به نام خدا

دانشکده مهندسی کامپیوتر دانشگاه اصفهان

شبیه‌ساز صفحه گسترده (Mini-Spreadsheet)

ساختمان داده‌ها - دکتر رضا رمضانی

طراحان:

پویان جای درنام، مهدی رضایی، علی طالبی، دریا ظهیری، امیررضا محمدی یگانه و زهرا مهدوی

پاییز ۱۴۰۴

مقدمه

پروژه‌ی کنونی به منظور طراحی یک شبیه‌ساز صفحه‌گسترده‌ی تعاملی[1] مبتنی بر کنسول تعریف شده است. سیستم با یک گرید[2] خالی با ابعاد مشخص آغاز به کار می‌کند. سپس دستورات را به‌صورت خط‌به‌خط از کاربر دریافت می‌نماید و تغییرات لازم را متناسب با دستور وارد شده، بر روی گرید اعمال می‌نماید.

در این پروژه، سلول‌ها می‌توانند مقدار ثابت، متن یا فرمول داشته باشند و فرمول‌ها باید تجزیه[3] و محاسبه شوند. همچنین در طی شبیه‌سازی، وابستگی‌ها باید مدیریت و هنگام بروز خطا، منتشر[4] شوند. پیاده‌سازی ساختمان داده‌های پشته و صف نیز باید به صورت دستی انجام شود.

اصطلاح شبیه‌سازی به معنای آن است که نیازی به طراحی واسط کاربری گرافیکی نیست و قابلیت‌های موردنظر می‌تواند با دریافت ورودی و نمایش خروجی متنی از طریق کنسول نیز انجام شود.

هدف پروژه

- شبیه‌سازی عملکرد یک صفحه‌گسترده‌ی ساده‌ی تعاملی در محاسبه‌ی مقادیر سلول‌ها.
- پیاده‌سازی و درک کاربرد عملی ساختمان‌داده‌های **پشته**[5] و **صف**[6] در پردازش داده‌ها.
- آشنایی با تبدیل و حل عبارت‌های ریاضی با استفاده از پشته.
- آشنایی با مدیریت وابستگی‌های پویا[7] در یک سیستم محاسباتی.
- تشخیص و مدیریت خطای وابستگی دورانی[8].
- آشنایی با الگوریتم‌های تجزیه کردن و ارزیابی عبارت‌های ریاضی[9].

---

[1] Interactive Spreadsheet Simulator
[2] Grid
[3] Parse
[4] Propagate
[5] Stack
[6] Queue
[7] Dynamic Dependency Management
[8] Circular Dependency
[9] Expression Evaluation

قابلیت‌ها

- محاسبه‌ی مقادیر سلول‌ها بر اساس مقادیر ثابت یا فرمول‌های ریاضی.
- پشتیبانی از فرمول‌های شامل عملگرها، مقادیر ثابت و ارجاع به سلول‌های دیگر.
- مدیریت و محاسبه‌ی مجدد و خودکار وابستگی‌های زنجیره‌ای پس از هر تغییر در سلول‌ها.
- رعایت اولویت‌بندی عملگرها در محاسبات فرمول‌ها.
- تشخیص، گزارش و انتشار خطاهای متداول (شامل خطای وابستگی دورانی).
- پشتیبانی از قابلیت پرکردن خودکار برای گسترش مقادیر یا فرمول‌ها در سلول‌های مجاور.

پیش‌نیازها

- آشنایی با ساختمان‌داده‌های پشته و صف.
- آشنایی با مبانی الگوریتم و پیاده‌سازی منطقی برنامه‌ها.
- آشنایی با عبارت‌های پایه‌ی ریاضی و اولویت عملگرها.
- آشنایی با مفاهیم برنامه‌نویسی شی‌گرا شامل کلاس‌ها، وراثت و چندریختی.
- تسلط نسبی بر مدیریت خطا و استثناها در کدنویسی.
- درک اصول کدنویسی تمیز و ساختارمند برای افزایش خوانایی و نگه‌داری کد.

ساختمان‌داده‌ها (۱۰۰ امتیاز)

- **پشته (Stack) (۴۰ امتیاز)**

برای ارزیابی عبارت‌ها و فرمول‌های ریاضی، مانند تبدیل Infix → Postfix و محاسبه‌ی عبارت‌های Postfix استفاده می‌شود.

