

Dawid Pszczółkowski
 Adrian Smykowski
 Paweł Sawicki

Zadanie 1.9

Ustalić naturalną n_{\max} . Wczytać $n \in \{0, 1, \dots, n_{\max}\}$ oraz różne węzły x_0, x_1, \dots, x_n i dowolne wartości A_0, A_1, \dots, A_n . Wyznaczyć w postaci Newtona wielomian interpolacyjny $P=P(x)$ taki, że $P(x_i)=A_i$ dla $i=0, 1, \dots, n$. Następnie, „dopóki użytkownik się nie znudzi”, wczytywać $j \in \{0, 1, \dots, n\}$ oraz t należące do R i obliczać wartość $P^{(j)}(t)$.

Postać Newtona wielomianu:

$$w(x) = \sum_{i=0}^n a_i \prod_{j=0}^{i-1} (x - x_j)$$

Wyznaczanie współczynników równań dzielonych:

$$f[x_i, \dots, x_{i+j+1}] = \frac{f[x_{i+1}, \dots, x_{i+j+1}] - f[x_i, \dots, x_{i+j}]}{x_{i+j+1} - x_i}$$

x_i	$f(x_i)$	$f[x_{i-1}, x_i]$	$f[x_{i-2}, x_{i-1}, x_i]$	\cdots	$f[x_{i-n}, \dots, x_i]$
x_0	$f(x_0)$				
x_1	$f(x_1)$	$f[x_0, x_1]$			
x_2	$f(x_2)$	$f[x_1, x_2]$	$f[x_0, x_1, x_2]$		
\vdots	\vdots	\vdots	\vdots	\ddots	
x_n	$f(x_n)$	$f[x_{n-1}, x_n]$	$f[x_{n-2}, x_{n-1}, x_n]$	\cdots	$f[x_0, \dots, x_n]$

Sprowadzenie wielomianu do postaci ogólnej.
 Obliczanie pochodnej j-tego rzędu rekurencyjnie

$$P^{(j)}(x) = (f^{(j-1)}(x))'$$

Podstawianie t do wzoru pochodnej i obliczanie.

*Wstępny opis algorytmu:

1. Pobieranie od użytkownika wartości n i kontrola.
 2. Wczytywanie wartości A_0, A_1, \dots, A_n
 3. Wczytywanie węzłów i kontrola unikalności każdego z węzłów X_0, X_1, \dots, X_n .
 4. Wypisywanie podanych punktów.
 5. Obliczanie współczynników i wypis ich.
 6. Wypisywanie wielomianu w postaci interpolacyjnej Newtona
 7. Sprawdzenie wielomianu do postaci ogólnej i wypisywanie go.
 8. Sprawdzenie chęci użytkownika do obliczenia pochodnej. Jeżeli użytkownik wpisze słowo POCHODNA następuje zapytanie o rząd pochodnej i wartość t od której program obliczy daną pochodną wypisując wyniki.
- Jeżeli użytkownik wpisze KONIEC nastąpi zakończenie programu.

*Przykład

$n_{\max}=2$

$A=\{2,5,7\}$

$X=\{0,1,-1\}$

Liczenie różnic dzielonych

$5-2/1-0=3$

$7-5/-1-1=-1$

$-1-3/-1-0=4$

Wyznaczanie współczynników 2,3,4

Wyznaczanie z wielomianu w postaci interpolacyjnej Newtona wielomianu w postaci ogólnej

$$2+3(x-0)+4(x-0)(x-1)=2+3x+4x^2-4x=4x^2-x+2$$

$$4x^2-x+2$$

$$P'=8x-1$$

$$P'(3)=23$$

$$P''=8$$

$$P''(3)=8$$

*Opis najważniejszych struktur ,funkcji , procedur

1.Struktura wielomianu potrzebna do sprowadzenia wielomianu w postaci interpolacyjnej do postaci ogólnej

```
struct wielomian{
double wspolczynniki;
int stopien_zm;
};
```

2.Wczytywanie n_max oraz sprawdzanie czy podana wartosc jest prawdziwa (wieksza od 0 i mniejsza od n podanego w programie)

```
int n_max;
while(1){
    cout<<"Podaj n_max dla jakiego bedzie dzialal program,\nn_max powinno byc
    wieksze od 0 a mniejsze od "<<n<<endl;
    cout<<"n_max :";
    cin>>n_max;
    if(n_max>0 && n_max<=n) break;
    else cout<<"BLAD!!! - zle podane n_max."<<endl;
    cout<<"\n";
    system("PAUSE");
    cout<<"\n";
}
```

3.Wczytywanie węzłów i sprawdzanie ich unikalności

```
int unikalnosc=1;
cout<<"Wczytywanie wezlow dla X_0,X_1,...,X_"<<n_max<<endl;
for(i=0;i<=n_max;i++){
    cout<<"Wczytaj X_"<<i<<": ";
    cin>>temp;
    j=0;
    while(j<=i){
        if(X[j]==temp){
            cout<<"Podany wezel juz istnieje !! - wezly powinny byc
            unikalne."<<endl;
            unikalnosc=0;
            break;
        }
        j++;
    }
    if(unikalnosc){
        X[i]=temp;
        i++;
    }
    unikalnosc=1;
}
```

