بسمه تعالی

پروژه پایانی مبانی برنامه‌نویسی

پاییز ۱۳۹۵

مقدمه

در این پروژه میخواهیم بازی تایپ کلمات را پیاده سازی کنیم. در بازی اصلی کلمه‌ای، از بالای صفحه وارد شده و به سمت پایین حرکت میکند، و کاربر باید قبل از خارج شدن کلمه از کادر، آن را تایپ کند. ما قصد داریم حالت ساده شده‌ای از این بازی را پیاده سازی کنیم. یکی از کاربردهای این بازی تخمین سرعت تایپ برای یک فرد می‌باشد، سرعت تایپ یک فرد با WPM اندازه‌گیری می‌گردد، این واحد تعداد کلمات تایپ شده بر دقیقه می‌باشد که روش‌های استانداردی برای آن وجود دارد ولی در این پروژه ما آن به صورت کاملا غیر استاندارد که در ادامه شرح داده می‌شود، محاسبه می‌کنیم.

تعریف پروژه

بازی تایپ کلمات

تعدادی فایل که شامل تعدادی از کلمات می‌باشند به شما داده می‌شوند. اسم این فایل‌ها به شکل level-?.txt است که در آن علامت ؟ با یک عدد جایگزین شده است مثلا level-2.txt یا level-20.txt. در اینجا فرض شده است که اگر فایل level-x.txt به شما داده شده است حتما همه فایلهای level-y.txt که y<x است نیز داده شده است. در این فایل‌ها کلمات با فاصله از یکدیگر جدا شده‌اند و همگی از حروف **کوچک انگلیسی** تشکیل شده‌اند. هر فایل معادل یک مرحله از بازی است. در ادامه کلیات بازی و در انتها با یک مثال جزییات آن شرح داده شده است.

با اجرای برنامه، قبل از هر کاری شما باید تعداد مراحل را از روی تعداد فایلهایی که به شما داده شده است بدست آورید. برای اینکار میتوانید یا از دستوراتی که لیست فایلها را میدهند مثلا dir در windows استفاده کنید یا با یک ترتیب فایلها را باز کنید و از روی فایلهایی که باز نمیشود تشخیص دهید که چند مرحله داریم.

در ابتدای بازی، کاربر بعد از وارد کردن اسم خود، یا بازی را که قبلا انجام داده است ادامه می‌دهد یا یک بازی جدید را شروع می‌کند. برای شروع بازی جدید، مرحله بازی را انتخاب می‌کند که یک عدد است، حداکثر مقداری که می‌تواند وارد کند توسط برنامه بر اساس شماره فایل‌های level-?.txt به وی نشان داده می‌شود (ورودی غلط با پیغام مناسب به کاربر اعلام می‌شود). سپس برنامه شما می‌بایست فایل مربوط به آن مرحله از بازی را باز کرده و تمامی کلمات آن را در یک لیست پیوندی ذخیره کنید تا دسترسی راحت‌تری به کلمات آن داشته باشید.

در ادامه به ترتیب کلمه‌ها را به صورت تصادفی از لیست پیوندی بیرون کشیده و آن را از لیست حذف می‌کنید. کلمه بیرون کشیده شده را در بالای صفحه نمایش می‌دهید. کاربر باید سعی کند کلمه مورد نظر را در کمترین زمان ممکن تایپ کند. بنابراین بعد از نمایش کلمه به کاربر، شما اندازه‌گیری زمان این کلمه را شروع می‌کنید. اگر حرفی که کاربر وارد کرده است صحیح باشد، آن حرف در کلمه نمایش داده شده تبدیل به حرف بزرگ می‌شود و به سراغ حرف بعدی می‌روید (جزییات در مثالی موجود است) در غیر این صورت منتظر هستید که کاربر حرف دیگری را وارد کند. زمانی که کاربر همه حروف یک کلمه را درست وارد کرد، زمان تایپ این کلمه را محاسبه کرده و در امتیازدهی که در ادامه شرح داده می‌شود استفاده می‌کنید. با نوشته شدن کل کلمه، کلمه بعدی نمایش داده میشود. این کار تا اتمام کلمات ادامه میابد. در انتها امتیاز کل این مرحله محاسبه شده و به کاربر نشان داده می‌شود و از وی سوال پرسیده می‌شود که آیا قصد دارد به مرحله بعد برود یا نه. اگر جواب مثبت

بود این فرایند مجددا برای مرحله جدید تکرار می‌شود در غیر این صورت برنامه خاتمه پیدا می‌کند و اگر کاربر بخواهد قبل از اتمام بازی، وضعیت فعلی آن ذخیره می‌شود.

