Объявляем объект наследник, у которого в качестве аргумента msg и его возвращаем.

Foo1 f = Foo1(msg);

return f;

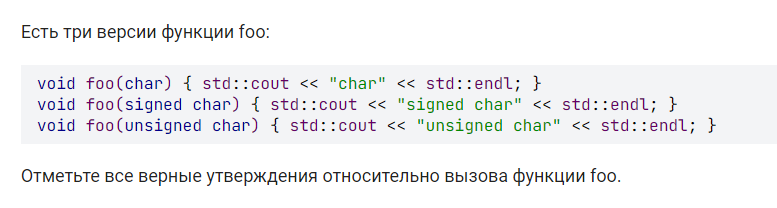
}

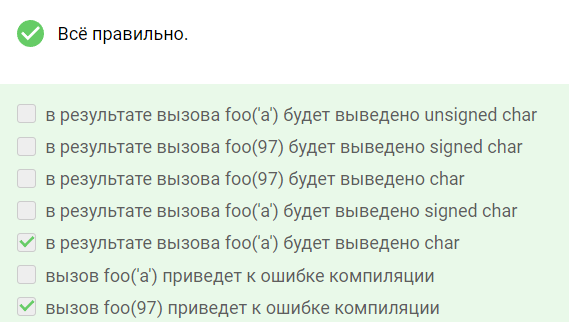
**Результат:**



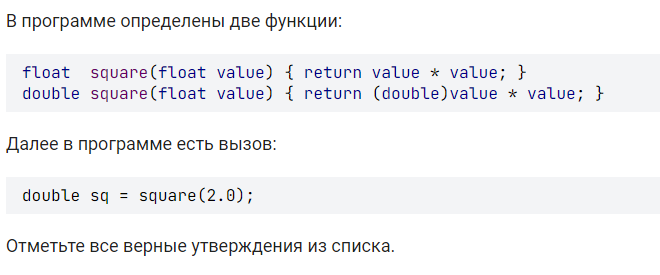
**Раздел 4.2 Перегрузка**

**Задача 1.**

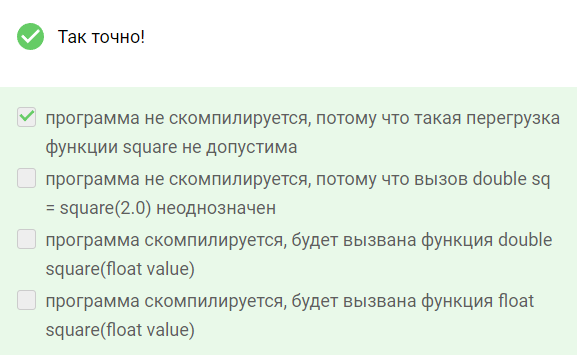
**Результат:**



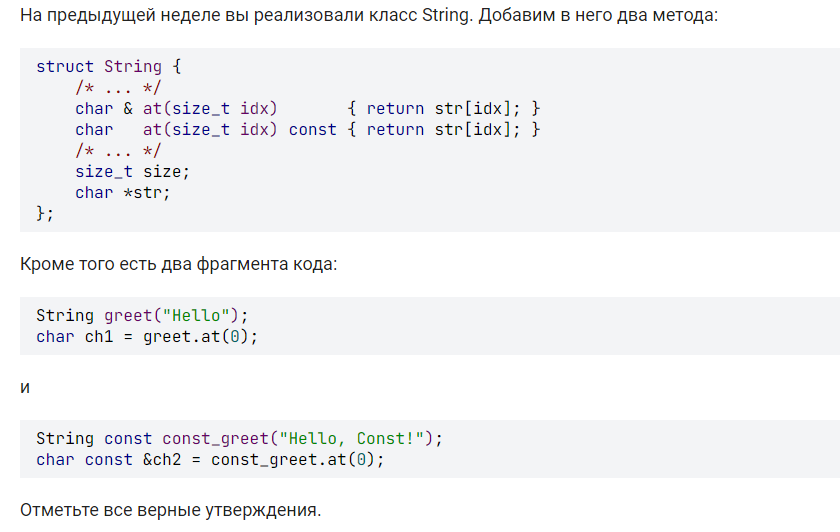
**Задача 2.**



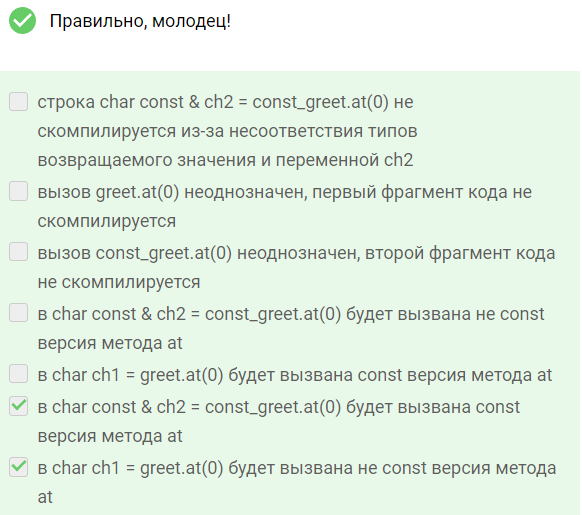
**Результат:**



**Задача 3.**



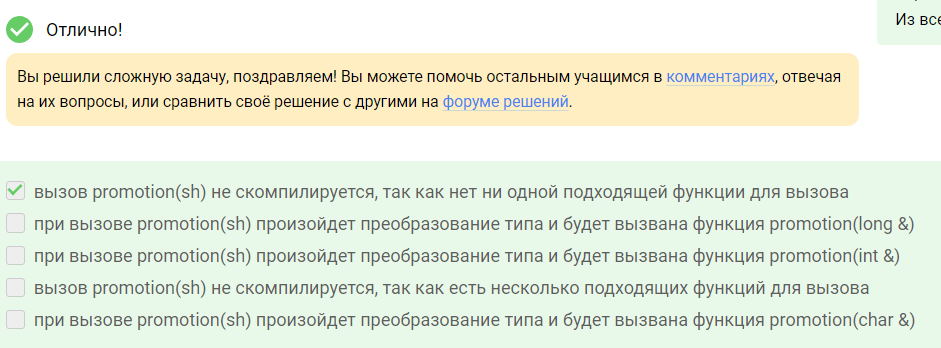
**Результат:**



**Задача 4.**



**Результат:**

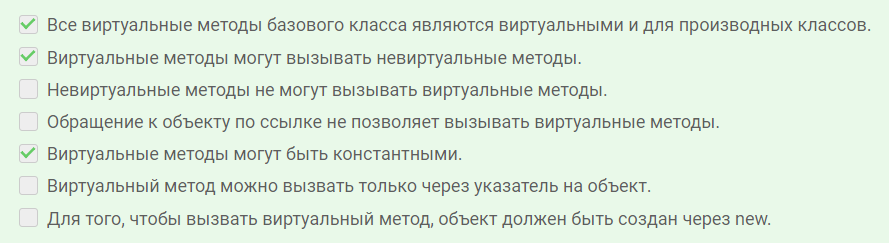


**Раздел 4.3 Виртуальные методы**

**Задача 1.**

Выберите все верные утверждения.

**Результат:**



**Задача 2.**

В этом задании вам предлагается разработать иерархию классов Expression для представления арифметических выражений. Конкретнее, вам нужно определить три класса: Expression — базовый класс иерархии, Number — для представления чисел и BinaryOperation — класс описывающий бинарную операцию (+, ­-, \* или /).

Класс Number должен хранить значение типа double.

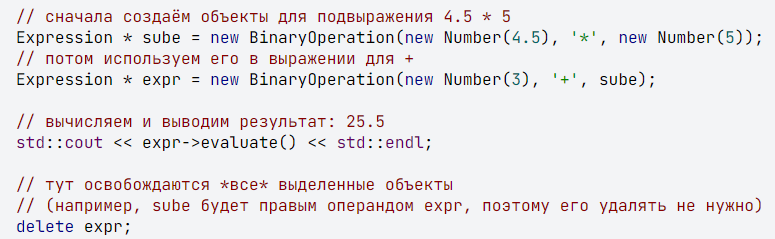
Класс BinaryOperation должен хранить указатель на левый и правый операнды, которые сами являются арифметическими выражениями, а также тип операции (+, ­-, \* или /), которую нужно над ними произвести.

Во всех классах должен быть метод evaluate, который возвращает значение типа double — значение соответствующего арифметического выражения, например, значение экземпляра типа Number — это число, которое он хранит, а если у вас есть объект BinaryOperation с операцией +, то нужно вычислить значения левого и правого операнда и вернуть их сумму.

В данном задании вам нужно расставить ключевое слово virtual там, где это необходимо, определить метод evaluate там, где его не хватает, а также реализовать деструкторы, там где они нужны.

