Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

**Санкт-Петербургский национальный исследовательский университет**

**информационных технологий, механики и оптики**

Факультет Фотоники и оптоинформатики

Кафедра Компьютерной фотоники и видеоинформатики

**Отчет по практике**

Выполнил: Потёмкин А.В.

Группа: v3310

Преподаватель: Кудрявцев А. С.

Санкт-Петербург, 2016

Оглавление

[Цель проведения практики: 3](#_Toc434505577)

[Задание №1. Знакомство с централизованной системой контроля версий Subversion (SVN) 4](#_Toc434505578)

[Задание №2. Знакомство с распределенной системой контроля версий MercurialHg. 8](#_Toc434505579)

[Приложение А. Слайды презентации с семинара по С++11 на тему: …………………….. 11](#_Toc434505586)

# Цель проведения практики:

Освоение навыков использования C++ и изучение приемов разработки программного обеспечения. Практика проходит в компьютерном классе и состоит из лекционных занятий и практических заданий.

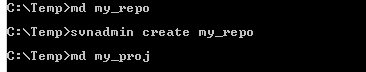
# Задание №1. Знакомство с централизованной системой контроля версий Subversion (SVN)

**Цель:** Знакомство с общими принципами работы в Subversion.

**Задачи:**

1. Научиться работать с svn в консоли.
2. Научиться работать с TortoiseSVN для Windows.

Создание репозитория и рабочей папки приведены на рис.1.1 и рис.1.2.

****

*Рис.1.1. Создание репозиториев.*

**

*Рис.1.2. Клонирование репозитория в рабочую папку.*

Добавление файла в систему контроля версий представлено на рис.1.3.

**

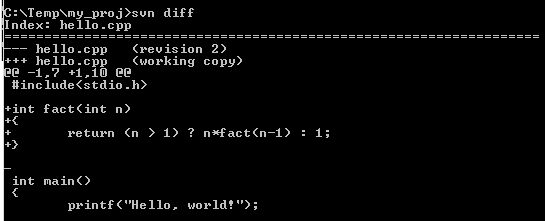
*Рис.1.3.Добавление файла в систему контроля версий.*

Внесение изменений в репозиторий и вывод svn находятся на рис.1.4.

**

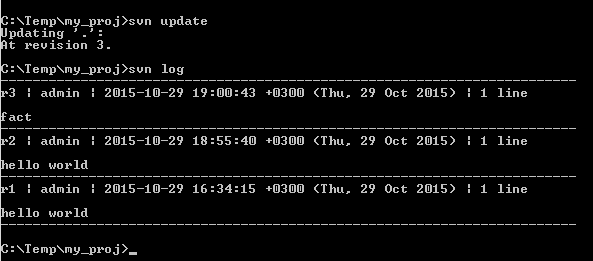
*Рис.1.4. Внесение изменений в репозиторий.*

Использование параметра diff. Этот параметр позволяет отследить изменения файлов относительно последней ревизии. Пример вывода svn diff приведено на рис.1.5.

**

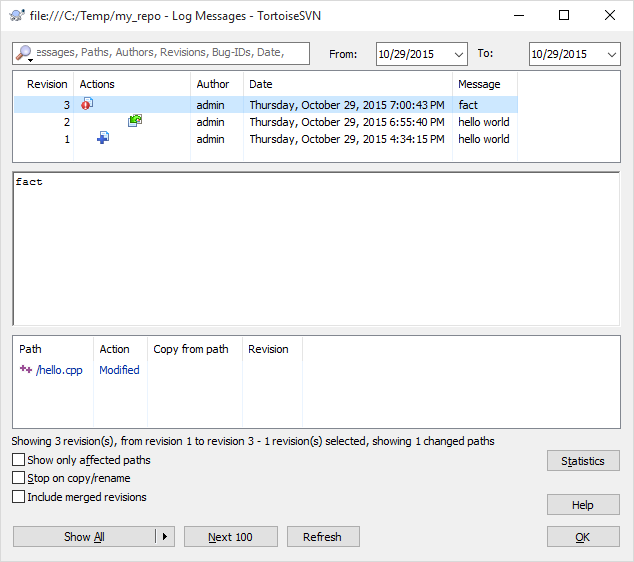
*Рис.1.5.Использование команды diff.*

Параметр log дает нам возможность показать изменения по ревизиям. После использования функции update (пример на рис.1.6)

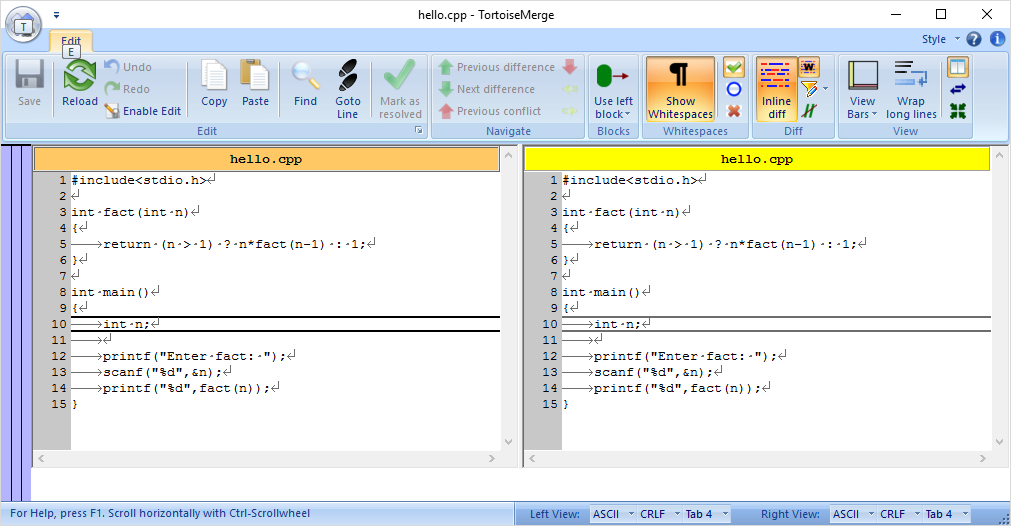
**

*Рис.1.6. Использование команды log.*

На рис.1.7, рис. 1.8 снимки экрана для TortoiseSVN.



*Рис.1.7. Функция log в TortoiseSVN.*



*Рис.1.8. Функция diff в TortoiseSVN.*

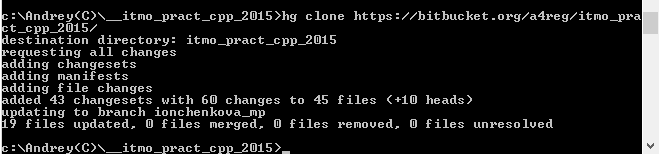
# Задание №2. Знакомство с распределенной системой контроля версий Mercurial.

**Цель:** Знакомство с общими принципами работы в Subversion.

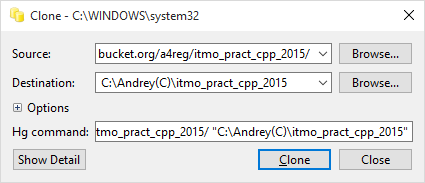
**Задачи:**

1. Познакомиться с общими принципами работы Mercurial.
2. Научиться работать с Mercurial в консоли (hg.exe).
3. Научиться работать с TortoiseHG для Windows.

Клонирование репозитория в консоли представлено на рис. 2.1а). В TortoiseHg на рис. 2.1.б).