به غیر از حروف کوچکی که کاربر برای تایپ کلمات وارد می‌کند، اگر در حین بازی حروف زیر وارد شوند، می‌بایست بازی به شرح جدول زیر واکنش نشان دهد: (توجه شود که این حروف بزرگ می‌باشند تا با حروف کلمات اشتباه نشوند.)

حرف	رویداد
Q	بازی اتمام می‌یابد.
P	بازی متوقف می‌شود. (یعنی محاسبه زمان متوقف می‌شود.)
R	بازی ادامه پیدا می‌کند. (یعنی محاسبه زمان از سر گرفته می‌شود.)

الگوریتم امتیازدهی

در این بازی، امتیاز کاربر طبق الگوریتم زیر محاسبه می‌شود

۱- امتیاز هر کلمه:

$$\text{امتیاز کلمه} = \frac{\text{تعداد حروف کلمه} * ۳ - \text{تعداد حروف نادرست تایپ شده}}{\text{زمان تایپ کلمه به ثانیه}}$$

۲- امتیاز هر مرحله:

$$\text{امتیاز مرحله} = \frac{\text{مجموع امتیاز کلمات}}{\text{تعداد کلمات}}$$

۳- امتیاز هر کاربر:

$$\text{امتیاز کاربر} = \text{مجموع امتیاز مراحل}$$

شما می‌بایست در **پایان هر مرحله و بازی (نیازی به نمایش امتیاز در هنگام بازی نیست)** امتیاز کاربر را نمایش دهید.

کاربران بازی

در ابتدا ورود به برنامه می‌بایست نام کاربر پرسیده شود، اگر کاربر قبلا وارد بازی شده باشد بتواند از آخرین **مرحله‌ای (نیازی به بازیابی کلمه نیست تنها می‌بایست آخرین مرحله و امتیاز کاربر تا آن مرحله بازیابی شود)** که در آن قرار داشته است ادامه دهد.

راهنمایی: برای این منظور می‌بایست از فایل‌ها استفاده کنید و بهتر است در این فایل برای هر کاربر یک struct در نظر بگیرید.

اتمام بازی

بازی به چندین طریق ممکن است به اتمام برسد

۱- کاربر حرف Q را وارد کند
۲- یک مرحله به اتمام برسد و کاربر نخواهد ادامه دهد
۳- کاربر همه مراحل را بازی کرده است و مرحله جدید وجود ندارد

۴- پنجره console که بازی در آن اجرا میشود بسته شود، یا سیستم عامل crash کند، یا کامپیوتر بسوزد یا .... (به صورت بازگشتی کهکشان راه شیری منفجر شود یا ....)

مدیریت کردن مرحله ۴ جزء وظایف برنامه شما نیست!!!! (۱۰۰۰ نمره اضافی در صورت مدیریت ☺) ولی در مراحل ۱ و ۲ و ۳ قبل از اینکه برنامه تمام شود از کاربر پرسیده می‌شود که آیا وضعیت فعلی آن ذخیره شود یا نه؟ اگر جواب مثبت باشد، مرحله‌ای که برنامه در اجرای بعدی از آن شروع خواهد شد (در حالت ۱ مرحله‌ای که الان در آن بازی می‌کرد، حالت ۲ مرحله بعدی و در حالت ۳ مرحله بعدی وجود ندارد) و امتیاز کاربر تا آخرین مرحله‌ای که به صورت کامل بازی شده است ذخیره می‌شود.

روند کلی

برای جمع بندی روند کلی بازی از ابتدا مرور می‌کنیم. ورودی‌های کاربر با **رنگ سبز** مشخص شده‌اند.