При выполнении этого задания учтите, что при уничтожении объекта BinaryOperation он отвечает за уничтожение левого и правого операндов (гарантируется, что они выделены в динамической памяти).

Например, выражению 3 + 4.5 \* 5 будет соответствовать следующий код:



**Код:**

struct Expression {

virtual double evaluate() const = 0;

virtual ~Expression() {}

};

struct Number: public Expression {

Number(double value): value(value) {}

double evaluate() const { return value; }

private:

double value;

};

struct BinaryOperation: Expression {

BinaryOperation(Expression const \* left, char op, Expression const \* right) : left(left), op(op), right(right) {}

double evaluate() const {

double t = left->evaluate();

switch(op) {

case '+':

t += right->evaluate();

break;

case '-':

t -= right->evaluate();

break;

case '\*':

t \*= right->evaluate();

break;

default:

t /= right->evaluate();

break;

}

return t;

}

~BinaryOperation() {

delete left;

delete right;

}

private:

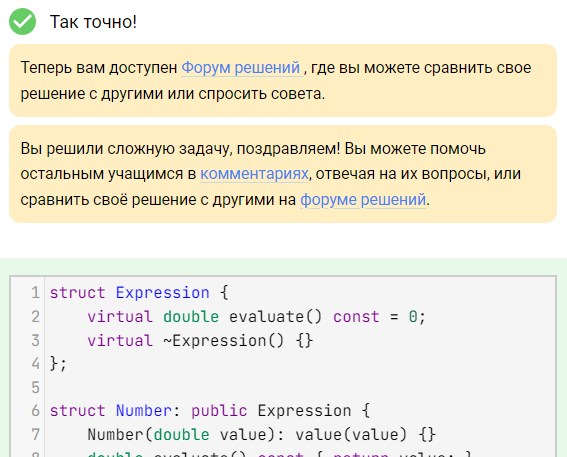
Expression const \* left;

Expression const \* right;

char op;

};

**Результат:**



**Раздел 4.4 Таблица виртуальных методов**

**Задача 1.**

Задание повышенной сложности) Предполагаемое решение этого задания не переносимо с точки зрения стандарта, однако оно проверено на различных версиях компиляторов g++/clang++/msvc. Решение этого задания не обязательно для получения диплома по этому курсу, однако мы считаем его полезным, чтобы лучше понять как устроены виртуальные функции внутри.

Вам требуется реализовать функцию, которая принимает на вход два указателя на базовый класс Expression, и возвращает true, если оба указателя указывают на самом деле на объекты одного и того же класса, и false в противном случае (т.е. если оба указателя указывают на BinaryOperation, то возвращается true, а если один из них указывает на Number, а второй на BinaryOperation, то false).

