

1. Код

```
#define _CRT_SECURE_NO_WARNINGS //removes visual studio warning about scanf unsafe usage

#include <iostream>
#include "stdio.h"
#include <conio.h>

using namespace std;

double power(double x, int n) {
    //calculates x^n
    double result = 1.0;
    for (int i = 0; i < n; i++) {
        result *= x;
    }
    return result;
}

double nthSum(double x, int n) {
    //calculates n-th sum
    double sum = 0.0;
    for (int k = 1; k <= n; k++) {
        sum += power(x, k + 1) / (k * k * (k + 1) * (k + 1));
    }
    return sum;
}

void print(double x, int n) {
    //this function calls nthSum to calculate sums and then prints them
    printf("n sum\n");
    for (int i = 1; i <= n; i++) {
        printf("%d %lf\n", i, nthSum(x, i));
    }
}

int main() {
    double x;
    int n;
    printf("enter real number x and positive integer n.\nthe program will calculate n\npartial sums\n");
    scanf("%lf%d", &x, &n);
    print(x, n);
    return 0;
}
```

2.

Проверим работу программы посчитав отдельно первые три частичные суммы (я считал в вольфраме) и сравнив результаты.

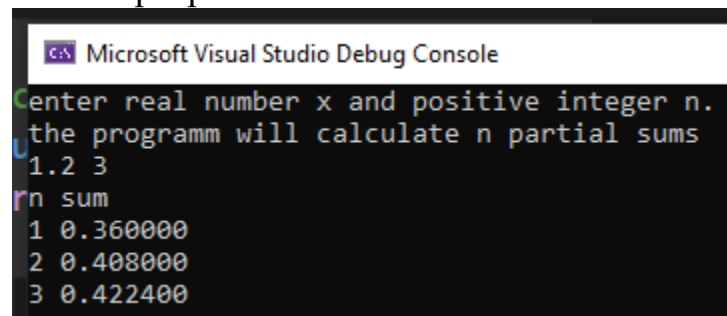
Возьмем $x=1.2$ и $n=3$

$$n=1: (1.2^2)/(1*1*2*2)=0.36$$

$$n=2: (1.2^2)/(1*1*2*2) + (1.2^3)/(2*2*3*3)=0.408$$

$$n=3: (1.2^2)/(1*1*2*2) + (1.2^3)/(2*2*3*3) + (1.2^4)/(3*3*4*4)=0.4224$$

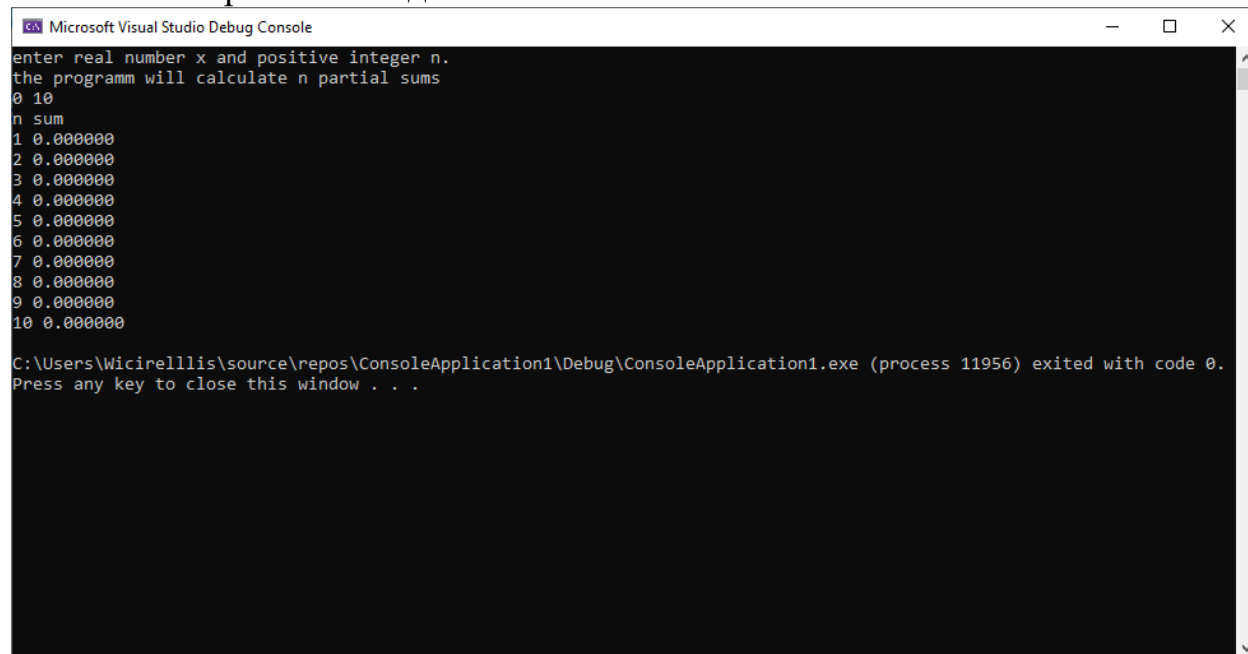
Ответ программы

