Zadanie indywidualne

Całkowanie metodą złożonych trapezów

Autor:
Marcin Smarzewski

Spis treści

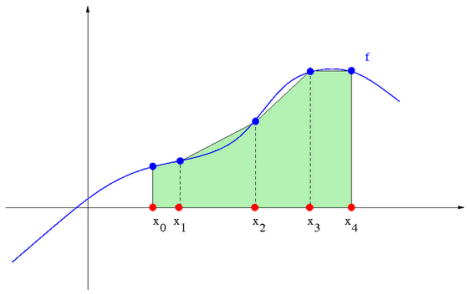
1 Całkowanie metodą złożonych trapezów

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

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



Wizualizacja metody

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 ctg(x) na przedziale [0,1] 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.

2 Działanie programu

2.1 Przetwarzanie danych

2.1.1 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 1.) Korzysta ona z algorytmu Shunting yard który w liście kroków 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

Źródło: https://en.wikipedia.org/wiki/Shunting_yard_algorithm

2.1.2 Przetwarzanie danych wejściowych z pliku testowego

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

Wartości teoretyczne zostały obliczone poprzez WolframAlpha:

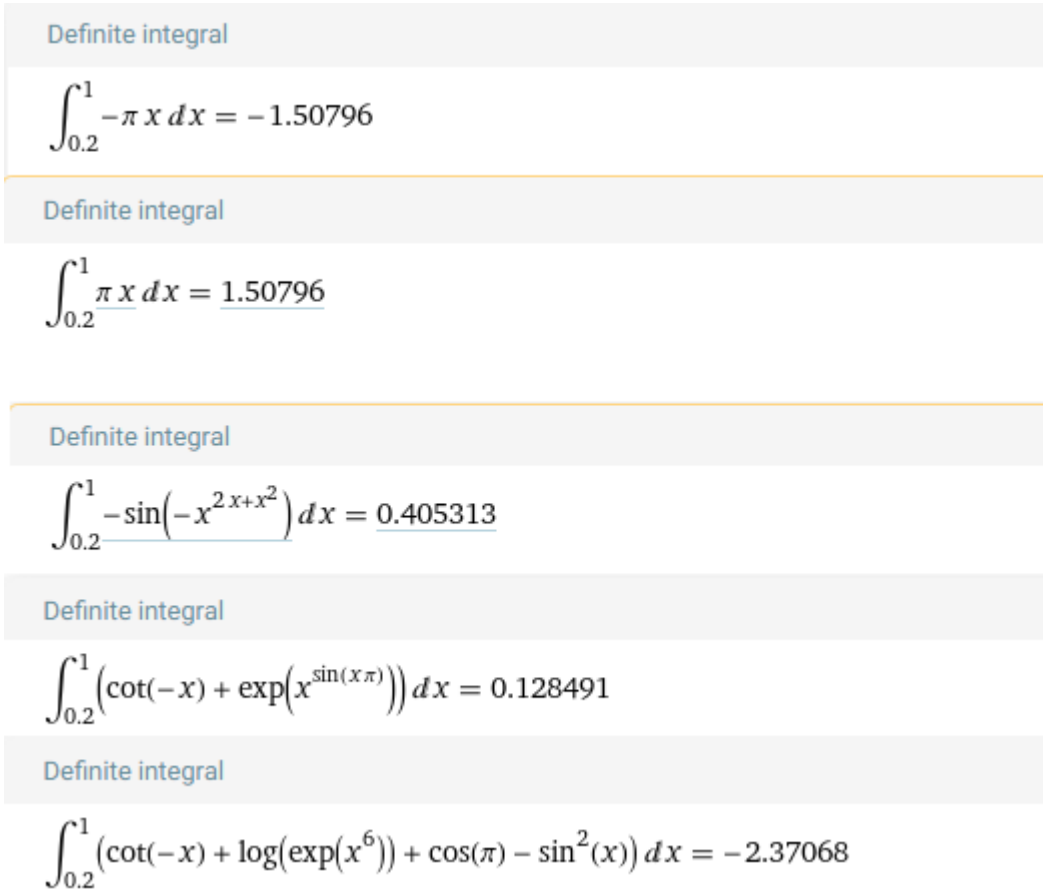


Figure 3: Źródło: https://www.wolframalpha.com/

Listing 1: 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 2). W funkcji przypisanie a=b po dodaniu aktualnego trapzeu do sumy pozwala zaoszczędzić niepotrzebnemu liczeniu ówcześnie policzonej funkcji(początek przedziału n+1 jest końcem przedzialu n).
Funkcja evalFun(listing3) oblicza wartość wyrażenia ONP dla aktualnego końca przedziału. Aby obliczyć funkcję elementarne(trygonometryczne, logarytmiczne itp.).