**Код:**

#include <iostream>

#include <iomanip>

#include <cstring>

class Base {

public:

int i;

virtual void method()

{

std::cout << "Base\n";

}

virtual void method2()

{

std::cout << "Base2\n";

}

virtual void method3()

{

std::cout << "Base3\n";

}

virtual void method4()

{

std::cout << "Base4\n";

}

};

class Delivered : public Base {

public:

long j;

virtual void method() override

{

std::cout << "Delivered\n";

}

virtual void method2() override

{

std::cout << "Delivered2\n";

}

virtual void method3() override

{

std::cout << "Delivered3\n";

}

virtual void method4() override

{

std::cout << "Delivered4\n";

}

};

void print\_data(void\* data, int size, int bytes\_per\_line = 8)

{

unsigned char\* data\_ = reinterpret\_cast<unsigned char\*>(data);

for (int i = 0; i < size; ++i) {

if (i % bytes\_per\_line == 0)

std::cout << '\n' << i << " | ";

std::cout << std::setfill('0') << std::setw(2) << std::hex << static\_cast<int>(data\_[i]) << " ";

}

std::cout << '\n';

}

void\* get\_vtable\_address(void\* object)

{

void\* vtable;

std::memcpy(&vtable, object, sizeof(void\*));

return vtable;

}

bool check\_equals(void\* obj1, void\* obj2)

{

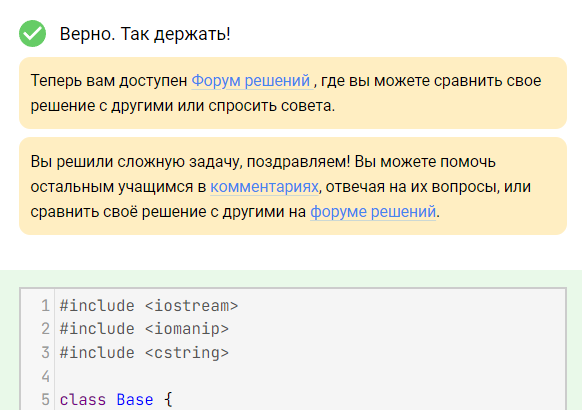
void\* vtable1 = get\_vtable\_address(obj1);

void\* vtable2 = get\_vtable\_address(obj2);

return vtable1 == vtable2;

}

**Результат:**



**Раздел 4.5 Объектно-ориентированное программирование**

**Задача 1.**

а прошлом степе вы познакомились с реализацией паттерна посетитель. Теперь пришло время воспользоваться этими знаниями. Ваш класс должен печатать (используя std::cout) текстовое представление арифметического выражения. Т.е. для объекта класса Number он должен напечатать число, которое в нем хранится, а для объекта класса BinaryOperation он должен напечатать левый операнд, затем операцию, а затем правый операнд.

Учтите, что операции + и - имеют меньший приоритет, чем операции \* и /, т. е. возможно вам придется печатать дополнительные круглые скобки, чтобы сохранить правильный порядок операций.

Классы иерархии Expression и абстрактный класс Visitor приведены в комментарии в шаблоне для удобства.

При проверке задания лишние пробелы или лишние скобки будут игнорироваться, т.е. вывод "1+2" эквивалентен выводу "( ( 1 ) + ( 2 ) )", а вот вывод "1 + 2 \* 3" не эквивалентен выводу "((1 + 2) \* 3)". Вы можете попытаться минимизировать количество скобок в выводе, но это не требуется.

Перед выполнением задания внимательно прочитайте содержимое предыдущего степа.

**Код:**

#include <iostream>

/\* Этот класс вам нужно реализовать \*/

struct PrintVisitor: Visitor {

void visitNumber(Number const \* number) {

double x = number->evaluate();

if(x < 0)

std::cout << "(" << x << ")";

else

std::cout << x ;

}

void visitBinaryOperation(BinaryOperation const \* bop) {

bool is\_prev = this->is; //исходное

char c = bop->get\_op();

bool is\_cur = is\_prev && (c == '-' || c == '+'); //снаруже \* или /, а у нас + или - => нужны скобки

if(is\_cur)

std::cout << "(";

this->is = c=='\*'||c=='/'; //сообщаем внутрь, что у нас приоритетная операция

bop->get\_left()->visit(this);

std::cout << c;

this->is = c=='\*'||c =='/'; //сообщаем внутрь, что у нас приоритетная операция

bop->get\_right()->visit(this);

if(is\_cur)

std::cout << ")";

this->is = is\_prev; //вернем

}

private:

bool is=false;

};

**Результат:**

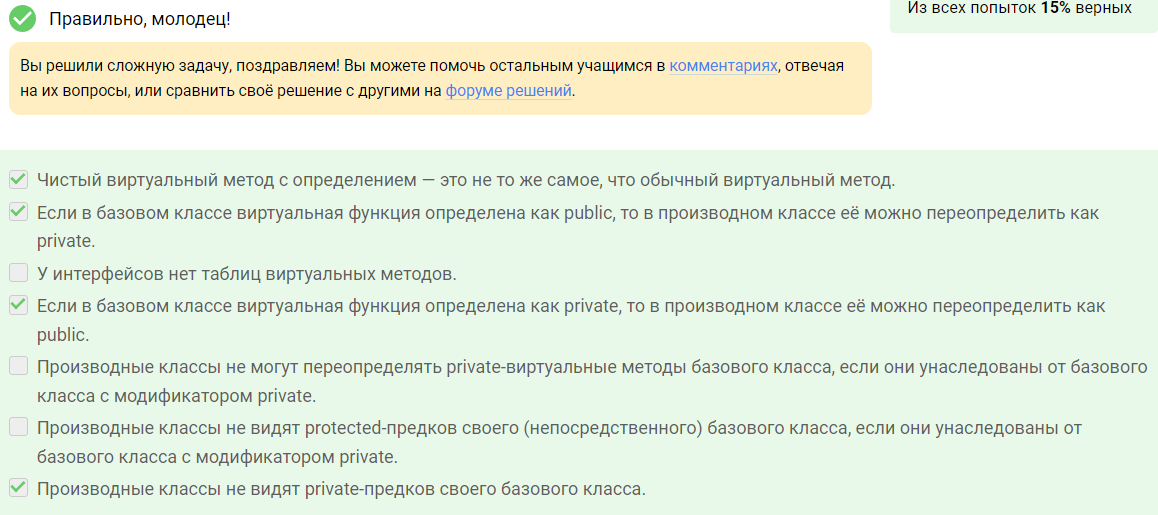


**Раздел 4.6 Особенности наследования в C++**

**Задача 1.**

Выберите все верные утверждения.

**Результат:**



**Раздел 5.2 Правила переопределения операторов**

**Задача 1.**

Вам дан класс Rational, который описывает рациональное число. В нем определены методы add, sub, mul и div, которые прибавляют к нему число, отнимают число, умножают на число и делят на число соответственно. Кроме того в нем определен метод neg, который меняет знак на противоположный.