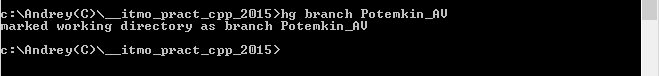
а)



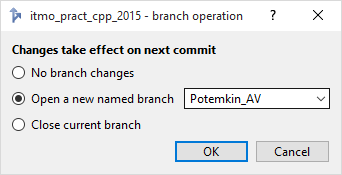
б)

*Рис.2.1. Клонирование репозитория.*

Для создания отдельной ветки использовалась команда branch, пример использования на рис.2.2а) в консоли, рис.2.2б) в TortoiseHg.

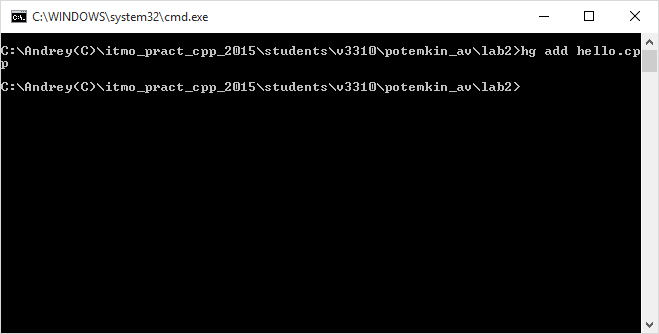


а)



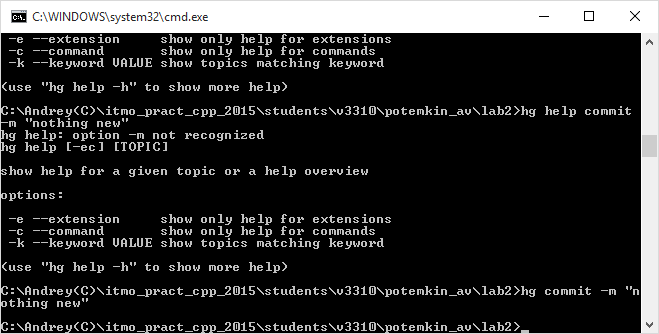
б)

Рис.2.2. Пример создания ветки.

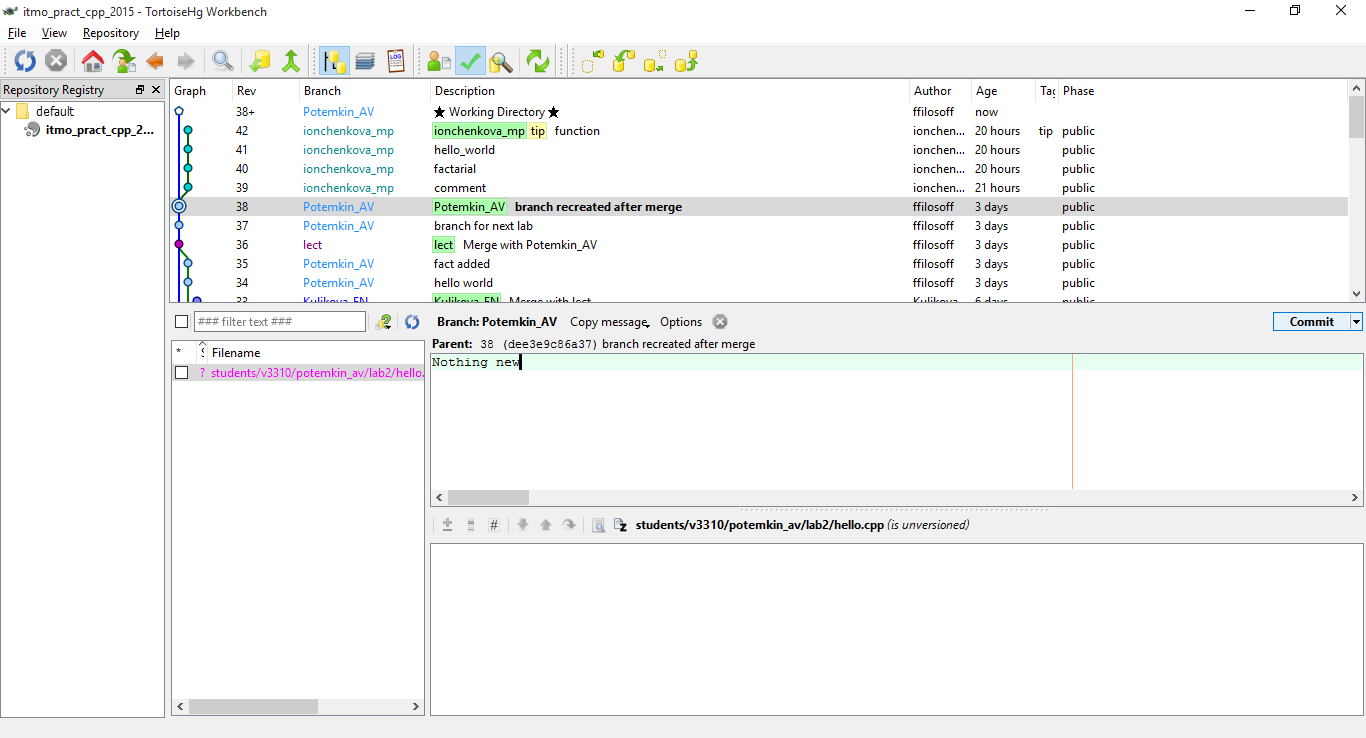
Добавление файлов осуществляется командой add в консоли. На рис.2.3 приведен пример использования консоли. В TortoiseHg добавление файлов происходит во время внесения изменений рис.2.4б).**

*Рис.2.3. Добавление файла в систему контроля версий.*

Внесение изменений в репозиторий при помощи команды commit рис.2.4а). В TortoiseHg – рис.2.4б).

**

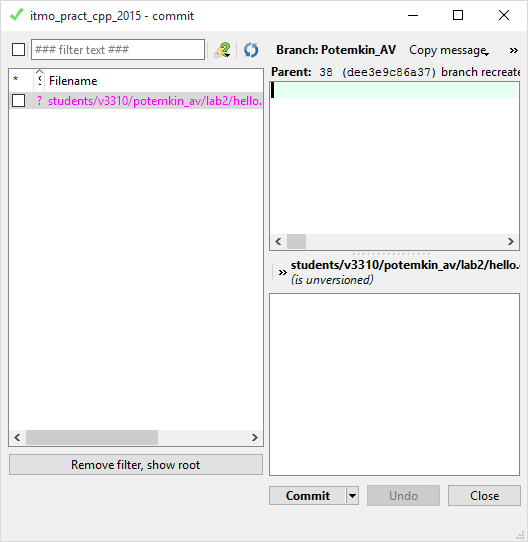
*а)*



*б)*

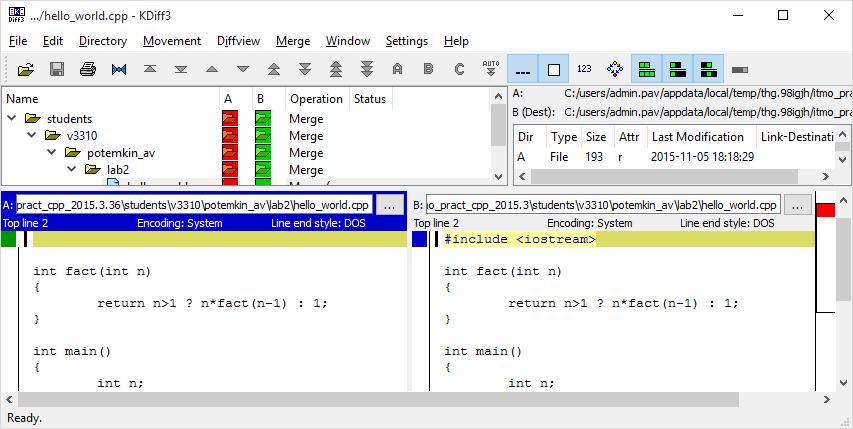
*Рис.2.4. Внесение изменениий в репозиторий.*

Окно внесения изменений, вызванное из проводника снятно на рис.2.5.