4.Obliczanie współczynników

```
for(i=0;i<=n_max;i++){
    wspolczynniki[i][0]=A[i];
}

for(i=1;i<=n_max;i++){
    for(j=1;j<=i;j++){
        wspolczynniki[i][j]=(wspolczynniki[i][j-1]-wspolczynniki[i-1][j-1])/
        (X[i]-X[i-j]);
    }
}
```

5.Deklaracja dynamicznych tablic

-nawiasy do przechowywania nawiasow

-2-wymiarowej tablicy wymnozone nawiasy do przechowywania wymnozonych nawiasow
-postac_ogolna do przechowywania wielomianu w postaci ogolnej uporządkowanej
-posredni do zapisania wszystkich nieuporządkowanych wartosci

```
wielomian *nawiasy= new wielomian[2*(n_max+1)];

wielomian *postac_ogolna = new wielomian[n_max+1];

wielomian *posredni=new wielomian[pow(2,n_max+1)+1];

wielomian **wymnozone_nawiasy = new wielomian *[n_max+1];
    for (j=1,i=0;i<n_max+1; i++,j++){
        wymnozone_nawiasy[i] = new wielomian [pow(2,j)];
```

6.Mnozenie poszczegolnych nawiasow

```
int nawiasowa=2;
wymnozone_nawiasy[0][0].wspolczynnik=nawiasy[0].wspolczynnik;
wymnozone_nawiasy[0][0].stopien_zm=nawiasy[0].stopien_zm;
wymnozone_nawiasy[0][1].wspolczynnik=nawiasy[1].wspolczynnik;
wymnozone_nawiasy[0][1].stopien_zm=nawiasy[1].stopien_zm;

    for (j=2,i=1; i < n_max+1; i++,j++){
        int z=0;
        k=0;
        for(;k<pow(2,j-1);z++,k++){
            wymnozone_nawiasy[i]
[z].wspolczynnik=nawiasy[nawiasowa].wspolczynnik*wymnozone_nawiasy[i-1][k].wspolczynnik;
            wymnozone_nawiasy[i]
[z].stopien_zm=nawiasy[nawiasowa].stopien_zm+wymnozone_nawiasy[i-1][k].stopien_zm;
        }
        nawiasowa++;
        k=0;
        for(;k<pow(2,j-1);z++,k++){
            wymnozone_nawiasy[i]
[z].wspolczynnik=nawiasy[nawiasowa].wspolczynnik*wymnozone_nawiasy[i-1][k].wspolczynnik;
            wymnozone_nawiasy[i]
[z].stopien_zm=nawiasy[nawiasowa].stopien_zm+wymnozone_nawiasy[i-1][k].stopien_zm;
        }
        nawiasowa++;
    }
```

7.Mnozenie nawiasow przez wspolczynniki

```
    for (j=1,i=0,k=1; i < n_max; i++,j++,k++ ){
        for(int z=0;z<pow(2,j);z++){
            wymnozone_nawiasy[i][z].wspolczynnik*=wspolczynniki[k][k];
        }
    }
```

8.Sprowadzanie wielomianu do postaci ogolnej

a)Najpierw do tablicy posredni wpisujemy wszystkie wyrazy

```
int z=1;
posredni[0].wspolczynnik=wspolczynniki[0][0];
posredni[0].stopien_zm=0;
    for(j=1,i=0;i<n_max+1;i++,j++){
        for (int w=0;w<pow(2,j)&& z<pow(2,n_max+1)+1;w++,z++){
            posredni[z].wspolczynnik=wymnozone_nawiasy[i][w].wspolczynnik;
            posredni[z].stopien_zm=wymnozone_nawiasy[i][w].stopien_zm;
        }
    }
```

b)Porządkowanie wyrazow do tablicy postac_ogolna

```
int najwyzsza_pot=n_max;
    for(i=0;i<n_max+1;i++){
```

```

        postac_ogolna[i].stopien_zm=najwyzsza_pot;
        for(j=0;j<pow(2,n_max+1)+1;j++){
            if(posredni[j].stopien_zm==najwyzsza_pot){
                postac_ogolna[i].wspolczynnik+=posredni[j].wspolczynnik;
            }
        }
        najwyzsza_pot--;
    }
}