Вам нужно определить операторы +=, -=, \*=, /= для класса Rational так, чтобы они могли принимать в качестве аргументов и объекты типа Rational и целые числа. Кроме того вам необходимо определить операторы унарного минуса и плюса для класса Rational.

**Код:**

struct Rational

{

Rational(int numerator = 0, int denominator = 1);

void add(Rational rational);

void sub(Rational rational);

void mul(Rational rational);

void div(Rational rational);

void neg();

void inv();

double to\_double() const;

Rational & operator+=(const Rational & rational) {

this->add(rational);

return \*this;

}

Rational & operator+=(int d) {

this->add(Rational(d));

return \*this;

}

Rational & operator-=(const Rational & rational) {

this->sub(rational);

return \*this;

}

Rational & operator-=(int d) {

this->sub(Rational(d));

return \*this;

}

Rational & operator\*=(const Rational & rational) {

this->mul(rational);

return \*this;

}

Rational & operator\*=(int d) {

this->mul(Rational(d));

return \*this;

}

Rational & operator/=(const Rational & rational) {

this->div(rational);

return \*this;

}

Rational & operator/=(int d) {

this->div(Rational(d));

return \*this;

}

Rational operator-() const { Rational r(this->numerator\_, this->denominator\_); r.neg(); return r; }

Rational operator+() const { Rational r(this->numerator\_, this->denominator\_); return r; }

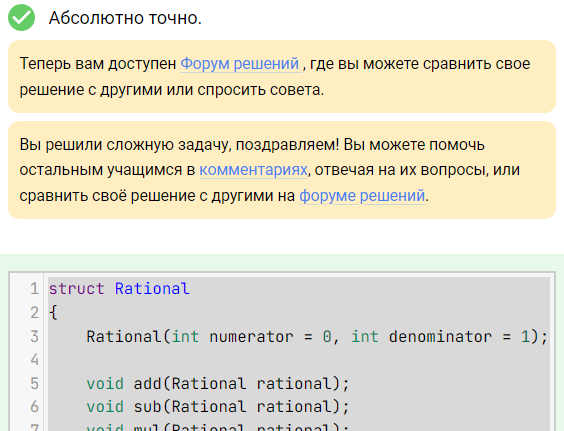
private:

int numerator\_;

unsigned denominator\_;

};

**Результат:**



**Задача 2.**

Определите для класса Rational операторы сложения, вычитания, умножения и деления, так чтобы объекты типа Rational можно было складывать (вычитать, умножать и делить) не только друг с другом но и с целыми числами.

**Код:**

struct Rational

{

Rational(int numerator = 0, int denominator = 1);

void add(Rational rational);

void sub(Rational rational);

void mul(Rational rational);

void div(Rational rational);

void neg();

void inv();

double to\_double() const;

Rational& operator+=(Rational rational);

Rational& operator-=(Rational rational);

Rational& operator\*=(Rational rational);

Rational& operator/=(Rational rational);

Rational operator-() const;

Rational operator+() const;

Rational operator+(const Rational & rational) const {

Rational r(this->numerator\_, this->denominator\_);

r.add(rational);

return r;

}

Rational operator+(int d) const {

Rational r(this->numerator\_, this->denominator\_);

r.add(Rational(d));

return r;

}

friend Rational operator+(int d, const Rational & rational) {

return rational + d;

}

Rational operator-(const Rational & rational) const {

Rational r(this->numerator\_, this->denominator\_);

r.sub(rational);

return r;

}

Rational operator-(double d) const {

Rational r(this->numerator\_, this->denominator\_);

r.sub(Rational(d));

return r;

}

friend Rational operator-(int d, const Rational & rational) {

return -(rational - d);

}

Rational operator\*(const Rational & rational) const {

Rational r(this->numerator\_, this->denominator\_);

r.mul(rational);

return r;

}

Rational operator\*(int d) const {

Rational r(this->numerator\_, this->denominator\_);

r.mul(Rational(d));

return r;

}

friend Rational operator\*(int d, const Rational & rational) {

return rational \* d;

}

Rational operator/(const Rational & rational) const {

Rational r(this->numerator\_, this->denominator\_);

r.div(rational);

return r;

}

Rational operator/(int d) const {

Rational r(this->numerator\_, this->denominator\_);

r.div(Rational(d));

return r;

}

friend Rational operator/(int d, const Rational & rational) {

return Rational(d)/rational;

}

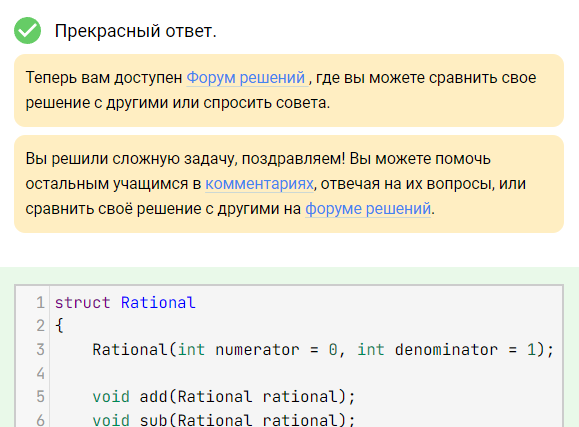
private:

int numerator\_;

unsigned denominator\_;

};

**Результат:**



**Задача 3.**

Еще одна важная группа операторов, которые полезно реализовать для класса рациональных чисел — это операторы сравнения. Реализуйте операторы <, <=, >, >=, ==, != для класса Rational так, чтобы можно было сравнивать объекты класса Rational не только друг с другом, но и с целыми числами.

При решении задания не используйте метод to\_double, он введен в класс, в первую очередь, для удобства тестирования. Вы можете определять любые вспомогательные методы или функции если необходимо.

