

TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI
VIỆN TOÁN ỨNG DỤNG VÀ TIN HỌC

—o0o—



BÁO CÁO KỸ THUẬT LẬP TRÌNH

MÃ HỌC PHẦN: MI3310

Sinh viên: Nguyễn Văn Nghiêm – 20206206

Lớp: CTTN Toán Tin – K65

Mã lớp học: 131456

Giảng viên: Nguyễn Thị Thanh Huyền

Hà Nội, tháng 8 năm 2022

Mục lục

1	Chủ đề báo cáo	3
2	Phân tích, thiết kế chương trình	3
2.1	Phân tích tổng quát	3
2.2	Thiết kế chương trình	4
3	Kết quả chương trình và mã nguồn	12
3.1	Kết quả chương trình	12
3.2	Kết quả chương trình trên tệp văn bản	25
3.3	Mã nguồn	28

1 Chủ đề báo cáo

Chủ đề 5: Viết chương trình xây dựng đa thức nội suy ứng với một bảng số cho trước theo công thức **Newton** (cả hai trường hợp nút nội suy cách đều và không cách đều).

Yêu cầu:

- 1) Tính và in bảng tỷ hiệu (hoặc sai phân) ra màn hình và tệp văn bản
- 2) Viết đa thức nội suy tìm được (cả 2 dạng tiến và lùi đối với công thức Newton)
- 3) Dùng sơ đồ Hoocne tính giá trị đa thức tìm được tại một điểm được nhập vào từ bàn phím. Cần đưa toàn bộ quá trình tính toán trong sơ đồ Hoocne ra màn hình và tệp văn bản

Yêu cầu:

- Mọi kết quả được hiển thị với số chữ số thập phân nhập vào từ bàn phím.
- In đầy đủ các kết quả (cả kết quả trung gian) ra màn hình và tệp văn bản.
- Chương trình có chức năng hiển thị kết quả từ tệp văn bản.
- Điều khiển chương trình bằng menu.

2 Phân tích, thiết kế chương trình

2.1 Phân tích tổng quát

Phân tích tổng quát bài toán:

1. Dữ liệu vào là 1 bảng số cho trước, số chữ số thập phân và một điểm nhập vào từ bàn phím,
2. Chương trình phải thực hiện được các chức năng:
 - Tính và in bảng tỷ hiệu hoặc bảng sai phân
 - Viết đa thức nội suy ở cả 2 dạng tiến và lùi, nút bất kì và nút cách đều

- Tính giá trị đa thức tìm được tại một điểm theo sơ đồ Hoocne
- In các kết quả và kết quả trung gian ra màn hình và tệp
- Các kết quả in ra với số chữ số thập phân đã nhập
- Điều khiển bằng menu

2.2 Thiết kế chương trình

– Trình bày ý chính của giải thuật bằng ngôn ngữ tự nhiên:

Để có góc nhìn tổng quát hơn về đa thức nội suy tìm được, ta xây dựng thêm 1 chức năng vẽ đồ thị hàm số 2 hàm : hàm chuẩn (nếu có) và đa thức nội suy tìm được

1. Xây dựng menu 2 nấc, nấc 1 gồm 3 chức năng:

1. Newton nút bất kì
2. Newton nút cách đều
3. Kết thúc

2. Với mỗi chức năng 1 và 2, xây dựng menu nấc 2 gồm 6 chức năng:

1. Bảng tỷ hiệu (bảng sai phân đối với Newton nút cách đều)
2. Đa thức nội suy tiến
3. Đa thức nội suy lùi
4. Tính giá trị đa thức tại một điểm
5. Vẽ đồ thị
6. Quay lại

– Tinh chỉnh từng bước:

Bước 1:

1. Input: Bảng số
2. Output: menu gồm các chức năng như bước trên, hiển thị các kết quả trên màn hình và tệp văn bản.

– Ý tưởng thuật toán:

1. Sử dụng công thức tính tỷ hiệu, sai phân để xây dựng bảng tỷ hiệu và bảng sai phân
2. Sử dụng công thức đa thức nội suy Newton để xây dựng đa thức nội suy:
 - + Sử dụng sơ đồ Hoocone để tính các giá trị tích: $\omega_k(x) = \prod_{i=0}^{k-1} (x - x_i)$
 \longrightarrow xây dựng 1 bảng tính tích lưu hệ số các giá trị tích $\omega_k(x)$
3. Sử dụng sơ đồ Hoocone để tính giá trị đa thức tại một điểm
4. Sử dụng thư viện đồ họa *graphics.h* để xây dựng menu và vẽ đồ thị

Bước 2: Đọc dữ liệu

1. Input: Bảng số nhập từ file, số chữ số thập phân nhập từ bàn phím
2. Output:
 - + Số chữ số thập phân *precision*
 - + Bảng số *inf* (gồm các cặp điểm X và FX tương ứng)

Bước 3: Xây dựng bảng tỷ hiệu (sai phân)

1. Input: Bảng số
2. Output: Bảng tỷ hiệu (sai phân)

– Ý tưởng thuật toán:

1. Tạo 1 bảng lưu bảng tỷ hiệu (bảng sai phân)
2. Gán cột đầu bằng các giá trị của *inf.X*, cột thứ 2 bằng các giá trị của *inf.FX*
3. Duyệt từ hàng thứ 1 (đánh số bắt đầu từ 0) đến cuối
4. Tại mỗi hàng tính giá trị các vị trí tương ứng theo công thức tỷ hiệu (sai phân)

Bước 4: Xây dựng bảng tính tích

1. Input: Bảng số

2. Output: Bảng tích

– Ý tưởng thuật toán:

1. Tạo 1 mảng lưu giá trị của bảng tích
2. Xây dựng bảng tính bằng các ứng dụng số đồ Hoocne nhân đa thức với đơn thức
3. Đối với đa thức nội suy Newton tiến mốc bất kì, tính tích từ x_0 đến x_{n-1}
4. Đối với đa thức nội suy Newton lùi mốc bất kì, tính tích từ x_n đến x_1
5. Đối với đa thức nội suy Newton tiến mốc cách đều, tính tích từ x đến $x(x-1)\dots(x-n+1)$
6. Đối với đa thức nội suy Newton lùi mốc cách đều, tính tích từ x đến $x(x+1)\dots(x+n-1)$

Bước 5: Xây dựng đa thức nội suy tiến (lùi)

1. Input: Bảng tỷ hiệu, bảng tích
2. Output: Đa thức nội suy tiến (lùi)

– Ý tưởng thuật toán:

1. Tạo 1 mảng lưu hệ số của đa thức nội suy
2. Từ bảng tỷ hiệu và bảng tích tính toán các hệ số của đa thức nội suy theo công thức nội suy Newton
3. Đối với đa thức nội suy Newton tiến, lựa chọn các giá trị đầu bảng của bảng tỷ hiệu và bảng tính tích tương ứng
4. Đối với đa thức nội suy Newton lùi, lựa chọn các giá trị cuối bảng của bảng tỷ hiệu và bảng tính tích tương ứng
5. Với nội suy với các mốc bất kì thì đa thức nội suy Newton tiến hay lùi đều giống nhau.

Bước 6: Tính giá trị đa thức tại một điểm

1. Input: Đa thức nội suy, một điểm nhập từ bàn phím
2. Output: Giá trị đa thức tại điểm đó

– Ý tưởng thuật toán:

1. Tính lại giá trị theo công thức đổi ẩn đối với nội suy đa thức tiến (lùi) cách đều
2. Ứng dụng sơ đồ Horner tính toán giá trị của đa thức

Bước 7: Vẽ đồ thị

1. Input: Hàm số chuẩn (nếu có), đa thức nội suy, đoạn $[a, b]$
2. Output: Đồ thị biểu diễn 2 hàm số trên đoạn $[a, b]$

– Ý tưởng thuật toán:

1. Sử dụng thư viện *graphics.h*
2. Tìm max, min trên đoạn $[a, b]$
3. Xác định tọa độ O, độ dài, độ rộng của mỗi pixel
4. Vẽ trục tọa độ
5. Tính giá trị của hàm số bắt đầu từ $x = a$, kết thúc tại $x = b$, bước nhảy là h tùy chọn
6. Vẽ và nối các điểm trên lại ta được hình dáng của đồ thị hàm số

Bước 8: Mã giả các hàm chính

Note:

1. Quy ước đối với các biến (hàm) dùng với nội suy Newton mốc cách đều sẽ thêm chữ equidistly (viết tắt equid) so với các biến (hàm) dùng với nội suy Newton mốc bất kỳ.
2. type NORMAL đại diện cho mốc bất kỳ, EQUID đại diện cho mốc cách đều
3. type FORWARD đại diện cho Newton tiến, BACKWARD đại diện cho Newton lùi

– Nhập dữ liệu từ file

Input: File

Output: Bảng số (X, FX) *inf* , *infEquid*

function readData(TYPE)

if TYPE = NORMAL **then** đọc file lưu vào inf

else TYPE = EQUID **then** đọc file lưu vào infEquid

– Tính bảng tỷ hiệu

Input: Bảng số *inf*

Output: Bảng tỷ hiệu *divDiffTable*

function div_diff_table()

 Gán cột đầu của bảng tỷ hiệu bằng các giá trị của inf.X

 Gán cột thứ 2 của bảng tỷ hiệu bằng các giá trị của inf.FX

for i = 1 **to** inf.size() - 1 **do**

for j = 2 **to** i + 1 **do**

$$\text{divDiffTable}[i][j] = (\text{divDiffTable}[i][j - 1] - \text{divDiffTable}[i - 1][j - 1]) / (\text{divDiffTable}[i][0] - \text{divDiffTable}[i - j + 1][0])$$

end

end

– Tính bảng sai phân

Input: Bảng số *infEquid*

Output: Bảng sai phân *diffTable*

function diff_table()

 Gán cột đầu của bảng tỷ hiệu bằng các giá trị của infEquid.X

 Gán cột thứ 2 của bảng tỷ hiệu bằng các giá trị của infEquid.FX

for i = 1 **to** infEquid.size() - 1 **do**

for j = 2 **to** i + 1 **do**

diffTable[i][j] = diffTable[i][j - 1] - diffTable[i - 1][j - 1]

end

end

– Tính bảng tích mốc bất kì

Input: Bảng số *inf*

Output: Bảng tích *mulTable*

function mul_table (TYPE)

if TYPE = FORWARD **then** type = 0 **else** type = inf.size() - 1

mulTable[0][0] = -inf[type].X, mulTable[0][1] = 1

for i = 1 **to** inf.size() - 2 **do**

mulTable[i][i + 1] = 1

for j = i **downto** 1 **do**

mulTable[i][j] = mulTable[i - 1][j - 1]

- inf[abs(type - i)].X * mulTable[i - 1][j]

end

mulTable[i][0] = - inf[abs(type - i)].X * mulTable[i - 1][0]

end

– Tính bảng tích mốc cách đều

Input: Bảng số *infEquid*

Output: Bảng tích *mulTableEquid*

function mul_table_equid (TYPE)

mulTableEquid[0][0] = 1

for i = 1 **to** infEquid.size() - 2 **do**

mulTable[i][i] = 1


```

function Newton_interpolation_equidistly (TYPE)

    mul_table_equid(TYPE)

    if TYPE = FORWARD then type = 0 else type = infEquid.size( ) - 1

    NewtonInterEquid[0] = diffTable[type][1]

    for i = 1 to infEquid.size( ) - 1 do

        for j = 1 to i do

            if TYPE = FORWARD then

                NewtonInterEquid[j] += diffTable[i][i + 1]

                    * mulTableEquid[i - 1][j - 1] / factorial(i)

            else

                NewtonInterEquid[j] += diffTable[type][i + 1]

                    * mulTableEquid[i - 1][j - 1] / factorial(i)

            end

        end

    end