۱. در ابتدا نام کاربر پرسیده می‌شود:

```
Player name: parham
```

۲. منو اصلی بازی نمایش داده می‌شود:

```
Hi parham :-) R U ready to play?!
[1] Play a new game
[2] Resume an old game
1
```

۳. در صورتی شروع یک بازی جدید انتخاب شود، کاربر می‌بایست مرحله شروع را انتخاب کند:

```
Okay, let's start
Please Enter the Level (now, I have at most 13 levels): 2
```

۴. با انتخاب مرحله بازی شروع می‌شود:

```
hello
h
Hello
a
Hello
b
Hello
e
HEllo
Q
```

```
:-( Exit?!
Save current state?[Y/N]
```

نکات پیاده‌سازی

- کاربر را خیلی حرف-گوش-کن در نظر نگیرید بنابراین هر ورودی دلخواهی را ممکن است وارد کند که برنامه شما باید آنرا مدیریت کند.
- حتما! حتما! حتما! حتما! برای هر کار مشخصی یک تابع بنویسید.
- هرگز! هیچگاه! Never، به هیچ وجه من الوجوه! کامپایل و تست برنامه را برای آخر کار نگذارید. هر تابعی که می‌نویسید کامپایل کنید و سعی کنید آنرا تست کنید.
- سعی کنید در زمانه فرجه یا زمان بین امتحانات بر روی نحوه پیاده‌سازی پروژه فکر کرده و تا حد ممکن آنرا انجام دهید.
- بهتر است برای اینکه با یک فایل خیلی بزرگ سر و کله نزنید، کد برنامه را به چندین فایل بشکنید.
- لزومی ندارد همه قابلیت‌های برنامه را از ابتدا بخواهید در نظر بگیرید. برنامه شما میتواند چند نسخه داشته باشد. بهتر است از یک نسخه خیلی ساده شروع کنید. مثلا حالتی که کاربری وجود ندارد و فقط یک مرحله وجود دارد. توسعه برنامه در چند نسخه اولا به شما کمک می‌کند که گام به گام برنامه را بهبود دهید. ثانیا در انتهای هر نسخه برنامه‌ای دارید که کار می‌کند و اگر به هر دلیل موفق به تکمیل همه قابلیت‌ها نشدید حداقل می‌توانید این نسخه را تحویل دهید.
- چند مورد هست که در کلاس درس به آنها اشاره نشده ولی برای انجام این پروژه لازم دارید مثلا اندازه‌گیری زمان بین دو مرحله از برنامه (در این جا از نمایش کلمه تا تایپ درست و کامل آن). برای این موارد باید متوسل/متوکل/... به Google شوید (که احتمالا سر از سایت stackoverflow درخواهید آورید). به این فرایند عادت کنید چرا که مابقی عمرتان به این نحوه خواهد گذشت ☺/☹

موارد امتیازی

- جدول امتیازات ۱۰ کاربر برتر را نمایش دهید.
- سیستم‌های امتیازدهی مختلفی وجود داشته باشد و کاربر بتواند از بین آن‌ها انتخاب کند (طراحی و پیاده‌سازی آن به عهده شما است)
- افزایش زیبایی محیط بازی (مثلا رنگی کردن حرفی که الان باید تایپ شود): برای استفاده بهتر از محیط کنسول و رسم شکل و تغییر رنگ در آن کتابخانه‌های مختلفی وجود دارد، مثلا در سیستم‌های UNIX محور می‌توانید از کتابخانه‌ی ncurses استفاده کنید و در سیستم‌های ویندوزی می‌توانید از کتابخانه‌ی windows.h استفاده نمایید.
- در نسخه‌ی اصلی بازی کلمات هر چند ثانیه یکبار یک خط به پایین می‌آیند، سعی کنید این ویژگی را پیاده‌سازی کنید. به عنوان راهنمایی می‌توانید تابع kbhint را در اینجا ببینید. (توجه شود این تابع تنها در سیستم‌های ویندوزی وجود دارد).

تحویل پروژه

- پروژه به صورت تک نفره باید انجام شود و کار گروهی پذیرفته نیست.
- پروژه را تا ساعت ۲۳ روز جمعه ۱ بهمن باید در Quera آپلود کنید.
- تحویل حضوری پروژه روز شنبه ۲ بهمن خواهد بود.