```

9. Wypis postaci ogólnej

```

    najwyzsza_pot=n_max;
    cout<<"Wielomian w postaci ogólnej:"<<endl;
    for(i=0;i<n_max+1;i++){
        if(postac_ogolna[i].stopien_zm>0){
            if(postac_ogolna[i].stopien_zm==1){
                if(postac_ogolna[i].wspolczynnik==1) cout<<"x";
                else{
                    if(postac_ogolna[i].wspolczynnik==-1)cout<<"-x";
                    else {
                        if(postac_ogolna[i].wspolczynnik>0)
                            cout<<"+"<<postac_ogolna[i].wspolczynnik<<"x";
                        else cout<<"-"<<-
                            postac_ogolna[i].wspolczynnik<<"x";
                    }
                }
            }
            else{
                if(postac_ogolna[i].wspolczynnik>0 &&
                    postac_ogolna[i].stopien_zm==najwyzsza_pot)cout<<postac_ogolna[i].wspolczynnik<<"x^"<<postac_ogolna[i].stopien_zm;
                else {
                    if(postac_ogolna[i].wspolczynnik<0) cout<<"-"<<-
                        postac_ogolna[i].wspolczynnik<<"x^"<<postac_ogolna[i].stopien_zm;
                    else
                        cout<<"+"<<postac_ogolna[i].wspolczynnik<<"x^"<<postac_ogolna[i].stopien_zm;
                }
            }
        }
        else {
            if(postac_ogolna[i].wspolczynnik>0)cout<<"+"<<postac_ogolna[i].wspolczynnik<<endl;
            else cout<<postac_ogolna[i].wspolczynnik<<endl;
        }
    }
    cout<<"\n";
    system("PAUSE");
    cout<<"\n";

```

10. Pętla sprawdzająca warunek do dalszego działania programu

```

while(1){
    cout<<"Wpisz POCHODNA -- by obliczyc pochodna wielomianu stopnia j od
    liczby t."<<endl;
    cout<<"Wpisz KONIEC -- by zakonczyc program."<<endl;
    cout<<"Twój wybór ->";
    cin>>warunek;
    if(strcmp(warunek,"KONIEC")==0) break;
    else{
        if(strcmp(warunek,"POCHODNA")!=0){
            cout<<"BLAD !! - ZLE PODANY WARUNEK DZIAŁANIA PROGRAMU."<<endl;
            cout<<"\n";
            system("PAUSE");
            cout<<"\n";
            //system("CLS");
        }
    }
}

```

```

        else{
            cout<<"Program obliczy pochodna wielomianu stopnia j od liczby
t."<<endl;

            cout<<"Podaj j (j powinno byc z zakresu od 0 do "<<n_max<<") :";
            cin>>j;
            if(j<0 || j>n_max){
                //system("CLS");
                cout<<"BLAD!! - ZLE PODANE j.\nTWOJE j POWINNO BYC Z
ZAKRESU OD 0 DO "<<n_max<<endl;

                cout<<"\n";
                system("PAUSE");
                cout<<"\n";
                //system("CLS");
                continue;
            }
            cout<<"Podaj t:";
            cin>>t;
            pochodna(postac_ogolna,j,t,n_max);
        }
    }
}

```

11.Funkcja liczaca pochodna rzadanego rzędu i wypisujaca wynik

```
void pochodna(wielomian *tab,int j, double t, int wielkosc){
```

```
    wielomian *kopia= new wielomian[wielkosc];
```

```
    cout<<"Pochodna rzędu "<<j<<": "<<endl;
```

```
    int stopien=j;
```

a)Kopiowanie tablicy postac_ogolna do kopii

```
    for(int i=0;i<wielkosc+1;i++) kopia[i]=tab[i];
```

b)Liczenie pochodnej

```
    while(j--){
```

```
        for(int i=0;i<wielkosc+1;i++){
```

```
            kopia[i].wspolczynnik*=kopia[i].stopien_zm;
```

```
            if(kopia[i].stopien_zm!=0) kopia[i].stopien_zm--;
```

```
        }
```

```
    }
```

```
    wielkosc-=stopien;
```

c)Wypis obliczonej pochodnej

```
    for(int i=0;i<wielkosc+1;i++){
```

```
        if(kopia[i].stopien_zm>0){
```

```
            if(kopia[i].stopien_zm==1){
```

```
                if(kopia[i].wspolczynnik==1) cout<<"x";
```

```
            else{
```

```
                if(kopia[i].wspolczynnik==-1)cout<<"-x";
```

```
            else {
```

```
                if(kopia[i].wspolczynnik>0)
```

```
                cout<<"+"<<kopia[i].wspolczynnik<<"x";
```

```
                else cout<<"-"<<-kopia[i].wspolczynnik<<"x";
```

```
            }
```

```
        }
```

```
    }
```

```
    else{
```

```
        if(kopia[i].wspolczynnik>0)cout<<kopia[i].wspolczynnik<<"x^"<<kopia[i].stopien_zm;
```

```
        else {
```

```
            if(kopia[i].wspolczynnik<0) cout<<"-"<<-
```

```
kopia[i].wspolczynnik<<"x^"<<kopia[i].stopien_zm;
```

```
            else
```

```
            cout<<"+"<<kopia[i].wspolczynnik<<"x^"<<kopia[i].stopien_zm;
```

```
        }
```

```

    }
}
else {
    if(kopia[i].wspolczynnik>0)cout<<"+"<<kopia[i].wspolczynnik<<endl;
    else cout<<kopia[i].wspolczynnik<<endl;
}
}

```

d) Liczenie pochodnej danego rzędu od wartości t

```

double wynik=0;
for(int i=0;i<wielkosc+1;i++){
    wynik += pow(t,kopia[i].stopien_zm)*kopia[i].wspolczynnik;
}
cout<<"P^"<<stopien<<"("<<t<<" )="<<wynik<<endl;
cout<<"\n";
}

```

*Opis „wejścia-wyjścia”

Użytkownik podaje n_max , którego w pętli jest sprawdzana poprawność by nie było mniejsze od 0 i większe od n zadeklarowanego w programie.