```

– Tính giá trị đa thức tại một điểm

Input: Đa thức P , điểm cần tính *point* nhập từ bàn phím

Output: giá trị tại $value = P(point)$

```

function cal_value (Poly P, TYPE1, TYPE2)

    if TYPE1 = EQUID and TYPE2 = FORWARD then

        point_t = (point - infEquid[0].X) / (infEquid[1].X - infEquid[0].X)

    else if TYPE1 = EQUID and TYPE2 = BACKWARD then

        point_t = (point - infEquid[infEquid.size( ) - 1].X)

            / (infEquid[1].X - infEquid[0].X)

    else point_t = point

    value = P[P.size( ) - 1]

```

```

for P.size( ) - 2 to 0 do

    value = value * point_t + P[i];

end

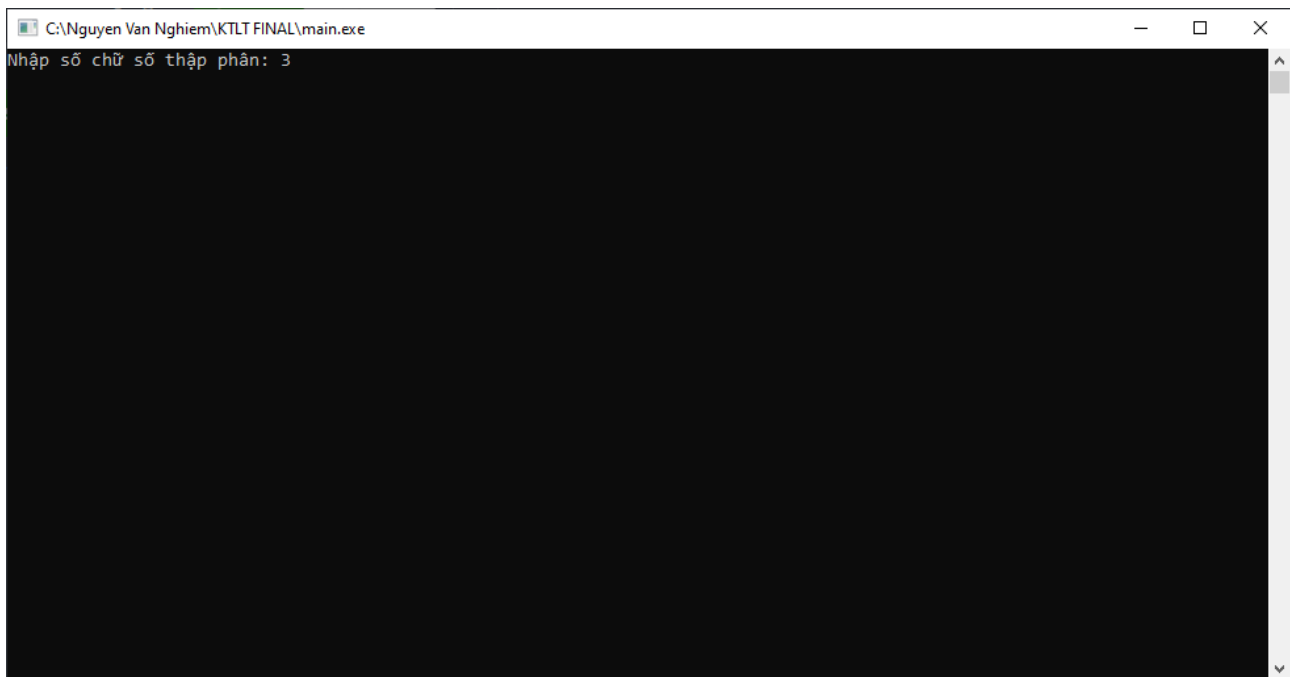
```

3 Kết quả chương trình và mã nguồn

3.1 Kết quả chương trình

- Ví dụ về nội suy Newton với bộ dữ liệu được trích từ hàm

$$\frac{1}{25x^2 + 1}$$



Hình 1: Khởi đầu khi chạy chương trình

- Sau khi nhập *precision* từ bàn phím, sẽ hiển thị thêm màn hình đồ họa *menu* điều khiển

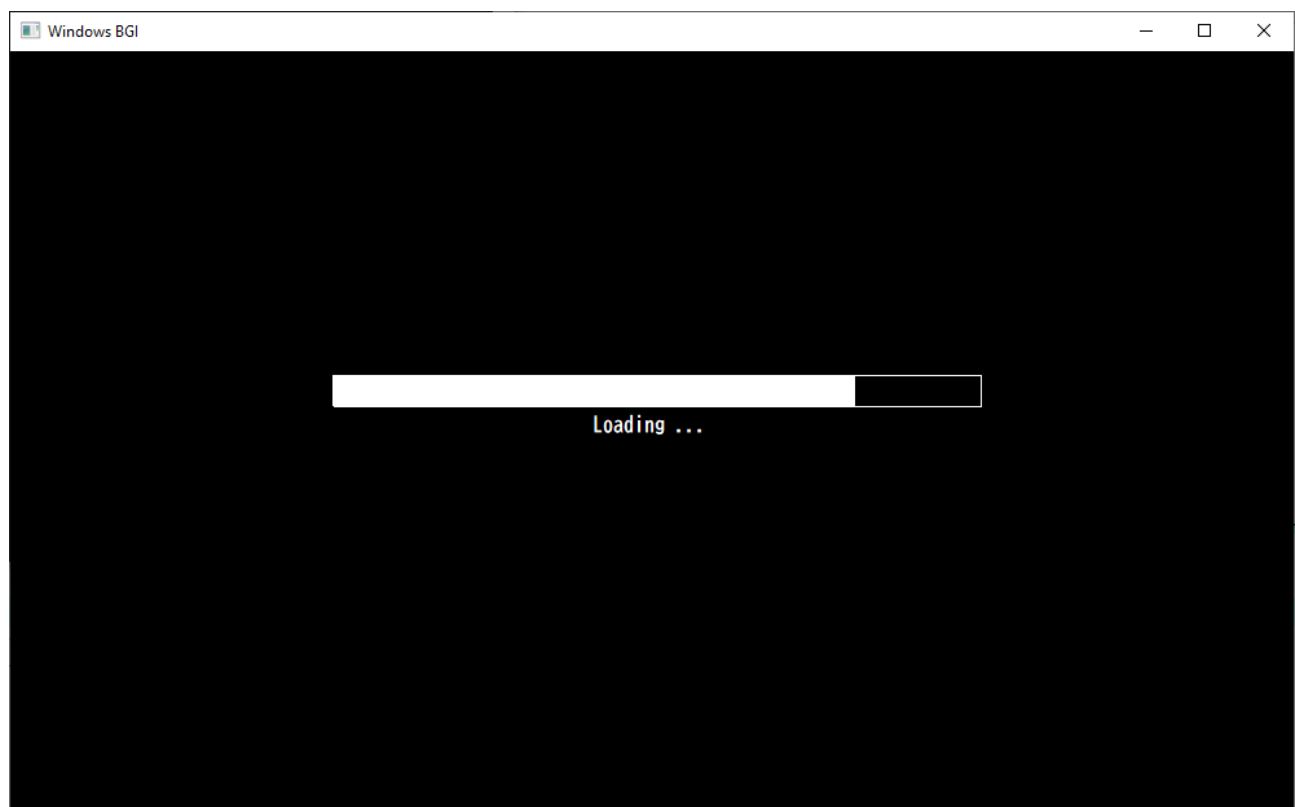
- Dùng các nút bàn phím "UP, DOWN" để di chuyển và "ENTER" để chọn chức năng:

- Khi chọn chức năng 1 "Noi suy moc bat ki" sẽ hiển thị ra bảng số được đọc từ file ra màn hình văn bản và menu tương ứng trên màn hình đồ họa, tương tự đối với chức năng 2 "Noi suy moc cach deu"

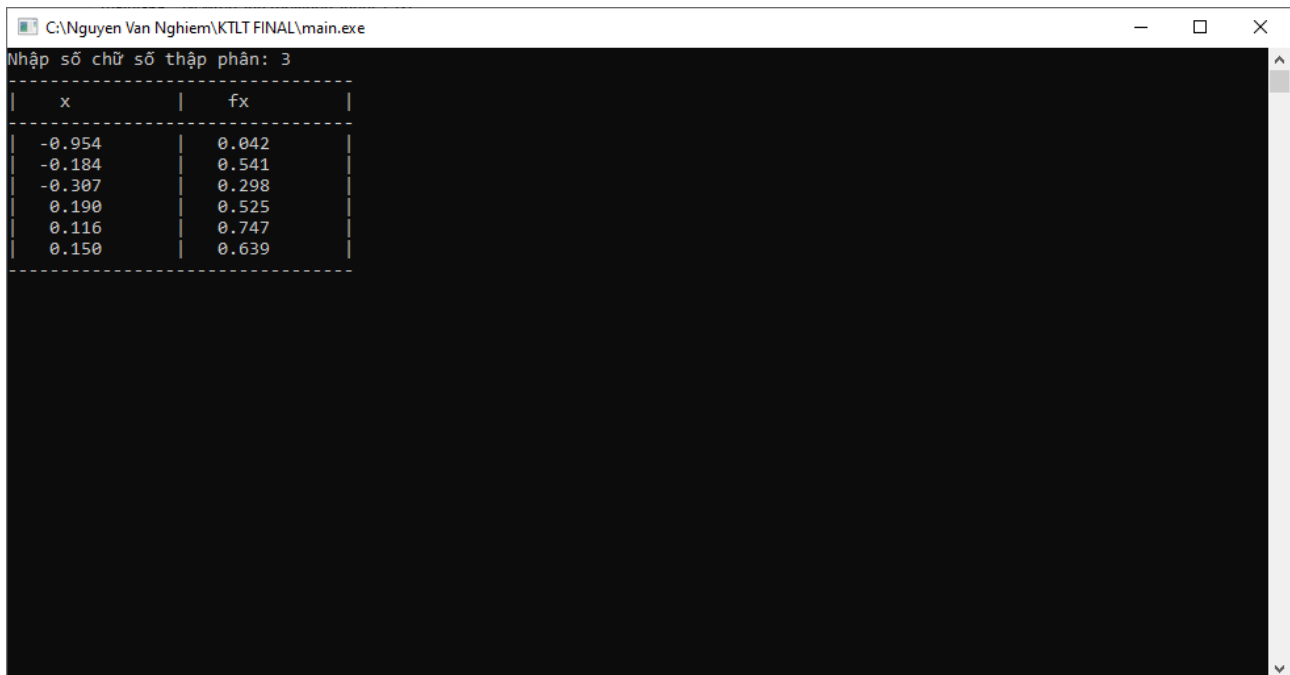


Hình 2: Màn hình menu được hiển thị

– Chọn các chức năng và các kết quả tương ứng được in ra màn hình văn bản và tệp



Hình 3: Màn hình loading được hiển thị sau khi chọn chức năng



Ngôn ngữ lập trình C++ được sử dụng để phát triển chương trình. Hình 4 thể hiện chức năng "Nói suy mơc bat ki" và bảng số hiển thị ra màn hình văn bản.

x	fx
-0.954	0.042
-0.184	0.541
-0.307	0.298
0.190	0.525
0.116	0.747
0.150	0.639

Hình 4: Chức năng "Nói suy mơc bat ki" và bảng số hiển thị ra màn hình văn bản



Hình 5: Chức năng "Nói suy mơc bat ki" và menu tương ứng hiển thị ra màn hình đồ họa

C:\Nguyễn Văn Nghiêm\KTLT FINAL\main.exe

Nhập số chữ số thập phân: 3

x	fx
-0.954	0.042
-0.184	0.541
-0.307	0.298
0.190	0.525
0.116	0.747
0.150	0.639

Bảng tỷ hiệu:

x	fx					
-0.954	0.042	0.000	0.000	0.000	0.000	0.000
-0.184	0.541	0.649	0.000	0.000	0.000	0.000
-0.307	0.298	1.981	2.060	0.000	0.000	0.000
0.190	0.525	0.456	-4.075	-5.364	0.000	0.000
0.116	0.747	-3.008	-8.185	-13.677	-7.770	0.000
0.150	0.639	-3.185	4.440	27.614	123.441	118.861

