**به نام خدا**

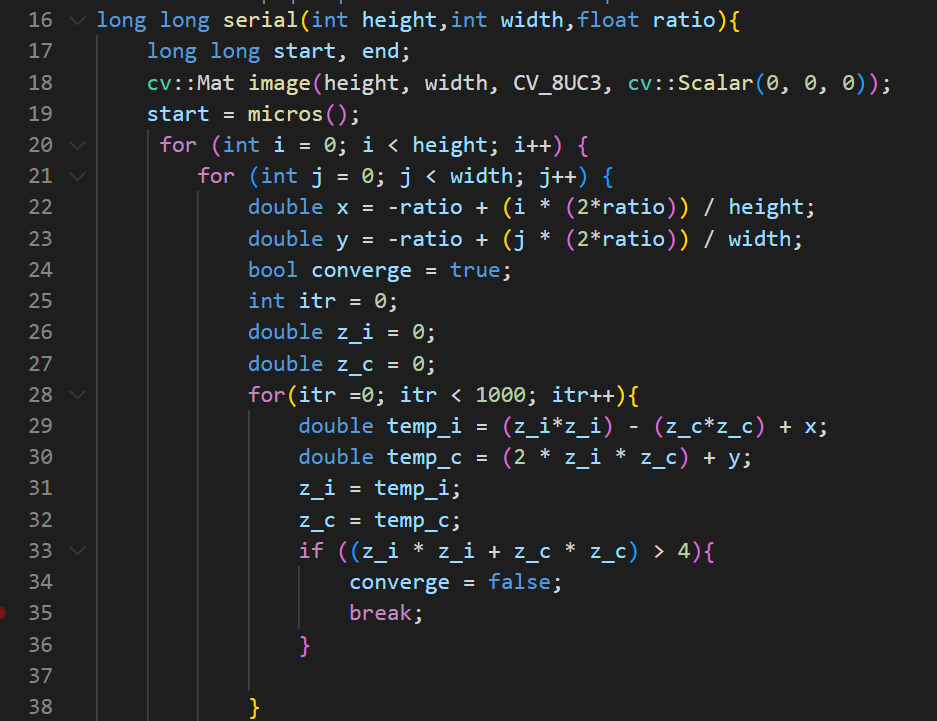
تمرین کامپیوتری دوم – موازی سازی در سطح نخ