- **صف (Queue) (۴۰ امتیاز)**

برای مدیریت ترتیب محاسبات سلول‌ها و حل وابستگی‌های زنجیره‌ای میان آن‌ها به کار می‌رود. این صف تضمین می‌کند که یک سلول پس از محاسبه‌ی تمام پیش‌نیازهایش ارزیابی شود.

  - می‌توانید یک صف حلقوی با آرایه و دو اندیس head و tail پیاده‌سازی کنید و یا یک لیست پیوندی ساده پیاده‌سازی کنید که دارای قابلیت‌های enqueue و dequeue می‌باشد.

- **آرایه (Array) (۲۰ امتیاز)**

توجه: پیاده‌سازی ساختمان‌داده‌های پشته و صف باید به‌صورت دستی انجام شود. استفاده از پیاده‌سازی‌های آماده در کتابخانه‌های استاندارد زبان‌های برنامه‌نویسی، موجب **کسر نمره‌ی کامل این بخش** خواهد شد.

ساختار و مدل‌های داده (۵۰ امتیاز)

برای مدیریت بهتر و تفکیک مسئولیت‌ها، توصیه می‌شود از کلاس‌ها یا ساختارهای مجزای زیر استفاده شود.

- کلاس **Cell**: کلاس نگهداری اطلاعات یک سلول شامل محتوای خام، مقدار نهایی، نوع داده، لیست وابستگی‌ها و وضعیت خطا.
- کلاس **Grid** یا **SpreadSheet**: کلاس مدیریت گرید (آرایه‌ی دو بعدی سلول‌ها) که مسئول اجرای محاسبات و مدیریت وابستگی‌ها است و فراخوانی ارزیابی مبتنی بر صف را انجام می‌دهد.
- کلاس **FormulaEvaluator**: کلاس مسئول تجزیه و محاسبه‌ی عبارات ریاضی با استفاده از پشته.

ویژگی‌ها و عملکردها

- **پشتیبانی از انواع محتوا**[10] **(۲۰ امتیاز)**
  - مقادیر عددی شامل اعداد صحیح و اعشاری (مثال ۱۰ یا ۲.۵۳)
  - فرمول‌ها: عبارات ریاضی شامل عملوندها و عملگرها
  - محاسبه‌ی مقادیر سلول‌ها بر اساس مقدارهای **متنی** محصور در Double Quotation مانند "Hello" از موارد **امتیازی** می‌باشد.
- **پشتیبانی از عملگرهای پایه (۵۰ امتیاز)**
  - جمع (+)، تفریق (-)، ضرب (*)، تقسیم (/).
- **پشتیبانی از عملوندها (۵۰ امتیاز)**
  - اعداد صحیح و اعشاری: با قالب استاندارد «بخش اعشاری . بخش صحیح».
  - ثابت‌های عددی: شامل عدد پی (PI = 3.14159) و عدد نپر (EN = 2.71828).
  - ارجاعات سلولی[11]: ارجاع به سلول‌های دیگر در گرید مانند A1، B5 یا Z10.
- **پشتیبانی از اولویت‌بندی محاسبات**[12] **(۵۰ امتیاز)**

سیستم باید قوانین ریاضی اولویت عملگرها را در ارزیابی فرمول‌ها رعایت کند؛ مانند اولویت ضرب و تقسیم نسبت به جمع و تفریق. پرانتزها نیز باید برای تغییر این اولویت‌بندی‌ها بررسی شوند.

---

[10] Content Types
[11] Cell References
[12] Operator Precedence

- **پشتیبانی از قابلیت پرکردن خودکار (AutoFill) (۴۰ امتیاز)**

به منظور گسترش مقادیر یا فرمول‌ها در سلول‌های مجاور از این قابلیت بهره گرفته می‌شود. این قابلیت به معنای آن است که وقتی یک مقدار یا فرمول در یک سلول نوشته می‌شود، می‌توان آن را به‌صورت خودکار در سلول‌های کناری (مثلاً سطر یا ستون مجاور) گسترش داد؛ بدون آنکه لازم باشد برای هر سلول، جداگانه آن مقدار یا فرمول وارد شود.