**

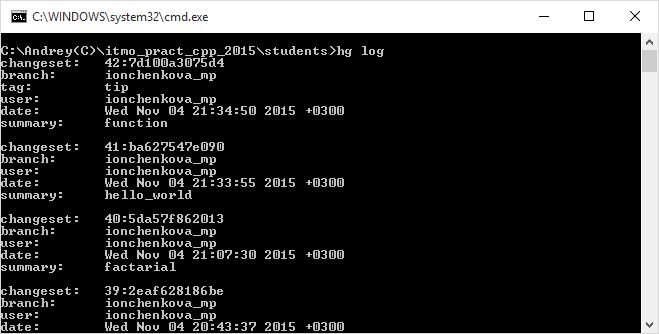
*Рис.2.5. Внесение изменений из проводника*

Окно изменений diff показана на рис.2.6.



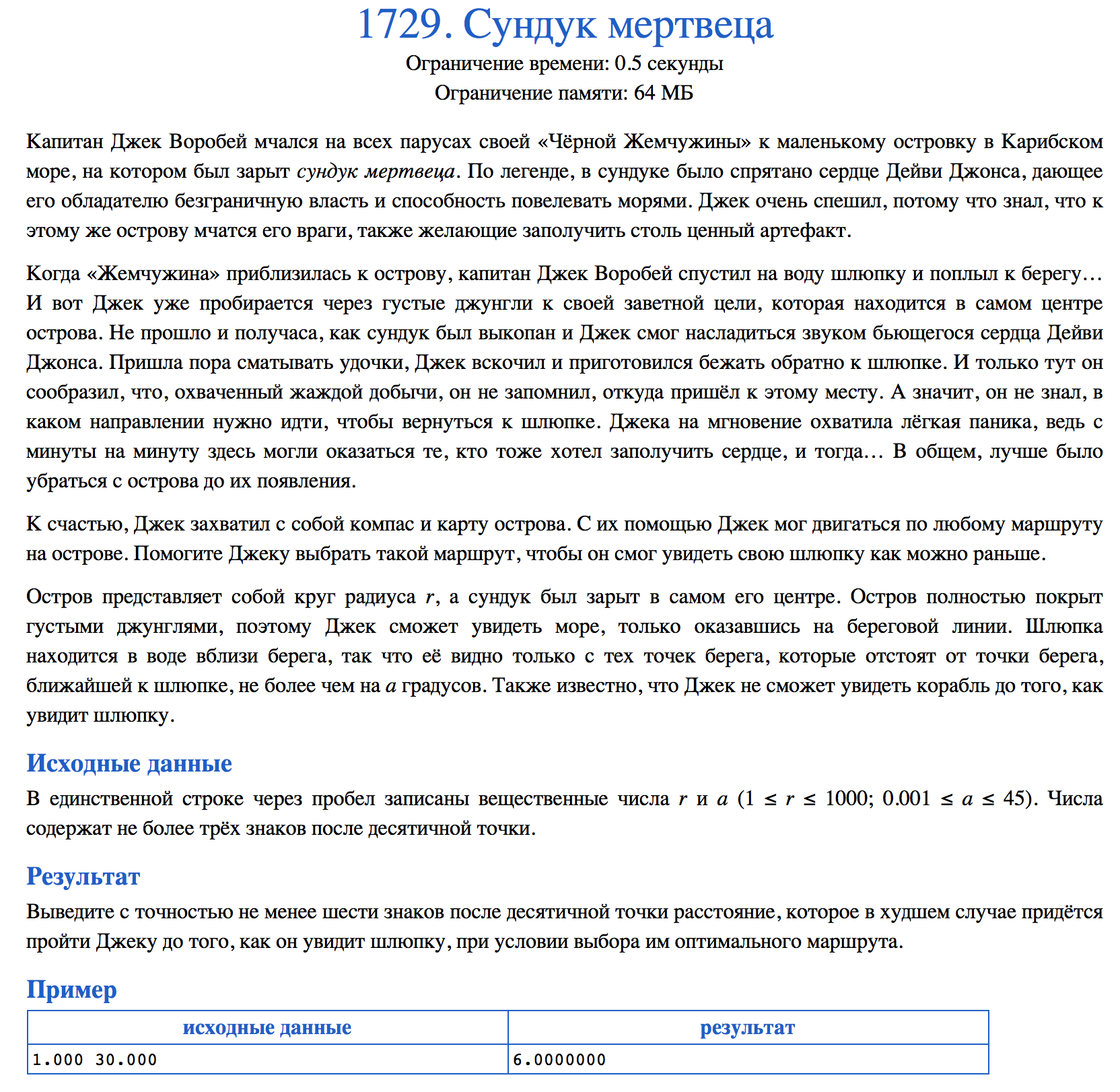
*Рис.2.6. Команда diff в TortoiseHg.*

Изменения команды log в консоли приведены на рис.2.7.