**Код:**

struct Rational {

public:

int numerator\_;

int denominator\_;

Rational(int numerator = 0, int denominator = 1);

void add(Rational rational);

void sub(Rational rational);

void mul(Rational rational);

void div(Rational rational);

void neg();

void inv();

double to\_double() const;

Rational& operator+=(Rational rational);

Rational& operator-=(Rational rational);

Rational& operator\*=(Rational rational);

Rational& operator/=(Rational rational);

Rational operator-() const;

Rational operator+() const;

bool operator == (Rational const & b) {

return (numerator\_ \* b.denominator\_== b.numerator\_ \* denominator\_);

}

bool operator == (int b) {

return (numerator\_ == (double)b \* denominator\_);

}

bool operator != (Rational const & b) {

return !(\*this == b);

}

bool operator != (int b) {

return !(\*this == b);

}

bool operator > (Rational const & b) {

return (numerator\_ \* b.denominator\_ > b.numerator\_ \* denominator\_);

}

bool operator > (int b) {

return (numerator\_ > (double)b \* denominator\_);

}

bool operator <= (Rational const & b) {

return !(\*this > b);

}

bool operator <= (int b) {

return !(\*this > b);

}

bool operator < (Rational const & b) {

return (numerator\_ \* b.denominator\_ < b.numerator\_ \* denominator\_);

}

bool operator < (int b) {

return (numerator\_ < (double)b \* denominator\_);

}

bool operator >= (Rational const & b) {

return !(\*this < b);

}

bool operator >= (int b) {

return !(\*this < b);

}

};

Rational operator+(Rational lhs, Rational rhs);

Rational operator-(Rational lhs, Rational rhs);

Rational operator\*(Rational lhs, Rational rhs);

Rational operator/(Rational lhs, Rational rhs);

bool operator == (int a, Rational const & b) {

return (b.numerator\_ == a \* b.denominator\_);

}

bool operator != (int a, Rational const & b) {

return !(a == b);

}

bool operator > (int a, Rational const & b) {

return (b.numerator\_ < (double)a \* b.denominator\_);

}

bool operator <= (int a, Rational const & b) {

return !(a > b);

}

bool operator < (int a, Rational const & b) {

return (b.numerator\_ > (double)a \* b.denominator\_);

}

bool operator >= (int a, Rational const & b) {

return !(a < b);

}

**Результат:**



**Задача 4.**

Добавьте в класс Rational оператор приведения к double. Все операторы из предыдущих заданий отсутствуют и реализовывать их не нужно. Метод to\_double можно использовать в этом задании.

**Код:**

struct Rational

{

Rational(int numerator = 0, int denominator = 1);

void add(Rational rational);

void sub(Rational rational);

void mul(Rational rational);

void div(Rational rational);

void neg();

void inv();

double to\_double() const;

operator double() const{

return to\_double();

}

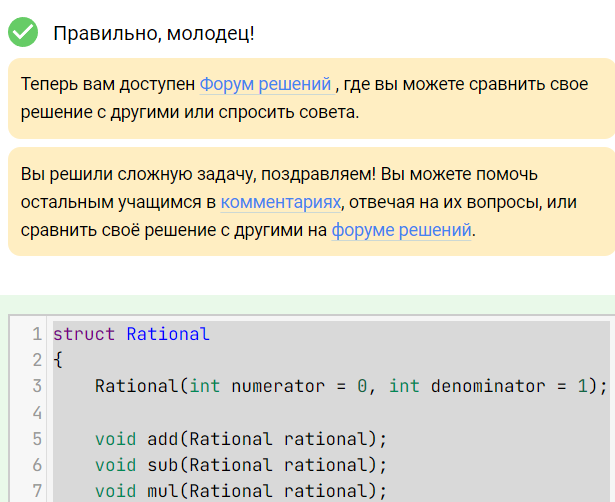
private:

int numerator\_;

int denominator\_;

};

**Результат:**



**Задача 5.**

В этой задаче вам требуется реализовать оператор [] для уже известного вам класса String. Однако на этот раз оператор должен реализовывать нестандартное поведение: оператор нужно реализовать таким образом, чтобы для объекта str класса String можно было писать str[i][j] и это выражение возвращало подстроку начинающуюся в позиции i (считая с 0) и заканчивающуюся в позиции j (не включая).

**Код:**

#include <cstddef> // size\_t

#include <iostream>

struct String {

String(const char \*str = "");

String(size\_t n, char c);

~String();

String(const String &other);

String &operator=(const String &other);

void append(const String &other);

class PreparedString {

public:

void write(std::ostream& out) const;

String operator[](unsigned int to) const;

PreparedString(const String& source\_in, unsigned int from\_in);

private:

const String& source;

unsigned int from;

};

void write(std::ostream& out) const;

PreparedString operator[](unsigned int from) const;

size\_t size;

char \*str;

};

void String::PreparedString::write(std::ostream& out) const

{

out << (this->source.str+this->from) << std::endl;

}

String String::PreparedString::operator[](unsigned int to) const

{

char \*tmp\_c\_str = new char[to-from+1];

for(char \*at\_src = source.str+from, \*at\_dst = tmp\_c\_str, \*last = source.str+to; at\_src != last; ++at\_src, ++at\_dst)

\*at\_dst = \*at\_src;

\*(tmp\_c\_str+to-from) = '\0';

String tmp\_string(tmp\_c\_str);

delete[] tmp\_c\_str;

return tmp\_string;

}

String::PreparedString::PreparedString(const String& source\_in, unsigned int from\_in): source(source\_in), from(from\_in)

{

}

String::PreparedString String::operator[](unsigned int from) const

{

return String::PreparedString(\*this,from);

}

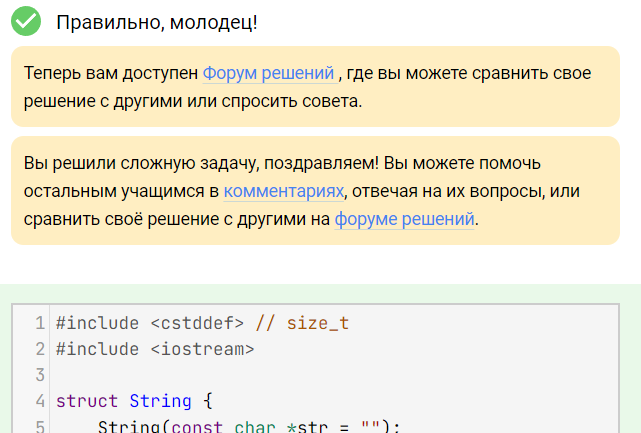
void String::write(std::ostream& out) const

{

out << this->str << std::endl;

}

**Результат:**



**Раздел 5.3 Умные указатели**

**Задача 1.**

Помните иерархию Expression, которую мы разрабатывали на четвёртой неделе? Реализуйте ScopedPtr, который будет работать с указателями на базовый класс Expression. В этом задании вам требуется реализовать методы get, release и reset, операторы \* и -> так, как это было описано в предыдущем степе. А также реализуйте конструктор ScopedPtr от указателя на Expression.

