

# Zadanie indywidualne

Całkowanie metodą złożonych trapezów

**Autor:**  
Marcin Smarzewski

## Spis treści

<b>1</b>	<b>Całkowanie metodą złożonych trapezów</b>	<b>2</b>
1.1	Wprowadzenie . . . . .	2
1.2	Wady i zalety rozwiązania . . . . .	2
<b>2</b>	<b>Działanie programu</b>	<b>4</b>
2.1	Schematy blokowe . . . . .	4
2.2	Przetwarzanie danych . . . . .	6
2.2.1	Przetwarzanie danych wejściowych z terminalu . . . . .	6
2.2.2	Przetwarzanie danych wejściowych z pliku testowego . . . . .	7
<b>3</b>	<b>Obliczanie całki</b>	<b>9</b>
3.1	Zaimplementowane funkcję oraz działania matematyczne . . . . .	11
3.1.1	Format danych . . . . .	11
3.1.2	Implementacja funkcji matematycznych . . . . .	11
<b>4</b>	<b>GitHub</b>	<b>12</b>
<b>5</b>	<b>Przykłady działania programu</b>	<b>12</b>
<b>6</b>	<b>Kod całego programu</b>	<b>13</b>

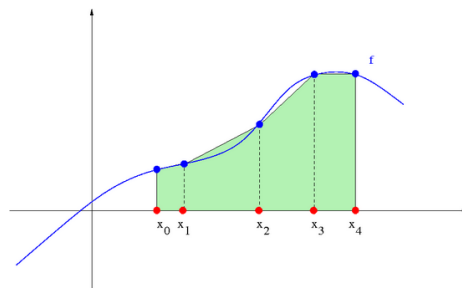
# 1 Całkowanie metodą złożonych trapezów

## 1.1 Wprowadzenie

Całkowanie złożoną metodą trapezów jest metodą numeryczną, mającą na celu obliczenie przybliżonej wartości całki oznaczonej. Metoda ta polega na podzieleniu przedziału na równe podprzedziały - im więcej podprzedziałów(trapezów) tym wynik będzie bardziej zbliżony do poprawnego. Przybliżenie wartości całki wygląda w następujący sposób:

1.  $a_n$  - wartość funkcji na początku n-tego podprzedziału(I podstawa trapezu)
2.  $b_n$  - wartość funkcji na końcu n-tego podprzedział(II podstawa trapezu)
3.  $h$  - długość podprzedziału (wysokość trapezu)
4.  $n$  - ilość trapezów

$$\int_a^b f(x) dx \approx \sum_{i=1}^n \frac{(a_i + b_i)h}{2}$$



Wizualizacja metody

Źródło: <https://wazniak.mimuw.edu.pl/index.php?title=MN14>

## 1.2 Wady i zalety rozwiązania

Głównymi wadami rozwiązania są następujące wady:

1. Rozwiązanie nie determinuje całkowalności funkcji na danym przedziale, lecz wypisze poprawny wynik. Przykładem może być funkcja  $\text{ctg}(x)$  na przedziale  $[-1,1]$ (Listing 1) oraz dokładności 5 program wypisze wartość liczbową gdy całka jnie jest zbieżna(listing 5).
2. Rozwiązanie jest w stanie przybliżyć wartość tej funkcji, ogranicza go dokładność oraz użyty typ danych, także w niektórych przypadkach wyniki będą różnić się od teoretycznych.

Głównymi zaletami rozwiązania są:

1. Szybkie przybliżenie wartości całki bez używania do tego wzorów matematycznych lub całkowania metodą analityczną. W przypadku bardziej skomplikowanych całek jest to często szybsze niż ich analityczne obliczenie.
2. Niektóre całki nie posiadają analitycznych rozwiązań, a więc rozwiązanie numeryczne jest dobrym sposobem na przybliżenie jej wartości.