**

*Рис.2.7. Команда изменений log.*

# Задание №3. (1729)

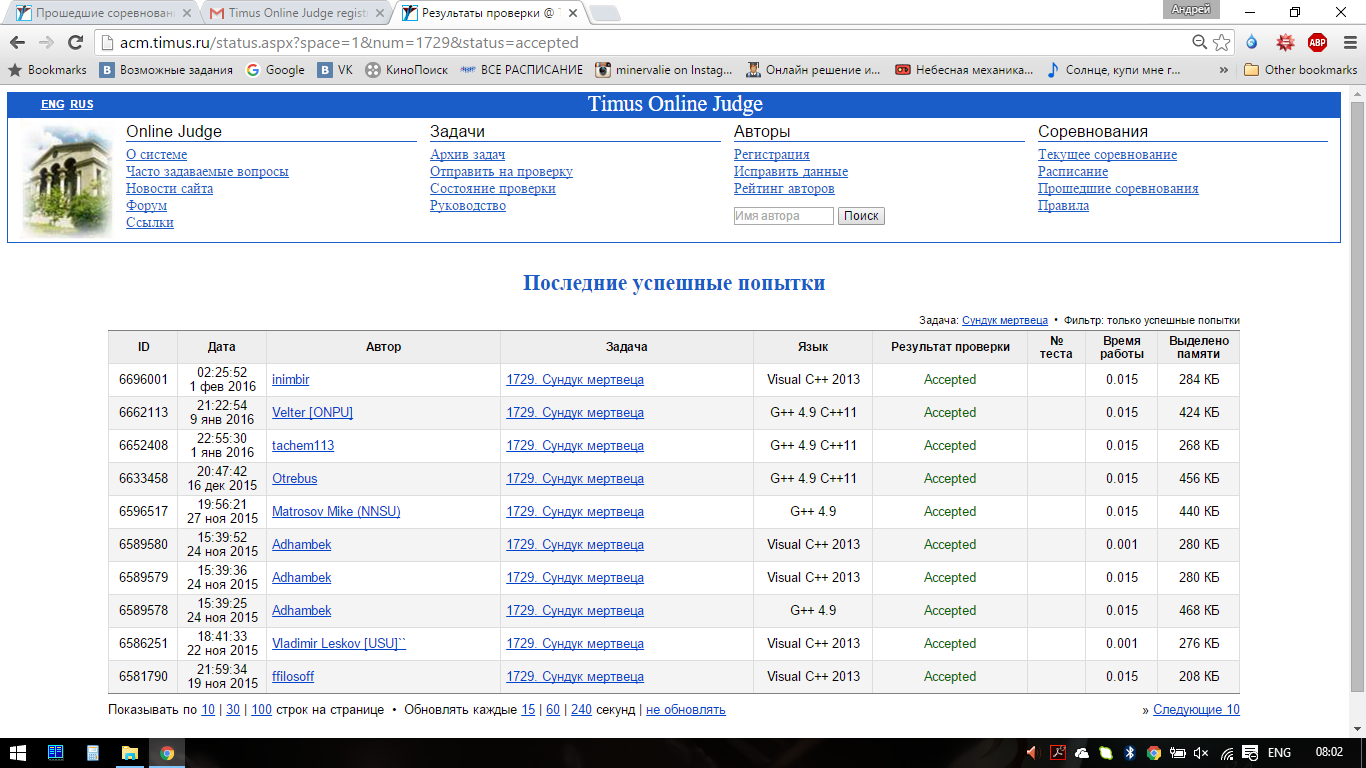


*Рис.3.1. Задача 1729.*

**Алгоритм:**

Считается угол слева. Далее рассчитывается количество полных углов. Считается расстояние. Считаются остаточные углы. Все суммируется, включая радиус острава.

**Скриншот с acm.timus.ru:**

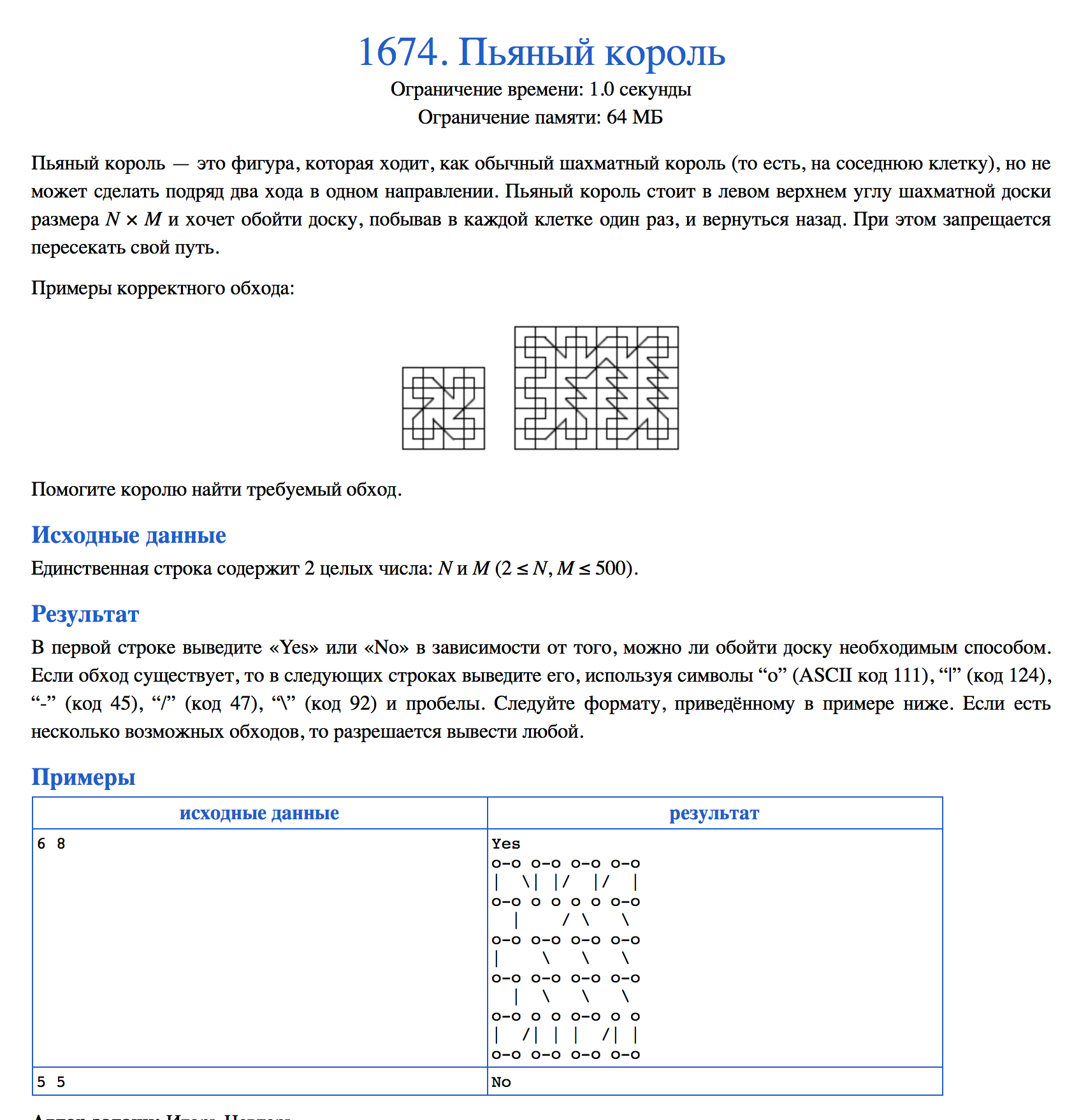


*Рис.3.2. Скриншот с подтверждением решения задачи*

**Код:**

#include **<iostream>**#include **<math.h>  
  
double** ToRad(**double** aDeg) {  
 **return** aDeg \* 3.14159265359 / 360.;  
}  
  
**double** Range(**double** aR, **double** aAngle) {  
 **return** 2 \* aR \* sin(aAngle);  
}  
  
**int** main() {  
 **double** r, a;  
 scanf(**"%lf%lf"**, &r, &a);  
*// first, walk to the edge of the island* **double** way = r;  
*// total angle of undiscovered edge* **double** angleLeft = 360.0 - 2.0 \* a;  
*// hops count on 2\*a "arc" line* **int** fullHops = (**int**) (angleLeft / (2.0 \* a));  
 way += Range(r, ToRad(2.0 \* a)) \* fullHops;  
*// remaining angle to hop* **double** angleRemain = angleLeft - fullHops \* (2.0 \* a);  
 way += Range(r, ToRad(angleRemain));  
 printf(**"%.6lf"**, way);  
}

