Temat: Całkowanie numeryczne metodą Romberga			
Wykonał: Marcin Fabrykowski	Wydział: FiIS	Kierunek Inf. Stos.	Grupa: grupa 3

1. Wstęp
   Metoda Romberga jest rekurencyjną metodą obliczania całek. Polega na tworzeniu tablic całkowych. Tablice te tworzy się na podstawie poniższego algorytmu:

   (a) $D_{0,0} = \frac{1}{2}[f(a) + f(b)]$;

   (b) $D_{n,0} = \frac{1}{2}D_{n-1,0} + h_n \sum_{i=1}^{2^{n-1}} f(x + (2i-1)h_n)$

   (c) $D_{n,k} = \frac{4^k D_{n,k-1} - D_{n-1,k-1}}{4^k - 1}$

   gdzie $h_n = \frac{b-a}{2^n}$

2. Wykonanie ćwiczenia
   Celem ćwiczenia jest wyznaczenie tablic całek dla funkcji

   (a) $\int_{-1}^{1} \frac{\sin(x)}{x} dx$ $(= 1.892166141)$, dla $n = 7$

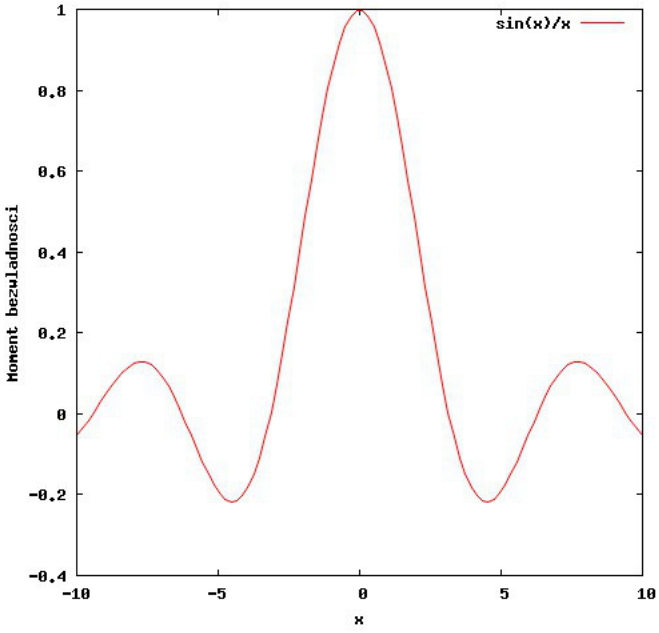
   (b) $\int_{0}^{\pi} \sin(5x)x^2 dx$ $(= 1.941920881)$, dla $n = 10$

   Wykresy powyższych funkcji widać na wykresach poniżej. Program realizujący powyższe zadanie:

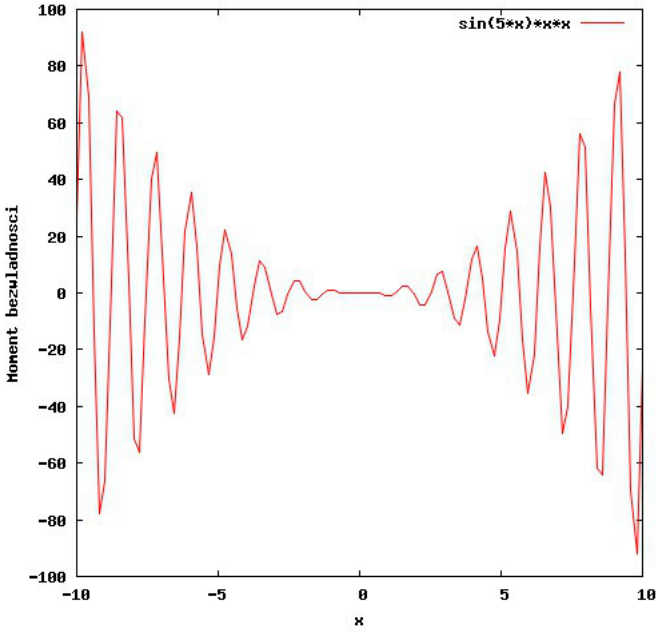
Listing 1: main.cpp

```
#include <iostream>
#include <cmath>
using namespace std;
float f1(float x);
float f2(float x);
int main()
{
    cout.precision(5);
```

Rysunek 1: f1(x)



Rysunek 2: f2(x)



```
    float (*f)(float);
    f=f2;
    int N=10;
    float a=0;
    float b=3.14;
    float** tab=new float*[N];
    int i;
    int j;
    for(i=0;i<N;i++)
    {
        tab[i]=new float[N];
    };
    tab[0][0]=(f(a)+f(b))/2;
    int n;
    int k;
    for(n=1;n<N;n++)
    {
        float h=(b-a)/pow(2,n);
        tab[n][0]=tab[n-1][0]/2;
        float sum=0;
        for(i=1;i<=pow(2,n-1);i++)
        {
            sum+=f(a+(2*i-1)*h);
        };
        sum*=h;
        tab[n][0]+=sum;
    };
    for(n=1;n<N;n++)
    {
        for(k=1;k<=n;k++)
        {
            tab[n][k]=(float)(pow(4,k)*tab[n][k-1]-tab[n-1][k-1])/(pow(4,k)-1);
        };
    };
    for(n=0;n<N;n++)
    {
        for(k=0;k<=n;k++)
        {
            cout<<tab[n][k]<<"\t";
        };
        cout<<endl;
    };
};
float f1(float x)
{
    return sin(x)/(x+0.00000001);
};
float f2(float x)
{
```

```
    return sin(5*x)*x*x;
};
```

Tabela dla $f1(x)$:

```
0.84147
0.42074 0.28049
1.1692 1.4187 1.4946
1.5338 1.6554 1.6712 1.674
1.7138 1.7738 1.7817 1.7834 1.7838
1.8032 1.833 1.8369 1.8378 1.838 1.8381
1.8477 1.8626 1.8645 1.865 1.8651 1.8651 1.8651
```

Tabela dla $f2(x)$:

```
0.039253
3.8895 5.1729
-1.4934 -3.2878 -3.8518
1.2573 2.1742 2.5383 2.6397
1.7761 1.949 1.934 1.9244 1.9216
1.8996 1.9407 1.9402 1.9403 1.9403 1.9403
1.9307 1.941 1.941 1.9411 1.9411 1.9411 1.9411
1.9387 1.9414 1.9415 1.9415 1.9415 1.9415 1.9415 1.9415
1.9409 1.9416 1.9417 1.9417 1.9417 1.9417 1.9417 1.9417 1.9417
1.9415 1.9417 1.9418 1.9418 1.9418 1.9418 1.9418 1.9418 1.9418 1.9418
```

3. Wnioski
   Można zauważyć, że w przypadku obu funkcji, wartości na diagonali oraz pierwszej kolumnie dają oczekiwane wartości od 7 iteracji. Wynik jest zadowalający, oraz otrzymany w krótkim czasie, co pokazuje zasadność używania tej metody.