برنامه‌سازی پیشرفته

تمرین کامپیوتری شماره ۱

مدرس: رامتین خسروی

طراحان: الهه خداوردی، شهریار عطار

مهلت تحویل: چهارشنبه ۱۸ مهر ۱۴۰۳، ساعت ۲۳:۵۵

مقدمه

هدف از این تمرین آشنایی شما با container-ها و iterator-هاست. در این تمرین شما یک نمونه ساده شده از یک برنامه مدیریت مالی را پیاده‌سازی کنید.

علاوه بر اجرای درست برنامه، جدا کردن مسئولیت‌ها به تابع‌های مختلف، تمیزی کد، استفاده از container-ها و iterator-ها اهمیت زیادی دارند. سعی کنید در پیاده‌سازی تمرین، توابع مختلفی تعریف کنید که هر کدام تنها یک کار انجام می‌دهند.

برنامه مدیریت مالی

در این برنامه شما قرار است برای مدیریت هزینه‌های روزانه، برنامه‌ای پیاده‌سازی کنید. در این برنامه هزینه مربوط به امور مختلف وارد شده و براساس مجموعه‌های مختلف، هزینه‌های مربوط به هر مجموعه ذخیره خواهد شد. به کمک این برنامه فرد می‌تواند میزان هزینه‌ها خود را برای هر مجموعه مدیریت کند.

تعریف مجموعه

با وارد کردن این دستور، می‌توانیم مجموعه‌های جدید ایجاد کرده و برای هر مجموعه زیرمجموعه‌هایی خواهیم داشت. تضمین می‌شود که هر مجموعه حداکثر پنج زیرمجموعه داشته باشد (می‌تواند هیچ زیرمجموعه‌ای نداشته باشد). همچنین تضمین می‌شود که نام‌های مجموعه‌ها یکتا می‌باشد و نیازی به بررسی این مورد ندارید.

قالب ورودی برای تعریف یک مجموعه جدید

```
new_category <category_name> <subcategory_1> <subcategory_2> … <subcategory_n>
```

برای مثال می‌توانیم یک مجموعه «حمل و نقل» با زیر بخش‌های مترو، اتوبوس و اسنپ تعریف کنیم. برای این کار از دستور زیر استفاده می‌کنیم.

مثال ورودی برای تعریف یک مجموعه جدید

```
new_category transportation subway bus snapp
```

اضافه کردن هزینه‌ها

با این دستور هزینه‌ها مربوط به هر بخش را اضافه می‌کنیم. اگر می‌خواهیم به یک مجموعه هزینه‌ها را اضافه کنیم تنها اسم آن بخش را می‌نویسیم، اما اگر خواستیم به یک زیرمجموعه هزینه‌ها را اضافه کنیم به صورت category/subcategory باید اسم‌ها را وارد کنیم (تضمین می‌شود در نام مجموعه‌ها کاراکتر / وجود ندارد).

قالب ورودی برای اضافه کردن هزینه‌ها

```
add_expense <item_of_a_category> <amount>
```

مثال ورودی برای اضافه کردن هزینه‌ها

```
add_expense transportation/snapp 1000
```

حذف کردن هزینه‌ها

با این دستور، آخرین هزینه ورودی یک بخش را حذف می‌کنیم. توجه داشته باشید که با این کار صرفا آخرین هزینه وارد شده حذف می‌شود و بقیه هزینه‌ها تغییری نمی‌کنند، همچنین ممکن است که چند بار این دستور وارد شود و چندین هزینه متوالی از یک بخش حذف شود ولی همواره تضمین می‌شود که هزینه‌ای برای حذف وجود دارد.

قالب ورودی برای حذف کردن هزینه‌ها

```
remove_expense <item_of_a_category>
```

مثال ورودی برای حذف کردن هزینه‌ها

```
remove_expense transportation/snapp
```

نمایش گزارش

با این دستور کاربر می‌تواند یک گزارش از موجودی و هزینه‌های مربوط به همه ی مجموعه‌ها را ببیند.
این گزارش به ترتیب شامل موارد زیر است:

- برای هر مجموعه موارد زیر به ترتیب نمایش داده خواهد شد:
  - نام مجموعه
  - جمع کل هزینه‌ها مربوط به زیر مجموعه‌ها (دقت کنید که هزینه‌های خود مجموعه جزو این بخش حساب می‌شوند)
  - جمع کل هزینه‌های هر زیرمجموعه

ترتیب مجموعه‌ها بر اساس نام به صورت صعودی، و ترتیب زیرمجموعه‌ها بر حسب مجموعه هزینه‌ها باشد به صورت نزولی می‌باشد، در صورتی که زیرمجموعه‌ها دارای هزینه برابر باشند بر اساس نام به صورت صعودی مرتب می‌شوند. پیش از نمایش اطلاعات هر زیرمجموعه، 4 فاصله (space) و یک کاراکتر dash (-) و سپس نام زیر مجموعه و سپس هزینه آن بخش را داریم، و پس از هر مجموعه 10 کاراکتر dash نیز خواهیم داشت، پس از آخرین مجموعه به جای کاراکتر dash از کاراکتر ستاره (asterisk) استفاده کنید. اگر مجموعه‌ای وجود نداشت نیز تنها کافی است یک خط خالی را چاپ کنید. اگر زیرمجموعه‌ای نیز وجود نداشت صرفا خط اول شامل Subcategories expenses: را چاپ کنید.
ممکن است پس از report نیز یک عدد وارد شده باشد، در این حالت باید زیرمجموعه‌هایی که مجموع هزینه‌هایشان از آن عدد کمتر است را از نمایش دادن در خروجی حذف کنید. (دقت کنید که تمام مجموعه‌ها در هر صورت نمایش داده شوند)