پیاده‌سازی: (۱۵۰ امتیاز)

در این پروژه، **سه بخش اصلی** برای پیاده‌سازی الگوریتمی مدنظر است:

۱. محاسبه‌ی عبارت‌ها و فرمول‌های ریاضی (۵۰ امتیاز)

این بخش مسئول ارزیابی فرمول‌های نوشته‌شده درون سلول‌ها است. یکی از روش‌های استاندارد برای انجام این کار، تبدیل عبارت Infix به Postfix با استفاده از پشته و سپس محاسبه‌ی عبارت Postfix با استفاده‌ی مجدد از پشته می‌باشد.

پیش از ارزیابی هر فرمول، برای مثال (A1 = B1 + C1)، باید مقادیر محاسبه‌شده‌ی سلول‌های ارجاع‌شده (در این مثال، B1 و C1) در جای مناسب جای‌گذاری شوند.

۲. مدیریت وابستگی و ارزیابی سلول‌ها (۵۰ امتیاز)

این بخش مسئول محاسبه‌ی زنجیره‌ای سلول‌ها پس از هر تغییر است. هنگامی که یک سلول مثلاً A1 به سلول‌های دیگر مانند B1 و C1 وابسته است، باید اطمینان حاصل شود که B1 و C1 قبل از A1 محاسبه شده‌اند. یکی از روش‌های مدیریت ترتیب محاسبات و تضمین پردازش سلول‌ها پس از تکمیل محاسبات پیش‌نیازهایشان، استفاده از ساختمان داده‌ی صف است.

۲,۱. مدیریت به‌روزرسانی و وابستگی

سیستم باید **وابستگی‌های بین سلول‌ها** را به‌صورت پویا مدیریت نماید؛ یعنی پس از هر دستور Set، وابستگی‌ها (ارجاعات بین سلول‌ها) به‌روز شوند. این بخش همچنین باید قادر باشد وابستگی‌های دورانی[۱۳] را تشخیص دهد و خطاها را منتشر کند؛ به طوری که اگر B1 خطا داشته باشد، A1 نیز خطای مربوطه را نشان دهد. پس از تنظیم موفقیت‌آمیز هر سلول، برنامه باید:

[13] Circular Dependency

- مقدار آن سلول و تمام سلول‌های وابسته به آن را، به‌صورت زنجیره‌ای، مجداً محاسبه و به‌روزرسانی کند؛
- جدول به‌روز شده را چاپ نماید. در صورت بروز خطا، باید مدیریت آن انجام شود.

سیستم در قابلیت AutoFill باید بتواند مقدار یا فرمول یک سلول را در محدوده‌ای از سلول‌های مجاور کپی کند. مثلاً اگر بنویسیم Fill(A1, B1:D1) و مقدار A1 برابر ۵ باشد، سیستم باید مقدار ۵ را در سلول‌های B1، C1 و D1 نیز قرار دهد. اگر A1 فرمولی مانند =B1+2 داشته باشد، همان فرمول در سلول‌های بعدی هم تکرار می‌شود تا مقادیرشان به‌طور خودکار پر شود.

۲,۲. مدیریت خطا، وابستگی دورانی و انتشار آن

در صورت بروز خطا در یک دستور (شامل: فرمول نامعتبر، ایجاد وابستگی دورانی یا خطای محاسباتی)، وضعیت آن سلول باید به حالت خطا تغییر یابد. این وضعیت خطا باید به تمام سلول‌هایی که به آن سلول خاص وابسته هستند، منتشر شود. در هنگام چاپ جدول، سلول‌هایی که در حالت خطا قرار دارند، باید با مقدار !#ERR نمایش داده شوند و زیر جدول، فهرستی از دلایل خطاها برای اطلاع به کاربر ارائه گردد.

۳. ذخیره‌سازی و دسترسی سریع به نتایج (۵۰ امتیاز)

مقادیر سلول‌ها (چه خام، چه محاسبه‌شده و چه دارای خطا) باید در ساختاری ذخیره شوند که دسترسی سریع به آن‌ها امکان‌پذیر باشد. استفاده از آرایه‌ی دو بعدی برای شبیه‌سازی گرید، دسترسی زمانی O(1) به محتوا یا مقدار هر سلول (با داشتن شماره سطر و ستون) را فراهم می‌کند.