# Задание №4. (1674)

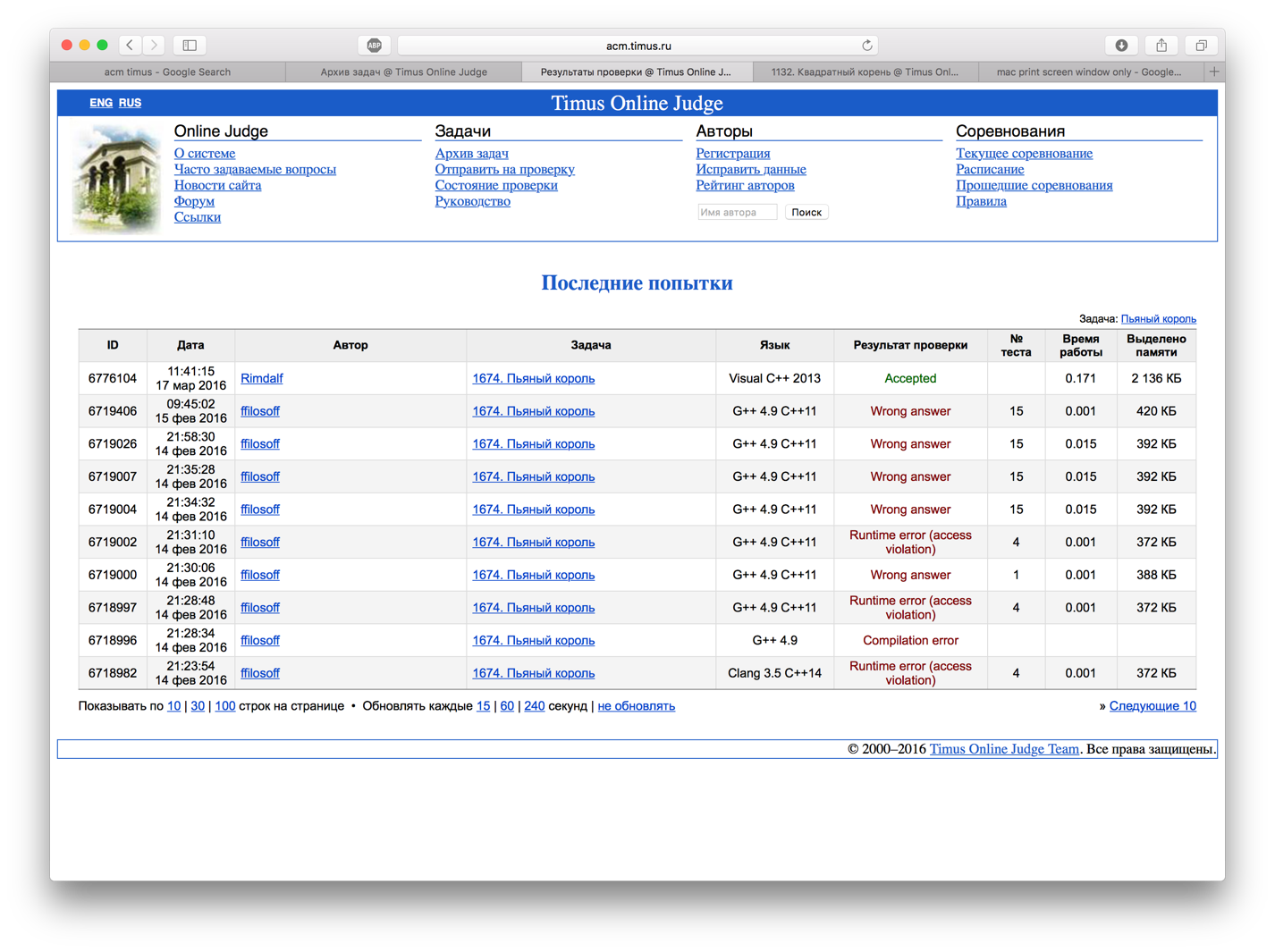


*Рис.4.1. Задача 1674.*

**Алгоритм:**

Неполное решение заключается в том, что мы формируем рисунок с приоритетом переходов в определенные стороны в зависимости от направления. При направлении вниз будет один набор приоритетов операций. При движении вниз – другой. Также будут отличны операции движения по верху.

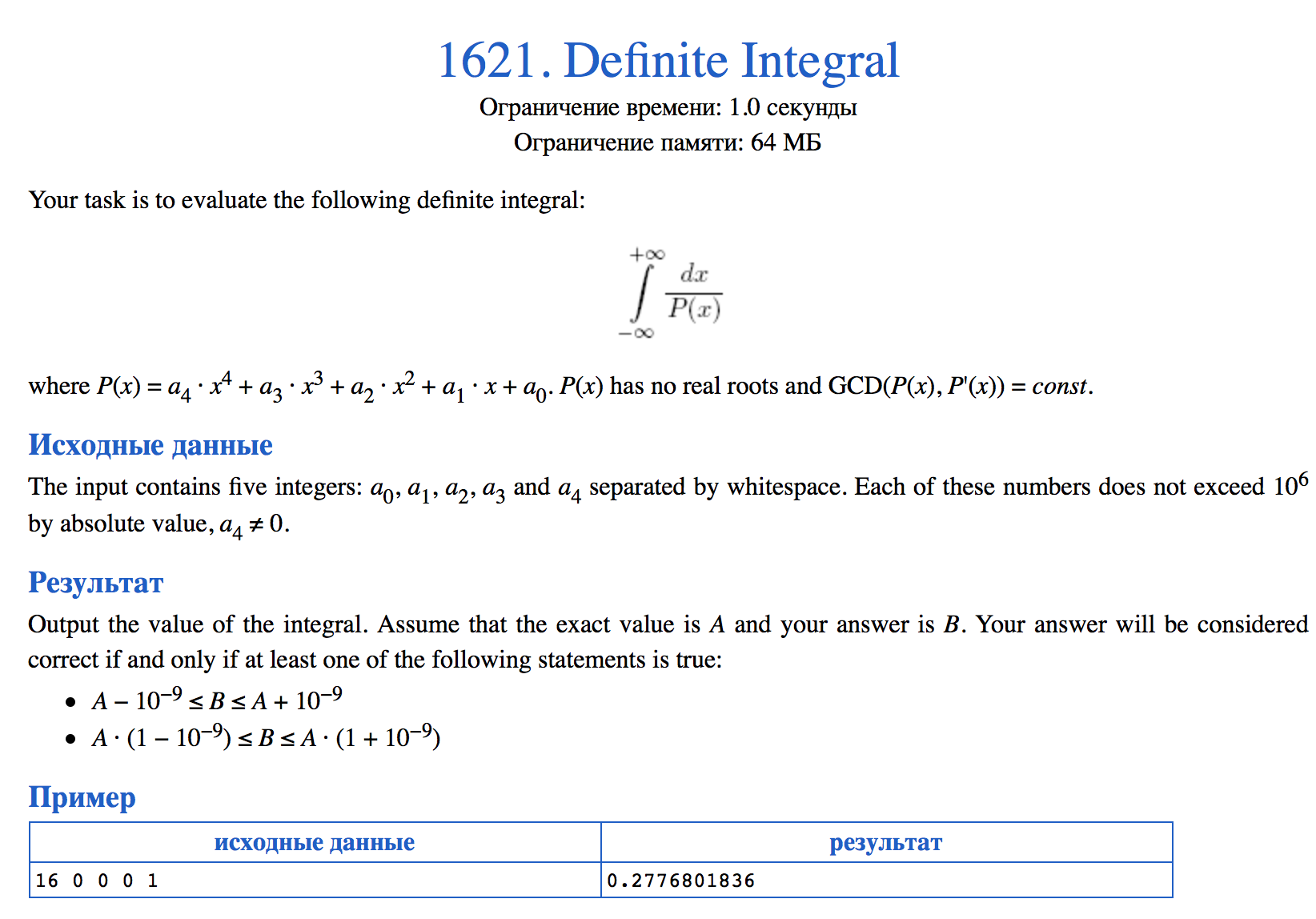
**Скриншот с acm.timus.ru:**

 *Рис.4.2. Скриншот с подтверждением неполного решения задачи*

**Код:**

*//  
// Created by Potemkin Andrey Vladimirovich on 14.02.16.  
//*#include **<vector>**#include **<iostream>**#include **<string>  
  
using namespace** std;  
  
**class** field {  
 **int** m\_;  
 **int** n\_;  
 **bool** reversed\_;  
 **bool** solution\_;  
 vector<vector<**char**>> matrix\_;*//field contains hubs and connections* **enum** direction {  
 ***down***,  
 ***up***,  
 };  
  
 **char** GetHub(**int** aY, **int** aX) {  
 **if**(aY >= 0 && aX >= 0 && aY < n\_ && aX < m\_) {  
 **return** matrix\_[2 \* aY][2 \* aX];  
 }  
 **return** 0;  
 }  
  