قالب ورودی برای نمایش گزارش

```
report <threshold>
```

قالب خروجی برای نمایش گزارش

```
Category: <category>
Total expenses: <total_expenses>
Subcategories expenses:
    - <subcategory_name>: <total_subcategory_expenses>
    - <subcategory_name>: <total_subcategory_expenses>
    - <subcategory_name>: <total_subcategory_expenses>
----------
Category: <category>
Total expenses: <total_expenses>
Subcategories expenses:
    - <subcategory_name>: <total_subcategory_expenses>
    - <subcategory_name>: <total_subcategory_expenses>
    - <subcategory_name>: <total_subcategory_expenses>
**********
```

مثال اول ورودی برای نمایش گزارش

```
report 100
```

مثال اول خروجی برای نمایش گزارش

```
Category: Transportation
Total expenses: 5000
Subcategories expenses:
    - snap: 3000
    - bus: 1950
----------
Category: University
Total expenses: 7000
Subcategories expenses:
    - fee: 6000
    - food: 400
    - books: 150
**********
```

مثال دوم ورودی برای نمایش گزارش

```
report
```

مثال دوم خروجی برای نمایش گزارش

```
Category: GiftShop
Total expenses: 6300
Subcategories expenses:
    - fee: 6000
    - books: 300
----------
Category: Transportation
Total expenses: 4450
Subcategories expenses:
    - snap: 3000
    - bus: 1400
    - taxi: 50
**********
```

نکات و نحوه تحویل

- کد خود را در قالب یک فایل با نام A1-SID.cpp در صفحه eLearn درس بارگذاری کنید که SID شماره دانشجویی شماست. برای مثال اگر شماره‌ی دانشجویی شما ۸۱۰۱۰۲۰۰۰ باشد، نام فایل شما باید A1-810102000.cpp باشد. دقت کنید که در صورتی که از قالب گفته شده پیروی نکنید از نمره شما کاسته می‌شود.
- برنامه شما باید در سیستم عامل لینوکس و با مترجم g++ با استاندارد c++20 ترجمه و در زمان معقول برای ورودی های آزمون اجرا شود.
- **بخش مهمی از ارزیابی برنامه شما درستی عملکرد آن است. بنابراین به اندازه کافی ورودی‌های آزمایشی طراحی کنید تا درستی خروجی در حالت‌های مختلف آزموده شود و دقت کنید که اگر برنامه شما با مثال‌های داده شده درست کار کند لزوما به معنای درستی در تمام سناریوها نیست.**
- درستی برنامه شما از طریق آزمون‌های خودکار سنجیده می‌شود. به این ترتیب، لازم است خروجی تولید شده از نظر بزرگی و کوچکی حروف، رعایت فاصله‌ها، عدم وجود خروجی‌های اضافه، ... دقیقا مانند نمونه‌های داده شده باشد. بنابراین پیشنهاد می‌شود که با استفاده از ابزارهایی مانند diff فرمت خروجی برنامه خود را با خروجی‌هایی که در اختیارتان قرار داده شده است مطابقت دهید.
- هدف این تمرین یادگیری شماست. لطفا تمرین را خودتان انجام دهید. در صورت کشف تقلب مطابق سیاست درس با آن برخورد خواهد شد.

نمرات

- تمیزی کد
  - رعایت کردن نام‌گذاری صحیح و انسجام[1]
  - عدم وجود کد تکراری
  - رعایت دندانه‌گذاری[2]
  - استفاده **صحیح** از متغیرهای ثابت[3] به جای Magic Value-ها[4]
  - استفاده **صحیح** از container-ها و iterator-ها به جای روش‌های قدیمی
  - توابع کوتاه که فقط یک کار را انجام می‌دهند و عبارت‌های لامبدا
- درستی کد
  - آزمون‌های خودکار

دقت کنید که موارد ذکر شده لزوما کل نمره شما را تشکیل نمی‌دهند و ممکن است با تغییراتی همراه باشند.

---

[1] Consistency

[2] Indentation

[3] Constant

[4] به مقادیر خاصی که در کد استفاده می‌شود و برای عملکرد صحیح کد ضروری است اما دلیل استفاده از آن‌ها مشخص نیست و قابل جایگزین شدن با یک ثابت با اسم مشخص جهت افزایش خوانایی هستند، magic value گفته می‌شود. برای آشنایی بیشتر با این مفهوم می‌توانید این لینک را مشاهده کنید.