توصیه‌های پیاده‌سازی

برای پیشبرد پروژه به‌صورت منظم و مرحله‌به‌مرحله، پیشنهاد می‌شود روند زیر را دنبال کنید:

۱. **پیاده‌سازی محاسبه‌ی عبارت‌های ریاضی برای یک سلول**

ابتدا الگوریتم ارزیابی فرمول‌های درون سلول را پیاده‌سازی کنید.

مثال) A1 = 5 * (3 + PI)

۲. **گسترش به چند سلول با ترتیب ساده**

برنامه را برای چند سلول آماده کنید؛ با فرض اینکه سلول‌ها به ترتیب منطقی و بدون وابستگی پیچیده وارد می‌شوند. در این مرحله، وابستگی‌های ساده بین سلول‌ها برقرار و محاسبه‌ی زنجیره‌ای اولیه تست می‌شود.

مثال) A1 = 10 → B1 = 5 → C1 = A1 + B1

۳. مدیریت وابستگی‌های درهم و نامرتب

در این مرحله، به سراغ وابستگی‌های پیچیده و نامرتب بروید. در این مرحله باید از ساختمان داده‌ی صف برای مدیریت ترتیب محاسبات و تضمین ارزیابی صحیح سلول‌ها استفاده شود.

مثال C1 = A1 + B1 → A1 = 10 → B1 = 5

٤. افزودن قابلیت تشخیص و انتشار خطا

- تشخیص وابستگی دورانی و خطاهای محاسباتی؛
- انتشار وضعیت خطا به سلول‌های وابسته؛
- نمایش !ERR# در جدول و ارائه‌ی لیست دلایل خطا در زیر جدول.

مدیریت خطا (۵۰ امتیاز)

خطاهای ممکن در این سیستم در **سه دسته** کلی طبقه‌بندی می‌شوند.

۱ .**خطاهای ورودی و فرمت** (Input / Format Error) **(۱۰ امتیاز)**

- فرمول با ساختار نامعتبر. مثال:
  - Mismatched Parentheses : C1 = 3 * 2 + 5)
  - Invalid Operator Sequence : B1 = + / 2
  - Missing Operand : C1 = 5 +

۲ .خطاهای محاسباتی (Arithmetic Error) **(۱۰ امتیاز)**

- خطاهای زمان اجرا مانند تقسیم بر صفر یا جذر عدد منفی. مثال:
  - Division by Zero : A1= B1/0
  - Invalid Argument for SQRT (Negative Number) : A1 = SQRT(-9)
  - Invalid Argument for LOG : B1 = LOG(-5) , A1 = LOG(0)
  - Invalid Exponentiation for Negative Base : A1 = (-4) ^ 0.5
  - Factorial Defined Only for Non-negative Integers : C1 = 2.5! , B1 = (-3)!
  - …

۳ .خطاهای وابستگی و ارجاع (Reference Error)

- **وابستگی دورانی (۱۰ امتیاز)**

زمانی که دو یا چند سلول به صورت مستقیم یا غیرمستقیم به یکدیگر وابسته باشند، این خطا پرتاب می‌شود.

- Circular Dependency: A1=B1+2 & B1=A1*2

- **ارجاع نامعتبر** (Bad Reference) **(۱۰ امتیاز)**

  ارجاع به سلولی خارج از ابعاد گرید. مثال:

  فرض کنید گرید موردنظر دارای ابعاد ۴×۵ است و ما می‌خواهیم به خانه A8 مقداری منتسب کنیم.

  - Bad Reference

- **ارجاع به مقدار نامعتبر** (Value Error) **(۱۰ امتیاز)**

  ارجاع به سلولی که حاوی مقدار متنی است و در یک عملیات ریاضی استفاده می‌شود. مثال

  - Value Error: A1= “Hello” & B1=A1+2

هدف‌های جانبی

- به‌کارگیری مفاهیم برنامه‌نویسی شی‌گرا[14] شامل کپسوله‌سازی، وراثت و چندریختی.
- رعایت اصول ساختارمندی و معماری نرم‌افزار مانند SOLID.
- افزایش قابلیت نگه‌داری، توسعه‌پذیری و خوانایی کد در طراحی نرم‌افزار.
- مدیریت خطاها و حالات استثنا[15] در جریان اجرای برنامه.
- رعایت اصول کدنویسی تمیز[16] برای خوانایی، فهم‌پذیری و نگهداری آسان کد ضروری است.
  - نام‌گذاری معنادار و واضح برای متغیرها و توابع؛
  - توابع کوتاه و تک‌مسئولیتی[17]؛
  - پرهیز از تکرار کد و استفاده از روش‌های متکی بر قابلیت استفاده مجدد[18]؛
  - سازماندهی منطقی کدها و مستندسازی مناسب برای هر کلاس و تابع.

