به نام خدا

مبانی برنامه سازی با C/C++

دانشکده م.کامپیوتر، دانشگاه صنعتی شریف

تمرین سری چهارم | مهلت تحویل: 7 خرداد 86

تمرین ها را تا نیمه شب زمان تحویل به آدرس ای-میل درس ارسال نمایید.

email	ce153c@gmail.com
subject	hw4-StudentID
Body	خالی بماند
Attachment	Main.cc || main.c || main.cpp

توجه کنید:

- در این تمرین برخلاف تمرین های قبلی به جای نوشتن چند برنامه کوچک می بایست یک برنامه بـزرگ تحویل دهید. به همین دلیل ممکن است در طی کد زدن دچار مشکلات بسیاری شـوید. بـه ایـن منظـور توصیه می کنیم کد زدن را از همین امروز شروع کنید!
- یکی از مهمترین مراحل در برنامه نویسی *Debug* کردن کد می باشد. اصول پایـه ای *Debugging* در کلاس های تمرین به شما آموزش داده خواهند شد. خداوند کد شما را از *bug*ها حفظ کناد![1]
- ضمن تشویق شما به همکاری فکری در حل این تمرین در صورت شباهت بیش از اندازه بین کـد شـما و دیگران نمره منفی دریافت خواهید کرد.
- پاسخ های خود را همراه با توضیحات مبسوط ارسال نمایید.

[1] دیباگ کردن در اصطلاح به تصحیح ایراد های یک شی، (عموما یک برنامه کامپیوتری) اطلاق می شود. یعنی شما (که لزوماً سازنده برنامه نیستید.) با خواندن کد ها و عملکرد برنامه به کمک ابزارهای خاص ایرادات برنامه را پیدا و اصلاح کنید. نقاط شکست هم به خطوطی می گویند که از نظر دیباگر (کسی که دیباگ می کند) نقطه هایی از برنامه هستند که چموش بازی در می آورند. این خطوط را می توان مشخص کرد تا برنامه در هنگام اجرای آنها متوقف شده و از آن پس تحت نظارت اجرا شود. در صورت داشتن یک ابزار دیباگ خوب حتی می توان روندی را که به ایجاد اشکال منتهی شده است را نیز پیدا کرد.

ریشه شناسی کلمه دیباگ *Debug* نیز جالب است. در افسانه ها(!) آمده است که روزی روزگاری کامپیوتر *ENIAC Mark I* در هنگام کار کردن دچار مشکل شد. اپراتور ها بعد از کلی تلاش توانستند علت این ایراد را پیدا کنند. یک حشره (*bug*) در درون مدار های ماشین رفته بود و همان جا جان داده و مانع از عملکرد صحیح شده بود. به همین شکل اپراتور بعد از اصلاح ماشین گفت : "*من یک bug (حشره) پیدا کردم*" و از آن به بعد به کلیه اشکالات کامپیوتر (که مطمئنا دیگر تقصیر حشرات نبوده است) به باگ تعبیر می شود و روند رفع اشکال را دیباگ می گویند.

پ.ن.اگرچه بسیاری از افراد فکر می کنند که ریشه کلمه *bug* به داستان بالا بر می گردد اما چنین نیست. در حقیقت کلمه باگ مدت ها قبل برای عموم اشکالات ماشین وجود داشت. ولی چرا من با گفتن حقیقت یک داستان و باور خوب و زیبای تاریخی را خراب کرده ام؟

امروزه یکی از مهمترین معضلات در تقریب زدن داده ها در ریاضیات کاربردی درونیـابی تعـدادی نقطـه(معرف یک سیگنال گسسته در زمان – ورودی) به کمک یک تابع (معرف یک سیگنال پیوسته در زمان – خروجی) می باشد. کلی ترین صورت یک مسئله درونیابی به صورت زیر است:

n نقطه $x_1, \ldots, x_n$ و $y_1,\ldots,y_n$ داده شده است. چند جمله ای $p_{n-1}(x)$ از درجه حـداکثر $n-1$ را به نحوی بیابید به طوری که برای هر $0<i\leq n$ داشته باشیم $p_{n-1}(x_i) = y_i$

برای مثال تابع $p_2(x) = 1 + 4x - 2x^2$ نقاط (1,3), (3,-5), (-2,-15) را درونیابی می نماید.
انواع مختلفی از درونیابی بر حسب کارایی و دقت آنها وجود دارند. هدف شما در این پروژه پیاده سازی چند درونیـابی کارامد می باشد.