Następnie użytkownik podaje n_max wartości i węzłów . Po czym na wyjściu otrzymuje zbiór punktów jakie wybrał, program w pętli sprawdza unikalność węzłów.

Program oblicza z podanych wartości współczynniki oraz je wyświetla.

Na wyjściu użytkownik otrzymuje również wielomian w postaci interpolacyjnej Newtona oraz w postaci ogólnej.

Kolejnym wejściem jakie podaje użytkownik są słowa określające warunek działania programu, POCHODNA – program wtedy oblicza pochodną zadanego rzędu j (większego od 0 a mniejszego od n_max) które podaje użytkownik od wartości t również podawanej przez użytkownika. Na wyjściu zostaje wyświetlona pochodna danego rzędu jak i wynik $P^{(j)}(t)$.

W wypadku wpisania przez użytkownika słowa Koniec program kończy swą pracę. Program sprawdza poprawność wpisywanych słów jeżeli wystąpi błąd zwróci komunikat o błędzie i poprosi o ponowne podanie warunku działania programu.

*Kod źródłowy

```

/*****AUTORZY*****/
//Dawid Pszczolkowski
//Adrian Smykowski
//Pawel Sawicki

/*****TRESC_ZADANIA*****/
//ZADANIE 1.9
//Ustalic naturalna n_max. Wczytac n nalezace do przedzialu {0,1...,n_max} oraz rozne
//wezly x_0,x_1,...x_n i dowolne wartosci A_0,A_1,...A_n. Wyznaczyc w postaci Newtona
//wielomian interpolacyjny P=P(x) taki, ze P(x_i)=A_i dla i=0,1,...n. Nastepnie
//"dopoki uzytkownik sie nie znudzi",wczytywac j nalezace do przedzialu {0,1,...n} oraz t
//nalezace do R
//i obliczac wartosc P^(j)(t).

#include<iostream>
#include<string.h>
#include<math.h>

using namespace std;

const int n=20; //ustalone n_max;

//Struktura wielomianu potrzebna do sprowadzenia wielomianu w postaci interpolacyjnej do
//postaci ogolnej
struct wielomian{
double wspolczynnik;
int stopien_zm;
};

int pow(int base,int w);

void pochodna(wielomian *tab,int j, double t, int wielkosc);

int main(int argc, char* argv[]){
int i,j;
double t;
//Wczytywanie n_max
int n_max;
while(1){
cout<<"Podaj n_max dla jakiego bedzie dzialal program,\nn_max powinno byc
wieksze od 0 a mniejsze od "<<n<<endl;
cout<<"n_max :";
cin>>n_max;
if(n_max>0 && n_max<=n) break;
else cout<<"BLAD!!! - zle podane n_max."<<endl;
cout<<"\n";
system("PAUSE");
cout<<"\n";
//system("CLS");
}

//Deklaracje dynamicznych tablic dla X_0 => X_n_max i A_0 => A_n_max oraz wczytywanie tych
liczb
double *A =new double[n_max+1];
double *X= new double[n_max+1];
cout<<"Wczytywanie wartosci dla A_0,A_1,...,A_"<<n_max<<endl;
for(i=0;i<=n_max;i++){
cout<<"Wczytaj A_"<<i<<": ";
cin>>A[i];
}
```



```

}
double temp;
int unikalnosc=1;
cout<<"Wczytywanie wezlow dla X_0,X_1,...,X_"<<n_max<<endl;
for(i=0;i<=n_max;){
    cout<<"Wczytaj X_"<<i<<": ";
    cin>>temp;
    j=0;
    while(j<=i){
        if(X[j]==temp){
            cout<<"Podany wezel juz istnieje !! - wezly powinny byc
unikalne."<<endl;
            unikalnosc=0;
            break;
        }
        j++;
    }
    if(unikalnosc){
        X[i]=temp;
        i++;
    }
    unikalnosc=1;
}
//system("CLS");
//Wypisywanie kontrolne podanych liczb
for(int i=0;i<=n_max;i++){
    cout<<"(X_"<<i<<","A_"<<i<<")=("<<X[i]<<","<<A[i]<<")"<<endl;
}
cout<<"\n";
system("PAUSE");
cout<<"\n";
//system("CLS");

/*****OBLICZANIE
WSPOLCZYNNIKOW*****/

//Tablica wspolczynniki
double wspolczynniki[n+1][n+1];

//Wypelnianie tablicy wspolczynniki zerami
for(i=0;i<=n_max;i++){
    for(int j=0;j<=n_max;j++){
        wspolczynniki[i][j]=0;
    }
}

//Obliczanie wspolczynniki
for(i=0;i<=n_max;i++){
    wspolczynniki[i][0]=A[i];

    for(i=1;i<=n_max;i++){
        for(j=1;j<=i;j++){
            wspolczynniki[i][j]=(wspolczynniki[i][j-1]-wspolczynniki[i-1][j-1])/
(X[i]-X[i-j]);
        }
    }

}

//Wypisywanie wspolczynniki
cout<<"Wspolczynniki:"<<endl;
for(i=0;i<=n_max;i++){
    cout<<wspolczynniki[i][i]<<" ";
}
cout<<"\n";