Funckja używa biblioteki <functional>. Dzięki niej nie musimy definiować oddzielnie przypadków wywołania danej funkcji, lecz możemy zdefiniować uniwersalną funkcję wywołującą(listing 4).

Listing 2: calculateIntegral

```
double calculateIntegral(double start, double stop,
                         int precision, string equation) {
    string RPNV=RPN(equation);
    double result=0,width,a=evalFun(RPNV,start),b;
    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 3: 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 operatorem lub
            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;
}
```

Listing 4: 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 to jakiekolwiek wyrażenie matematyczne które korzysta z zaimplementowanych funckji 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 sinx 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

Listing 5: oraz liczbą trapezów wynoszącą 5]ctg(x) w przedziale [-1,1] oraz liczbą trapezów wynoszącą 5

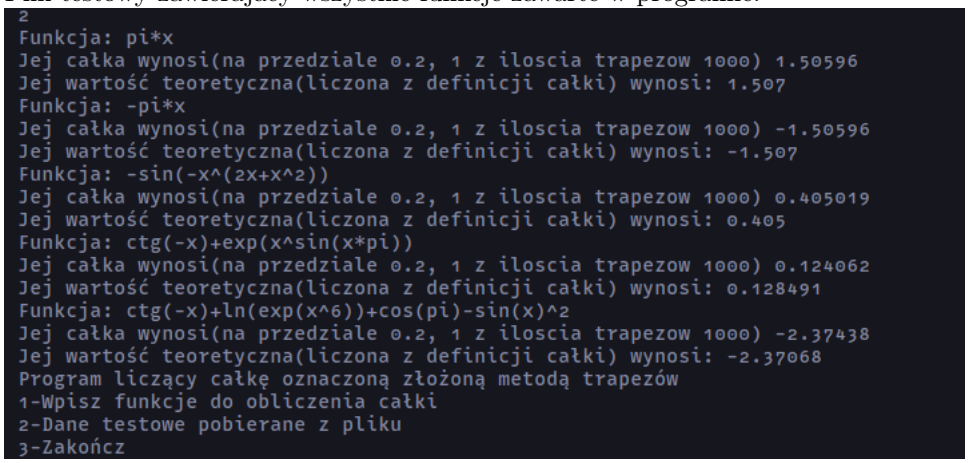
```
Program liczacy calke oznaczona zlozona metoda trapezow
1--Wpisz funkcje do obliczenia calki
2--Dane testowe pobierane z pliku
3--Zakoncz
1
Wpisz funkcje ktora chcesz obliczyc
ctg(x)
Wpisz poczatek przedzialu
-1
Wpisz koniec przedzialu
1
Wpisz ilosc trapezow
5
Calka oznaczona na podanym przedziale wynosi: -0.67759
```

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 zawierajacy wszystkie funkcje zawarte w programie:



6 Schematy blokowe

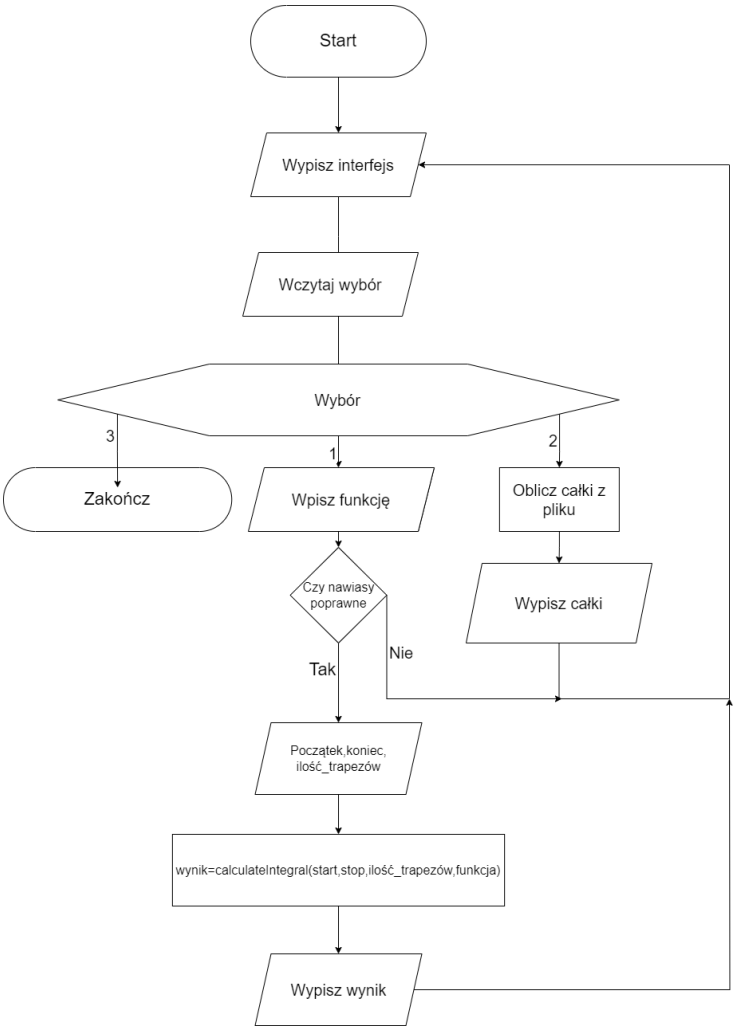


Figure 4: Schemat blokowy programu

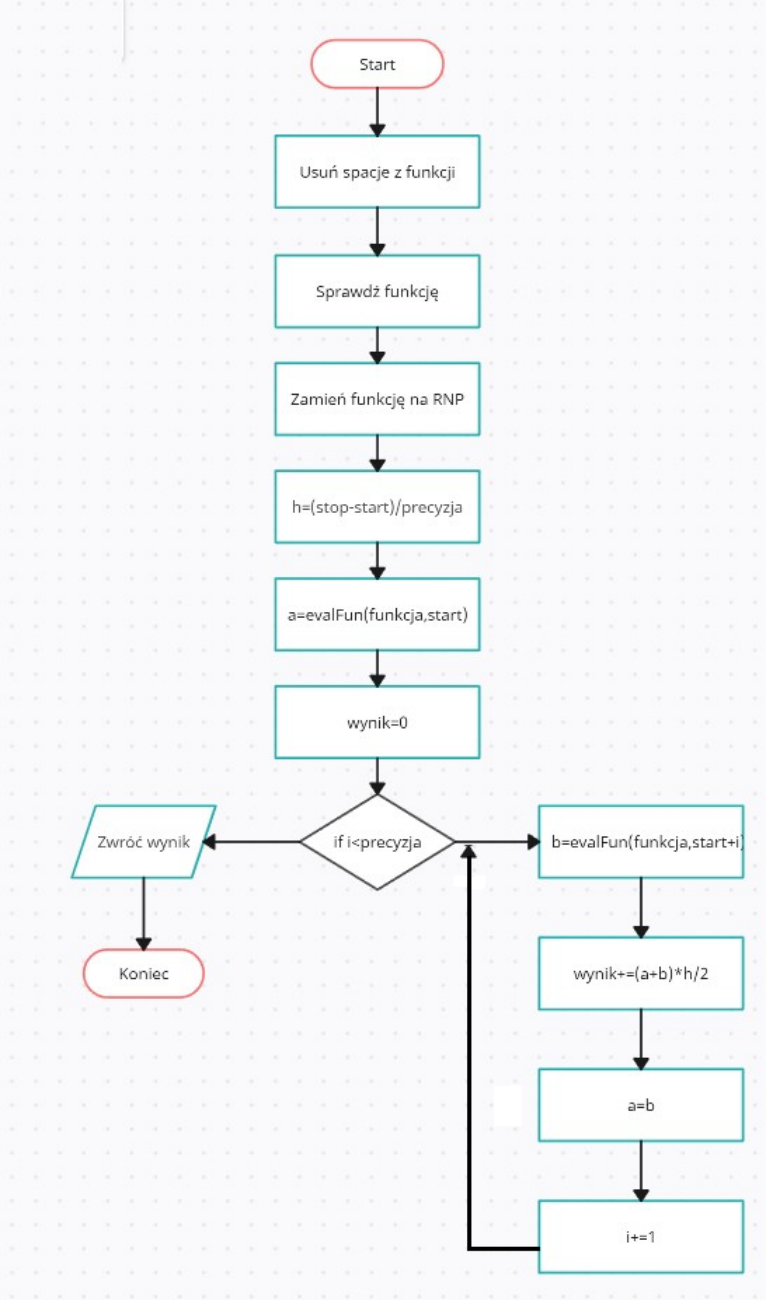


Figure 5: Schemat blokowy calculateIntegral()

7 Kod całego programu

Listing 6: Caly program

```
Program liczacy calke oznaczona zlozona metoda trapezow
#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;
    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)){
            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"){
            result+=curr+" ";
        }
        else if(isOperator(curr)){
            if(ostack.empty()||curr=="(") ostack.push(curr);
            else{
                while(!ostack.empty()&&rnpPrio(curr)<=rnpPrio(ostack.top())&&ost
                    result+=ostack.top()+" ";
                    ostack.pop();
                }
                if(curr==")") ostack.pop();
                else ostack.push(curr);
            }
        }
    }
    while(!ostack.empty()){
        result+=ostack.top()+" ";
        ostack.pop();
    }
    return result;
}
double calculateIntegral(double start, double stop, int precision, string equati
    string RPNV=RPN(equation);
    double result=0,width,a=evalFun(RPNV,start),b;
    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;
}

int initInterface(){
    while(true){