فربد عظیم محسنی 810100187

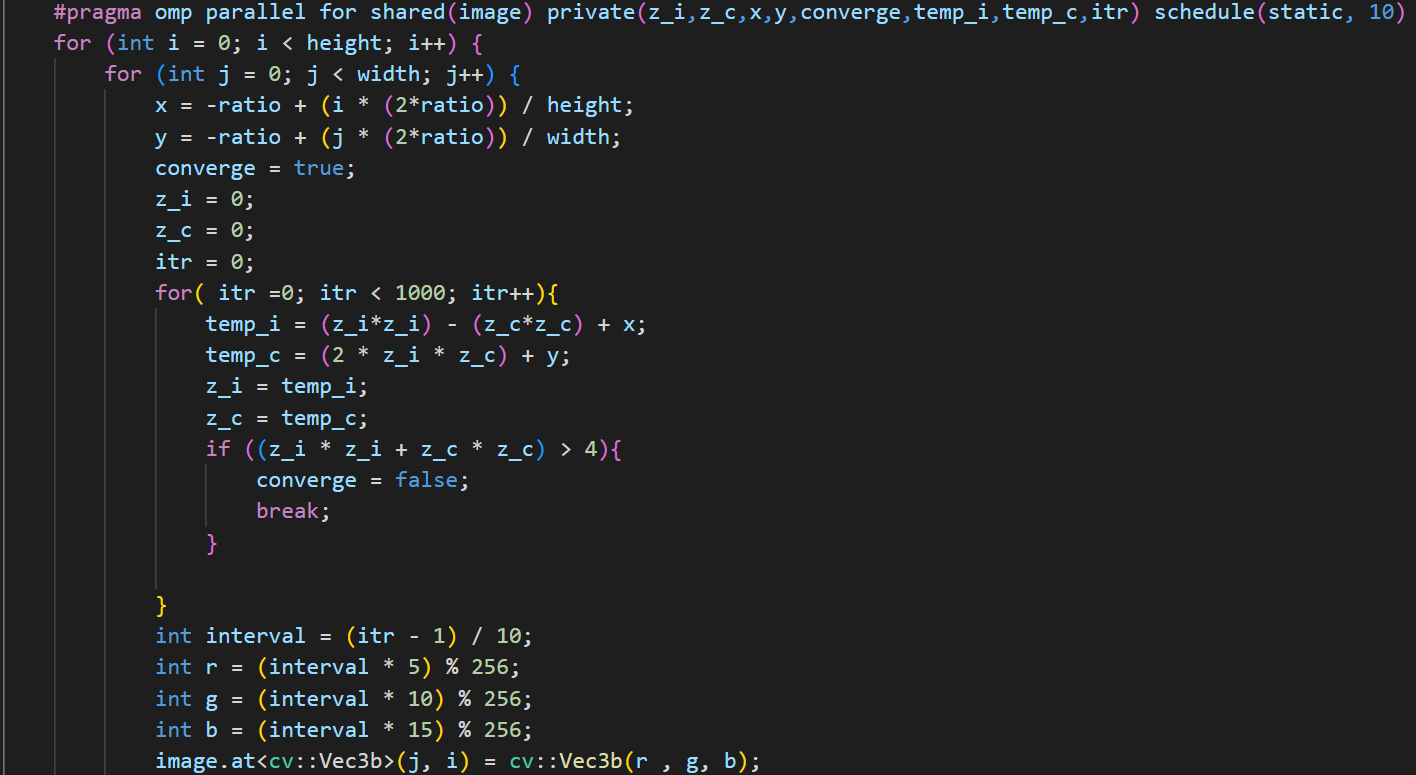
امیرحسین راحتی 81010144

* سوال اول:

در این بخش به محاسبه مجموعهSet Mandelbrot میپردازیم . این سری ، اعدادی مختلط در فضای دو بعدی هستند که به ازای یک عدد ثابت z ، اگر به توان 2 برسند و با c جمع شوند و برای هزار بار این کار را تکرار کنیم ، کوچکتر از 4 و همگرا میشوند . این کار را به ازای c های مختلف تکرار میکنیم . در بخش سریال قطعه کد زیر عملیات محاسبات را انجام میدهد. سپس برای رنگی کردن خروجی بر اساس تکراری که واگرایی رخ داده ، از یک مپینگ بین عدد تکرار و رنگ پیکسل ها استفاده میکنیم:

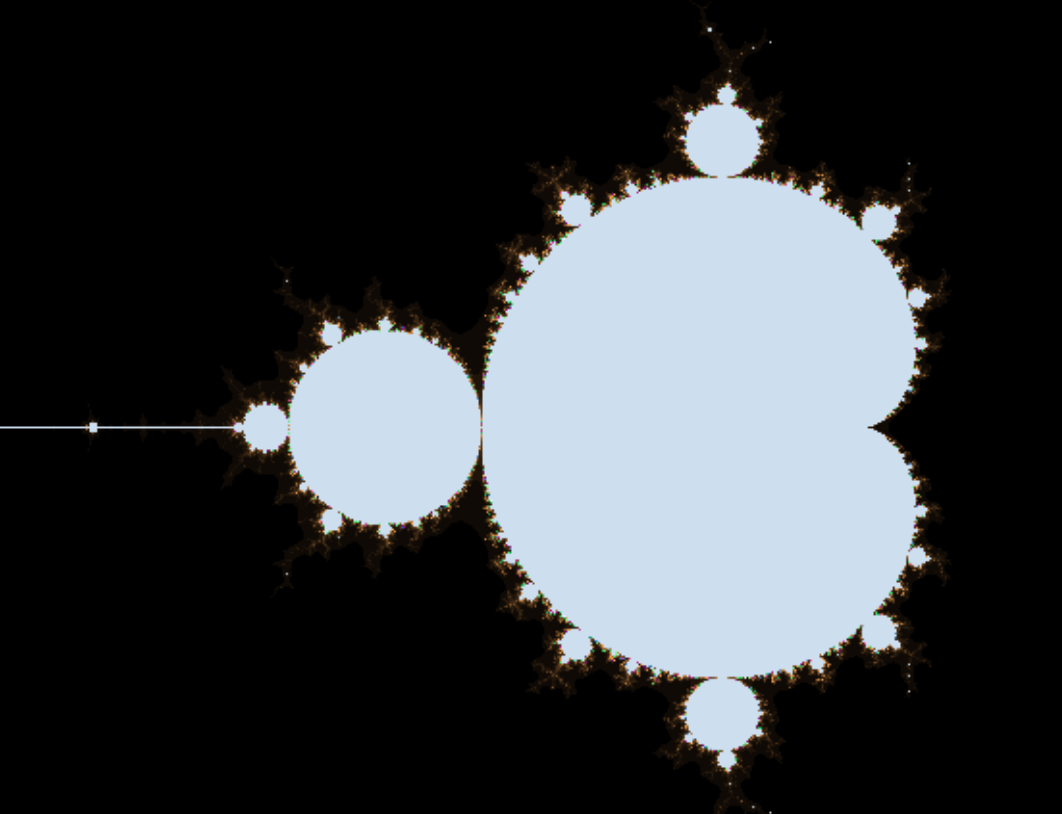


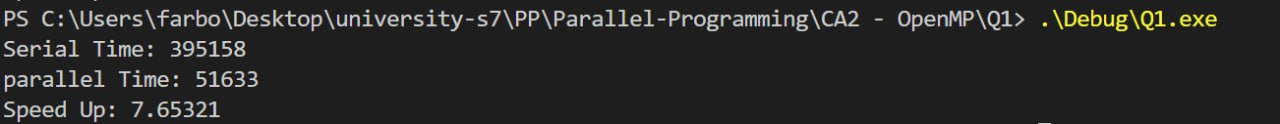
برای موازی سازی این کد ، با استفاده از کتابخانه openMP ، حلقه فور بیرونی را بین ترد ها تقسیم میکنیم .



تصویر و متغیر های داخل حلقه مثل مقادیر z به صورت private هستند و تکرار حلقه هارا به صورت static بین ترد ها تقسیم میکنیم .

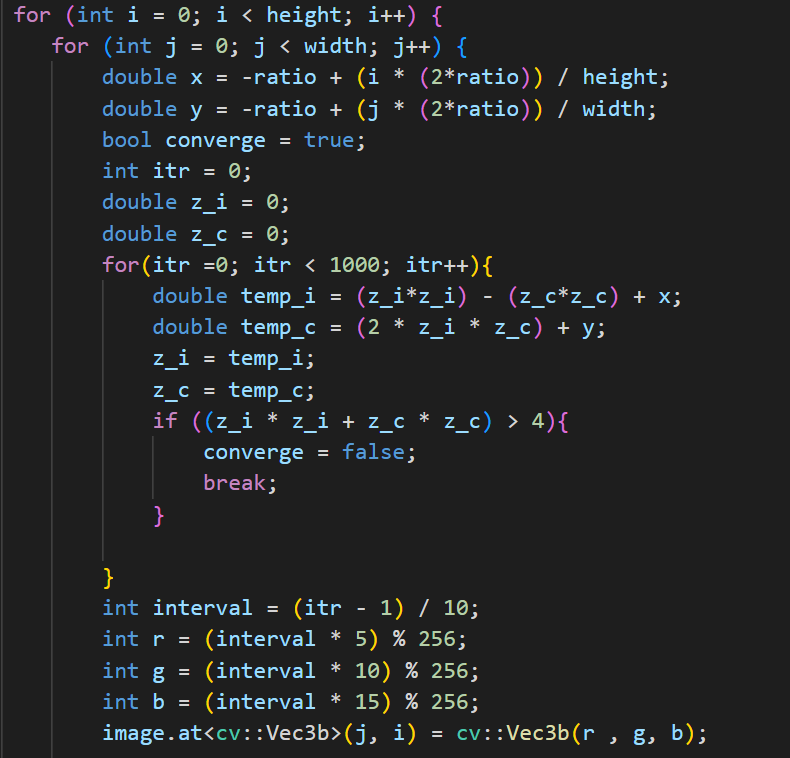
بخش امتیازی : برای zoom in و zoom out مقدار width و height را در حلقه داینامیک کردیم که می توان با تغییر آن زوم را تغییر داد . خروجی :