پیچیدگی و بهینه‌سازی (۵۰ امتیاز)

- پیچیدگی زمانی و فضایی الگوریتم را در بدترین حالت بررسی کنید. (۲۵ امتیاز)
  - فرض کنید گرید ابعاد R × C دارد و پس از N دستورSet، محاسبه مجدد انجام می‌شود. پیچیدگی را به‌صورت مختصر توضیح دهید.

---

[14] Object-Oriented Programming
[15] Exception Handling
[16] Clean Code
[17] Single Responsibility
[18] Reusability

- روش‌های بهینه‌سازی (۲۵ امتیاز)
  - جلوگیری از محاسبه‌ی مجدد کل گرید.
  - محاسبه‌ی مجدد تنها سلول‌های وابسته پس از تغییر یک سلول.
  - استفاده از ساختارهای داده‌ای مناسب؛ مثلاً استفاده از صف برای مدیریت ترتیب محاسبات به منظور کاهش پیچیدگی زمانی
  - ذخیره‌سازی مقادیر محاسبه‌شده برای کاهش مصرف حافظه و دسترسی سریع‌تر

ورودی و خروجی

برنامه به‌صورت تعاملی اجرا می‌شود. در ابتدای برنامه، کاربر مقادیر R و C را وارد می‌کند و یک جدول به ابعاد $R \times C$ چاپ می‌شود. R معادل تعداد سطر و C معادل تعداد ستون است. سپس برنامه وارد حلقه دریافت دستورات می‌شود تا زمانی که دستور خروج توسط کاربر وارد شود.

ورودی

۱. خط اول — ابعاد گرید

دریافت دو عدد R و C (به ترتیب تعداد سطرها و ستون‌ها)

$$1 \leq R,C \leq 26$$

۲. دستورات — حلقه تعاملی

پس از دریافت ابعاد گرید، برنامه وارد حلقه دریافت دستور می‌شود و تا زمانی که کاربر دستور خروج را وارد نماید، منتظر ورودی کاربر می‌ماند. دستورات قابل قبول عبارتند از:

- تنظیم سلول (Set)
  - فرمت دستور: [Cell_Address] = [Content]

    فضاهای اطراف ’=‘ نادیده گرفته می‌شوند.
  - [Cell_Address]: آدرس استاندارد سلول، مانند A1، B5 و Z20.
  - [Content]: می‌تواند یکی از موارد زیر باشد:
    - مقدار متنی (String) داخل Double Quotation؛ مانند A1 = “Hello”.
    - مقدار عددی ثابت: عدد صحیح یا اعشاری؛ مانند A1 = 2 یا C5 = 2.36.
    - فرمول با مقادیر ثابت: شامل مقادیر ثابت و عملگرها؛ مانند A1 = 2*3.
    - فرمول وابسته: شامل ارجاع به سلول‌های دیگر؛ مانند A1 = B1*2 یا D1 = C1 + (A1 / PI).

- خروج (Exit)
  - دستوری مشخص مانند exit که اجرای برنامه را خاتمه می‌دهد.