Hình 6: Chức năng bảng tỷ hiệu



Hình 7: Tiếp tục chọn các chức năng khác trên menu

C:\Nguyễn Văn Nghiêm\KTLT FINAL\main.exe

Bảng tỷ hiệu:

x	fx					
-0.954	0.042	0.000	0.000	0.000	0.000	0.000
-0.184	0.541	0.649	0.000	0.000	0.000	0.000
-0.307	0.298	1.981	2.060	0.000	0.000	0.000
0.190	0.525	0.456	-4.075	-5.364	0.000	0.000
0.116	0.747	-3.008	-8.185	-13.677	-7.770	0.000
0.150	0.639	-3.185	4.440	27.614	123.441	118.861

Bảng tính tích:

0.954	1.000	0.000	0.000	0.000	0.000
0.176	1.138	1.000	0.000	0.000	0.000
0.054	0.525	1.444	1.000	0.000	0.000
-0.010	-0.046	0.250	1.254	1.000	0.000
0.001	-0.005	-0.075	0.104	1.138	1.000

--> $f(x) = 0.042$
 --> $f(x) = 0.661 + 0.649x$
 --> $f(x) = 1.023 + 2.993x + 2.060x^2$
 --> $f(x) = 0.734 + 0.179x - 5.687x^2 - 5.364x^3$
 --> $f(x) = 0.813 + 0.536x - 7.627x^2 - 15.108x^3 - 7.770x^4$
 --> $f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5$
 Đa thức nội suy tiến là:
 $f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5$

Hình 8: Chức năng đa thức nội suy tiến, hiển thị ra bảng tích và các bước trung gian

C:\Nguyễn Văn Nghiêm\KTLT FINAL\main.exe

0.001	-0.005	-0.075	0.104	1.138	1.000
-------	--------	--------	-------	-------	-------

--> $f(x) = 0.042$
 --> $f(x) = 0.661 + 0.649x$
 --> $f(x) = 1.023 + 2.993x + 2.060x^2$
 --> $f(x) = 0.734 + 0.179x - 5.687x^2 - 5.364x^3$
 --> $f(x) = 0.813 + 0.536x - 7.627x^2 - 15.108x^3 - 7.770x^4$
 --> $f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5$
 Đa thức nội suy tiến là:
 $f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5$

Bảng tính tích:

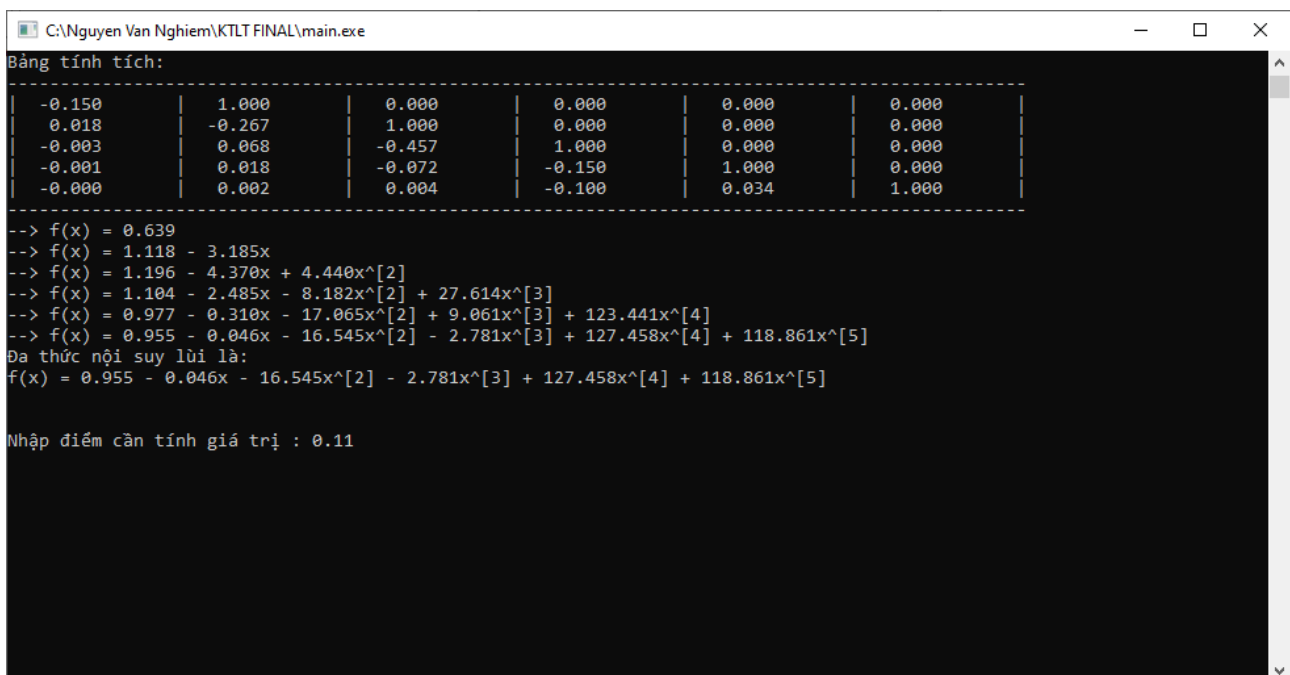
-0.150	1.000	0.000	0.000	0.000	0.000
0.018	-0.267	1.000	0.000	0.000	0.000
-0.003	0.068	-0.457	1.000	0.000	0.000
-0.001	0.018	-0.072	-0.150	1.000	0.000
-0.000	0.002	0.004	-0.100	0.034	1.000

--> $f(x) = 0.639$
 --> $f(x) = 1.118 - 3.185x$
 --> $f(x) = 1.196 - 4.370x + 4.440x^2$
 --> $f(x) = 1.104 - 2.485x - 8.182x^2 + 27.614x^3$
 --> $f(x) = 0.977 - 0.310x - 17.065x^2 + 9.061x^3 + 123.441x^4$
 --> $f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5$
 Đa thức nội suy lùi là:
 $f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5$

Hình 9: Chức năng đa thức nội suy lùi, hiển thị ra bảng tích và các bước trung gian



Hình 10: Chọn chức năng tính giá trị tại một điểm



Hình 11: Yêu cầu nhập điểm cần tính tại màn hình văn bản

```

C:\Nguyễn Văn Nghiêm\KTLT FINAL\main.exe
-0.003 | 0.068 | -0.457 | 1.000 | 0.000 | 0.000 |
-0.001 | 0.018 | -0.072 | -0.150 | 1.000 | 0.000 |
-0.000 | 0.002 | 0.004 | -0.100 | 0.034 | 1.000 |
-----
--> f(x) = 0.639
--> f(x) = 1.118 - 3.185x
--> f(x) = 1.196 - 4.370x + 4.440x^2
--> f(x) = 1.104 - 2.485x - 8.182x^2 + 27.614x^3
--> f(x) = 0.977 - 0.310x - 17.065x^2 + 9.061x^3 + 123.441x^4
--> f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5
Đa thức nội suy lùi là:
f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5

Nhập điểm cần tính giá trị : 0.11

Sơ đồ Hoocone:
-----
| coeff | 118.861 | 127.458 | -2.781 | -16.545 | -0.046 | 0.955 |
| 0.110 | 118.861 | 140.533 | 12.677 | -15.151 | -1.713 | 0.767 |
-----

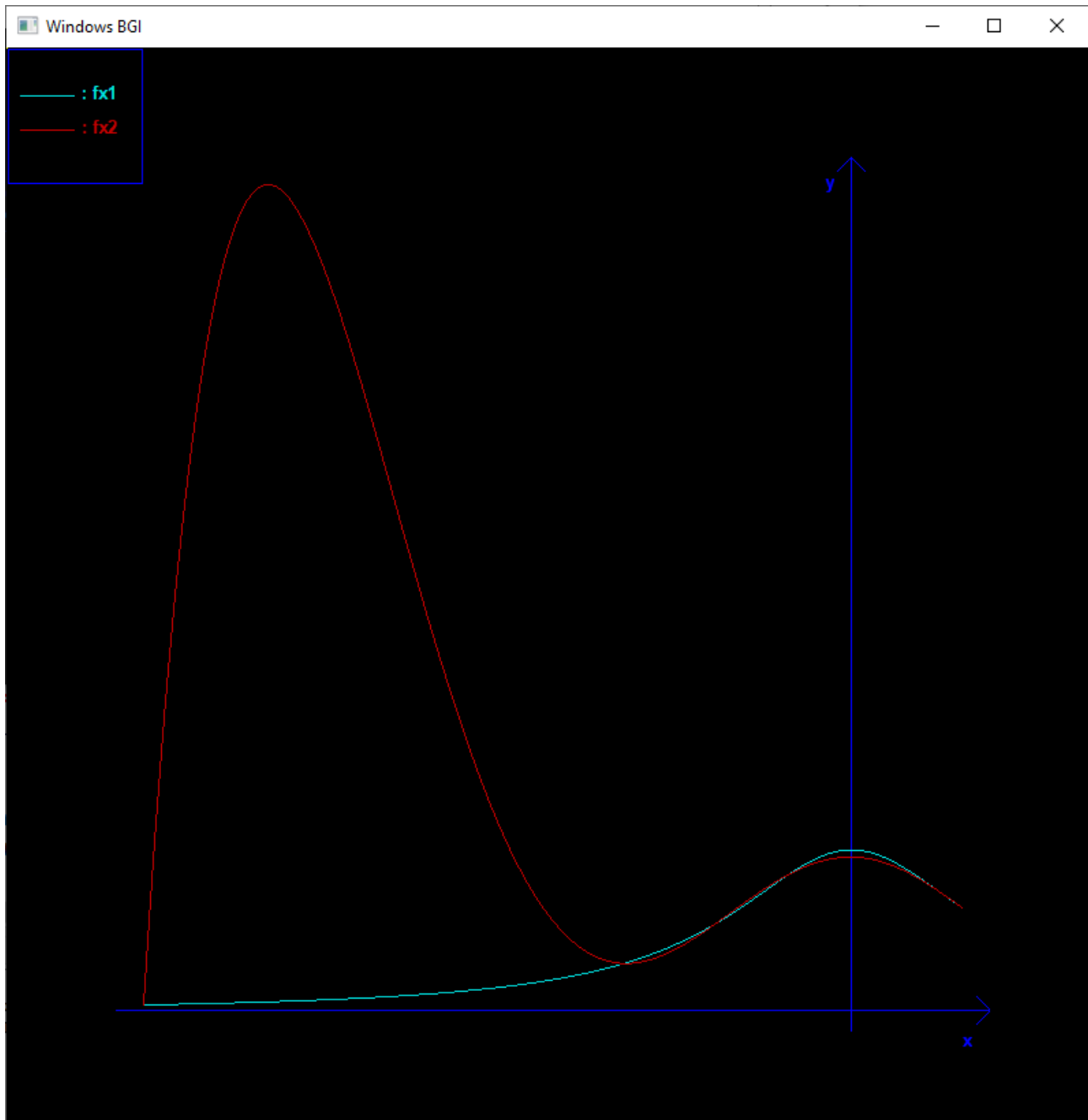
value = 118.861 * 0.110 + 127.458 = 140.533
value = 140.533 * 0.110 + -2.781 = 12.677
value = 12.677 * 0.110 + -16.545 = -15.151
value = -15.151 * 0.110 + -0.046 = -1.713
value = -1.713 * 0.110 + 0.955 = 0.767
--> f(0.110) = 0.767