Listing 1:  $\int \text{ctg}(x)$  na przedziale  $[-1;1]$

```
Program liczący całkę oznaczoną złożoną metodą trapezów
1—Wpisz funkcję do obliczenia całki
2—Dane testowe pobierane z pliku
3—Zakończ
1
Wpisz funkcję którą chcesz obliczyć
ctg(x)
Wpisz początek przedziału
-1
Wpisz koniec przedziału
1
Wpisz ilość trapezów
5
Całka oznaczona na podanym przedziale wynosi: -0.67759
```

## 2 Działanie programu

### 2.1 Schematy blokowe

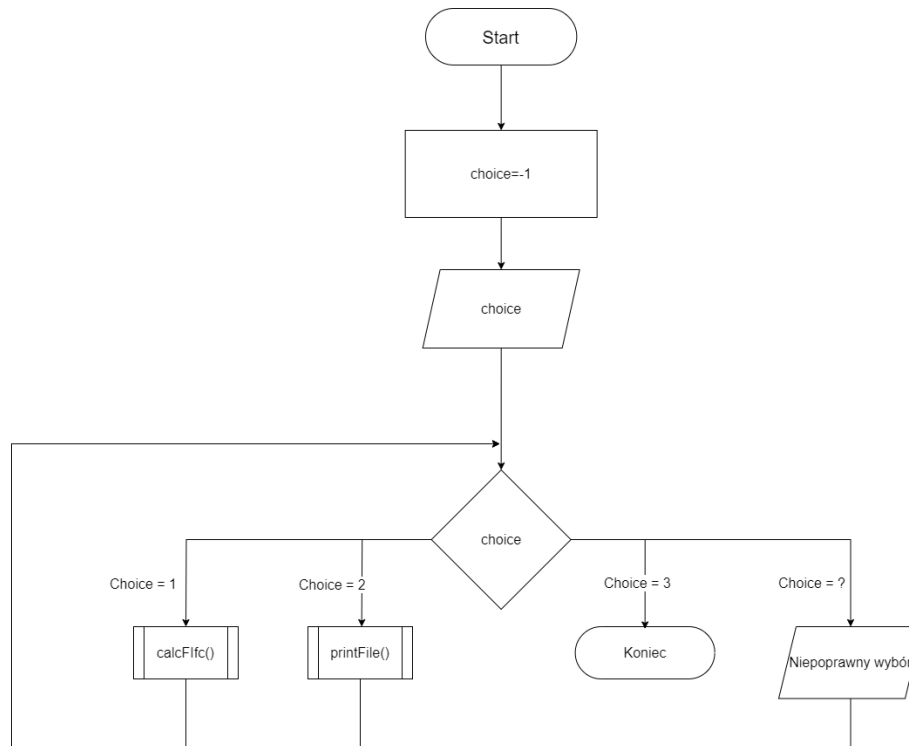


Figure 2: Schemat blokowy programu

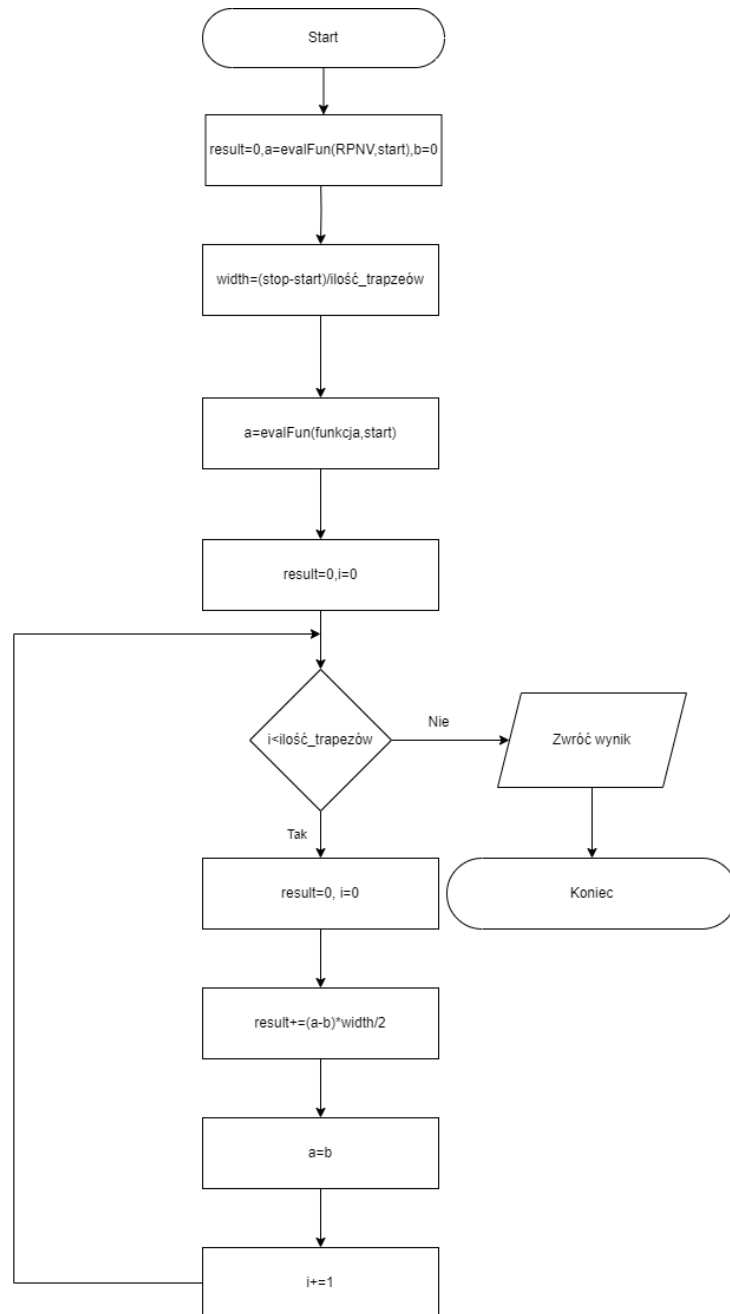


Figure 3: Schemat blokowy calculateIntegral()

## 2.2 Przetwarzanie danych

### 2.2.1 Przetwarzanie danych wejściowych z terminalu

Dane wejściowe wpisane w terminalu są parsowane ze względu na kolejność wykonywania działań która nie musi być zachowana przez użytkownika. Funkcja wpisana przez użytkownika zamieniana jest na odwrotną notację polską w funkcji RPN(listing 2.) Korzysta ona z lekko zmodyfikowanego algorytmu Shunting yard, w rezultacie algorytm przedstawia się następująco:

1. Pobierz element
2. Jeśli aktualny element jest x'em lub jest liczbą->zwróć go na wyjściu
3. Jeśli element jest operatorem i stos jest pusty lub jest nawiasem otwierającym -> wrzuć go na stos