خروجی

- خروجی اولیه (Initial Print)
  - پس از دریافت ابعاد و قبل از اولین دستور، کل گرید چاپ می‌شود.
  - تمام سلول‌ها با '–' (وضعیت مقداردهی نشده) نشان داده می‌شوند.
- خروجی پس از هر دستور (Automatic Update)
  - پس از دریافت هر دستور set (موفق یا مواجه شده خطا)، کل گرید بر اساس آخرین تغییرات مجدداً محاسبه و چاپ می‌شود.
- قالب‌بندی چاپ گرید به صورت زیر می‌باشد:
  - چاپ کل گرید (R × C)
  - سلول‌ها در هر سطر با کاراکتر کاما (',') از هم جدا می‌شوند.
  - مقادیر عددی به صورت صحیح و اعداد اعشاری تا حداکثر دو رقم اعشار چاپ می‌شوند.
  - مقادیر متنی: بدون کوتیشن‌ها چاپ می‌شوند.
  - سلول‌های خالی: سلول‌هایی که هنوز مقداردهی نشده‌اند، با '–' نمایش داده می‌شوند.
  - سلول‌های مواجه شده با خطا: سلول‌های دارای خطا با !ERR# نمایش داده می‌شوند.
- گزارش خطا[19]
  - پس از چاپ گرید، اگر خطایی در سلول‌ها وجود داشته باشد، لیستی از سلول‌هایی که منشأ اصلی خطا[20] هستند، به همراه آدرس و نوع خطا در خطوط جداگانه چاپ شود.

توجه: برای ایجاد درک بهتری از نحوه اعمال ورودی‌ها و نمونه‌هایی از خروجی‌های متناسب مورد انتظار، یک مستند جداگانه از چندین نمونه ورودی و خروجی در اختیار دانشجویان قرار خواهد گرفت.

[19] Error Reporting
[20] Root Cause

موارد امتیازی

- **پشتیبانی از قابلیت اضافه کردن سطر و ستون به جدول**

  راهنمایی: در این مورد باید به وابستگی‌ها و به‌روزرسانی فرمول سلول‌های جدول دقت کرد.

- **پشتیبانی از عملگرهای پیشرفته**

  پشتیبانی از عملگرهایی مانند **توان** (^)، **فاکتوریل** (!)، **لگاریتم** و... از موارد **امتیازی** می‌باشد.

- **پشتیبانی از توابع تجمعی و محدوده‌ها**

  پشتیبانی از توابع تجمعی و توابعی که بر روی محدوده‌ای از سلول‌ها عمل می‌کنند (مانند SUM و AVG).

- **پشتیبانی از قابلیت تاریخچه جدول**

  پشتیبانی از قابلیت برگرداندن (Undo یا Ctrl+Z) جدول به حالت قبل از اجرای دستور پیشین

- **پشتیبانی از نمایش توابع (رسم نمودار)**

  امکان رسم نمودار داده‌های یک محدوده

- **پشتیبانی از رابط کاربری گرافیکی**

  پیاده‌سازی یک رابط کاربری گرافیکی (GUI) ساده برای نمایش گرید و ورود داده‌ها.

- **سایر موارد**

  گسترش قابلیت‌ها بسته به خلاقیت، مانند:

  - نحوه ذخیره و بارگذاری فایل
  - قالب‌بندی سلول‌ها
  - امکانات خلاقانه دیگر

توجه: بخش‌های امتیازی پروژه شامل ارزیابی با تست‌کیس نمی‌شوند و دانشجویان باید برای موارد امتیازی پروژه خود، در صورت امکان، تست‌کیس‌های مناسب بسازند و هنگام ارائه از آنها استفاده نمایند.

نکات تکمیلی

- **زبان پیاده‌سازی:** زبان‌های قابل ارزیابی شامل C++، Java و C# می‌باشد.
- **مهلت اتمام:** مهلت تکمیل پیاده‌سازی این پروژه تا روز یکشنبه ۲۵ آبان‌ماه ساعت ۰۶:۰۰ صبح می‌باشد.
- **گروهبندی:** این پروژه به‌صورت **فردی** انجام می‌شود.
- **بستر پیاده‌سازی:** توسعه و پیاده‌سازی این پروژه در بستر **گیت‌هاب** و **سامانه کوئرا** انجام می‌شود.

- روند توسعه پروژه در قالب کامیت‌های **متوالی و معنادار** بر روی برنچ Spreadsheet و تحلیل پیچیدگی روی برنچ Analysis باید انجام گیرد.
- ساخت **برنچ‌های متعدد** و رعایت **اصول نوشتار صحیح متن کامیت** توصیه می‌شود.

- **نحوه ارزیابی:** ارزیابی عملکرد پروژه به‌صورت تست‌کیس-محور و همچنین **ارائه حضوری** انجام می‌شود.
  - ارزیابی نهایی در قالب ارائه ۲۰ دقیقه‌ای و براساس **بارم‌بندی بخش‌های مختلف مستند** انجام می‌شود.