```

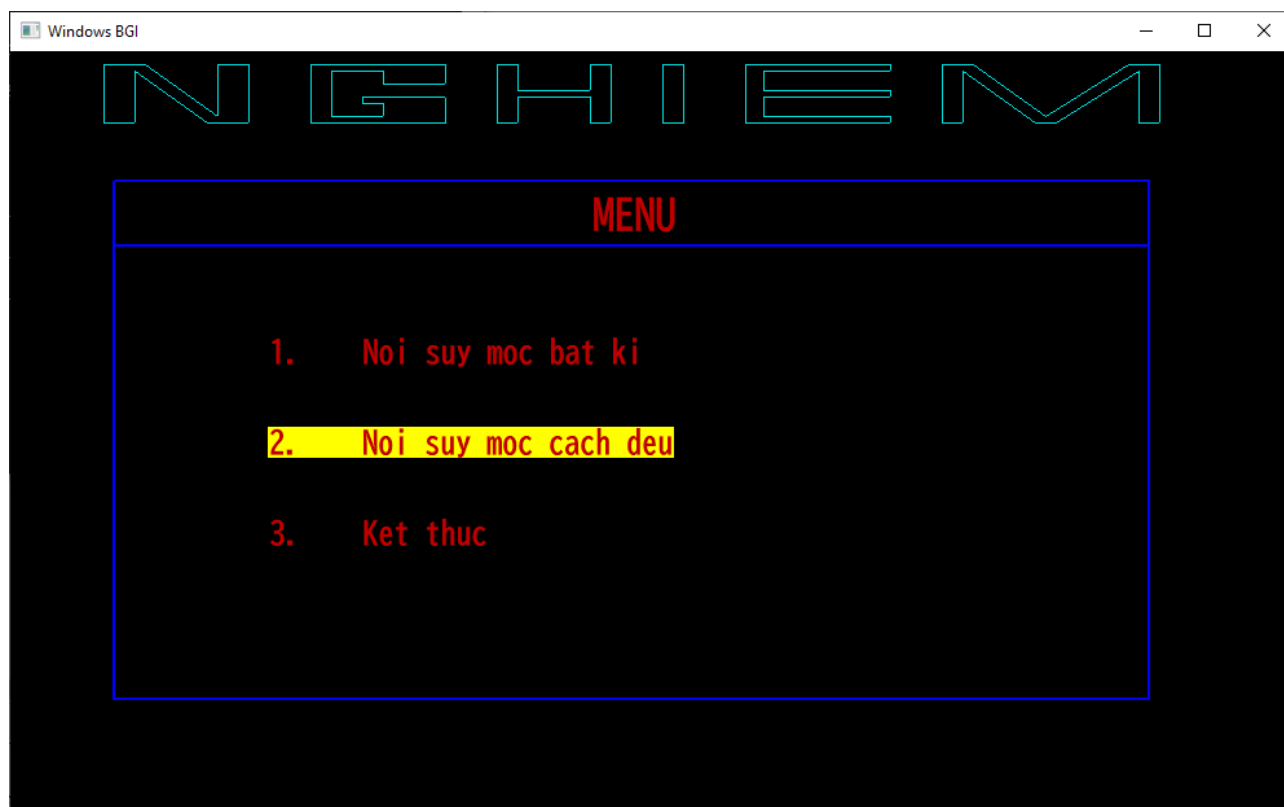
Hình 12: ENTER sau khi nhập xong, hiển thị ra sơ đồ Hoocone và các bước tính



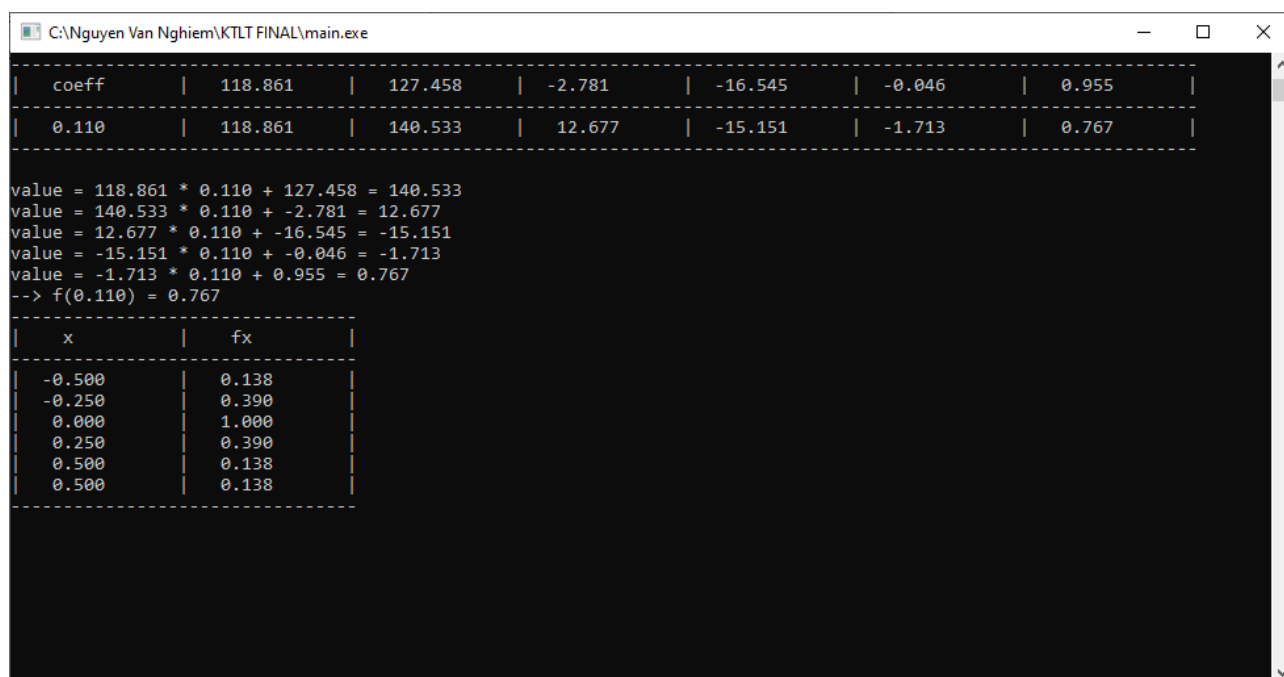
Hình 13: Chức năng vẽ đồ thị



Hình 14: Đồ thị được vẽ trên màn hình đồ họa, **fx2** là đồ thị đa thức nội suy



Hình 15: Chọn chức năng Quay lại menu đầu và chọn chức năng "Noi suy moc cach deu"



Hình 16: Chức năng "Noi suy moc cach deu", bảng số được hiển thị trên màn hình văn bản



Hình 17: Chức năng "Noi suy moc cach deu", menu được hiển thị trên màn hình đồ họa

Bảng sai phân:

x	fx						
-0.500	0.138	0.000	0.000	0.000	0.000	0.000	0.000
-0.250	0.390	0.252	0.000	0.000	0.000	0.000	0.000
0.000	1.000	0.610	0.358	0.000	0.000	0.000	0.000
0.250	0.390	-0.610	-1.220	-1.577	0.000	0.000	0.000
0.500	0.138	-0.252	0.358	1.577	3.154	0.000	0.000
0.500	0.138	0.000	0.252	-0.105	-1.682	-4.837	

Hình 18: Chức năng bảng sai phân

```

C:\Nguyen Van Nghiem\KTLT FINAL\main.exe
--> f(t) = 0.138
--> f(t) = 0.138 + 0.252t
--> f(t) = 0.138 + 0.074t + 0.179t^2
--> f(t) = 0.138 - 0.452t + 0.967t^2 - 0.263t^3
--> f(t) = 0.138 - 1.241t + 2.413t^2 - 1.051t^3 + 0.131t^4
--> f(t) = 0.138 - 2.208t + 4.428t^2 - 2.462t^3 + 0.534t^4 - 0.040t^5
Đa thức nội suy tiến mốc cách đều là :
f(t) = 0.138 - 2.208t + 4.428t^2 - 2.462t^3 + 0.534t^4 - 0.040t^5

Bảng tính tích:
-----
| 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| 2.000 | 3.000 | 1.000 | 0.000 | 0.000 |
| 6.000 | 11.000 | 6.000 | 1.000 | 0.000 |
| 24.000 | 50.000 | 35.000 | 10.000 | 1.000 |
-----

--> f(t) = 0.138
--> f(t) = 0.138
--> f(t) = 0.138 + 0.126t + 0.126t^2
--> f(t) = 0.138 + 0.091t + 0.074t^2 - 0.018t^3
--> f(t) = 0.138 - 0.329t - 0.698t^2 - 0.438t^3 - 0.070t^4
--> f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5
Đa thức nội suy lùi mốc cách đều là :
f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5

```

Hình 19: Chức năng đa thức nội suy tiến, lùi mốc cách đều

```

C:\Nguyen Van Nghiem\KTLT FINAL\main.exe
--> f(t) = 0.138
--> f(t) = 0.138 + 0.126t + 0.126t^2
--> f(t) = 0.138 + 0.091t + 0.074t^2 - 0.018t^3
--> f(t) = 0.138 - 0.329t - 0.698t^2 - 0.438t^3 - 0.070t^4
--> f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5
Đa thức nội suy lùi mốc cách đều là :
f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5

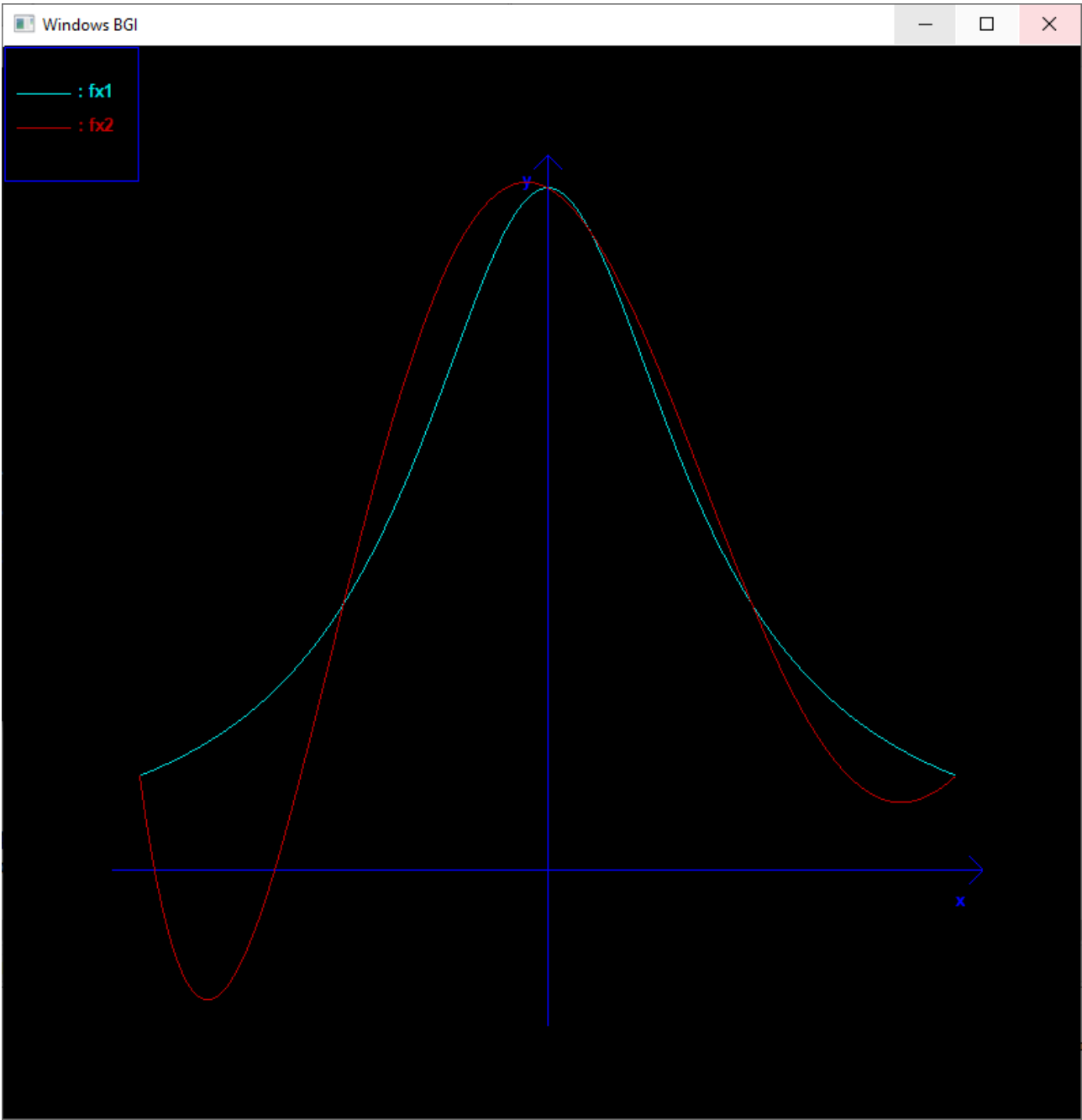
Nhập điểm cần tính giá trị : 0.12

Sơ đồ Hoocne:
-----
| coeff | -0.040 | 0.534 | -2.462 | 4.428 | -2.208 | 0.138 |
| 2.480 | -0.040 | 0.435 | -1.384 | 0.995 | 0.259 | 0.780 |
-----

value = -0.040 * 2.480 + 0.534 = 0.435
value = 0.435 * 2.480 + -2.462 = -1.384
value = -1.384 * 2.480 + 4.428 = 0.995
value = 0.995 * 2.480 + -2.208 = 0.259
value = 0.259 * 2.480 + 0.138 = 0.780
--> f(2.480) = 0.780