- **درونیابی به کمک ماتریس واندرموند[۲]:**

فرض کنید چند جمله ای p(x) به صورت زیر باشد:

$$p_{n-1}(x) = a_1 + a_2x + a_3x^2 + \ldots + a_nx^{n-1} = y$$

بدیهی است که با حل معادله ماتریسی زیر می توان ضرایب a_i را بدست آورد:

$$\begin{bmatrix} 1 & x_1 & x_1^2 & \cdots & x_1^{n-1} \\ 1 & x_2 & x_2^2 & \cdots & x_2^{n-1} \\ 1 & x_3 & x_3^2 & \cdots & x_3^{n-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \cdots & x_n^{n-1} \end{bmatrix} \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ \vdots \\ a_n \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix}$$

حل و بررسی این ماتریس بر عهده شما!

---

[2] *Vandermonde Matrix*

- **درونیابی چند جمله ای و درونیابی هرمیت مکعبی:**

درونیابی به کمک ماتریس واندوموند شیوه سرراستی برای تقریب است اما ایراد اساسی آن کندی بیش از حد است. در این راستا تقریب های بهتر و کارامد تری ارائه شده است. نمونه از این تقریب ها تقریب هرمیت درجه سه می باشد.

فرق اساسی این درونیابی با موارد مشابه به روش واندرموند دخیل کردن مشتقات تابع در محاسبه تابع است تا نمودار های هموارتری بدست آید[۳].

به صورت خلاصه در مسئله درونیابی هرمیت شما می بایست با داشتن مقادیر x_r, x_l, y_r, y_l و مقادیر مشتقات s_r,s_l که در آن r,l به ترتیب به معنای چپ و راست می باشند ضرایب a,b,c,d را در معادله زیر پیدا نمایید:

$$q(z) = a + b(z-x_L) + c(z-x_L)^2 + d(z-x_L)^2(z-x_R)$$

البته با توجه به اینکه:

$$q(x_L) = y_L \qquad q(x_R) = y_R$$
$$q'(x_L) = s_L \qquad q'(x_R) = y_R$$

برای اینکار با در نظر گرفتن مشتق عبارت فوق

$$q'(z) = b + 2c(z-x_L) + d(2(z-x_L)(z-x_R) + (z-x_L)^2)$$

ضرایب a,b,c,d به آسانی بدست خواهند آمد.

---

[3] تعریف دقیق اینکه واقعاً یک نمودار هموار smooth به چه معنی می باشد را در ریاضیات 2 ~~فرا خواهید گرفت~~ یاد گرفته اید (ولی). در حال حاضر تنها به شهود خود در این زمینه تکیه کنید.

ورودی/خروجی:

ورودی خـود را از standard input بخوانیـد و خروجـی تـان را بـر روی standard output بنویسید.[4]

1. **ورودی برنامه:**

ورودی برنامه شما ابتدا عدد n (تعداد نقاط) و بعـد از ان n زوج نقطـه بـه شـکل x_i,y_i می باشد.

2. **خروجی های برنامه:**

- برنامه شما می بایست درونیابی نقاط ورودی را به کمک ماتریس واندرموند و روش هرمیت درونیابی کند. توجه کنید که باید هر دو درونیابی را پیاده سـازی نماییـد. در انتها چند جمله ای که حاصل از درونیابی نقطه ها هستند را بنویسید.

3. **نمره اضافی:**

- پیاده سازی موارد زیر مشمول نمره اضافی خواهد بود:

کشیدن نمودار چند جمله ای درونیابی شده بر روی صفحه مـانیتور بـا کمـک کتابخانه های گرافیکی موجود برای C/C++

موفق باشید.

پروفسور شاپور:

[4] برای این کار از *header file* های *stdio.h*(برای زبان C) و *iostream* (برای زبان C++) استفاده کنید.