4. Jeśli element jest operatorem(lub funkcją) o **WIĘKSZYM** priorytecie niż aktualny element na górze stosu -> wrzuć go na stos
5. Jeśli element jest operatorem(lub funkcją) o **MNIEJSZYM** priorytecie niż aktualny element na górze stosu -> zwracaj elementy ze stosu na wyjściu do napotkania elementu o mniejszym bądź równym priorytecie lub wyczerpaniu stosu lub do napotkania nawiasu otwierającego
6. Jeśli element jest nawiasem zamykającym -> zwracaj elementy ze stosu do napotkania nawiasu otwierającego -> zdejmij nawias otwierający ze stosu
7. Jeśli można pobrać następny element -> przejdź do kroku 1

Algorytm opracowany na podstawie algorytmu:

[https://en.wikipedia.org/wiki/Shunting\\_yard\\_algorithm](https://en.wikipedia.org/wiki/Shunting_yard_algorithm)

### 2.2.2 Przetwarzanie danych wejściowych z pliku testowego

Przetwarzanie samych danych(funcji) jest identyczne jak w przypadku danych przyjętych z terminala. Format pliku testowego wygląda następująco:  
Wiersz 2n - Funkcja której wartość będzie obliczona przez program(gdzie  $n > 0$ )  
Wiersz 2n+1 - Wartość funcji obliczona oraz wpisana przez użytkownika(gdzie  $n > 1$ )

Wartości teoretyczne zostały obliczone poprzez WolframAlpha:

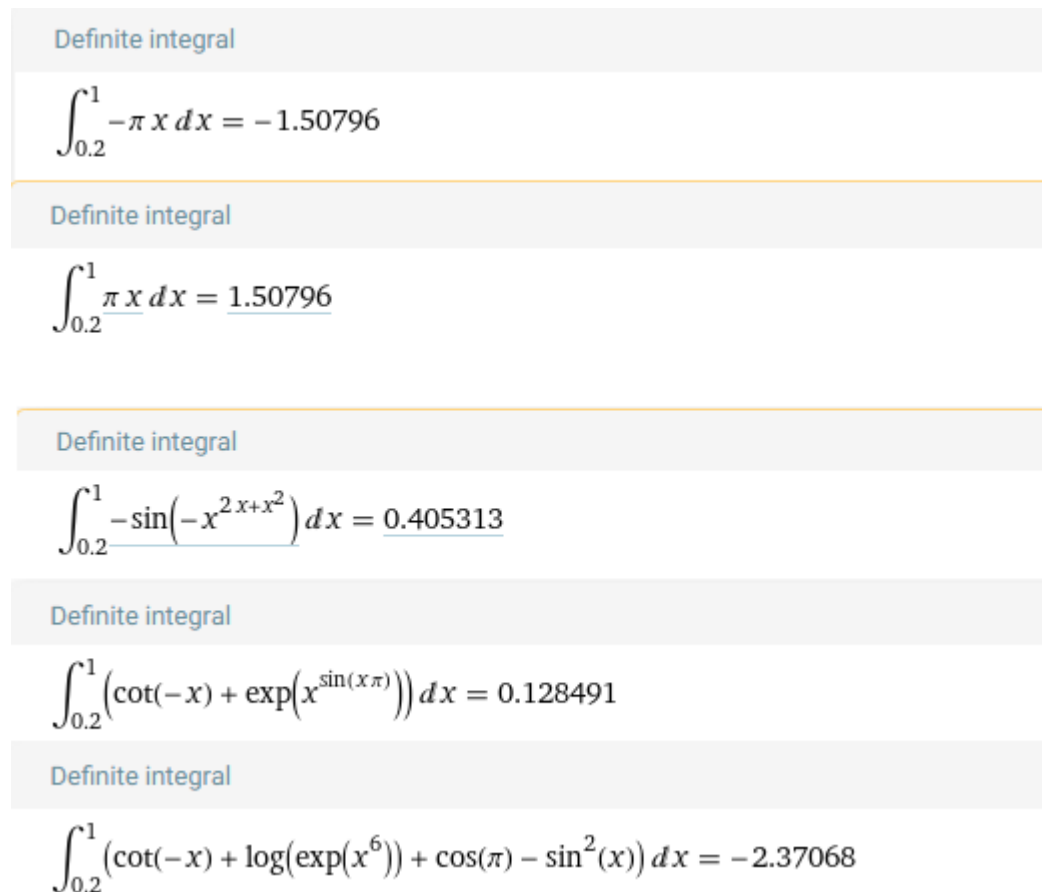


Figure 5: Źródło: <https://www.wolframalpha.com/>



Listing 2: Funkcja RPN

```

string RPN(string equation){           //checkString(string) usuwa spacje oraz
equation=checkString(equation); //dodaje * miedzy cyframi oraz funkcjami
    equation=fixSpaces(equation);      //fixSpaces(string) dodaje spacje
    cout<<endl<<equation<<endl;       //wokol operatorow
    stack<string>ostack;
    string curr,result="";
    istringstream temp(equation);
    while(temp>>curr){
        if(isdigit(curr[0])||curr=="x"){
            result+=curr+"_";
        }
        else if(isOperator(curr)){
            if(ostack.empty()||curr=="(") ostack.push(curr);
            else{
                while(!ostack.empty()
&& rnpPrio(curr)<=rnpPrio(ostack.top())
&&ostack.top()!="("){
                    result+=ostack.top()+"_";
                    ostack.pop();
                }
                if(curr==")") ostack.pop();
                else ostack.push(curr);
            }
        }
    }
    while(!ostack.empty()){
        result+=ostack.top()+"_";
        ostack.pop();
    }
    return result;
}

```

### 3 Obliczanie całki

Po przetworzeniu danych wejściowych na odwrotną notację polską program oblicza sumę funkcji

$$\int_a^b f(x) dx \approx \sum_{i=1}^n \frac{(a_n + b_n)h}{2}$$

Oblicza to w funkcji `calculateIntegral`(listing 3). W funkcji przypisanie `a=b` po dodaniu aktualnego trapezu do sumy pozwala zaoszczędzić niepotrzebnemu liczeniu ówczśnie policzonej funkcji(początek przedziału `n+1` jest końcem przedziału `n`).