```

Hình 20: Chức năng tính giá trị đa thức tại 1 điểm



Hình 21: Chức năng vẽ đồ thị



Hình 22: Quay lại và chọn Kết thúc để kết thúc chương trình

```

C:\Nguyễn Văn Nghiêm\KT LT FINAL\main.exe
--> f(t) = 0.138
--> f(t) = 0.138 + 0.126t + 0.126t^2
--> f(t) = 0.138 + 0.091t + 0.074t^2 - 0.018t^3
--> f(t) = 0.138 - 0.329t - 0.698t^2 - 0.438t^3 - 0.070t^4
--> f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5
Đa thức nội suy lùi mốc cách đều là :
f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5

Nhập điểm cần tính giá trị : 0.12

Sơ đồ Hoocne:
-----
|  coeff   | -0.040 |  0.534 | -2.462 |  4.428 | -2.208 |  0.138 |
|-----|
|  2.480   | -0.040 |  0.435 | -1.384 |  0.995 |  0.259 |  0.780 |
|-----|

value = -0.040 * 2.480 + 0.534 = 0.435
value = 0.435 * 2.480 + -2.462 = -1.384
value = -1.384 * 2.480 + 4.428 = 0.995
value = 0.995 * 2.480 + -2.208 = 0.259
value = 0.259 * 2.480 + 0.138 = 0.780
--> f(2.480) = 0.780

-----
Process exited after 656.5 seconds with return value 0
Press any key to continue . . .

```

Hình 23: Kết thúc chương trình, màn hình đồ họa sẽ tự đóng

3.2 Kết quả chương trình trên tệp văn bản

```

Newton_interpolation_output.txt - Notepad
File Edit Format View Help
Nhập số chữ số thập phân: 3

| x | fx |
|---|---|
| -0.954 | 0.042 |
| -0.184 | 0.541 |
| -0.307 | 0.298 |
| 0.190 | 0.525 |
| 0.116 | 0.747 |
| 0.150 | 0.639 |

Bảng tỷ hiệu:

| x | fx | | | | | |
|---|---|---|---|---|---|---|
| -0.954 | 0.042 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| -0.184 | 0.541 | 0.649 | 0.000 | 0.000 | 0.000 | 0.000 |
| -0.307 | 0.298 | 1.981 | 2.060 | 0.000 | 0.000 | 0.000 |
| 0.190 | 0.525 | 0.456 | -4.075 | -5.364 | 0.000 | 0.000 |
| 0.116 | 0.747 | -3.008 | -8.185 | -13.677 | -7.770 | 0.000 |
| 0.150 | 0.639 | -3.185 | 4.440 | 27.614 | 123.441 | 118.861 |

Bảng tính tích:

| 0.954 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 0.176 | 1.138 | 1.000 | 0.000 | 0.000 | 0.000 |
| 0.054 | 0.525 | 1.444 | 1.000 | 0.000 | 0.000 |
| -0.010 | -0.046 | 0.250 | 1.254 | 1.000 | 0.000 |
| 0.001 | -0.005 | -0.075 | 0.104 | 1.138 | 1.000 |

--> f(x) = 0.042
--> f(x) = 0.661 + 0.649x
--> f(x) = 1.023 + 2.993x + 2.060x^2
--> f(x) = 0.734 + 0.179x - 5.687x^2 - 5.364x^3
--> f(x) = 0.813 + 0.536x - 7.627x^2 - 15.108x^3 - 7.770x^4
--> f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5
Đa thức nội suy tiến là:
f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5

Bảng tính tích:

| -0.150 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 0.018 | -0.267 | 1.000 | 0.000 | 0.000 | 0.000 |
| -0.003 | 0.068 | -0.457 | 1.000 | 0.000 | 0.000 |
| -0.001 | 0.018 | -0.072 | -0.150 | 1.000 | 0.000 |
| -0.000 | 0.002 | 0.004 | -0.100 | 0.034 | 1.000

```

```

Newton_interpolation_output.txt - Notepad
File Edit Format View Help
Bảng tính tích:
-----
| -0.150 | 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 0.018 | -0.267 | 1.000 | 0.000 | 0.000 | 0.000 |
| -0.003 | 0.068 | -0.457 | 1.000 | 0.000 | 0.000 |
| -0.001 | 0.018 | -0.072 | -0.150 | 1.000 | 0.000 |
| -0.000 | 0.002 | 0.004 | -0.100 | 0.034 | 1.000 |
-----

--> f(x) = 0.639
--> f(x) = 1.118 - 3.185x
--> f(x) = 1.196 - 4.370x + 4.440x^2
--> f(x) = 1.104 - 2.485x - 8.182x^2 + 27.614x^3
--> f(x) = 0.977 - 0.310x - 17.065x^2 + 9.061x^3 + 123.441x^4
--> f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5
Đa thức nội suy lùi là:
f(x) = 0.955 - 0.046x - 16.545x^2 - 2.781x^3 + 127.458x^4 + 118.861x^5

Nhập điểm cần tính giá trị :0.110

Sơ đồ Hoocone:
-----
| coeff | 118.861 | 127.458 | -2.781 | -16.545 | -0.046 | 0.955 |
| 0.110 | 118.861 | 140.533 | 12.677 | -15.151 | -1.713 | 0.767 |
-----

value = 118.861 * 0.110 + 127.458 = 140.533
value = 140.533 * 0.110 + -2.781 = 12.677
value = 12.677 * 0.110 + -16.545 = -15.151
value = -15.151 * 0.110 + -0.046 = -1.713
value = -1.713 * 0.110 + 0.955 = 0.767
--> f(0.110) = 0.767

-----
| x | fx |
-----
| -0.500 | 0.138 |
| -0.250 | 0.390 |
| 0.000 | 1.000 |
| 0.250 | 0.390 |
| 0.500 | 0.138 |
| 0.500 | 0.138 |
-----

Bảng sai phân:
-----
| x | fx |
-----
| -0.500 | 0.138 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| -0.250 | 0.390 | 0.252 | 0.000 | 0.000 | 0.000 | 0.000 |
-----

```

```

Newton_interpolation_output.txt - Notepad
File Edit Format View Help
-----
Bảng sai phân:
-----
| x | fx | | | | | | |
| -0.500 | 0.138 | 0.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| -0.250 | 0.390 | 0.252 | 0.000 | 0.000 | 0.000 | 0.000 |
| 0.000 | 1.000 | 0.610 | 0.358 | 0.000 | 0.000 | 0.000 |
| 0.250 | 0.390 | -0.610 | -1.220 | -1.577 | 0.000 | 0.000 |
| 0.500 | 0.138 | -0.252 | 0.358 | 1.577 | 3.154 | 0.000 |
| 0.500 | 0.138 | 0.000 | 0.252 | -0.105 | -1.682 | -4.837 |
-----

Bảng tính tích:
-----
| 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| -1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| 2.000 | -3.000 | 1.000 | 0.000 | 0.000 |
| -6.000 | 11.000 | -6.000 | 1.000 | 0.000 |
| 24.000 | -50.000 | 35.000 | -10.000 | 1.000 |
-----

--> f(t) = 0.138
--> f(t) = 0.138 + 0.252t
--> f(t) = 0.138 + 0.074t + 0.179t^2
--> f(t) = 0.138 - 0.452t + 0.967t^2 - 0.263t^3
--> f(t) = 0.138 - 1.241t + 2.413t^2 - 1.051t^3 + 0.131t^4
--> f(t) = 0.138 - 2.208t + 4.428t^2 - 2.462t^3 + 0.534t^4 - 0.040t^5
Đa thức nội suy tiến mốc cách đều là :
f(t) = 0.138 - 2.208t + 4.428t^2 - 2.462t^3 + 0.534t^4 - 0.040t^5

Bảng tính tích:
-----
| 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| 2.000 | 3.000 | 1.000 | 0.000 | 0.000 |
| 6.000 | 11.000 | 6.000 | 1.000 | 0.000 |
| 24.000 | 50.000 | 35.000 | 10.000 | 1.000 |
-----

--> f(t) = 0.138
--> f(t) = 0.138
--> f(t) = 0.138 + 0.126t + 0.126t^2
--> f(t) = 0.138 + 0.091t + 0.074t^2 - 0.018t^3
--> f(t) = 0.138 - 0.329t - 0.698t^2 - 0.438t^3 - 0.070t^4
--> f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5
Đa thức nội suy lùi mốc cách đều là :
f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5

```

```

Newton_interpolation_output.txt - Notepad
File Edit Format View Help
| 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| -1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| 2.000 | -3.000 | 1.000 | 0.000 | 0.000 |
| -6.000 | 11.000 | -6.000 | 1.000 | 0.000 |
| 24.000 | -50.000 | 35.000 | -10.000 | 1.000 |
-----
--> f(t) = 0.138
--> f(t) = 0.138 + 0.252t
--> f(t) = 0.138 + 0.074t + 0.179t^2
--> f(t) = 0.138 - 0.452t + 0.967t^2 - 0.263t^3
--> f(t) = 0.138 - 1.241t + 2.413t^2 - 1.051t^3 + 0.131t^4
--> f(t) = 0.138 - 2.208t + 4.428t^2 - 2.462t^3 + 0.534t^4 - 0.040t^5
Đa thức nội suy tiến mốc cách đều là :
f(t) = 0.138 - 2.208t + 4.428t^2 - 2.462t^3 + 0.534t^4 - 0.040t^5

Bảng tính tích:
-----
| 1.000 | 0.000 | 0.000 | 0.000 | 0.000 |
| 1.000 | 1.000 | 0.000 | 0.000 | 0.000 |
| 2.000 | 3.000 | 1.000 | 0.000 | 0.000 |
| 6.000 | 11.000 | 6.000 | 1.000 | 0.000 |
| 24.000 | 50.000 | 35.000 | 10.000 | 1.000 |
-----
--> f(t) = 0.138
--> f(t) = 0.138
--> f(t) = 0.138 + 0.126t + 0.126t^2
--> f(t) = 0.138 + 0.091t + 0.074t^2 - 0.018t^3
--> f(t) = 0.138 - 0.329t - 0.698t^2 - 0.438t^3 - 0.070t^4
--> f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5
Đa thức nội suy lùi mốc cách đều là :
f(t) = 0.138 - 1.297t - 2.713t^2 - 1.849t^3 - 0.473t^4 - 0.040t^5

Nhập điểm cần tính giá trị :0.120

Sơ đồ Hoocne:
-----
| coeff | -0.040 | 0.534 | -2.462 | 4.428 | -2.208 | 0.138 |
| 2.480 | -0.040 | 0.435 | -1.384 | 0.995 | 0.259 | 0.780 |
-----

value = -0.040 * 2.480 + 0.534 = 0.435
value = 0.435 * 2.480 + -2.462 = -1.384
value = -1.384 * 2.480 + 4.428 = 0.995
value = 0.995 * 2.480 + -2.208 = 0.259
value = 0.259 * 2.480 + 0.138 = 0.780
--> f(2.480) = 0.780