به ازای 800x800 مقدار speedup حدودا 7 برابر بدست می آید:  


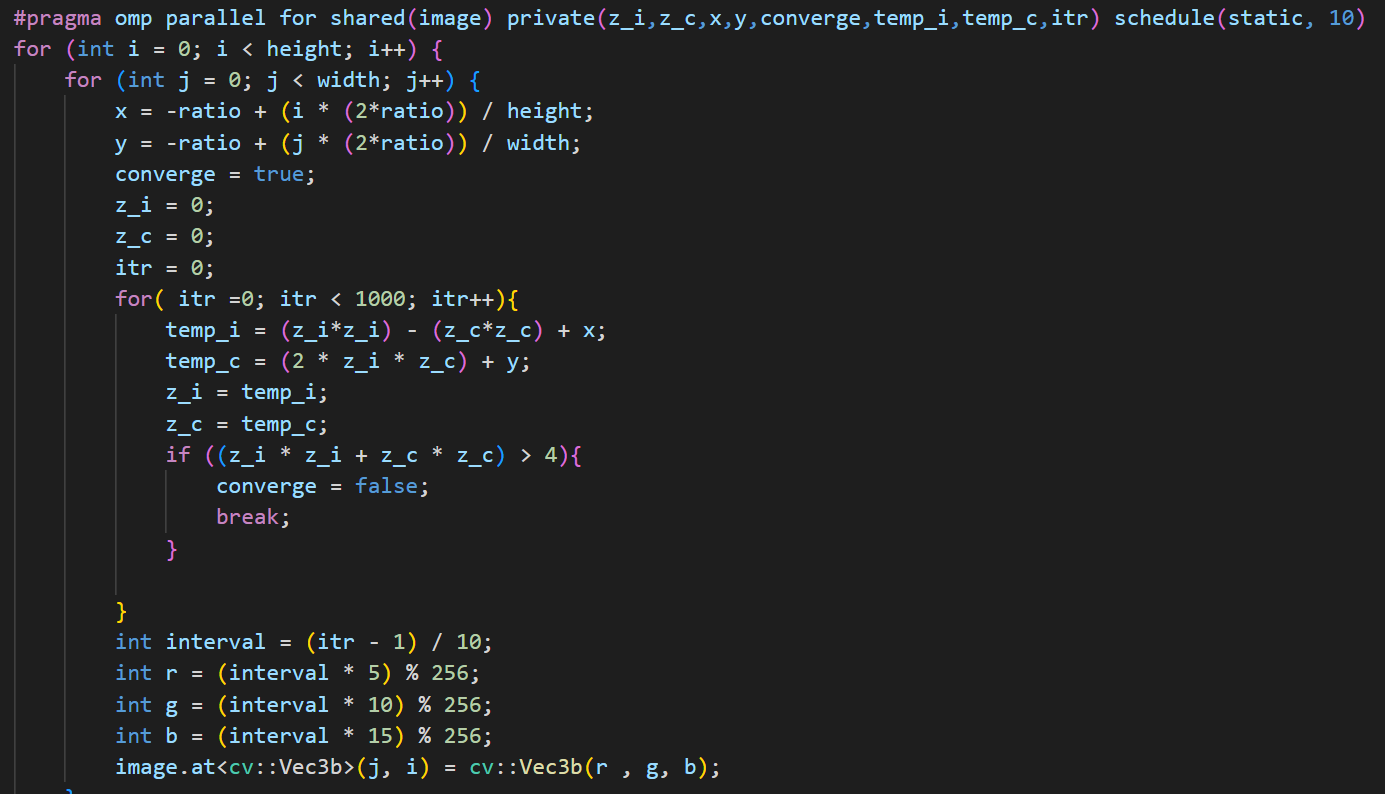
* سوال دوم :