**Код:**

struct Expression;

struct Number;

struct BinaryOperation;

struct ScopedPtr {

// реализуйте следующие методы:

//

explicit ScopedPtr(Expression \*ptr = 0) {

this->ptr\_=ptr;

}

~ScopedPtr() {

delete ptr\_;

}

Expression\* get() const {

return ptr\_;

}

Expression\* release() {

Expression \* ptr = this->ptr\_;

this->ptr\_ = NULL;

return ptr;

}

void reset(Expression \*ptr = 0) {

delete this->ptr\_;

this->ptr\_ = ptr;

}

Expression& operator\*() const {

return \*ptr\_;

}

Expression\* operator->() const {

return ptr\_;

}

private:

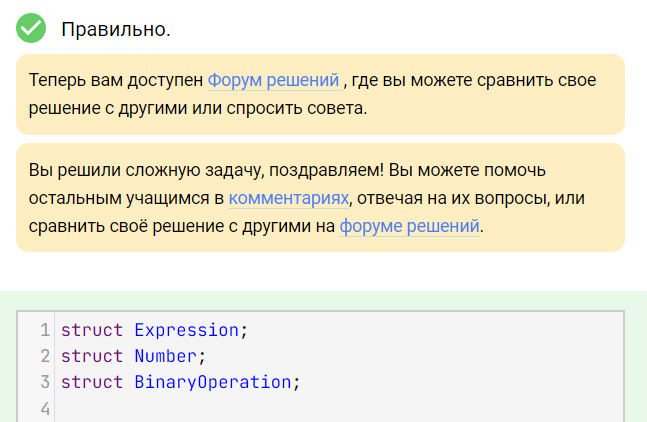
ScopedPtr& operator=(const ScopedPtr & other) {} //оператор присваивания

ScopedPtr(const ScopedPtr & other) {} //конструктор копирования

Expression \*ptr\_;

};

**Результат:**



**Задача 2.**

Реализуйте класс SharedPtr как описано ранее. Задание немного сложнее, чем кажется на первый взгляд. Уделите особое внимание "граничным случаям" — нулевой указатель, присваивание самому себе, вызов reset на нулевом SharedPtr и прочее.

**Код:**

struct Expression;

struct Number;

struct BinaryOperation;

struct Help {

Help(Expression \*ptr=NULL) {

this->ptr = ptr;

cnt = 1;

}

~Help() {

delete ptr;

}

Expression \* ptr;

mutable int cnt ;

};

struct SharedPtr {

// реализуйте следующие методы

//

explicit SharedPtr(Expression \*ptr = 0) {

if(ptr != 0)

h = new Help(ptr);

else

h = NULL;

}

~SharedPtr() {

if(h!=NULL) {

--(h->cnt);

if(h->cnt == 0)

delete h;

}

}

SharedPtr(const SharedPtr & other) {

this->h = other.h;

if(this->h!=NULL && this->h->cnt)

++(this->h->cnt);

}

SharedPtr& operator=(const SharedPtr & other) {

if(this!=&other){

this->~SharedPtr();

this->h = other.h;

if(this->h != NULL)

++(this->h->cnt);

}

}

Expression\* get() const {

if(this->h != NULL)

return this->h->ptr;

else

return NULL;

}

void reset(Expression \*ptr = 0) {

this->~SharedPtr();

if(ptr != 0)

h = new Help(ptr);

else

h = NULL;

}

Expression& operator\*() const {

if(this->h != NULL)

return \*(this->h->ptr);

else

return \*(Expression \*)NULL;

}

Expression \* operator->() const {

if(this->h != NULL)

return this->h->ptr;

else

return NULL;

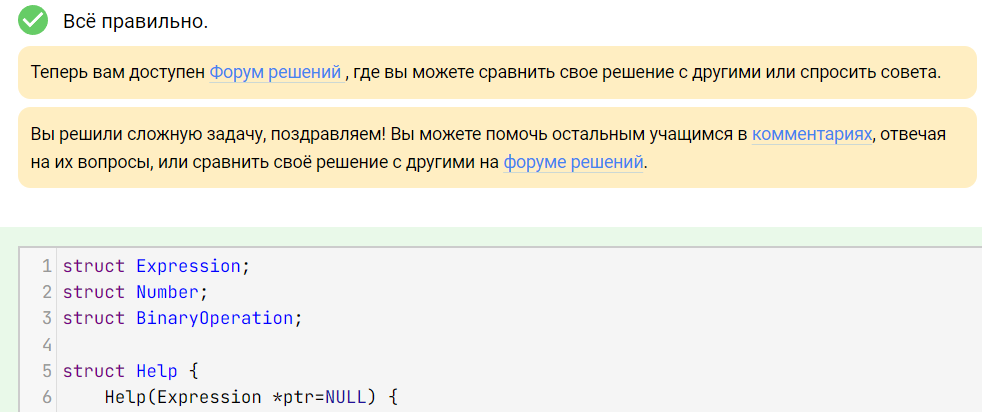
}

private:

Help \* h=NULL;

};

**Результат:**



**Раздел 6.1 Шаблоны классов**

**Задача 1.**

Реализуйте шаблонную версию класса Array. Список всех операций, которые должен поддерживать класс Array, приведен в шаблоне кода.