```

3.3 Mã nguồn

```

/* library */
#include <iostream>
#include <graphics.h>
#include <windows.h>
#include <iomanip>
#include <vector>
#include <math.h>

#define X first
#define FX second

```

```

#define FOR_EACH(x, v)      for(auto x : v)
#define FOR(a, b, i)        for(int i = a; i < b; i++)
#define FOR_(a, b, i)        for(int i = a; i >= b; i--)
#define DRAW(a)              FOR(0, a, i) printf("-");
#define DRAWFILE(nameFile, a) FOR(0, a, i) fprintf(nameFile, "-");

#define KEY_ENTER  13

#define NORMAL      0
#define EQUID        1

#define FORWARD     1
#define BACKWARD    2

#define MAX_CHOICE_1 3
#define MAX_CHOICE_2 6
#define MAX_CHOICE_3 6

#define WIDTH        1000
#define HEIGHT        600

#define FIELDWIDTH    30

#define FUNCTION_1    1
#define FUNCTION_2    2
#define FUNCTION_3    3
#define FUNCTION_4    4
#define FUNCTION_5    5
#define FUNCTION_6    6

using namespace std;

typedef vector< pair <double, double> > Inf;
typedef vector< vector <double> > Table;
typedef vector<double> Poly;

const char inFile[2][100] = {"Newton_interpolation_input.txt",
    "Newton_interpolation_input_2.txt"};
const char outFile[100] = {"Newton_interpolation_output.txt"};

const char menu_1[MAX_CHOICE_1][50] = {"1. Noi suy moc bat ki",
    "2.    Noi suy moc cach deu",
    "3.    Ket thuc"};

const char menu_2[MAX_CHOICE_2][50] = {"1. Bang ty hieu",
    "2.    Da thuc noi suy tien",
    "3.    Da thuc noi suy lui",
    "4.    Tinh gia tri da thuc tai mot diem",
    "5.    Ve do thi",
    "6.    Quay lai"};

const char menu_3[MAX_CHOICE_3][50] = {"1. Bang sai phan",
    "2.    Da thuc noi suy tien moc cach deu",

```

```

        "3.   Da thuc noi suy lui moc cach deu",
        "4.   Tinh gia tri da thuc tai mot diem",
        "5.   Ve do thi",
        "6.   Quay lai"};

/* prototypes */

void readData(int);
void print_poly(Poly, char);

void frame(void);
void frame_load(void);
void set_color(int, int);

void Newton_interpolation(void);
void print_Newton_interpolation(int);
void div_diff_table(void);
void print_div_diff_table(void);
void mul_table(int);
void print_mul_table(void);

void Newton_interpolation_equidistly(int);
void print_Newton_interpolation_equidistly(int);
void diff_table(void);
void print_diff_table(void);
void mul_table_equid(int);
void print_mul_table_equid(void);
long long factorial(int n);

void cal_value(Poly, int, int);
void print_cal_value(Poly);

double fx1(double);
double fx2(Poly, double);
double fx3(Poly, double);
void draw_graph(double (*)(double), double (*)(Poly, double), Poly, double, double);

void menu(void);
int show_menu_1(void);
int show_menu_2(void);
int show_menu_3(void);
void highlight(int, int, char*);

/* Global variables */
int precision;
Inf inf, infEquid;
Table divDiffTable, diffTable, mulTable, mulTableEquid;
Poly NewtonInter, NewtonInterEquid;
bool run = true, runMenu1 = false, runMenu2 = false, runMenu3 = false;
FILE* fileOut = fopen(outFile, "w");

int main(){
    printf("Nhap so chu so thap phan: ");

```

```

    fprintf(fileOut, "Nhap so chu so thập phân: ");
    scanf("%d", &precision);
    fprintf(fileOut, "%d\n", precision);

    initwindow(WIDTH, HEIGHT);
    while(run) menu();
    fclose(fileOut);
}

void frame(void){
    setlinestyle(0, 0, 2);
    setcolor(BLUE);
    rectangle(80, 100, 880, 500);
    line(80, 150, 880, 150);
    setcolor(CYAN);
    setlinestyle(0, 0, 1);

    // N
    line(72, 10, 104, 10);
    line(72, 55, 96, 55);
    line(72, 10, 72, 55);
    line(96, 15, 96, 55);
    line(96, 15, 152, 55);
    line(104, 10, 160, 50);
    line(160, 10, 160, 50);
    line(152, 55, 184, 55);
    line(184, 10, 184, 55);
    line(160, 10, 184, 10);
    // G
    line(232, 10, 336, 10);
    line(248, 15, 288, 15);
    line(288, 15, 288, 25);
    line(288, 25, 336, 25);
    line(336, 10, 336, 25);
    line(232, 10, 232, 55);
    line(248, 15, 248, 50);
    line(232, 55, 336, 55);
    line(336, 55, 336, 35);
    line(272, 35, 272, 40);
    line(272, 35, 336, 35);
    line(272, 40, 288, 40);
    line(288, 40, 288, 50);
    line(248, 50, 288, 50);
    // H
    line(376, 10, 392, 10);
    line(448, 10, 464, 10);
    line(376, 10, 376, 55);
    line(464, 10, 464, 55);
    line(376, 55, 392, 55);
    line(448, 55, 464, 55);
    line(392, 10, 392, 30);
    line(392, 35, 392, 55);
    line(448, 10, 448, 30);

```

```

    line(448, 35, 448, 55);
    line(392, 30, 448, 30);
    line(392, 35, 448, 35);
    // I
    rectangle(504, 10, 520, 55);
    // E
    line(568, 10, 680, 10);
    line(568, 55, 680, 55);
    line(582, 50, 680, 50);
    line(582, 35, 680, 35);
    line(582, 30, 680, 30);
    line(582, 15, 680, 15);
    line(568, 10, 568, 55);
    line(582, 15, 582, 30);
    line(582, 35, 582, 50);
    line(680, 10, 680, 15);
    line(680, 30, 680, 35);
    line(680, 50, 680, 55);
    // M
    line(720, 10, 720, 55);
    line(736, 15, 736, 55);
    line(872, 15, 872, 55);
    line(888, 10, 888, 55);
    line(720, 10, 744, 10);
    line(864, 10, 888, 10);
    line(720, 55, 736, 55);
    line(792, 55, 808, 55);
    line(872, 55, 888, 55);
    line(744, 10, 800, 50);
    line(800, 50, 864, 10);
    line(736, 15, 792, 55);
    line(808, 55, 872, 15);
}

void frame_load(void){
    cleardevice();

    settextstyle(0, 0, 2);
    setcolor(WHITE);
    outtextxy(450, 280, (char*)"Loading ...");

    settextstyle(0, 0, 3);
    set_color(WHITE, WHITE);
    rectangle(250, 250, 750, 274);
    for(int i = 250; i < 740; i += 10){
        delay(1);
        if(i == 650) delay(200);
        outtextxy(i, 250, (char*)" ");
    }
    setbkcolor(BLACK);
}

void set_color(int colorText, int colorBkText){

```



```

    setcolor(colorText);
    setbkcolor(colorBkText);
}

void menu(void){
    int choice1 = 0, choice2 = 0, choice3 = 0;
    runMenu1 = true;
    while(runMenu1){
        choice1 = show_menu_1();
        switch (choice1){
            case FUNCTION_1: frame_load();
                           readData(NORMAL);
                           runMenu2 = true;
                           runMenu1 = false; break;
            case FUNCTION_2: frame_load();
                           readData(EQUID);
                           runMenu3 = true;
                           runMenu1 = false; break;
            case FUNCTION_3: runMenu1 = false;
                           run = false; break;
        }
    }

    while(runMenu2){
        div_diff_table();
        choice2 = show_menu_2();
        switch (choice2){
            case FUNCTION_1: frame_load();
                           print_div_diff_table(); break;
            case FUNCTION_2: frame_load();
                           print_Newton_interpolation(FORWARD); break;
            case FUNCTION_3: frame_load();
                           print_Newton_interpolation(BACKWARD); break;
            case FUNCTION_4: frame_load();
                           Newton_interpolation();
                           cal_value(NewtonInter, NORMAL, FORWARD); break;
            case FUNCTION_5: frame_load();
                           Newton_interpolation();
                           draw_graph(fx1, fx2, NewtonInter, inf[0].X, inf[inf.size() -
                               1].X); break;
            case FUNCTION_6: frame_load();
                           runMenu2 = false; break;
        }
    }

    while(runMenu3){
        diff_table();
        choice3 = show_menu_3();
        switch (choice3){
            case FUNCTION_1: frame_load();
                           print_diff_table(); break;
            case FUNCTION_2: frame_load();
                           print_Newton_interpolation_equidistly(FORWARD); break;
        }
    }
}

```

```

        case FUNCTION_3: frame_load();
            print_Newton_interpolation_equidistly(BACKWARD);          break;
        case FUNCTION_4: frame_load();
            Newton_interpolation_equidistly(FORWARD);
            cal_value(NewtonInterEquid, EQUID, FORWARD);          break;
        case FUNCTION_5: frame_load();
            Newton_interpolation_equidistly(FORWARD);
            draw_graph(fx1, fx3, NewtonInterEquid, infEquid[0].X,
                infEquid[infEquid.size() - 1].X); break;
        case FUNCTION_6: frame_load();
            runMenu3 = false;                                         break;
    }
}
}

/* Display menu and select function */
int show_menu_1(void){
    cleardevice();
    frame();

    setcolor(RED);
    settextstyle(0, 0, 4);
    outtextxy(450, 110, (char*)"MENU");
    settextstyle(0, 0, THICK_WIDTH);

    int curFunction = FUNCTION_1;
    MOVE_MENU_1:
    for(int i = 220, j = FUNCTION_1; j <= MAX_CHOICE_1; i += 70, j += 1){
        if(j == curFunction) highlight(200, i, (char *)menu_1[j - 1]);
        else outtextxy(200, i, (char*)menu_1[j - 1]);
    }
    do{
        int n = getch();
        if(n == KEY_UP && curFunction >= FUNCTION_2){
            curFunction--;
            goto MOVE_MENU_1;
        }
        else if(n == KEY_DOWN && curFunction < MAX_CHOICE_1){
            curFunction++;
            goto MOVE_MENU_1;
        }
        else if(n == KEY_ENTER) return curFunction;
    }
    while(true);
}

int show_menu_2(void){
    cleardevice();
    frame();

    setcolor(RED);
    settextstyle(0, 0, 4);
    outtextxy(350, 110, (char*)"Noi suy moc bat ki");