در این بخش به محاسبه مجموعه Julia Set میپردازیم. این سری نیز مشابه سوال قبل است . با این تفاوت که اینبار مقدار cثابت است و مقدار z را برای آن عدد مختلط تغییر میدهیم . قطعه کد بخش سریال به صورت زیر است :



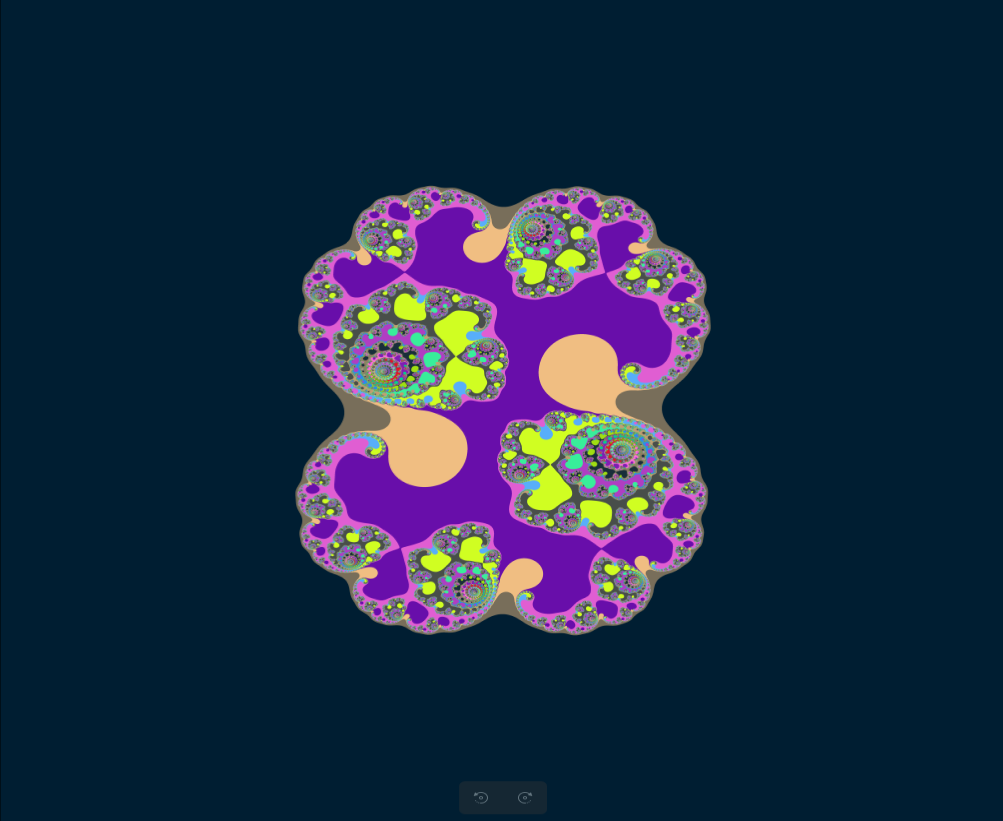
در نهایت برای رنگی شدن خروجی بر اساس تعداد دفعات تکرار ، آن را به یک رنگ پیکسل مپ میکنیم .