 **void** SetHub(**int** aY, **int** aX, **int** aOldY = -1, **int** aOldX = -1) {  
 **if**(aY >= 0 && aX >= 0 && aY < n\_ && aX < m\_) {  
 matrix\_[2 \* aY][2 \* aX] = **'o'**; *// here set hub  
 //set connection, will bug if the start and end point are the same* **if**(aOldX >= 0 && aOldX < m\_ && aOldY >= 0 && aOldY < n\_) {  
 **int** dx = aY - aOldY;  
 **int** dy = aX - aOldX;  
 **char** dir = **'-'**;  
 **if** ((dx > 0 && dy > 0) || (dx < 0 && dy < 0)){  
 dir = **'\\'**;  
 }  
 **if** ((dx < 0 && dy > 0) || (dx > 0 && dy < 0)){  
 dir = **'/'**;  
 }  
 **if** (dx && !dy) {  
 dir = **'|'**;  
 }  
 **if** (dy && !dx) {  
 dir = **'-'**;  
 }  
 matrix\_[ dx + aOldY \* 2 ][dy + aOldX \* 2 ] = dir;  
 }  
 }  
 **return**;  
 }  
**public**:  
 field(**int** aM, **int** aN){  
 m\_ = aM;  
 n\_ = aN;  
 solution\_ = **false**;  
 reversed\_ = **false**;  
*// Solve();* }  
  
 **bool** Solve(){  
 *// when field is 2-by-2* **if**(m\_ == 2 && n\_ == 2){  
 matrix\_ = vector<vector<**char**>> (2 \* n\_ - 1,vector<**char**> (2 \* m\_ - 1,**' '**));  
 SetHub(0, 0);  
 SetHub(0, 1, 0, 0);  
 SetHub(1, 1, 0, 1);  
 SetHub(1, 0, 1, 1);  
 SetHub(0, 0, 1, 0);  
 solution\_ = **true**;  
 **return true**;  
 }  
 *// field less or equal than 3 => no solution* **if**((m\_ % 2 != 0 && n\_ % 2 != 0) || (m\_ <= 3 || n\_ <= 3) ) {  
 solution\_ = **false**;  
 **return false**;  
 }  
 **if**(m\_ % 4 > 1 && n\_ % 4 > 1) {  
 solution\_ = **false**;  
 **return false**;  
 }  
 **int** parity = 0; *// pairity for 3 instead 2 by 2 subfields* **if**( m\_ % 4 == 0){  
 matrix\_ = vector<vector<**char**>> (2 \* n\_ - 1,vector<**char**> (2 \* m\_ - 1,**' '**));  
 parity = 1;  
 } **else** {  
 *// find solution with rotated matrix, anyway doesn't matter how to print* **if** (m\_ % 2 != 0) {  
 matrix\_ = vector<vector<**char**>> (2 \* n\_ - 1,vector<**char**> (2 \* m\_ - 1,**' '**));  
 } **else** {  
 swap(n\_, m\_);  
 reversed\_ = **true**;  
 matrix\_ = vector<vector<**char**>>(2 \* n\_ - 1, vector<**char**>(2 \* m\_ - 1, **' '**));  
 }  
 }  
 *// start positions for this algo only 1 and 0...* **int** start\_y = 1;  
 **int** start\_x = 0;  
 SetHub(start\_y, start\_x);  
 **int** y = 2;  
 **int** x = 1;  
 SetHub(y, x, start\_y, start\_x);  
 direction current\_directon = ***up***;  
 **char** last\_directon = **'|'**;  
 **int** pos = -1;  
 **int** count = (n\_ - 2) \* 2;  
 **int** right = 0,left = 0,top = 1, bottom = n\_;  
 *//main part of field (except top 2 by n\_)* **while** (pos < m\_ / 2 + 1) {  
 **if**(count >= (n\_ - (top + 1)) \* (right - left)) {  
 pos ++;  
 right = 2 \* (pos + 1) + !parity;  
 left = 2 \* pos + !parity;  
 **if**(!parity && pos == 0){  
 right = 3;  
 left = 0;  
 }  
 top = 1;  
 bottom = n\_;  
 count = 0;  
 **if**(right > m\_) {  
 **break**;  
 }  
 **if**(current\_directon == ***up***) {  
 current\_directon = ***down***;  
 } **else** {  
 current\_directon = ***up***;  
 }  
 }  
 count ++;  
 **int** old\_y = y;  
 **int** old\_x = x;  
 *// priority of directons /\|- are very important* **if** (current\_directon == ***up***) {  
 **if** (y + 1 < bottom && last\_directon != **'|'** && GetHub(y + 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y++;  
 } **else if** (y + 1 < bottom && x - 1 >= left && last\_directon != **'/'** && GetHub(y + 1, x - 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y++;  
 x--;  
 } **else if** (x + 1 < right && y + 1 < bottom && last\_directon != **'\\'** && GetHub(y + 1, x + 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y++;  
 x++;  
 } **else if** (x - 1 >= left && last\_directon != **'-'** && GetHub(y, x - 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x--;  
 } **else if** (x + 1 < right && last\_directon != **'-'** && GetHub(y, x + 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x++;  
 } **else if** (x - 1 >= left && last\_directon != **'\\'** && y - 1 > top && GetHub(y - 1, x - 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y--;  
 x--;  
 } **else if** (y - 1 > top && last\_directon != **'|'** && GetHub(y - 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y--;  
 } **else if** (x + 1 < right && last\_directon != **'/'** && y - 1 > top && GetHub(y - 1, x + 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y--;  
 x++;  
 }  
 } **else if** (current\_directon == ***down***) {  
 **if** (y - 1 > top && last\_directon != **'|'** && GetHub(y - 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y--;  
 } **else if** (y - 1 > top && x - 1 >= left && last\_directon != **'\\'** && GetHub(y - 1, x - 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y--;  
 x--;  
 } **else if** (x + 1 < right && y - 1 > top && last\_directon != **'/'** && GetHub(y - 1, x + 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y--;  
 x++;  
 } **else if** (x - 1 >= left && last\_directon != **'-'** && GetHub(y, x - 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x--;  
 } **else if** (x + 1 < right && last\_directon != **'-'** && GetHub(y, x + 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x++;  
 } **else if** (x - 1 >= left && last\_directon != **'/'** && y + 1 < bottom && GetHub(y + 1, x - 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y++;  
 x--;  
 } **else if** (y + 1 < bottom && last\_directon != **'|'** && GetHub(y + 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y++;  
 } **else if** (x + 1 < right && last\_directon != **'\\'** && y + 1 < bottom && GetHub(y + 1, x + 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y++;  
 x++;  
 }  
 }  
 *// setting up hub on field* **if**(!(y==old\_y && x==old\_x) ) {  
 SetHub(y, x, old\_y, old\_x);  
 }  
 }  
 *//handle transition to top part* right = m\_;  
 left = 0;  
 top = -1;  
 bottom = 2;  
 *//* **int** old\_x = x,  
 old\_y = y;  
 **if** (y - 1 > top && x - 1 >= left && last\_directon != **'\\'** && GetHub(y - 1, x - 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y--;  
 x--;  
 } **else if** (x + 1 < right && y - 1 > top && last\_directon != **'/'** && GetHub(y - 1, x + 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y--;  
 x++;  
 } **else if** (x - 1 >= left && last\_directon != **'-'** && GetHub(y, x - 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x--;  
 } **else if** (x + 1 < right && last\_directon != **'-'** && GetHub(y, x + 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x++;  
 } **else if** (x - 1 >= left && last\_directon != **'/'** && y + 1 < bottom && GetHub(y + 1, x - 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y++;  
 x--;  
 } **else if** (y + 1 < bottom && last\_directon != **'|'** && GetHub(y + 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y++;  
 } **else if** (y - 1 > top && last\_directon != **'|'** && GetHub(y - 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y--;  
 } **else if** (x + 1 < right && last\_directon != **'\\'** && y + 1 < bottom && GetHub(y + 1, x + 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y++;  
 x++;  
 }  
 **if**(!(y==old\_y && x==old\_x) ) {  
 SetHub(y, x, old\_y, old\_x);  
 }  
 count = 0;  
 *// handle top* **while**(x > 0 && count < 2 \* m\_) {  
 **int** old\_x = x;  
 **int** old\_y = y;  
 **if** (x + 1 < right && last\_directon != **'-'** && GetHub(y, x + 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x++;  
 } **else if** (x + 1 < right && last\_directon != **'\\'** && y + 1 < bottom && GetHub(y + 1, x + 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y++;  
 x++;  
 } **else if** (x + 1 < right && y - 1 > top && last\_directon != **'/'** && GetHub(y - 1, x + 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y--;  
 x++;  
 } **else if** (y - 1 > top && last\_directon != **'|'** && GetHub(y - 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y--;  
 } **else if** (y + 1 < bottom && last\_directon != **'|'** && GetHub(y + 1, x) != **'o'**) {  
 last\_directon = **'|'**;  
 y++;  
 } **else if** (y - 1 > top && x - 1 >= left && last\_directon != **'\\'** && GetHub(y - 1, x - 1) != **'o'**) {  
 last\_directon = **'\\'**;  
 y--;  
 x--;  
 } **else if** (x - 1 >= left && last\_directon != **'/'** && y + 1 < bottom && GetHub(y + 1, x - 1) != **'o'**) {  
 last\_directon = **'/'**;  
 y++;  
 x--;  
 } **else if** (x - 1 >= left && last\_directon != **'-'** && GetHub(y, x - 1) != **'o'**) {  
 last\_directon = **'-'**;  
 x--;  
 }  
 **if**(!(y==old\_y && x==old\_x) ) {  
 SetHub(y, x, old\_y, old\_x);  
 }  
 count++;  
 }  
 SetHub(start\_y,start\_x,y,x);  
 solution\_ = **true**;  
 **return true**;  
 }  
  
