به نام خداوند جان و خرد


# پروژه‌ی نهایی

گزارش کار

حل سودوکو به‌کمک پشته

**درس:**

ساختمان‌های داده

**استاد:**

استاد محرم منصوری‌زاده

**نویسنده:**

محمد امین چیتگرها

(۹۸۱۲۳۵۸۰۱۵)

زمستان ۱۳۹۹

اسفند ماه

# پیاده‌سازی و الگوریتم حل

این برنامه، با نام زودوکو، برای حل تند یک جدول سودوکو ساخته شده است. در پیاده‌سازی، تنها از نگه‌دارنده‌هایی شامل پشته و آرایه استفاده شده است.

در این‌جا، دو کلاس اصلی با نقش‌های زیر هستند:

- App: مدیریت کلیت برنامه، ورودی‌ها و خروجی‌ها (چه فایل و چه رابط خط‌فرمان)، خواندن و نوشتن در فایل‌ها و مدیریت برخی خطاها (برخی خطاها نیز در تابع `main()` مدیریت‌شده و نمایانده می‌شوند).
- SudokuSolver: حل‌کننده‌ی یک جدول سودوکو. تابع اصلی این کلاس، `solve()` است که پردازش حل جدول را انجام می‌دهد.

برای حل، برنامه ابتدا یک‌بار همه‌ی خانه‌های جدول را پیمایش می‌کند. این پیمایش، برای یافتن اطلاعات خانه‌های خالی و مقدارهای موجود در هر بلوک است. منظور از بلوک، یک سطر، یک ستون یا یک مربع ۳×۳ است؛ و می‌دانیم که در هر بلوک، هر عدد دقیقا یک‌بار باید باشد.

داده‌های تک‌تک خانه‌های خالی، درون یک پشته، به ترتیب دیده‌شدن، گذاشته می‌شوند. منظور از داده‌های هر خانه‌ی خالی، مکان جدولی آن و مقدارهای ممکن آن است (که در ابتدا خالی است). مقدارهای موجود در هر بلوک نیز، برای دسترسی سریع، در آرایه‌ای ۹تایی از بولین نگه داشته می‌شوند. به بیان دیگر، برای نمونه، برای همه‌ی ۹ سطر جدول، آرایه‌ای ۹تایی هست که هر عضو آن نمایش‌دهنده‌ی وجود آن مقدار در آن سطر می‌باشد؛ و به همین ترتیب برای دو نوع بلوک دیگر.

در این مرحله، با پیمایش در خانه‌های خالی، مقدارهای ممکن هریک، با توجه به مقدارهای (نا)موجود در بلوک‌های سه‌گانه، تنظیم می‌شوند. نوع این مجموعه مقدارها نیز پشته است.

لازم به ذکر است که، برای هر پشته، یک پشته‌ی پشتیبان نیز در نظر گرفته شده است، تا بتوان بدون استفاده از روش بازگشتی، پردازش‌ها را به شیوه‌ی برنامه‌نویسی پویا (DP) انجام داد (برای جابجایی از یک وضعیت به وضعیت دیگر).

در این‌جا، که خانه‌های خالی و مقدارهای ممکن هریک از آن‌ها مشخص‌اند، و پیش از شروع مرحله‌ی اصلی حل، به‌کمک ۹ پشته، پشته‌ی خانه‌های خالی بر پایه‌ی شمار مقدارهای ممکن هر خانه‌ی خالی مرتب می‌شوند. بدین ترتیب، خانه‌های خالی‌ای که تنها یک مقدار ممکن می‌توانند به خود بگیرند، در بالای پشته (دسترسی زودتر)، و برعکس آن‌هایی که مقدارهای ممکن بیش‌تری دارند، در پایین پشته دیده می‌شوند. دلیل این کار، افزایش احتمال انتخاب درست برای هر خانه‌ی خالی است.

سپس، خانه‌های خالی شروع به پر شدن می‌کنند. به‌کمک آزمون و خطا، مقدارهای ممکن هر خانه‌ی خالی گذاشته شده، و سپس به خانه‌ی بعدی رفته می‌شود. این عمل تا جایی ادامه پیدا می‌کند که یا

خانه‌های خالی پایان یابد (که در این حالت جدول به درستی پر شده است)، یا این‌که خانه‌ای یافته شود که هیچ مقدار ممکنی برای آن وجود نداشته باشد. در این صورت، دو قدم به عقب برگشته، و شبیه بازگشتی، مقدارهای امتحان‌نشده، امتحان می‌شوند. اهمیت پشته‌های پشتیبان، هنگام بازگشت از مسیری اشتباه مشخص می‌گردد؛ با توجه به این‌که پیاده‌سازی نیز بازگشتی نیست.

