Basics of Programming Mini Project

Winter 2026

Course Instructor: Dr. Saeed Reza Kheradpisheh

1. مقدمه و آشنایی با بازی

مینی‌پروژه‌ی امسال بازی اتللو (Othello) یا همان Reversi هست. این بازی ، یک بازی دو نفره استراتژیک است و روی یک صفحه شطرنجی انجام می‌شود. هدف نهایی این است که در پایان بازی، تعداد مهره‌های شما روی صفحه بیشتر از مهره‌های حریف باشد.

تاریخچه

بازی اُتلُّو ریشه در بازی قدیمی‌تری به نام ریورسی (Reversi) دارد که در سال ۱۸۸۳ در انگلستان توسط لوئیس واترمن و جان مولت مستقل از هم ابداع شد؛ هر دو یکدیگر را متقلب می‌دانستند. این بازی در پایان قرن نوزدهم در انگلستان محبوبیت یافت.

نسخهٔ مدرن این بازی با نام اُتلُّو (Othello) در سال ۱۹۷۱ توسط گورو هاسگاوا یک فروشندهٔ ۳۸ ساله در ژاپن، اختراع و ثبت اختراع شد. نام «اُتلُّو» از نمایشنامهٔ شکسپیر الهام گرفته شده و نماد تضاد رنگ‌های سیاه و سفید و همچنین «حسادت» («وحش چشم‌سبز» در نمایشنامه) است. رنگ سبز صفحهٔ بازی نیز به میدان نبرد اُتلُّو اشاره دارد.

هاسگاوا در سال ۱۹۷۳ انجمن ژاپن اُتلُّو را تأسیس کرد و اولین مسابقهٔ ملی را برگزار نمود. شرکت ژاپنی Tsukuda Original نیز همان سال بازی را به‌صورت تجاری در ژاپن عرضه کرد که موفقیت فوری یافت. در سال ۱۹۷۵، شرکت Gabriel Industries آن را در آمریکا منتشر کرد. تاکنون بیش از ۴۰ میلیون نسخه در بیش از ۱۰۰ کشور فروخته شده و فروش جهانی آن از ۶۰۰ میلیون دلار فراتر رفته است.

چشم انداز

در این مینی‌پروژه، انتظار می‌رود بازی اتللو (Othello) را به‌صورت یک برنامهٔ کنسولی بر اساس مشخصات و محدودیت‌های ذکر شده در این راهنما پیاده‌سازی کنید. هدف اصلی این است که بتوانید منطق یک بازی دو نفره را در یک محیط متنی پیاده‌سازی کرده و قوانین اصلی آن — از جمله قرار دادن مهره، وارونه کردن مهره‌های حریف، و تعیین برنده در پایان بازی — را به درستی پیاده‌سازی نمایید. این پروژه فرصتی برای به‌کارگیری مفاهیمی مانند حلقه‌ها، شرط‌ها، توابع و دیگر مفاهیم پایه ای برنامه نویسی است.

شروع بازی

بازی با یک شبکه مربعی (معمولا 8×8) شروع می‌شود. در ابتدای بازی، 4 مهره در مرکز صفحه قرار می‌گیرند (دو مهره سیاه و دو مهره سفید به صورت ضربدری). بازی توسط مهره سیاه آغاز می‌شود.

قوانین حرکت

شما باید مهره خود را در خانه‌ای خالی قرار دهید که بتوانید یک یا چند مهره حریف را بین مهره جدید خود و مهره‌های قبلی‌تان «محاصره» کنید. این محاصره می‌تواند در راستای افقی، عمودی یا مورب رخ دهد.

تغییر رنگ مهره‌ها

تمام مهره‌های حریف که در محاصره قرار می‌گیرند، برمی‌گردند (Flip می‌شوند) و به رنگ مهره شما در می‌آیند.

رد شدن نوبت و پایان بازی

اگر بازیکنی در نوبت خود، هیچ حرکت معتبری نداشته باشد، نوبت او رد می‌شود و حریف دوباره بازی می‌کند. اگر هر دو بازیکن نتوانند حرکتی انجام دهند، بازی پایان می‌یابد و تعداد مهره‌ها شمارش شده و برنده مشخص می‌شود.

بازی آنلاین و جزئیات قوانین

برای انجام آنلاین بازی می‌توانید به این لینک مراجعه کنید.
همچنین می‌توانید شرح کامل قوانین بازی را در این سایت مطالعه کنید.