Funkcja `evalFun`(listing 4) oblicza wartość wyrażenia ONP dla aktualnego końca przedziału. Aby obliczyć funkcję elementarne(trigonometryczne, logarytmiczne itp.).

Funkcja używa biblioteki `<functional>`. Dzięki niej nie musimy definiować oddzielnie przypadków wywołania danej funkcji, lecz możemy zdefiniować uniwersalną funkcję wywołującą(listing 4).

Listing 3: `calculateIntegral`

```
double calculateIntegral(double start , double stop ,  
    int precision , string equation) {  
    string RPNV=RPN(equation);  
    double result=0,a=evalFun(RPNV,start),b=0;  
    double width =(stop-start)/precision;  
    for(int i=0;i<precision;i++){  
        b=evalFun(RPNV,start+width*i);  
        result+=((a+b)*width) / 2;  
        a=b;  
    }  
    return result;  
}
```

Listing 4: evalFun

```

double evalFun(string RPNV,double value){
    stack<string>operrands;
    double result=0,pres,num1,num2;
    istringstream temp(RPNV);
    string x;
    while(temp>>x)
    {
        if(isdigit(x[0])){
            operrands.push(x);
        }
        else if(x=="x"){
            operrands.push(to_string(value));
        }
        else if(x=="pi") operrands.push(to_string(M_PI));
        else if(isFunc(x)){
            pres=eval(func[x],stod(operrands.top()));
            operrands.pop();
            operrands.push(to_string(pres));
        }
        else if(isOperator(x)){ //Sprawdza czy dany element jest
            num2=stod(operrands.top()); //operatorem lub funkcj
            operrands.pop();
            num1=stod(operrands.top());
            operrands.pop();
            pres=basicOperation(x,num1,num2);
            operrands.push(to_string(pres));
        }
    }
    if(operrands.size()==1) result+=stod(operrands.top());
    return result;
}

```

Listing 5: Przeliczanie funkcji zwracającej typ double oraz przyjmującej typ double w zależności od zmiennej x

```

double eval(function<double(double)> func, double x)
    return func(x);
}

```

### 3.1 Zaimplementowane funkcję oraz działania matematyczne

W programie zostały zaimplementowane podstawowe działania arytmetyczne (dodawanie, odejmowanie, dzielenie oraz mnożenie) oprócz tego zostały dodane wszystkie funkcję trygonometryczne (bez funkcji cyklometrycznych), funkcja logarytmu naturalnego, pierwiastka kwadratowego, funkcja wykładnicza oraz potęgowanie. Zostały one zdefiniowane w globalnej mapie `func` (listing 6).

#### 3.1.1 Format danych

Program jest w stanie przeliczyć jakiekolwiek wyrażenie matematyczne które korzysta z zaimplementowanych funkcji oraz operacji matematycznych. Przy czym format danych ma pare reguł:

1. Argumenty funkcji zawsze piszemy w nawiasach tzn.  $\sin(x)$  jest poprawnym zapisem przy czym  $\sin x$  nie.
2. Przed funkcją oraz  $x$ 'em może stać cyfra co będzie poprawnym zapisem (funkcja `checkString` wstawi tam znak mnożenia)
3. Gdy nawiasy nie będą domknięte program odrzuci funkcję wypisując przy tym odpowiedni błąd
4. Program powinien korzystać tylko ze zdefiniowanych funkcji ze względu że nie obsługuje on błędów odczytu, nie obliczy funkcji  $\arcsin$  ze względu na to że nie jest ona w nim zadeklarowana - program najprawdopodobniej zakończy się przy takim wczytaniu

#### 3.1.2 Implementacja funkcji matematycznych

Jedną z najważniejszych funkcji w programie która nie została dotychczas omówiona jest inicjacja mapy globalnej zdefiniowanych funkcji matematycznych. Składa się ona z nazwy typu `string` danej funkcji oraz przypisania do niej wskaźnika funkcji która ją oblicz. Większość użytych funkcji jest zdefiniowana w podstawowych bibliotekach matematycznych dla języka C++. Reszta z nich jest zdefiniowana jest lambdą. Metoda ta ułatwia przyszłą skalowalność programu.