```

```

    settextstyle(0, 0, THICK_WIDTH);

    int curFunction = FUNCTION_1;
    MOVE_MENU_2:
    for(int i = 200, j = FUNCTION_1; j <= MAX_CHOICE_2; i += 50, j += 1){
        if(j == curFunction) highlight(200, i, (char *)menu_2[j - 1]);
        else outtextxy(200, i, (char*)menu_2[j - 1]);
    }
    do{
        int n = getch();
        if(n == KEY_UP && curFunction >= FUNCTION_2){
            curFunction--;
            goto MOVE_MENU_2;
        }
        else if(n == KEY_DOWN && curFunction < MAX_CHOICE_2){
            curFunction++;
            goto MOVE_MENU_2;
        }
        else if(n == KEY_ENTER) return curFunction;
    }
    while(true);
}

int show_menu_3(void){
    cleardevice();
    frame();

    setcolor(RED);
    settextstyle(0, 0, 4);
    outtextxy(345, 110, (char*)"Noi suy moc cach deu");
    settextstyle(0, 0, THICK_WIDTH);

    int curFunction = FUNCTION_1;
    MOVE_MENU_3:
    for(int i = 200, j = FUNCTION_1; j <= MAX_CHOICE_3; i += 50, j += 1){
        if(j == curFunction) highlight(200, i, (char *)menu_3[j - 1]);
        else outtextxy(200, i, (char*)menu_3[j - 1]);
    }
    do{
        int n = getch();
        if(n == KEY_UP && curFunction >= FUNCTION_2){
            curFunction--;
            goto MOVE_MENU_3;
        }
        else if(n == KEY_DOWN && curFunction < MAX_CHOICE_2){
            curFunction++;
            goto MOVE_MENU_3;
        }
        else if(n == KEY_ENTER) return curFunction;
    }
    while(true);
}

```

```

void highlight(int x, int y, char* text){
    setbkcolor(YELLOW);
    outtextxy(x, y, text);
    setbkcolor(BLACK);
}

/* get data from file */
void readData(int TYPE){
    inf.clear(); infEquid.clear();
    // open file, get and print data
    FILE* fileIn = fopen(TYPE == NORMAL ? inFile[NORMAL] : inFile[EQUID], "r");

    DRAW(FIELDWIDTH + 3); DRAWFILE(fileOut, FIELDWIDTH + 3);
    printf("\n|   %-*s|   %-*s|\n", FIELDWIDTH/2 - 4, "x", FIELDWIDTH/2 - 4, "fx");
    fprintf(fileOut, "\n|   %-*s|   %-*s|\n", FIELDWIDTH/2 - 4, "x", FIELDWIDTH/2 - 4, "fx");
    DRAW(FIELDWIDTH + 3); DRAWFILE(fileOut, FIELDWIDTH + 3);

    double x, fx;
    while(!feof(fileIn)){
        fscanf(fileIn, "%lf %lf", &x, &fx);
        printf("\n| %-*.*lf| %-*.*lf|", FIELDWIDTH/2 - 2, precision, x,
            FIELDWIDTH/2 - 2, precision, fx);
        fprintf(fileOut, "\n| %-*.*lf| %-*.*lf|", FIELDWIDTH/2 - 2, precision, x,
            FIELDWIDTH/2 - 2, precision, fx);
        TYPE == NORMAL ? inf.push_back({x, fx}) : infEquid.push_back({x, fx});
    }
    printf("\n"); fprintf(fileOut, "\n");
    DRAW(FIELDWIDTH + 3); DRAWFILE(fileOut, FIELDWIDTH + 3);
    printf("\n"); fprintf(fileOut, "\n");
    fclose(fileIn);
}

/* calculate divided differences table */
void div_diff_table(void){
    divDiffTable.clear();
    FOR(0, inf.size(), i){
        divDiffTable.push_back(vector<double> (inf.size() + 1, 0));
        divDiffTable[i][0] = inf[i].X;
        divDiffTable[i][1] = inf[i].FX;
    }
    FOR(1, inf.size(), i){
        FOR(2, i + 2, j){
            divDiffTable[i][j] = (divDiffTable[i][j - 1] - divDiffTable[i - 1][j - 1]) / (divDiffTable[i][0] - divDiffTable[i - 1][0]);
        }
    }
}

/* print divided differences table */
void print_div_diff_table(void){
    printf("\nBang ty hieu:\n");
    fprintf(fileOut, "\nBang ty hieu:\n");
}

```

```

DRAW(FIELDWIDTH/2*(inf.size() + 1) + inf.size() + 2); DRAWFILE(fileOut,
    FIELDWIDTH/2*(inf.size() + 1) + inf.size() + 2);
printf("\n|  %-*s|  %-*s|*s\n", FIELDWIDTH/2 - 4, "x", FIELDWIDTH/2 - 4,
    "fx", FIELDWIDTH/2*(inf.size() - 1) + inf.size() - 1, "|");
fprintf(fileOut, "\n|  %-*s|  %-*s|*s\n", FIELDWIDTH/2 - 4, "x", FIELDWIDTH/2
    - 4, "fx", FIELDWIDTH/2*(inf.size() - 1) + inf.size() - 1, "|");
DRAW(FIELDWIDTH/2*(inf.size() + 1) + inf.size() + 2); DRAWFILE(fileOut,
    FIELDWIDTH/2*(inf.size() + 1) + inf.size() + 2);

FOR_EACH(v, divDiffTable){
    printf("\n|"); fprintf(fileOut, "\n|");
    FOR(0, v.size(), i){
        printf(" %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
        fprintf(fileOut, " %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
    }
}
printf("\n"); fprintf(fileOut, "\n");
DRAW(FIELDWIDTH/2*(inf.size() + 1) + inf.size() + 2); DRAWFILE(fileOut,
    FIELDWIDTH/2*(inf.size() + 1) + inf.size() + 2);
printf("\n"); fprintf(fileOut, "\n");
}

/* calculate multiply table */
void mul_table(int TYPE){
    mulTable.clear();
    int type = (TYPE == FORWARD ? 0 : inf.size() - 1);
    mulTable.push_back(vector<double> (inf.size(), 0));
    mulTable[0][0] = -inf[type].X;
    mulTable[0][1] = 1;
    FOR(1, inf.size() - 1, i){ // use Hoocner schema
        mulTable.push_back(vector<double> (inf.size(), 0));
        mulTable[i][i + 1] = 1;
        FOR_(i, 1, j){
            mulTable[i][j] = mulTable[i - 1][j - 1] - inf[abs(type - i)].X *
                mulTable[i - 1][j];
        }
        mulTable[i][0] = - inf[abs(type - i)].X * mulTable[i - 1][0];
    }
}

/* print multiply table */
void print_mul_table(void){
    printf("\nBang tinh tich: \n");
    fprintf(fileOut, "\nBang tinh tich: \n");

    DRAW(FIELDWIDTH/2*inf.size() + inf.size() + 1); DRAWFILE(fileOut,
        FIELDWIDTH/2*inf.size() + inf.size() + 1);
    FOR_EACH(v, mulTable){
        printf("\n|"); fprintf(fileOut, "\n|");
        FOR(0, v.size(), i){
            printf(" %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
            fprintf(fileOut, " %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
        }
    }
}

```

```

    }
    printf("\n"); fprintf(fileOut, "\n");
    DRAW(FIELDWIDTH/2*inf.size() + inf.size() + 1); DRAWFILE(fileOut,
        FIELDWIDTH/2*inf.size() + inf.size() + 1);
}

/* Newton Interpolation formula */
void Newton_interpolation(void){
    NewtonInter.clear();
    mul_table(FORWARD);
    NewtonInter.push_back(divDiffTable[0][1]);
    FOR(1, inf.size(), i){
        NewtonInter.push_back(0);
        FOR(0, i + 1, j){
            NewtonInter[j] += divDiffTable[i][i + 1] * mulTable[i - 1][j];
        }
    }
}

/* print Newton Interpolation formula */
void print_Newton_interpolation(int TYPE){
    NewtonInter.clear();
    mul_table(TYPE);
    print_mul_table();
    printf("\n"); fprintf(fileOut, "\n");

    int type = (TYPE == FORWARD ? 0 : inf.size() - 1);
    NewtonInter.push_back(divDiffTable[type][1]);
    printf("--> f(x) = %.1f\n", precision, NewtonInter[0]);
    fprintf(fileOut, "--> f(x) = %.1f\n", precision, NewtonInter[0]);

    FOR(1, inf.size(), i){
        NewtonInter.push_back(0);
        FOR(0, i + 1, j){
            if(TYPE == FORWARD) NewtonInter[j] += divDiffTable[i][i + 1] * mulTable[i - 1][j];
            else NewtonInter[j] += divDiffTable[type][i + 1] * mulTable[i - 1][j];
        }
        printf("--> ");
        fprintf(fileOut, "--> ");
        print_poly(NewtonInter, 'x');
    }
    if(TYPE == FORWARD) {
        printf("Da thuc noi suy tien la: \n");
        fprintf(fileOut, "Da thuc noi suy tien la: \n");
    }
    else{
        printf("Da thuc noi suy lui la: \n");
        fprintf(fileOut, "Da thuc noi suy lui la: \n");
    }
    print_poly(NewtonInter, 'x');
    printf("\n"); fprintf(fileOut, "\n");
}

```

```

/* print polynomial */
void print_poly(Poly P, char c){
    printf("f(%c) = ", c);
    fprintf(fileOut, "f(%c) = ", c);

    FOR(0, P.size(), deg){
        if(P[deg] == 0) continue;
        if(deg == 0){
            printf("%.1f", precision, P[deg]);
            fprintf(fileOut, "%.1f", precision, P[deg]);
        }
        else if(P[deg] > 0){
            printf(" + ");
            fprintf(fileOut, " + ");
            if(P[deg] != 1){
                printf("%.1f", precision, P[deg]);
                fprintf(fileOut, "%.1f", precision, P[deg]);
            }
            if(deg == 1){
                printf("%c", c);
                fprintf(fileOut, "%c", c);
            }
            else{
                printf("%c^[%d]", c, deg);
                fprintf(fileOut, "%c^[%d]", c, deg);
            }
        }
        else if(P[deg] < 0){
            printf(" - ");
            fprintf(fileOut, " - ");
            if(P[deg] != -1){
                printf("%.1f", precision, -P[deg]);
                fprintf(fileOut, "%.1f", precision, -P[deg]);
            }
            if(deg == 1){
                printf("%c", c);
                fprintf(fileOut, "%c", c);
            }
            else{
                printf("%c^[%d]", c, deg);
                fprintf(fileOut, "%c^[%d]", c, deg);
            }
        }
    }
    printf("\n"); fprintf(fileOut, "\n");
}

/* calculate the polynomial value at one point */
void cal_value(Poly P, int TYPE1, int TYPE2){
    printf("\n"); fprintf(fileOut, "\n");
    double point;
    printf("Nhap diem can tinh gia tri: ");

```

```

scanf("%lf", &point);
fprintf(fileOut, "Nhap diem can tinh gia tri:");
fprintf(fileOut, "%. *lf\n", precision, point);

printf("\n"); fprintf(fileOut, "\n");
printf("So do Hoocne: \n");
fprintf(fileOut, "So do Hoocne: \n");

DRAW(FIELDWIDTH/2*(P.size() + 1) + P.size() + 2); DRAWFILE(fileOut,
    FIELDWIDTH/2*(P.size() + 1) + P.size() + 2);
printf("\n"); fprintf(fileOut, "\n");
printf("| %- *s|", FIELDWIDTH/2 - 3, "coeff");
fprintf(fileOut, "| %- *s|", FIELDWIDTH/2 - 3, "coeff");

FOR_(P.size() - 1, 0, i){
    printf(" % - *.*lf|", FIELDWIDTH/2 - 2, precision, P[i]);
    fprintf(fileOut, " % - *.*lf|", FIELDWIDTH/2 - 2, precision, P[i]);
}
printf("\n"); fprintf(fileOut, "\n");

DRAW(FIELDWIDTH/2*(P.size() + 1) + P.size() + 2); DRAWFILE(fileOut,
    FIELDWIDTH/2*(P.size() + 1) + P.size() + 2);
printf("\n"); fprintf(fileOut, "\n");

double point_t;
if(TYPE1 == EQUID && TYPE2 == FORWARD){
    point_t = (point - infEquid[0].X) / (infEquid[1].X - infEquid[0].X);
}
else if (TYPE1 == EQUID && TYPE2 == BACKWARD){
    point_t = (point - infEquid[infEquid.size() - 1].X) / (infEquid[1].X -
        infEquid[0].X);
}
else point_t = point;
double value = P[P.size() - 1];

printf("| % - *.*lf|", FIELDWIDTH/2 - 2, precision, point_t);
fprintf(fileOut, "| % - *.*lf|", FIELDWIDTH/2 - 2, precision, point_t);

printf(" % - *.*lf|", FIELDWIDTH/2 - 2, precision, value);
fprintf(fileOut, " % - *.*lf|", FIELDWIDTH/2 - 2, precision, value);

double value_temp = value;
FOR_(P.size() - 2, 0, i){
    value_temp = value_temp * point_t + P[i];
    printf(" % - *.*lf|", FIELDWIDTH/2 - 2, precision, value_temp);
    fprintf(fileOut, " % - *.*lf|", FIELDWIDTH/2 - 2, precision, value_temp);
}
printf("\n"); fprintf(fileOut, "\n");
DRAW(FIELDWIDTH/2*(P.size() + 1) + P.size() + 2); DRAWFILE(fileOut,
    FIELDWIDTH/2*(P.size() + 1) + P.size() + 2);
printf("\n"); fprintf(fileOut, "\n");
printf("\n"); fprintf(fileOut, "\n");
FOR_(P.size() - 2, 0, i){