        cout<<"Program licz cy ca k oznaczon z o on metod trapez w\
        int choice,schoice,precision;
        double start,stop,result;
        string input;
        cin>>choice;
        if(choice==1){
            cout<<"Wpisz funkcje kt r chcesz obliczy "<<endl;
            cin.ignore();
            getline(cin,input);
            if(!checkParentheses(input)) cout<<"B dne dane(niedomkni ty/nieo
```

```
            else{
            cout<<"Wpisz pocz tek przedzia u"<<endl;
            cin>>start;
            cout<<"Wpisz koniec przedzia u"<<endl;
            cin>>stop;
            if(start>stop){
                cout<<"Start przedzia u jest wi kszy od ko ca czy chcesz zam
                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;
                }

            }
            }
        }else if(choice==2){
            ifstream test("test.txt");
            string teststr,descstr;
            while(getline(test,teststr)&&getline(test,descstr)){
                cout<<"Funkcja: "<<teststr<<endl<<"Jej ca ka wynosi(na przedzia
            }
            test.close();
        }else if(choice==3){
            break;
        }else{
            cout<<"Niepoprawny wyb r"<<endl;
        }
    }
    return 0;
}
int main() {
    system("chcp 1250>>null");
    setlocale(LC_CTYPE, "Polish");
    initInterface();
```

```
    return 0;
}
```