```

```

cout<<"\n";
system("PAUSE");
cout<<"\n";
//system("CLS");

//Wypisywanie wielomianu w postaci Newtona i dodawanie współczynników i stopni zmiennych do
tablicy W
cout<<"Wielomian w postaci interpolacyjnej Newtona:"<<endl;
cout<<wspolczynniki[0][0];
int k=1,c=1;
while(k<n_max+1){
if(wspolczynniki[k][k]<0)cout<<"-"<<-wspolczynniki[k][k];
else cout<<"+"<<wspolczynniki[k][k];
for(i=0;i<k;i++){
if(X[i]<0) cout<<"(x+"<<-X[i]<<")";
else cout<<"(x-"<<X[i]<<")";
}
k+=1;
}
cout<<endl;
cout<<"\n";
system("PAUSE");
cout<<"\n";
//system("CLS");

/*****TWORZENIE_WIELOMIANU
_W_POSTACI_OGOLNEJ*****/

//Deklaracja tablicy nawias do przechowywania nawiasów i 2-wymiarowej tablicy
wymnozone_nawiasy do przechowywania wymnożonych nawiasów
wielomian *nawiasy= new wielomian[2*(n_max+1)];

wielomian *postac_ogolna = new wielomian[n_max+1];

wielomian *posredni=new wielomian[pow(2,n_max+1)+1];

wielomian **wymnozone_nawiasy = new wielomian *[n_max+1];
for (j=1,i=0;i<n_max+1; i++,j++)
    wymnozone_nawiasy[i] = new wielomian [pow(2,j)];

//Wypełnianie tablicy posredni początkowymi wartościami
for(i=0;i<pow(2,n_max+1)+1;i++){
    posredni[i].wspolczynnik=0;
    posredni[i].stopien_zm=0;
}

//Wypełnianie tablicy postac_ogolna początkowymi wartościami
for(i=0;i<n_max+1;i++){
    postac_ogolna[i].wspolczynnik=0;
    postac_ogolna[i].stopien_zm=0;
}

//Wypełnianie tablicy nawiasy początkowymi wartościami
for(i=0;i<2*(n_max);i++){
    nawiasy[i].wspolczynnik=1;
    nawiasy[i].stopien_zm=0;
}

//Wypełnianie 2-wymiarowej tablicy wymnozone_nawiasy początkowymi wartościami
for (j=1,i=0; i < n_max; i++,j++ )
    for(int z=0;z<pow(2,j);z++){
        wymnozone_nawiasy[i][z].wspolczynnik=1;
        wymnozone_nawiasy[i][z].stopien_zm=0;
    }
}