Listing 6: Mapa funkcji matematycznych

```
map<string, double (*)(double)> initFunc(){
    map<string, double (*)(double)> func;
    func["sin"] = sin;
    func["cos"] = cos;
    func["tg"] = tan;
    func["ctg"] = [](double x){return 1/tan(x); };
    func["exp"] = exp;
    func["ln"] = log;
    func["sqrt"] = sqrt;
    func["pi"] = [](double){return M_PI;};
    return func;
}
map<string, double (*)(double)> func=initFunc();

}
```

## 4 GitHub

W pracy nad projektem wykorzystałem GitHub'a - platformy hostingowej dzięki której byłem w stanie pracować na tym samym kodzie z wielu urządzeń bez potrzeby przenoszenia kodu innymi drogami. Prostota obsługi dawała mi komfort pracy a także brak martwienia się zgrywaniem kodu między urządzeniami. Z platformy korzystałem już przy kilku projektach i niewyobrażam sobie pracy indywidualnej, a co dopiero zespołowej która jest ułatwiona poprzez możliwość pracowania na tym samym kodzie przez wiele osób jednocześnie.

## 5 Przykłady działania programu

Plik testowy zawierający wszystkie funkcje zawarte w programie:

```
2
Funkcja: pi*x
Jej całka wynosi(na przedziale 0.2, 1 z iloscia trapezow 1000) 1.50596
Jej wartość teoretyczna(liczona z definicji całki) wynosi: 1.507
Funkcja: -pi*x
Jej całka wynosi(na przedziale 0.2, 1 z iloscia trapezow 1000) -1.50596
Jej wartość teoretyczna(liczona z definicji całki) wynosi: -1.507
Funkcja: -sin(-x^(2x+x^2))
Jej całka wynosi(na przedziale 0.2, 1 z iloscia trapezow 1000) 0.405019
Jej wartość teoretyczna(liczona z definicji całki) wynosi: 0.405
Funkcja: ctg(-x)+exp(x*sin(x*pi))
Jej całka wynosi(na przedziale 0.2, 1 z iloscia trapezow 1000) 0.124062
Jej wartość teoretyczna(liczona z definicji całki) wynosi: 0.128491
Funkcja: ctg(-x)+ln(exp(x^6))+cos(pi)-sin(x)^2
Jej całka wynosi(na przedziale 0.2, 1 z iloscia trapezow 1000) -2.37438
Jej wartość teoretyczna(liczona z definicji całki) wynosi: -2.37068
Program liczący całkę oznaczoną złożoną metodą trapezów
1-Wpisz funkcje do obliczenia całki
2-Dane testowe pobierane z pliku
3-Zakończ
```