2 - توضیحات پیاده‌سازی

در این بخش، یک تقسیم‌بندی پیشنهادی برای پیاده‌سازی بازی داده شده است، اما شما می‌توانید پروژه را طبق برنامه‌ریزی خود یا با هماهنگی منتور پیش ببرید.

فاز 1: منو (Menu)

ابتدا باید یک منوی اصلی ایجاد شود که شامل گزینه‌های زیر باشد:

- New Game
- Load Game (امتیازی)
- Help
- Game History
- Exit

1- New Game

پس از انتخاب شروع بازی، کاربر دو گزینه خواهد داشت:

1. بازی تک‌نفره (Single Player): در این حالت، کاربر نام خود را وارد می‌کند و بازیکن مقابل یک ربات خواهد بود.
2. بازی دونفره (Two Player): در حالت دونفره، از کاربر خواسته می‌شود نام هر دو بازیکن را وارد کند.

2- Load Game (امتیازی)

با انتخاب این گزینه، آخرین بازی ناتمام که ذخیره شده است بارگذاری می‌شود و بازیکنان می‌توانند ادامه بازی را انجام دهند.

3- Help

با انتخاب این گزینه، توضیحی برای کاربر ارائه می‌شود تا نحوه استفاده از منو و قوانین بازی را متوجه شود. (مثلا نحوه‌ی حرکت در صفحه بازی، کارکرد دکمه‌های کیبورد و ...)

4- Game History

در این بخش سوابق بازی‌های انجام‌شده نمایش داده شود.
برای هر بازی، اطلاعات زیر باید ثبت و نمایش داده شود:

- نام بازیکن اول
- نام بازیکن دوم
- تعداد مهره‌های هر بازیکن در پایان بازی
- نام بازیکن برنده (یا مساوی بودن بازی)
- تاریخ و زمان انجام بازی

لیست بازی‌ها باید بر اساس تاریخ انجام بازی (جدیدترین بازی در ابتدا) مرتب شود.

5- Exit

با انتخاب این گزینه برنامه در ترمینال بسته می‌شود.

فاز 2: محیط بازی

بازی باید کاملاً در محیط ترمینال (Console / Terminal) پیاده‌سازی شود. استفاده از هرگونه کتابخانه گرافیکی مجاز نیست.

محیط بازی باید شامل موارد زیر باشد:

- شبکه‌ای از خانه‌ها: یک جدول 8×8 که نمایانگر صفحه بازی است.
- نمادها: برای نمایش مهره‌ها و خانه‌های خالی از کاراکترهای مناسب استفاده کنید:
  - خانه خالی: نشان‌دهنده فضایی که هنوز مهره‌ای در آن نیست.
  - مهره سیاه: مثلا ● یا B (یا هر کاراکتر مشخص دیگر).
  - مهره سفید: مثلا ○ یا W.
  - پیشنهاد حرکت (امتیازی): مکان‌هایی که بازیکن مجاز است مهره بگذارد (مثلاً با * مشخص شود).
- نوبت : اسم یا رنگ بازیکنی که باید حرکت کند را نمایش دهید.

فاز 3: شروع بازی و چیدمان اولیه

بعد از شروع بازی، جدول مستطیلی شکل مطابق فاز قبل ساخته می‌شود.

چیدمان اولیه مهره‌ها:

- دقیقا 4 مهره در مرکز جدول (خانه‌های سطر 4 و 5 و ستون 4 و 5) قرار می‌گیرند:
  - خانه (4,4) و (5,5): سفید
  - خانه (4,5) و (5,4): سیاه
- نوبت اولیه همیشه با مهره سیاه است.

حرکت بازیکن:

- بازیکن باید از کلیدهایی مانند W-A-S-D یا Arrow Keys، برای حرکت روی صفحه (انتخاب خانه) استفاده کند.

فاز 4: منطق حرکت و برگرداندن مهره‌ها

هر خانه‌ای که بازیکن انتخاب می‌کند باید شرایط زیر را داشته باشد:

- خالی باشد.
- باعث محاصره حداقل یک مهره حریف در یکی از جهات (افقی، عمودی، مورب) شود.

زمانی که کاربر یک خانه معتبر را انتخاب میکند:

- مهره در آن خانه قرار می‌گیرد.
- تمام مهره‌های حریف که در مسیر محاصره هستند تغییر رنگ می‌دهند (Flip).
- نوبت به بازیکن بعدی منتقل می‌شود.
- اگر بازیکن بعدی حرکتی نداشت، پیامی مانند "Pass" به او نمایش داده می‌شود و سپس نوبت او رد می‌شود.