 **void** print() {  
 **if**(solution\_) {  
 cout << **"Yes"** << endl;  
 **if**(!reversed\_) {  
 **for** (**int** i = 0; i < matrix\_.size(); i++) {  
 **for** (**int** j = 0; j < matrix\_[i].size(); j++) {  
 cout << matrix\_[i][j];  
 }  
 cout << endl;  
 }  
 } **else** {  
 **for** (**int** i = 0; i < matrix\_[0].size(); i++) {  
 **for** (**int** j = 0; j < matrix\_.size(); j++) {  
 **if**(matrix\_[j][i] == **'-'**) {  
 cout << **'|'**;  
 } **else if**(matrix\_[j][i] == **'|'**) {  
 cout << **'-'**;  
 } **else if**(matrix\_[j][i] == **'/'**) {  
 cout << **'/'**;  
 } **else if**(matrix\_[j][i] == **'\\'**) {  
 cout << **'\\'**;  
 } **else** {  
 cout << matrix\_[j][i];  
 }  
 }  
 cout << endl;  
 }  
 }  
 } **else** {  
 cout << **"No"**;  
 }  
 }  
};  
  
**int** main() {  
 **int** N,M;  
 cin >> N >> M;  
 field sol(M,N);  
 sol.Solve();  
 sol.print();  
 **return** 0;  
}

# Задание №5. (1621)



*Рис.5.1. Задача 1621.*

**Алгоритм:**

Данный интеграл можно представить как (5.1).

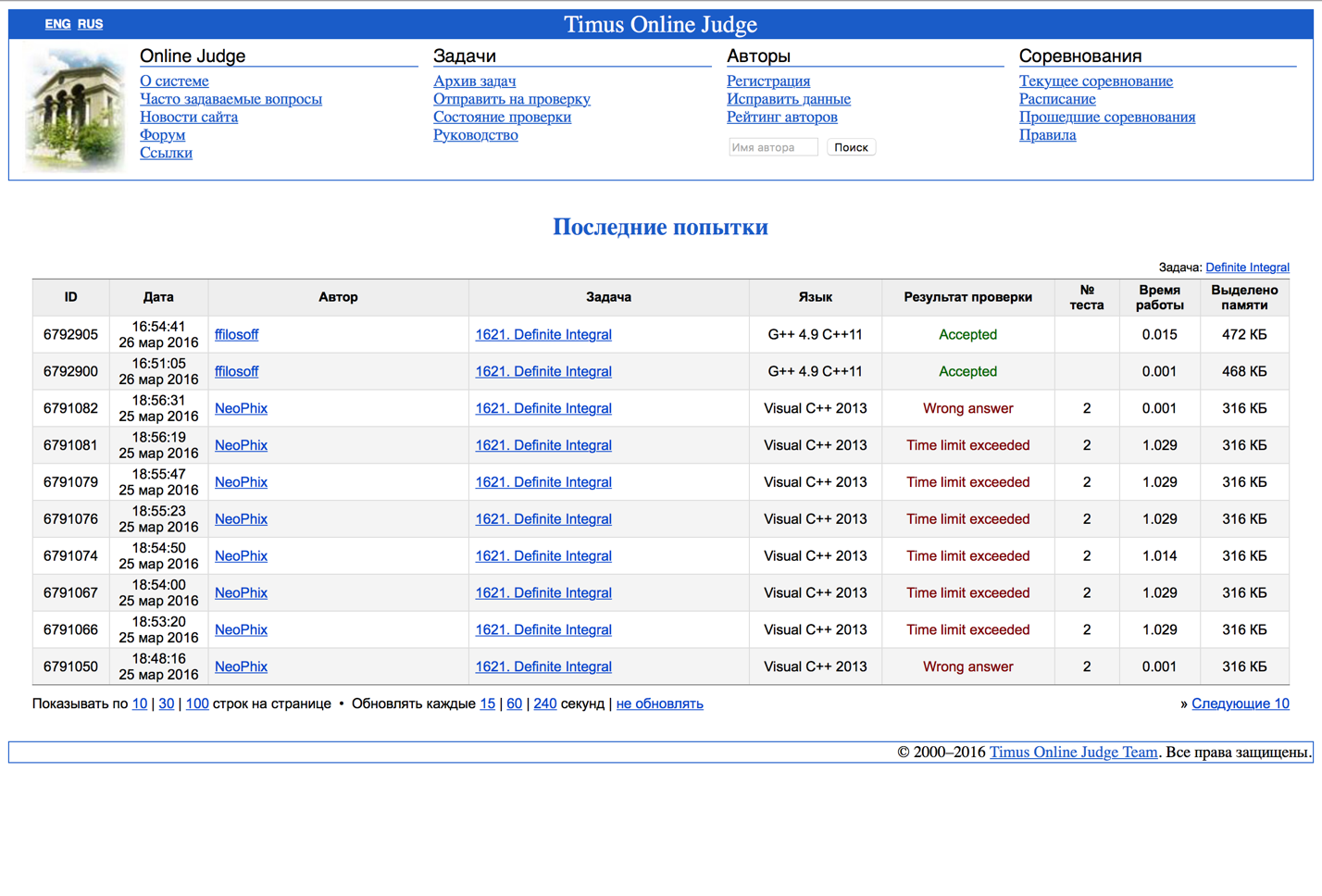
, (5.1)

где – корни полинома .