## 6 Kod całego programu

Listing 7: Caly program

```
#include <iostream>
#include <functional>
#include <cmath>
#include <math.h>
#include <string>
#include <map>
#include <stack>
#include <vector>
#include <sstream>
#include <fstream>
#include <iomanip>

using namespace std;
double basicOperation(string op, double num1, double num2){
    if(op=="^") return pow(num1,num2);
    if(op=="+") return num1+num2;
    if(op=="-") return num1-num2;
    if(op=="*") return num1*num2;
    if(op=="/") return num1/num2;
    return 0;
}

double eval(function<double(double)> func, double x) {
    return func(x);
}

bool checkParentheses(string input){
    int p1=0,p2=0;
    for(char op:input){
        if(op=='(') p1++;
        else if(op==')') p2++;
    }
    if(p2==p1) return true;
    return false;
}

map<string, double(*)(double)> initFunc(){
    map<string, double(*)(double)> func;
    func["sin"] = sin;
    func["cos"] = cos;
    func["tg"] = tan;
    func["ctg"] = [](double x){return 1/tan(x); };
    func["exp"] = exp;
```

```

    func["ln"] = log;
    func["sqrt"] = sqrt;
    func["pi"] = [](double){return M_PI;};
    return func;
}
map<string, double(*)(> double)> func=initFunc();
bool isFunc(string pFunc){
    return func.find(pFunc)!=func.end();
}
int rnpPrio(string op){
    if(isFunc(op)) return 4;
    else if(op=="^") return 3;
    else if(op=="|" || op=="*") return 2;
    else if(op=="+" || op=="-") return 1;
    else return 0;
}
bool isOperator(string isOp) {
    return (isOp == "+" || isOp == "-" || isOp == "*" || isOp == "/" || isOp ==
}
string fixSpaces(string input){
    stringstream str;
    for(int i=0;i<input.length();i++){
        if(isOperator(input.substr(i,1))) str<<"_"<<input[i]<<"_";
        else if(input[i]!='_') str<<input[i];
    }
    return str.str();
}

double evalFun(string RPNV, double value){
    stack<string>operrands;
    double result=0,pres,num1,num2;
    istream temp(RPNV);
    string x;
    while(temp>>x)
    {
        if(isdigit(x[0])){
            operrands.push(x);
        }
        else if(x=="x"){
            operrands.push(to_string(value));
        }
        else if(x=="pi") operrands.push(to_string(M_PI));
        else if(isFunc(x)){
            pres=eval(func[x], stod(operrands.top()));
            operrands.pop();
            operrands.push(to_string(pres));
        }
    }
}

```

```

    }
    else if(isOperator(x)){
        num2=stod(operrands.top());
        operrands.pop();
        num1=stod(operrands.top());
        operrands.pop();
        pres=basicOperation(x,num1,num2);
        operrands.push(to_string(pres));
    }
}
if(operrands.size()==1) result+=stod(operrands.top());
return result;
}
string checkString(string input){
    bool digit=false, alpha=false, op=false;
    if(input[0]=='-') input.insert(0,"0");
    int len=input.length(), i=0;
    while(i<len){
        char temp=input[i];
        if(input[i]=='(' && input[i+1]=='-') input.insert(i+1,"0");
        if(isdigit(input[i])){
            digit=true;
            alpha=false;
        }
        else if(isalpha(input[i])){
            alpha=true;
            if(digit) input.insert(i,"*");
            digit=false;
        }
        else if(isOperator(input.substr(i,1))){
            op=true;
            alpha=false;
            digit=false;
        }
        i++;
    }
    return input;
}
string RPN(string equation){
    equation=checkString(equation);
    equation=fixSpaces(equation);
    stack<string>ostack;
    string curr, result="";
    istringstream temp(equation);
    while(temp>>curr){
        if(isdigit(curr[0]) || curr=="x"){

```





```

        cin>>schoice;
        if( choice==1) swap(start ,stop);
    }
    else{
        cout<<"Wpisz ilo    _trapez w"<<endl;
        cin>>precision;
        try
        {
            cout<<setprecision(5);
            cout<<"Calka_oznaczona_na_podanym_przedziale_wynosi:_";
            cout<<calculateIntegral(start ,stop ,precision ,input)<<endl;

        }
        catch(const std::exception& e)
        {
            cout<<"Wyst pi _b    d "<<endl;
        }
    }
}

}

void printFile(){
    ifstream test("test.txt");
    string teststr ,descstr;
    while(getline(test ,teststr)&&getline(test ,descstr)){
        cout<<"Funkcja:_ "<<teststr<<endl<<"Jej _ca _ka _wynosi(na_przedziale_0
    }
    test.close();
}

void prompt(){
    int choice = -1;
    while(choice!=3){
        cout<<"Program_licz cy _ca k _oznaczon _z o on _metod _trapez
        cin>>choice;
        if( choice==1)
            calcFIfc();
        else if( choice==2)
            printFile();
        else if( choice==3){
            return;
        }else{
            cout<<"Niepoprawny_wyb r "<<endl;
        }
    }
}
}

```

```
int main() {  
    system("chcp_1250>>null");  
    setlocale(LC_CTYPE, "Polish");  
    prompt();  
    return 0;  
}  
}
```