فاز 5: شرایط پایان بازی

بازی زمانی پایان می‌یابد که هیچ یک از دو بازیکن حرکت معتبری نداشته باشد.
پس از پایان بازی:

- تعداد مهره‌های هر بازیکن شمارش می‌شود.
- بازیکنی که مهره‌های بیشتری دارد، برنده اعلام می‌شود.
- در صورت تساوی، بازی مساوی اعلام خواهد شد.
- نتیجه و جزییات بازی طبق موارد ذکر شده در قسمت Game History ذخیره می‌شود.
- کاربر پس از پایان بازی به منوی اصلی بازگردانده می‌شود.

فاز 6: حالت تک‌نفره و پیاده‌سازی ربات

در حالت تک‌نفره، بازی باید شامل یک ربات ساده باشد.

ربات پایه (الزامی)

- ربات باید در هر نوبت خود، تمام حرکات ممکن و معتبر خود را شناسایی کند.
  سپس به‌صورت تصادفی (Random) یکی از این حرکات را انتخاب و اجرا کند.
- در صورتی که ربات هیچ حرکت معتبری نداشته باشد، باید Pass کند.

ربات پیشرفته (امتیازی)

به‌عنوان بخش امتیازی پروژه، ربات می‌تواند از یک الگوریتم مؤثرتر برای تصمیم‌گیری استفاده کند، از جمله:

- انتخاب حرکتی که بیشترین تعداد مهره را برمی‌گرداند
- اولویت دادن به گوشه‌ها و لبه‌های صفحه
- استفاده از یک سیستم امتیازدهی ساده برای ارزیابی حرکت‌ها
- یا پیاده‌سازی الگوریتم‌هایی مانند Minimax

پیاده‌سازی ربات پیشرفته اختیاری است، اما امتیاز اضافه خواهد داشت.

فاز 7: ذخیره سازی نتایج

اطلاعات بازی‌های پایان‌یافته (Game History) نباید با بستن برنامه پاک شوند. این اطلاعات شامل:

- نام بازیکن اول و دوم
- تعداد مهره‌های هر بازیکن
- نام بازیکن برنده یا اعلام مساوی
- تاریخ و زمان انجام بازی

این اطلاعات باید در یک فایل متنی ذخیره شوند و در هر بار اجرای برنامه خوانده شوند.

ذخیره‌سازی یک بازی تمام نشده و بازیابی آن (امتیازی):
ذخیره‌ی وضعیت هر بازی در حال اجرا، و بازیابی مجدد آن از طریق گزینه Load Game .

3. نکات مهم

- در کدنویسی، قواعد کلین کد را رعایت کنید.
- کد تکراری ننویسید (DRY : Don't Repeat Yourself) و از توابع برای رفع این مشکل استفاده کنید.
- برای درک بهتر از منطق و قوانین بازی از طریق لینک گذاشته شده بازی را امتحان کنید تا آشنایی بیشتری از بازی داشته باشید.
- پس از نوشتن هر بخش از کد، قبل از ادامه و رفتن به بخش بعد، آن قسمت را تست کنید. انجام تست‌ها به صورت مرحله به مرحله، رفع اشکال و توسعه برنامه را راحت‌تر می‌کند.
- برای نگهداری و استفاده از اطلاعات بازی حتما باید از **استراکت** استفاده کنید.
- برای دریافت تنها یک کاراکتر از کیبورد بدون نیاز به فشرده شدن دکمه‌ی Enter، باید از تابع getch و یا روش‌های معادل آن استفاده کنید.
- پاک‌کردن صفحه ترمینال:
    - ویندوز:

```
system("cls");
```

    - مک و لینوکس:

```
system("clear");
```

این دستورات باعث می‌شوند که تمام محتوای ترمینال پاک شود و رابط کاربری مرتب‌تر به نظر برسد.

4. توابع و کتابخانه‌های کاربردی

- تابع random

از کتابخانه <cstdlib> استفاده می‌کند و به منظور تولید اعداد تصادفی استفاده می‌شود. این تابع یک عدد تصادفی در بازه مشخص را تولید می‌کند. برای استفاده از تابع random، ابتدا باید کتابخانه مربوطه را به کد خود اضافه کنید:

```
#include <cstdlib>
#include <ctime>
```

سپس در ابتدای برنامه یا تابع main، باید seed تابع را مقداردهی اولیه کنید:

```
srand(static_cast<unsigned int>(time(nullptr)));
```

می‌توانید تابع rand را برای تولید اعداد تصادفی استفاده کنید. مثلا برای تولید عدد تصادفی بین 1 تا 100:

```
int randomNum = rand() % 100 + 1;
```

- تابع getch

برای دریافت یک کاراکتر از ورودی بدون نمایش آن در کنسول استفاده می‌شود. برای استفاده از تابع getch در ویندوز، ابتدا باید کتابخانه مربوطه را به کد خود اضافه کنید:

```
#include <conio.h>
```

می‌توانید از تابع getch برای دریافت یک کاراکتر بدون نمایش آن استفاده کنید:

```
char usertInput = getch()
```

- کاراکتر های Unicode:

برای اینکه بتوانید طراحی قشنگی در ترمینال نمایش دهید باید از کارکترهای box-drawing (لینک) و همچنین دیگر کاراکترهای یونیکد (لینک) استفاده کنید.