A screenshot of the Microsoft Visual Studio Debug Console. The title bar reads "Microsoft Visual Studio Debug Console". The console text is as follows:
Enter real number x and positive integer n.
the programm will calculate n partial sums
1.2 3
n sum
1 0.360000
2 0.408000
3 0.422400
The text is displayed in a monospaced font with some color coding: "Enter" is green, "the programm" is blue, "n sum" is pink, and the numbers 1, 2, 3 are green.

Совпадает!

3.

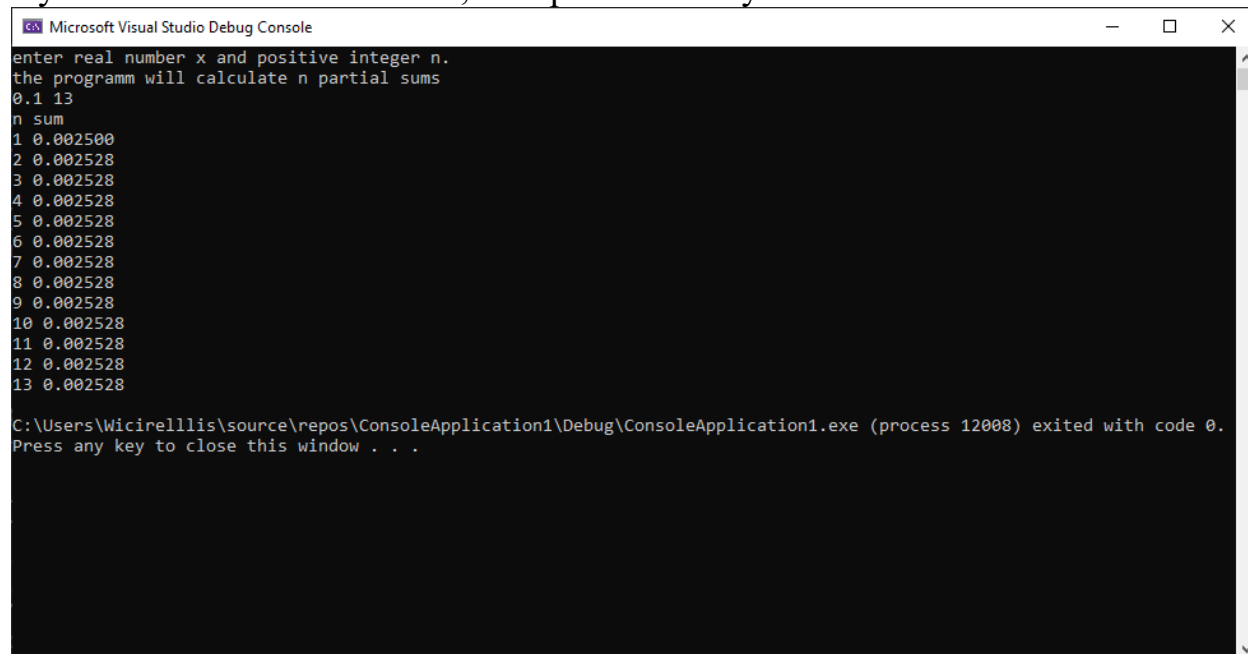
Несколько скринов вывода



```
Microsoft Visual Studio Debug Console
enter real number x and positive integer n.
the program will calculate n partial sums
0 10
n sum
1 0.000000
2 0.000000
3 0.000000
4 0.000000
5 0.000000
6 0.000000
7 0.000000
8 0.000000
9 0.000000
10 0.000000

C:\Users\wicirelllis\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (process 11956) exited with code 0.
Press any key to close this window . . .
```

Нули т.к. в числителе есть x, который все обнуляет.



```
Microsoft Visual Studio Debug Console
enter real number x and positive integer n.
the program will calculate n partial sums
0.1 13
n sum
1 0.002500
2 0.002528
3 0.002528
4 0.002528
5 0.002528
6 0.002528
7 0.002528
8 0.002528
9 0.002528
10 0.002528
11 0.002528
12 0.002528
13 0.002528

C:\Users\wicirelllis\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (process 12008) exited with code 0.
Press any key to close this window . . .
```