**Код:**

#include <cstddef>

template <typename T>

class Array

{

private:

size\_t size\_;

T \*data\_;

public:

// Конструктор класса

explicit Array(size\_t size = 0, const T &value = T()) : size\_(size), data\_(new T[size\_])

{

for (size\_t i = 0; i < size\_; ++i)

{

data\_[i] = value;

}

}

// Конструктор копирования

Array(const Array &other) : size\_(other.size\_), data\_(new T[size\_])

{

for (size\_t i = 0; i < size\_; ++i)

{

data\_[i] = other.data\_[i];

}

}

// Деструктор

~Array()

{

delete[] data\_;

}

// Оператор присваивания

Array &operator=(const Array &other)

{

if (this != &other)

{

delete[] data\_;

size\_ = other.size\_;

data\_ = new T[size\_];

for (size\_t i = 0; i < size\_; ++i)

{

data\_[i] = other.data\_[i];

}

}

return \*this;

}

// Метод возвращает размер массива

size\_t size() const

{

return size\_;

}

// Оператор доступа по индексу

T &operator[](size\_t index)

{

return data\_[index];

}

// Константная версия оператора доступа по индексу

const T &operator[](size\_t index) const

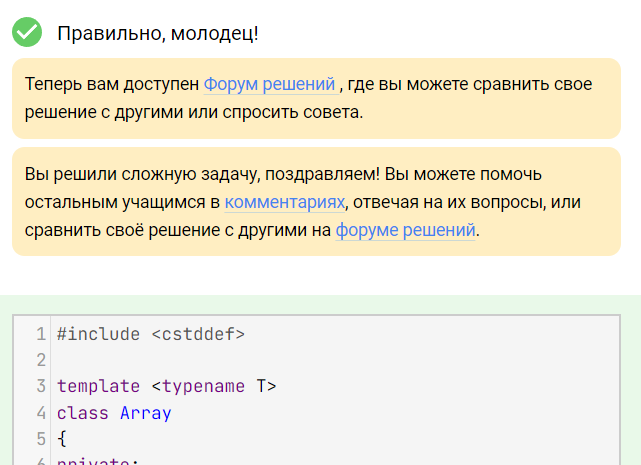
{

return data\_[index];

}

};

**Результат:**



**Задача 2.**

В предыдущей версии предполагается, что для типа T определены оператор присваивания и конструктор по умолчанию. При создании шаблонных классов контейнеров (вроде Array и не только) разумно стараться минимизировать требования к типам шаблонных параметров. Поэтому усложним задачу, реализуйте класс Array не полагаясь на то, что для типа T определен оператор присваивания и конструктор по умолчанию. Конструктор копирования у типа T есть.

Hints: используйте placement new и явный вызов деструктора (смотрите урок 3.4), чтобы создавать и уничтожать объекты, аллоцировать правильно выровненную память можно с помощью new char[N \* sizeof(T)], где N - количество элементов массива.

**Код:**

#include <cstddef>

template <typename T>

class Array

{

public:

Array(size\_t size, const T& value = T())

{

size\_ = size;

//Так как не существует конструктора по умолчанию, выделяем память через char(1 байт \* Размер T) и приводим //к нужному нам типу

data\_ = (T\*) new char[size\_ \* sizeof(T)];

//Потом проходим по массиву и явно вызываем конструктор с параметром

for (size\_t i = 0; i != size\_; ++i)

new (data\_ + i) T(value);

//У типа T Есть конструктор с параметром и нет оператора присваивания

//Не можем написать data[i] = value;

//Память уже выделена выше, поэтому используем Placement new с конструктором с параметром

}

Array()

{

size\_ = 0;

data\_ = 0;

}

Array(const Array & obj)

{

size\_ = obj.size();

//Абсолютно такая же логика для конструктора копирования

data\_ = (T\*) new char[size\_ \* sizeof(T)];

for (size\_t i = 0; i != size\_; ++i)

new (data\_ + i) T(obj[i]);

}

Array& operator=(const Array & obj)

{

if (this != &obj)

{

//Выкидываем то что было через наш недо деструктор

mem\_free();

//см конструктор

size\_ = obj.size();

data\_ = (T\*) new char[size\_ \* sizeof(T)];

for (size\_t i = 0; i != size\_; ++i)

new (data\_ + i) T(obj[i]);

}

return \*this;

}

size\_t size() const

{

return size\_;

}

T& operator[](size\_t i)

{

return \*(data\_ + i);

}

const T& operator[](size\_t i) const

{

return \*(data\_ + i);

}

~Array()

{

//Не можем сделать delete [] data\_ т. к выделяли память через бубен

mem\_free();

}

void mem\_free()

{

for (size\_t i = 0; i != size\_; ++i)

data\_[i].~T(); // Явно вызываем деструктор у каждого элемента

delete [] (char \*) data\_; // Выделяли память через [] char? Удаляем через [] char!

}

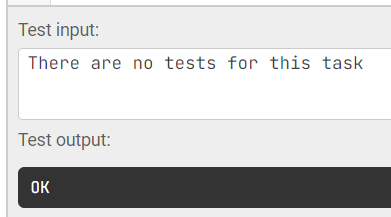
private :

size\_t size\_ ;

T \* data\_ ;

};

**Результат:**



**Раздел 6.2 Шаблоны функций**

**Задача 1.**

Реализуйте функцию копирования элементов copy\_n из массива источника типа U\* в целевой массив типа T\*, где T и U произвольные типы, для которых определено преобразование из U в T. На вход функция принимает два указателя и количество элементов, которые необходимо скопировать.