Тогда решением интеграла будет (5.2):

. (5.2)

**Скриншот с acm.timus.ru:**



*Рис.5.2. Скриншот с подтверждением решения задачи*

**Код:**

#include **<iostream>**#include **<cmath>**#include **<iomanip>**#include **<complex>**#include **<vector>**#include **<functional>  
  
using namespace** std;  
  
**typedef** complex<**long double**> ComplexNumber;  
  
*//Ferrari formula for fining roots*vector<ComplexNumber> GetRoots( **long double** a, **long double** b, **long double** c, **long double** d, **long double** e) {  
 vector<ComplexNumber> x(4);  
 ComplexNumber alfa = -3.L \* pow(b, 2.L) / (8.L \* pow(a, 2.L)) + c / a;  
 ComplexNumber betta = pow(b, 3.L) / (8.L \* pow(a, 3.L)) - b \* c / (2.L \* pow(a, 2.L)) + d / a;  
 ComplexNumber gamma = -3.L \* pow(b, 4.L) / (256.L \* pow(a, 4.L)) +  
 pow(b, 2.L) \* c / (16.L \* pow(a, 3.L)) - b \* d / (4.L \* pow(a, 2.L)) + e / a;  
 **if**(abs(betta) < pow(10.L,-9.L)){  
 x[0] = -b/(4.L\*a)+sqrt((-alfa+sqrt(pow(alfa,2.L)-4.L\*gamma))/2.L);  
 x[1] = -b/(4.L\*a)+sqrt((-alfa-sqrt(pow(alfa,2.L)-4.L\*gamma))/2.L);  
 x[2] = -b/(4.L\*a)-sqrt((-alfa+sqrt(pow(alfa,2.L)-4.L\*gamma))/2.L);  
 x[3] = -b/(4.L\*a)-sqrt((-alfa-sqrt(pow(alfa,2.L)-4.L\*gamma))/2.L);  
 } **else** {  
 ComplexNumber P = -pow(alfa, 2.L) / 12.L - gamma;  
 ComplexNumber Q = -pow(alfa, 3.L) / 108.L + alfa \* gamma / 3.L - pow(betta, 2.L) / 8.L;  
 ComplexNumber R = -Q / 2.L + pow(pow(Q, 2.L) / 4.L + pow(P, 3.L) / 27.L, 0.5L);  
 ComplexNumber U = pow(R, 1.L / 3.L);  
 ComplexNumber y = -5.L \* alfa / 6.L + U;  
 **if** (abs(U) < pow(10.L, -9.L)) {  
 y += pow(Q, 1.L / 3.L);  
 } **else** {  
 y += -P / (3.L \* U);  
 }  
 ComplexNumber W = pow(alfa + 2.L \* y, 0.5L);  
  
 x[0] = (-b / (4.L \* a) + (W - pow(-(3.L \* alfa + 2.L \* y + 2.L \* betta / W), 1.L / 2.L)) / 2.L);  
 x[1] = (-b / (4.L \* a) + (W + pow(-(3.L \* alfa + 2.L \* y + 2.L \* betta / W), 1.L / 2.L)) / 2.L);  
 x[2] = (-b / (4.L \* a) + (-W - pow(-(3.L \* alfa + 2.L \* y - 2.L \* betta / W), 1.L / 2.L)) / 2.L);  
 x[3] = (-b / (4.L \* a) + (-W + pow(-(3.L \* alfa + 2.L \* y - 2.L \* betta / W), 1.L / 2.L)) / 2.L);  
 }  
 **return** x;  
}  
  
  
**long double** SolveIntegral( **long double** a4, **long double** a3, **long double** a2, **long double** a1, **long double** a0) {  
 ComplexNumber ans = 0.L;  
 vector<ComplexNumber> omega = GetRoots(a4,a3,a2,a1,a0);  
  
 ComplexNumber i(0.L,1.L);  
 ComplexNumber a = omega[0];  
 ComplexNumber b = omega[1];  
 ComplexNumber c = omega[2];  
 ComplexNumber d = omega[3];  
 *// analitic formula for integral* ans = 3.14159265358979323846264338327950288419716939937510582097L \* i \*(  
 + (a.imag() >= 0.0L ? 1.L : -1.L) /((a - b) \* (a - c) \* (a - d))  
 + (b.imag() >= 0.0L ? 1.L : -1.L) /((b - a) \* (b - c) \* (b - d))  
 + (c.imag() >= 0.0L ? 1.L : -1.L) /((c - a) \* (c - b) \* (c - d))  
 + (d.imag() >= 0.0L ? 1.L : -1.L) /((d - a) \* (d - b) \* (d - c))  
 );  
 ans = abs(ans)/a4;  
 **return** ans.real();  
}  
  
**void** PrintTest(**bool** expr) {  
 **static long long** test\_num = 1;  
 cout << **"Test "** << test\_num << **": "**;  
 **if**(expr) {  
 cout << **"PASSED"**;  
 } **else** {  
 cout << **"Fail"**;  
 }  
 cout << endl;  
 test\_num++;  
};  
  
**void** test() {  
 **long double** accuracy = pow(10,-9);  
 PrintTest(abs(SolveIntegral(1, 0, 0, 0, 16) - 0.2776801836L) < accuracy);  
 PrintTest(abs(SolveIntegral(1, 0, 1000000, 100099, 999999) - 3.1423994906657376L/1000000.L) < accuracy);  
 PrintTest(abs(SolveIntegral(1, 0, 0, 0, 999999) - 0.00007024819999656384712594L) < accuracy);  
 PrintTest(abs(SolveIntegral(-1, 0, 0, 0, -999999) - (-0.00007024819999656384712594L)) < accuracy);  
 PrintTest(abs(SolveIntegral(1, 0, 1600000, 0, 1) - (0.002483645514171064038404671L)) < accuracy);  
  
  
 *//*}  
  
**int** main() {  
 **long double** a4 = 1, a3 = 0, a2 = 1000000, a1 = 100099, a0 = 999999;  
 test();  
 cin >> a0 >> a1 >> a2 >> a3 >> a4;  
 **long double** ans = SolveIntegral(a4, a3, a2, a1, a0) ;  
 cout << setprecision(15) << ans;  
 **return** 0;  
}

## Для 3й и последующих л/р:

1. Условия задания.

2. Схема алгоритма решения задачи (словесное описание пути решения и/или математическая модель задачи).

3. Текст программы.

4. Исходные данные, используемые при вводе в программу.

5. Результаты выполнения программы.

## Дополнительные указания

- Условие задания с сайта timus приводить полностью (включая номер и название задания, описание, исходные данные, результат и пример), лучше помещать задание в отчет как рисунок;

- Помещать в отчет скриншот сайта timus с результатами проверки, включающий логин автора, номер задания и результат выполнения;

- Предоставлять для проверки отчет, сохраненный в формате pdf (т.к. в разных версиях MS Word документ может отображаться по-разному, внося искажения в оформление отчета, что снижает оценку за оформление).

-Бумажная версия отчета предоставляется в конце семестра.

## Оформление отчета по ГОСТ 7.32-200

Основные требования и рекомендации (составлено по наиболее частым ошибкам):

- Отчет должен иметь титульный лист, оглавление и содержать отчеты по лабораторным работам, а также слайды презентации.

- Страницы должны быть пронумерованы.

- Для заголовков, подзаголовков, абзацев основного текста, списков и подписей рисунков/таблиц рекомендуется создать и использовать стили. Это позволит обеспечить единообразное оформление и возможность быстрой корректировки оформления.

- Рисунки и таблицы должны иметь подписи. Не допускается расположение рисунка/таблицы и его подписи на разных страницах.

# Задание №2 ……

# Приложение А. Слайды презентации с семинара по С++11 на тему: ……………………..