Суммы одинаковые начиная со второй, т.к. слагаемые становятся достаточно малы и теряются из-за ограничения на кол-во знаков в выводе.

```
Microsoft Visual Studio Debug Console
enter real number x and positive integer n.
the programm will calculate n partial sums
5 7
n sum
1 6.250000
2 9.722222
3 14.062500
4 21.875000
5 39.236111
6 83.524660
7 208.086203

C:\Users\wicirelllis\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (process 9464) exited with code 0.
Press any key to close this window . . .

Microsoft Visual Studio Debug Console
enter real number x and positive integer n.
the programm will calculate n partial sums
60 10
n sum
1 900.000000
2 6900.000000
3 96900.000000
4 2040900.000000
5 53880900.000000
6 1640819675.510204
7 55200003348.979591
8 1999200003348.979492
9 76648800003348.984375
10 3074971576862853.000000

C:\Users\wicirelllis\source\repos\ConsoleApplication1\Debug\ConsoleApplication1.exe (process 3076) exited with code 0.
Press any key to close this window . . .
```

4. Вопросы

1) Какой формат данных нужно использовать, чтобы хранить значение -50000?

long(32-битовый целый тип)

все дело в том, что -50000 меньше чем -32768.

число отрицательное, поэтому unsigned тип не подойдет.

int может не подойти, т.к. int *может* быть размером в два байта и тогда он представляет числа от -32768 до 32767. кажется, стандарт говорит, что int это по крайней мере 16 бит, но многие современные компиляторы используют 32 бита (=4 байта).

2) Что будет выведено в консоль, если в программе присутствует следующий код:

```
float i = 12345.12345;  
float b = 456.1265;  
printf("2.2%f", i);  
printf(" %2.2f", b);
```

Я подозреваю, что ответ это
2.212345.12345 456.13

Но если скопировать код VS, то получится
2.212345.123047 456.13

Давайте попробуем разобраться почему это происходит.

Я поправил форматирование и поставил нормальные кавычки, чтобы код компилировался. Чтобы узнать вывод можно просто скомпилировать код (добавив инклюды и майн).

В форматном выводе **%f** используется для чисел с плавающей точкой. Например float/double. По-умолчанию печатаются все цифры слева от точки и **шесть** цифр справа от точки. Чтобы выравнивать вывод или обрезать лишние знаки после точки используются параметры после **%**. Посмотрим на **%2.2f**. Здесь первая двойка говорит, что будет напечатано **не меньше** двух символов, если печатемое число короче, то слева добавятся пробелы. Вторая двойка означает, что после запятой остаются только два знака. Причем, вроде бы, происходит *округление* до двух знаков, а не просто отбрасывание лишних знаков.

Посмотрим на строку `float i = 12345.12345;` Как я понимаю, компилятор считает число после равенства `double`-ом, а затем приводит его к `float`-у и на этом теряет точность и из-за этого печатается не то, что можно было ожидать. Со второй строкой так же.

Т.е. в `printf("2.2%f", i);` 2.2 это просто строка, а в `printf(" %2.2f", b);` 2.2 это параметры вывода и в этом основная разница.

3) Приведите пример (примеры) кода, когда необходимо провести множественную проверку: значение условия может быть "1","2","3","4"

Тут есть два очевидных метода сделать необходимое. Либо `switch`, либо куча `if`-ов. `Switch` предпочтительнее, т.к. читаемость выше.

Можно посмотреть на такой код:

```
switch (i) {
case 1:cout << "1";
    break;
case 2:cout << "2";
    break;
case 3:cout << "3";
    break;
case 4:cout << "4";
    break;
}
```

В зависимости от переменной `i` он напечатает соответствующее сообщение. `break` нужны чтобы программа печатала только одно сообщение.

Вариант с вложенными `if`-ами

```
if (i == 1) {
    cout << "1";
}
else if (i == 2) {
    cout << "2";
}
else if (i == 3) {
    cout << "3";
}
else if (i == 4) {
    cout << "4";
}
```

Можно выкинуть все else:

```
if (i == 1) {  
    cout << "1";  
}  
if (i == 2) {  
    cout << "2";  
}  
if (i == 3) {  
    cout << "3";  
}  
if (i == 4) {  
    cout << "4";  
}
```

Разница в том, что без else каждое условие проверится. А с else, если мы нашли нужное значение, то оставшиеся if-ы проверяться не будут.