```



```

    printf("value = %.*lf * %.*lf + %.*lf = %.*lf\n", precision, value,
           precision, point_t, precision, P[i], precision, value * point_t + P[i]);
    fprintf(fileOut, "value = %.*lf * %.*lf + %.*lf = %.*lf\n", precision, value,
            precision, point_t, precision, P[i], precision, value * point_t + P[i]);
    value = value * point_t + P[i];
}
printf("--> f(%.*lf) = %.*lf", precision, point_t, precision, value);
fprintf(fileOut, "--> f(%.*lf) = %.*lf", precision, point_t, precision, value);
printf("\n"); fprintf(fileOut, "\n");
}

/* calculate differences table */
void diff_table(void){
    diffTable.clear();
    FOR(0, infEquid.size(), i){
        diffTable.push_back(vector<double> (infEquid.size() + 1, 0));
        diffTable[i][0] = infEquid[i].X; // cot X
        diffTable[i][1] = infEquid[i].FX;
    }
    FOR(1, infEquid.size(), i){
        FOR(2, i + 2, j){
            diffTable[i][j] = diffTable[i][j - 1] - diffTable[i - 1][j - 1];
        }
    }
}

/* print differences table */
void print_diff_table(void){
    printf("\nBang sai phan: \n");
    fprintf(fileOut, "\nbang sai phan: \n");
    DRAW(FIELDWIDTH/2*(infEquid.size() + 1) + infEquid.size() + 2);
    DRAWFILE(fileOut, FIELDWIDTH/2*(infEquid.size() + 1) + infEquid.size() + 2);
    printf("\n|  %-*s|  %-*s|*s\n", FIELDWIDTH/2 - 4, "x", FIELDWIDTH/2 - 4,
           "fx", FIELDWIDTH/2*(infEquid.size() - 1) + infEquid.size() - 1, "|");
    fprintf(fileOut, "\n|  %-*s|  %-*s|*s\n", FIELDWIDTH/2 - 4, "x", FIELDWIDTH/2
            - 4, "fx", FIELDWIDTH/2*(infEquid.size() - 1) + infEquid.size() - 1, "|");
    DRAW(FIELDWIDTH/2*(infEquid.size() + 1) + infEquid.size() + 2);
    DRAWFILE(fileOut, FIELDWIDTH/2*(infEquid.size() + 1) + infEquid.size() + 2);

    FOR_EACH(v, diffTable){
        printf("\n|"); fprintf(fileOut, "\n|");
        FOR(0, v.size(), i){
            printf(" %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
            fprintf(fileOut, " %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
        }
    }
    printf("\n"); fprintf(fileOut, "\n");
    DRAW(FIELDWIDTH/2*(infEquid.size() + 1) + infEquid.size() + 2);
    DRAWFILE(fileOut, FIELDWIDTH/2*(infEquid.size() + 1) + infEquid.size() + 2);
    printf("\n"); fprintf(fileOut, "\n");
}

/* calculate multiply table with x equidistant*/

```

```

void mul_table_equid(int TYPE){
    mulTableEquid.clear();
    mulTableEquid.push_back(vector<double> (infEquid.size() - 1, 0));
    mulTableEquid[0][0] = 1;
    FOR(1, infEquid.size() - 1, i){
        mulTableEquid.push_back(vector<double> (infEquid.size() - 1, 0));
        mulTableEquid[i][i] = 1;
        int type = TYPE == FORWARD ? i : -i;
        FOR_(i - 1, 1, j){
            mulTableEquid[i][j] = mulTableEquid[i - 1][j - 1] - type *
                mulTableEquid[i - 1][j];
        }
        mulTableEquid[i][0] = - type * mulTableEquid[i - 1][0];
    }
}

/* print multiply table with x equidistant*/
void print_mul_table_equid(void){
    printf("\nBang tinh tich: \n");
    fprintf(fileOut, "\nBang tinh tich: \n");

    DRAW(FIELDWIDTH/2*(infEquid.size() - 1) + infEquid.size()); DRAWFILE(fileOut,
        FIELDWIDTH/2*(infEquid.size() - 1) + infEquid.size());
    FOR_EACH(v, mulTableEquid){
        printf("\n|"); fprintf(fileOut, "\n|");
        FOR(0, v.size(), i){
            printf(" %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
            fprintf(fileOut, " %-*.*lf|", FIELDWIDTH/2 - 2, precision, v[i]);
        }
    }
    printf("\n"); fprintf(fileOut, "\n");
    DRAW(FIELDWIDTH/2*(infEquid.size() - 1) + infEquid.size()); DRAWFILE(fileOut,
        FIELDWIDTH/2*(infEquid.size() - 1) + infEquid.size());
}

/* calculate factorial*/
long long factorial(int n){
    if(n == 1) return 1;
    else return n * factorial(n - 1);
}

/* Newton Interpolation formula with x equidistant*/
void Newton_interpolation_equidistly(int TYPE){
    NewtonInterEquid.clear();
    mul_table_equid(TYPE);
    int type = (TYPE == FORWARD ? 0 : infEquid.size() - 1);
    NewtonInterEquid.push_back(diffTable[type][1]);

    FOR(1, infEquid.size(), i){
        NewtonInterEquid.push_back(0);
        FOR(1, i + 1, j){
            if(TYPE == FORWARD) NewtonInterEquid[j] += diffTable[i][i + 1] *
                mulTableEquid[i - 1][j - 1] / factorial(i);
        }
    }
}

```

```

        else NewtonInterEquid[j] += diffTable[type][i + 1] * mulTableEquid[i -
            1][j - 1] / factorial(i);
    }
}

/* print Newton Interpolation formula with x equidistant*/
void print_Newton_interpolation_equidistly(int TYPE){
    NewtonInterEquid.clear();
    mul_table_equid(TYPE);
    print_mul_table_equid();

    printf("\n"); fprintf(fileOut, "\n");

    int type = (TYPE == FORWARD ? 0 : infEquid.size() - 1);
    NewtonInterEquid.push_back(diffTable[type][1]);
    printf("--> f(t) = %.*lf\n", precision, NewtonInterEquid[0]);
    fprintf(fileOut, "--> f(t) = %.*lf\n", precision, NewtonInterEquid[0]);

    FOR(1, infEquid.size(), i){
        NewtonInterEquid.push_back(0);
        FOR(1, i + 1, j){
            if(TYPE == FORWARD) NewtonInterEquid[j] += diffTable[i][i + 1] *
                mulTableEquid[i - 1][j - 1] / factorial(i);
            else NewtonInterEquid[j] += diffTable[infEquid.size() - 1][i + 1] *
                mulTableEquid[i - 1][j - 1] / factorial(i);
        }
        printf("--> ");
        fprintf(fileOut, "--> ");
        print_poly(NewtonInterEquid, 't');
    }
    if(TYPE == FORWARD) {
        printf("Da thuc noi suy tien moc cach deu la : \n");
        fprintf(fileOut, "Da thuc noi suy tien moc cach deu la : \n");
    }
    else{
        printf("Da thuc noi suy lui moc cach deu la : \n");
        fprintf(fileOut, "Da thuc noi suy lui moc cach deu la : \n");
    }
    print_poly(NewtonInterEquid, 't');
    printf("\n"); fprintf(fileOut, "\n");
}

// standard function
double fx1(double x){
    return 1 / (25 * pow(x, 2) + 1);
    // return 0;
}

// Newton polynomial interpolation
double fx2(Poly P, double x){
    double value = P[P.size() - 1];
    FOR_(P.size() - 2, 0, i){

```

```

        value = value * x + P[i];
    }
    return value;
}

// Newton polynomial interpolation with x equidistant
double fx3(Poly P, double x){
    double t = (x - infEquid[0].X)/(infEquid[1].X-infEquid[0].X);
    double value = P[P.size() - 1];
    FOR_(P.size() - 2, 0, i){
        value = value * t + P[i];
    }
    return value;
}

// draw functions f1 and f2 in paragraphs [a, b] on the new graphic console
void draw_graph(double (*f1)(double x), double (*f2)(Poly P, double x), Poly P,
    double a, double b){
    closegraph();
    initwindow(800, 800);

    int x1 = 100, y1 = 100, x2 = 700, y2 = 700;
    double fx1, fx2, maxf = f1(a), minf = f1(a), h = 0.01, r = a;

    if(f1(a) > f2(P, a)) maxf = f1(a), minf = f2(P, a);
    else maxf = f2(P, a), minf = f1(a);

    while(r <= b){
        fx1 = f1(r);
        fx2 = f2(P, r);
        if(fx1 < minf) minf = fx1;
        if(fx1 > maxf) maxf = fx1;
        if(fx2 < minf) minf = fx2;
        if(fx2 > maxf) maxf = fx2;
        r += h;
    }

    double n = x2 - x1, m = y2 - y1;
    double k = (maxf - minf)/m;
    h = (b - a)/n;
    double x0 = x1 + (int)(-a/h + 0.5), y0 = y2 - (int)(-minf/k + 0.5);

    setcolor(BLUE);
    line(x1 - 20, y0, x2 + 20, y0);
    line(x2 + 10, y0 - 10, x2 + 20, y0);
    line(x2 + 10, y0 + 10, x2 + 20, y0);

    line(x0, y1 - 20, x0, y2 + 20);
    line(x0 - 10, y1 - 10, x0, y1 - 20);
    line(x0 + 10, y1 - 10, x0, y1 - 20);

    settextstyle(0, 0, 3);
    outtextxy(x2, y0 + 14, (char*)"x");

```

```
outtextxy(x0 - 18, y1 - 10, (char*)"y");

rectangle(1, 1, 99, 99);

setcolor(CYAN);
line(10, 35, 50, 35);
outtextxy(55, 25, (char*)" : fx1");

setcolor(RED);
line(10, 60, 50, 60);
outtextxy(55, 50, (char*)" : fx2");

setcolor(CYAN);
FOR(0, n + 1, i){
    double x = x1 + i;
    fx1 = f1(a + i*h);
    double y = (int)((fx1 - minf)/k + 0.5) + y1;
    y = y2 - y + y1;
    if(i == 0) moveto(x, y);
    else lineto(x, y);
}

setcolor(RED);
FOR(0, n + 1, i){
    double x = x1 + i;
    fx2 = f2(P, a + i*h);
    double y = (int)((fx2 - minf)/k + 0.5) + y1;
    y = y2 - y + y1;
    if(i == 0) moveto(x, y);
    else lineto(x, y);
}
getch();
closegraph();
initwindow(WIDTH, HEIGHT);
}
```