**Код:**

#include <cstddef>

#include <iostream>

template <class U, class T>

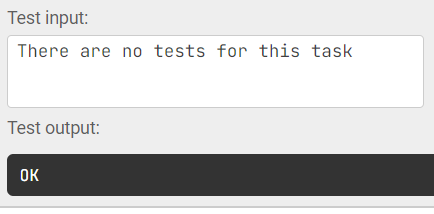
void copy\_n(T\* mass1, U\* mass2, size\_t e) {

for (int q = 0; q != e; q++)

mass1[q] = (T)mass2[q];

}

**Результат:**



**Задача 2.**

Реализуйте шаблонную функцию minimum, которая находит минимальный элемент, который хранится в экземпляре шаблонного класса Array, при этом типовой параметр шаблона Array может быть произвольным. Чтобы сравнивать объекты произвольного типа, на вход функции также будет передаваться компаратор, в качестве компаратора может выступать функция или объект класса с перегруженным оператором "()"

**Код:**

#include <cstddef>

template <typename T>

class Array

{

public:

explicit Array(size\_t size = 0, const T& value = T());

Array(const Array& other);

~Array();

Array& operator=(Array other);

void swap(Array &other);

size\_t size() const;

T& operator[](size\_t idx);

const T& operator[](size\_t idx) const;

private:

size\_t size\_;

T \*data\_;

};

// put your code here

template<typename T, typename Cmp>T minimum(Array<T> a, Cmp less) {

T min = a[0];

size\_t l = a.size();

for(size\_t i = 1; i < l; ++i)

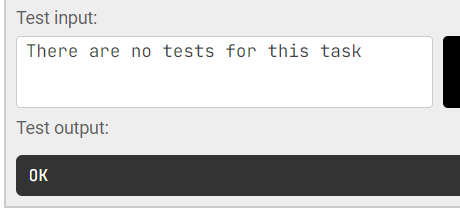
if(less(a[i],min))

min = a[i];

return min;

}

**Результат:**



**Задача 3.**

Шаблонный класс Array может хранить объекты любого типа, для которого определён конструктор копирования, в том числе и другой Array, например, Array< Array<int> >. Глубина вложенности может быть произвольной. Напишите шаблонную функцию (или несколько) flatten, которая принимает на вход такой "многомерный" Array неизвестной заранее глубины вложенности и выводит в поток out через пробел все элементы, хранящиеся на самом нижнем уровне.

**Код:**

#include <iostream>

template <typename Type>

void flatten(const Array<Type>& array, std::ostream& out)

{

for (size\_t i=0; i != array.size(); ++i){

out << array[i] << " ";

}

}

template <typename Type>

void flatten(const Array< Array<Type> >& array, std::ostream& out)

{

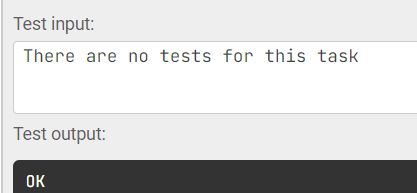
for (size\_t i=0; i != array.size(); ++i){

flatten(array[i], out);

}

}

**Результат:**



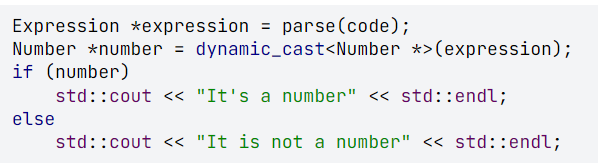
**Задача 4.**

В первом уроке вы реализовали простой шаблон ValueHolder, в этом задании мы используем его чтобы написать класс Any (интересно, что не шаблонный), который позволяет хранить значения любого типа! Например, вы сможете создать массив объектов типа Any, и сохранять в них int-ы, double-ы или даже объекты Array. Подробности в шаблоне кода.

Hint: в нешаблонном классе Any могут быть шаблонные методы, например, шаблонный конструктор.

Для реализации вам может потребоваться оператор dynamic\_cast.

С++ предоставляет и более простой способ узнать конкретный класс по указателю (или ссылке) на базовый класс. Для этого можно использовать оператор dynamic\_cast. Например, если у вас есть указатель Expression \*, и вы хотите узнать, указывает ли этот указатель на самом деле на объект Number, то сделать это можно так:



Если expression действительно указывает на объект Number (или на один из его наследников, но в нашем примере их нет), то оператор dynamic\_cast<Number \*> вернет правильный указатель. Если expression указывает не на класс Number, то будет возвращен нулевой указатель. Т. е. если в переменной number хранится нулевой указатель, значит expression не указывает на Number на самом деле.

Для правильной работы оператора dynamic\_cast в нашем примере требуется, чтобы класс Expression был полиморфным, т. е. в нем должна быть как минимум одна виртуальная функция (например, деструктор). Стандарт определяет работу dynamic\_cast и с не полиморфными типами, но в этом случае никаких проверок типа времени исполнения выполняться не будет - часто, это не то, что требуется.

Если передать оператору dynamic\_cast нулевой указатель, то он просто вернет нулевой указатель нужного типа, поэтому проверять указатель перед передачей в dynamic\_cast не нужно.

**Код:**

struct ICloneable;

template <typename T>

struct ValueHolder;

class Any{

public:

Any():ptr(0) {}

~Any() {

delete ptr;

}

Any(const Any & other):ptr(other.ptr ? other.ptr->clone() : 0) {}

template<typename T>Any(const T val):ptr(new ValueHolder<T>(val)) {}

Any & operator=(const Any & other) {

if(this != &other) {

delete ptr;

ptr = other.ptr ? other.ptr->clone() : 0;

}

return \*this;

}

template<typename T>Any & operator=(const T & val) {

delete ptr;

ptr = new ValueHolder<T>(val);

return \*this;

}

template<typename T>T \* cast() {

if(ptr == 0)

return 0;

else {

ValueHolder<T>\* pvh = dynamic\_cast<ValueHolder<T>\*>(ptr);

if(pvh == 0)

return 0;

else

return &(pvh->data\_);

}

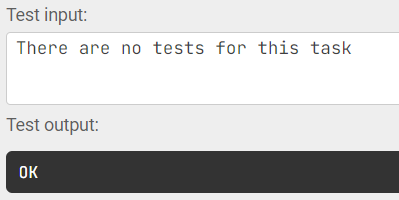
}

private:

ICloneable \* ptr;

};

**Результат:**



**Раздел 6.3 Специализация шаблонов**

**Задача 1.**

В качестве упражнения на частичную специализацию шаблонов классов вам предлагается реализовать простой шаблон SameType. Этот шаблон не содержит никаких методов, а только одно статическое константное поле типа bool, с именем value. Шаблон принимает два типовых параметра, и если два типовых параметра шаблона являются одним и тем же типом, то статическое поле value должно хранить значение true, в противном случае значение false.

**Код:**

#include <type\_traits>

template <class A, class B>

struct SameType

{

static const bool value = std::is\_same<A, B>::value;

};

**Результат:**