در هر آزمایش نیز، مقدارهای موجود در بلوک‌های جدول بروز می‌گردند؛ و برای همه‌ی مقدارهای ممکن هر یک از خانه‌های خالی بعدی، این موجود بودن، پیش از جای‌گذاری مقدار، بررسی می‌گردد. به بیان ساده‌تر، اگر در خانه‌ی خالی‌ای در سطر اول ۲ گذاشته شود، و در آینده در خانه‌ی خالی دیگری که در همان سطر اول است نیز ۲ جزو مقدارهای ممکن باشد، پیش از قرار داده‌شدن، بررسی می‌شود که در بلوک‌های مشترک (در این‌جا مثلا سطر اول) چنین مقداری قبلا گذاشته نشده باشد.

البته، قطعا کد نشان‌دهنده‌ی جزییات و توضیحات بسیار بیش‌تری است. در کد، بسیار تلاش شده است نام‌های پرمفهوم (تابع‌ها، نوع‌ها و متغییرها) و روند روان تابع‌ها و پیمانه‌ای و کوتاه بودن آن‌ها وجود داشته باشد تا فهم آن هر چه بیش‌تر ساده گردد. هم‌چنین، برای توضیحات کامل‌تر، در بخش هدر کلاس `SudokuSolver`، توضیحات درون‌کدیِ بسیار گسترده‌ای داده شده است.

## روند کلی برنامه و نمونه خروجی

برنامه، برای ورودی گرفتن جدول، از فایل با قالب CSV بهره می‌برد. تا زمانی که بخواهید، برنامه از شما مسیر فایل ورودی را می‌گیرد، سودوکو را حل می‌کند، در صورت نیاز خروجی را نمایش داده و به درخواست شما (کاربر)، خروجی را در یک فایل CSV دیگر (یا همان فایل ورودی) ذخیره می‌کند.

نمونه‌ی نمایی کلی از روند اجرای برنامه و خروجی‌ها و ورودی‌های آن

# موارد دیگر...

## کامپایل کردن

برای این کار، می‌توانید از دستورهای زیر به ترتیب استفاده کنید:

```
cmake -B build
make -C build
```

سپس، با اجرای دستور زیر، برنامه اجرا خواهد شد:

```
./build/Zudoku
```

## ابزارهای استفاده‌شده

گیت، دستیار همیشگی من، در روند رشد این برنامه نیز کنار من بود و به من کمک کرد. هم‌چنین، در این برنامه از سه کتاب‌خانه (به صورت پیوند ایستا) استفاده شده است؛ که مهم‌ترین آن ابزار خواندن فایل‌های CSV بود (می‌توانید جزییات این کتاب‌خانه‌ها را در فایل `.gitmodules` ببینید). این کتاب‌خانه‌ها در پوشه‌ی lib هستند.

## ویژگی‌های برنامه

### حساس بودن به خطاها و ورودی

برنامه به حد خوبی به اشتباه بودن ورودی حساس است. جدول نادرست (مانند جدولی که دو عدد ۳ در یک سطر دارد)، فایل CSV نادرست یا ناموجود، و خطاهای دیگر همگی گرفته شده و به کاربر بازخورد داده می‌شوند.

### طراحی کد جامع و زیبا و ساده

کد به شیوه‌ی بسیار خوبی ساده است و می‌توان روند برنامه را از آن فهمید. هم‌چنین تلاش شده است در طراحی کلاس‌ها و تابع‌ها، نکته‌های ریز نیز در نظر گرفته شوند.

### بهینه‌شده

برنامه در بسیاری از بخش‌ها بهینه‌سازی بسیاری انجام شده است. برای نمونه، برای انتقال یک مقدار از یک پشته به پشته‌ی دیگر، تابع بهینه‌شده‌ای برای همین در نظر گرفته شده است که به جای عملیات رونوشت، از عملیات انتقال استفاده می‌کند.

## مثال‌های آماده

برای سادگی کار، مثال‌هایی در پوشه‌ی examples در قالب CSV هستند (در دو دسته‌ی درست و نادرست) که می‌توانید برای صرفه‌جویی در زمان، از آن‌ها نیز برای آزمایش استفاده کنید.

## گواهی‌نامه



موفق باشید.