```

```

//Wypelnianie tablicy nawias odpowiednimi wartosciami
for(j=0,i=0;i<2*(n_max+1);i++){
    if(i%2==0){
        nawiasy[i].stopien_zm=1;
    }
    else{
        if(X[j]==0) nawiasy[i].wspolczynnik=0;
        else
            nawiasy[i].wspolczynnik=-X[j];
        j++;
    }
}

//Mnozenie poszczegolnych nawiasow
int nawiasowa=2;
wymnozone_nawiasy[0][0].wspolczynnik=nawiasy[0].wspolczynnik;
wymnozone_nawiasy[0][0].stopien_zm=nawiasy[0].stopien_zm;
wymnozone_nawiasy[0][1].wspolczynnik=nawiasy[1].wspolczynnik;
wymnozone_nawiasy[0][1].stopien_zm=nawiasy[1].stopien_zm;

    for (j=2,i=1; i < n_max+1; i++,j++){
        int z=0;
        k=0;
        for(;k<pow(2,j-1);z++,k++){
            wymnozone_nawiasy[i]
[z].wspolczynnik=nawiasy[nawiasowa].wspolczynnik*wymnozone_nawiasy[i-1][k].wspolczynnik;
            wymnozone_nawiasy[i]
[z].stopien_zm=nawiasy[nawiasowa].stopien_zm+wymnozone_nawiasy[i-1][k].stopien_zm;
        }
        nawiasowa++;
        k=0;
        for(;k<pow(2,j-1);z++,k++){
            wymnozone_nawiasy[i]
[z].wspolczynnik=nawiasy[nawiasowa].wspolczynnik*wymnozone_nawiasy[i-1][k].wspolczynnik;
            wymnozone_nawiasy[i]
[z].stopien_zm=nawiasy[nawiasowa].stopien_zm+wymnozone_nawiasy[i-1][k].stopien_zm;
        }
        nawiasowa++;
    }

//Mnozenie nawiasow przez wspolczynniki
for (j=1,i=0,k=1; i < n_max; i++,j++,k++){
    for(int z=0;z<pow(2,j);z++){
        wymnozone_nawiasy[i][z].wspolczynnik*=wspolczynniki[k][k];
    }
}

//Sprawdzanie wielomianu do postaci ogolnej
//Najpierw do tablicy posredni wpisujemy wszystkie wyrazy
int z=1;
posredni[0].wspolczynnik=wspolczynniki[0][0];
posredni[0].stopien_zm=0;
for(j=1,i=0;i<n_max+1;i++,j++){
    for (int w=0;w<pow(2,j)&& z<pow(2,n_max+1)+1;w++,z++){
        posredni[z].wspolczynnik=wymnozone_nawiasy[i][w].wspolczynnik;
        posredni[z].stopien_zm=wymnozone_nawiasy[i][w].stopien_zm;
    }
}

//Pozadkowanie wyrazow do tablicy postac_ogolna
int najwyzsza_pot=n_max;
for(i=0;i<n_max+1;i++){
    postac_ogolna[i].stopien_zm=najwyzsza_pot;
    for(j=0;j<pow(2,n_max+1)+1;j++){

```

```

        if(posredni[j].stopien_zm==najwyzsza_pot){
            postac_ogolna[i].wspolczynnik+=posredni[j].wspolczynnik;
        }
    }
    najwyzsza_pot--;
}

//Wypis postaci ogolnej
najwyzsza_pot=n_max;
cout<<"Wielomian w postaci ogolnej:"<<endl;
for(i=0;i<n_max+1;i++){
    if(postac_ogolna[i].stopien_zm>0){
        if(postac_ogolna[i].stopien_zm==1){
            if(postac_ogolna[i].wspolczynnik==1) cout<<"x";
            else{
                if(postac_ogolna[i].wspolczynnik==-1)cout<<"-x";
                else {
                    if(postac_ogolna[i].wspolczynnik>0)
cout<<"+"<<postac_ogolna[i].wspolczynnik<<"x";
                    else cout<<"-"<<-
postac_ogolna[i].wspolczynnik<<"x";
                }
            }
        }
        else{
            if(postac_ogolna[i].wspolczynnik>0 &&
postac_ogolna[i].stopien_zm==najwyzsza_pot)cout<<postac_ogolna[i].wspolczynnik<<"x^"<<postac
_ogolna[i].stopien_zm;
            else {
                if(postac_ogolna[i].wspolczynnik<0) cout<<"-"<<-
postac_ogolna[i].wspolczynnik<<"x^"<<postac_ogolna[i].stopien_zm;
                else
cout<<"+"<<postac_ogolna[i].wspolczynnik<<"x^"<<postac_ogolna[i].stopien_zm;
            }
        }
    }
    else {
        if(postac_ogolna[i].wspolczynnik>0)cout<<"+"<<postac_ogolna[i].wspolczynnik<<endl;
        else cout<<postac_ogolna[i].wspolczynnik<<endl;
    }
}

cout<<"\n";
system("PAUSE");
cout<<"\n";
/
*****RÓŻNICZKOWANIE*****
*****/
char warunek[9];
while(1){
    cout<<"Wpisz POCHODNA -- by obliczyc pochodna wielomianu stopnia j od
liczby t."<<endl;
    cout<<"Wpisz KONIEC -- by zakonczyc program."<<endl;
    cout<<"Twoj wybor ->";
    cin>>warunek;
    if(strcmp(warunek,"KONIEC")==0) break;
    else{
        if(strcmp(warunek,"POCHODNA")!=0){
            cout<<"BLAD !! - ZLE PODANY WARUNEK DZIALANIA PROGRAMU."<<endl;
            cout<<"\n";
            system("PAUSE");
            cout<<"\n";
            //system("CLS");
        }
    }
}

```



```

cout<<"+"<<kopia[i].wspolczynnik<<"x";
                                }
                                }
                                }
                                else{
                                if(kopia[i].wspolczynnik>0)cout<<kopia[i].wspolczynnik<<"x^"<<kopia[i].stopien_zm;
                                else {
                                if(kopia[i].wspolczynnik<0) cout<<"-"<<-
kopia[i].wspolczynnik<<"x^"<<kopia[i].stopien_zm;
                                else
                                cout<<"+"<<kopia[i].wspolczynnik<<"x^"<<kopia[i].stopien_zm;
                                }
                                }
                                }
                                else {
                                if(kopia[i].wspolczynnik>0)cout<<"+"<<kopia[i].wspolczynnik<<endl;
                                else cout<<kopia[i].wspolczynnik<<endl;
                                }
                                }

//Liczenie pochodnej danego rzędu od wartosi t
double wynik=0;
for(int i=0;i<wielkosc+1;i++){
    wynik += pow(t,kopia[i].stopien_zm)*kopia[i].wspolczynnik;
}
cout<<"P^"<<stopien<<"("<<t<<"")="<<wynik<<endl;
cout<<"\n";
}