5. موارد امتیازی

- ذخیره بازی (Save): در حین بازی، کاربر باید بتواند بازی را ذخیره کند (مثلا با زدن یک کلید خاص یا خروج از محیط بازی). وضعیت کامل صفحه و نوبت بازیکن باید در فایل ذخیره شود تا در اجرای بعدی با گزینه Load Game بازیابی شود.
- پیاده‌سازی الگوریتم های پیشرفته برای ربات، مانند Minimax
- زیبایی بصری: استفاده از رنگ‌بندی (ANSI Colors) برای مهره‌های سیاه و سفید و رابط کاربری جذاب.
- انتخاب سایز صفحه: به جای 8×8 ثابت، کاربر بتواند سایز زوج دیگری (مثلا 6×6 یا 10×10) را انتخاب کند.
- نمایش حرکات مجاز (Hint): در نوبت هر بازیکن، خانه‌هایی که می‌تواند در آن‌ها مهره بگذارد با یک علامت خاص (مثلاً * کمرنگ) مشخص شوند.
- پخش مجدد (Replay): پس از پایان بازی، امکان مشاهده دوباره تمام حرکات بازی از ابتدا تا انتها وجود داشته باشد.
- نمایش جدول امتیاز ها بر اساس تعداد برد هر بازیکن
- و هر ایده‌ی خلاقانه‌ای که پیاده‌سازی آن به تولید یک بازی بهتر کمک می‌کند.

6. ارزیابی

ارزیابی پروژه بازی اتللو بر اساس کیفیت پیاده‌سازی، صحت منطق بازی و میزان کامل بودن قابلیت‌های اصلی انجام می‌شود. در این ارزیابی، انتظار می‌رود قوانین بازی اتللو به‌صورت صحیح و کامل پیاده‌سازی شده باشند،از ارزیابی جمله تشخیص حرکات مجاز، برگرداندن مهره‌ها در تمامی جهات، مدیریت نوبت بازیکنان، تشخیص پایان بازی و تعیین برنده. همچنین مدیریت ورودی‌های نامعتبر از جمله موارد مهم در سنجش صحت منطق بازی محسوب می‌شود.

علاوه بر منطق بازی، کیفیت کد و ساختار پیاده‌سازی نیز نقش مهمی در ارزیابی دارد. خوانایی کد، نام‌گذاری مناسب متغیرها و توابع، تفکیک صحیح بخش‌های مختلف برنامه (منطق بازی، ورودی‌ها و نمایش)و پرهیز از کد تکراری

علاوه بر منطق بازی، کیفیت کد و ساختار پیاده‌سازی نیز نقش مهمی در ارزیابی دارد. خوانایی کد، نام‌گذاری مناسب متغیرها و توابع، تفکیک صحیح بخش‌های مختلف برنامه (منطق بازی، ورودی‌ها و نمایش)، پرهیز از کد تکراری و رعایت اصول کدنویسی تمیز از جمله معیارهای اصلی این بخش هستند. رابط کاربری بازی نیز باید به‌صورت واضح و قابل‌فهم طراحی شده باشد و نمایش صفحه بازی، وضعیت نوبت‌ها و پیام‌های مربوط به بازی به شکل مناسب انجام شود.

در کنار قابلیت‌های اصلی، پروژه می‌تواند شامل بهبودها یا ویژگی‌های تکمیلی باشد که نشان‌دهنده تلاش بیشتر، خلاقیت یا درک عمیق‌تر از مسئله هستند. افزودن قابلیت‌هایی مانند ربات پیشرفته، تنظیم سطح سختی، یا ایجاد تغییرات معنادار در منطق یا طراحی بازی می‌تواند در ارزیابی نهایی تاثیر مثبت داشته باشد.

نمره دهی هر فرد توسط منتور خودش انجام خواهد شد.

ددلاین پروژه دوشنبه، 15 دی ماه می‌باشد.

موفق باشید.