```

*Przykłady wywołań programu:

1.Pobieranie od użytkownika n i kontrola.

```
D:\Projekty\MetodyObliczeniowe\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe

Podaj n_max dla jakiego bedzie dzialal program,
n_max powinno byc wieksze od 0 a mniejsze od 20
n_max :-3
BLAD!!! - zle podane n_max.

Aby kontynuować, naciśnij dowolny klawisz . . .

Podaj n_max dla jakiego bedzie dzialal program,
n_max powinno byc wieksze od 0 a mniejsze od 20
n_max :21
BLAD!!! - zle podane n_max.

Aby kontynuować, naciśnij dowolny klawisz . . .

Podaj n_max dla jakiego bedzie dzialal program,
n_max powinno byc wieksze od 0 a mniejsze od 20
n_max :4
Wczytywanie wartosci dla A_0,A_1,...,A_4
Wczytaj A_0:
```

2. Wczytywanie wartości A_0, A_1, \dots, A_n
3. Wczytywanie węzłów i kontrola unikalności każdego z węzłów X_0, X_1, \dots, X_n .
4. Wypisywanie podanych punktów.

```
D:\Projekty\MetodyObliczeniowe\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe

Podaj n_max dla jakiego bedzie dzialal program,
n_max powinno byc wieksze od 0 a mniejsze od 20
n_max :3
Wczytywanie wartosci dla A_0,A_1,...,A_3
Wczytaj A_0: 2
Wczytaj A_1: 5
Wczytaj A_2: 7
Wczytaj A_3: 4
Wczytywanie wezlow dla X_0,X_1,...,X_3
Wczytaj X_0: 0
Wczytaj X_1: 0
Podany wezel juz istnieje ?? - wezly powinny byc unikalne.
Wczytaj X_1: 1
Wczytaj X_2: -1
Wczytaj X_3: 2
<X_0,A_0>=<0,2>
<X_1,A_1>=<1,5>
<X_2,A_2>=<-1,7>
<X_3,A_3>=<2,4>

Aby kontynuować, naciśnij dowolny klawisz . . . _
```

5. Obliczanie współczynników i wypis ich.

```
D:\Projekty\MetodyObliczeniowe\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe
n_max powinno byc wieksze od 0 a mniejsze od 20
n_max :3
Wczytywanie wartosci dla A_0,A_1,...,A_3
Wczytaj A_0: 2
Wczytaj A_1: 5
Wczytaj A_2: 7
Wczytaj A_3: 4
Wczytywanie wezlow dla X_0,X_1,...,X_3
Wczytaj X_0: 0
Wczytaj X_1: 0
Podany wezel juz istnieje !! - wezly powinny byc unikalne.
Wczytaj X_1: 1
Wczytaj X_2: -1
Wczytaj X_3: 2
<X_0,A_0>=<0,2>
<X_1,A_1>=<1,5>
<X_2,A_2>=<-1,7>
<X_3,A_3>=<2,4>

Aby kontynuować, naciśnij dowolny klawisz . . .

Wspolczynniki:
2 3 4 -2

Aby kontynuować, naciśnij dowolny klawisz . . .
```

6. Wypisywanie wielomianu w postaci interpolacyjnej Newtona


```
D:\Projekty\MetodyObliczeniowe\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe
Wczytaj A_2: 7
Wczytaj A_3: 4
Wczytywanie wezlow dla X_0,X_1,...,X_3
Wczytaj X_0: 0
Wczytaj X_1: 0
Podany wezel juz istnieje !! - wezly powinny byc unikalne.
Wczytaj X_1: 1
Wczytaj X_2: -1
Wczytaj X_3: 2
<X_0,A_0>=<0,2>
<X_1,A_1>=<1,5>
<X_2,A_2>=<-1,7>
<X_3,A_3>=<2,4>

Aby kontynuować, naciśnij dowolny klawisz . . .

Wspolczynniki:
2 3 4 -2

Aby kontynuować, naciśnij dowolny klawisz . . .

Wielomian w postaci interpolacyjnej Newtona:
2+3<x-0>+4<x-0><x-1>-2<x-0><x-1><x+1>

Aby kontynuować, naciśnij dowolny klawisz . . .
```

7.Sprowadzenie wielomianu do postaci ogólnej i wypisywanie go.

```
D:\Projekty\MetodyObliczeniowe\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe
Podany wezel juz istnieje !! - wezly powinny byc unikalne.
Wczytaj X_1: 1
Wczytaj X_2: -1
Wczytaj X_3: 2
<X_0,A_0>=<0,2>
<X_1,A_1>=<1,5>
<X_2,A_2>=<-1,7>
<X_3,A_3>=<2,4>

Aby kontynuować, naciśnij dowolny klawisz . . .

Wspolczynniki:
2 3 4 -2

Aby kontynuować, naciśnij dowolny klawisz . . .

Wielomian w postaci interpolacyjnej Newtona:
2+3<x-0>+4<x-0><x-1>-2<x-0><x-1><x+1>

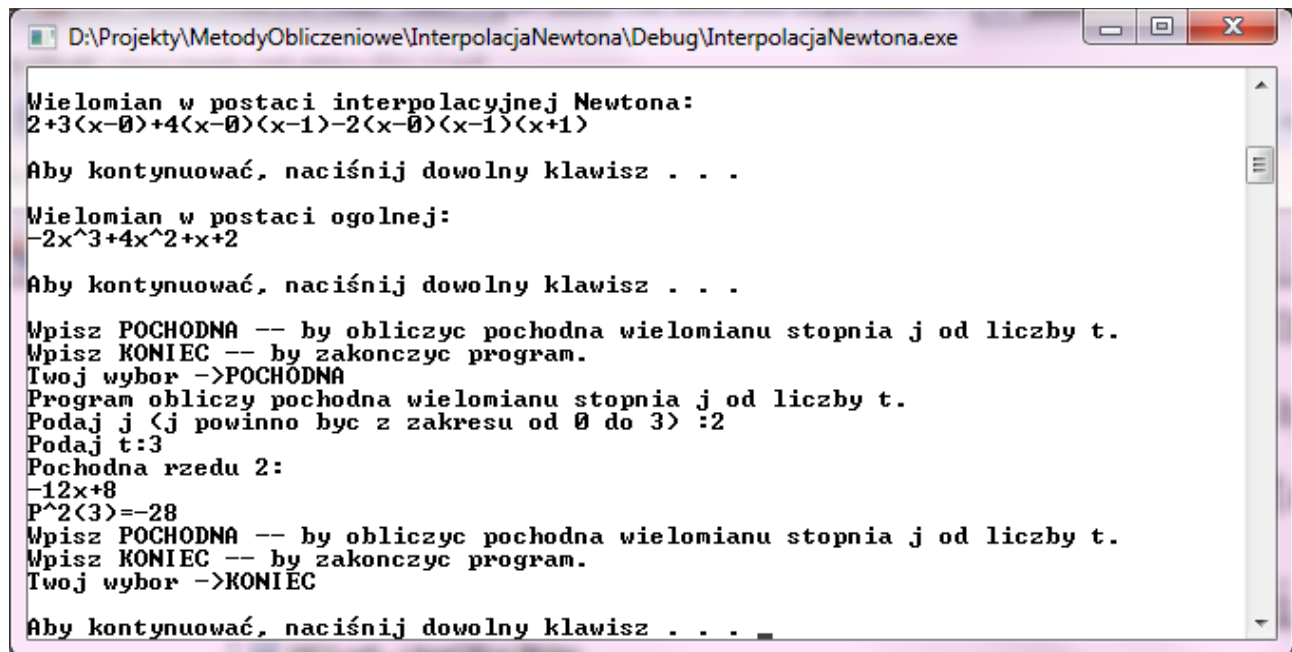
Aby kontynuować, naciśnij dowolny klawisz . . .

Wielomian w postaci ogólnej:
-2x^3+4x^2+x+2

Aby kontynuować, naciśnij dowolny klawisz . . .
```

8.Sprawdzenie chęci użytkownika do obliczenia pochodnej. Jeżeli użytkownik wpisze słowo POCHODNA następuje zapytanie o rząd pochodnej i wartość t od której program obliczy daną pochodną wypisując wyniki.

Jeżeli użytkownik wpisze KONIEC nastąpi zakończenie programu.



```
D:\Projekty\MetodyObliczeniowe\InterpolacjaNewtona\Debug\InterpolacjaNewtona.exe

Wielomian w postaci interpolacyjnej Newtona:
2+3(x-0)+4(x-0)(x-1)-2(x-0)(x-1)(x+1)

Aby kontynuować, naciśnij dowolny klawisz . . .

Wielomian w postaci ogólnej:
-2x^3+4x^2+x+2

Aby kontynuować, naciśnij dowolny klawisz . . .

Wpisz POCHODNA -- by obliczyć pochodną wielomianu stopnia j od liczby t.
Wpisz KONIEC -- by zakończyć program.
Twój wybór ->POCHODNA
Program obliczy pochodną wielomianu stopnia j od liczby t.
Podaj j (j powinno być z zakresu od 0 do 3) :2
Podaj t:3
Pochodna rzędu 2:
-12x+8
P^2(3)=-28
Wpisz POCHODNA -- by obliczyć pochodną wielomianu stopnia j od liczby t.
Wpisz KONIEC -- by zakończyć program.
Twój wybór ->KONIEC

Aby kontynuować, naciśnij dowolny klawisz . . .
```