در بخش موازی با استفاده از openMP ، قطعه کد به صورت زیر میشود:

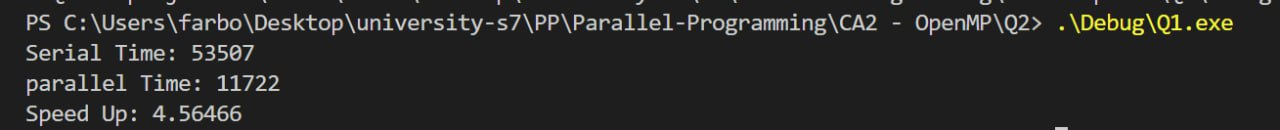


متغیر هایی مثل z و c و خود آبجکت تصویر مشترک هستند و تعداد تکرار های حلقه فور بیرونی هم به صورت static بین ترد ها تقسیم میشود

خروجی تصویر برنامه برای دو حالت موازی و سریال به صورت زیر خواهد بود:



به ازای 800x800 مقدار speedup حدودا 4.5 برابر بدست می آید:

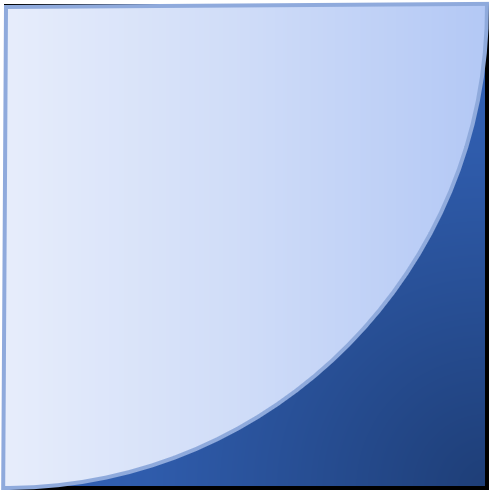


* سوال سوم :

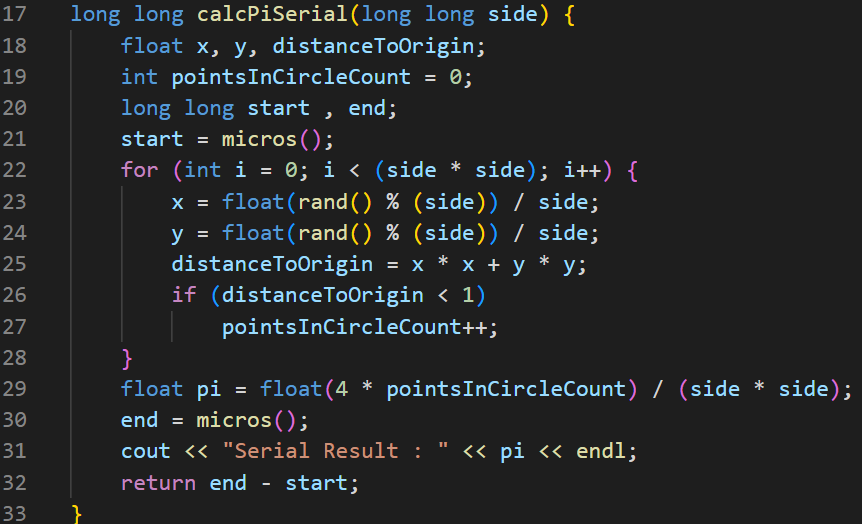
در روش تست مونته کارلو ، هدف این است که با تکرار تصادفی یا غیر تصادفی تست با ورودی های مختلف و تعداد تکرار خیلی زیاد ، بتوانیم تمام حالات سیستم را تست کنیم .

در این مثال ، برای تخمین عدد π باید تعداد زیادی نقطه در فضای دو بعدی ایجاد کنیم .

ما در این سوال به جای استفاده از دایره کامل داخل یک مربع ، به علت تقارن دایره و مربع و همچنین سادگی محاسبات ، یک ربع دایره به شعاع یک داخل یک مربع به طول ضلع یک در نظر گرفتیم .



حالا راس پایین سمت راست را نقطه مبدا مختصات در نظر میگیریم و نقاطی را با حداکثر طول و عرض یک به صورت رندوم ایجاد میکنیم که داخل مربع بیوفتند . سپس فاصله آن ها را با مرکز حساب میکنیم . هر نقطه ای که فاصله ای کمتر مساوی یک تا مرکز داشته باشد داخل ربع دایره است .

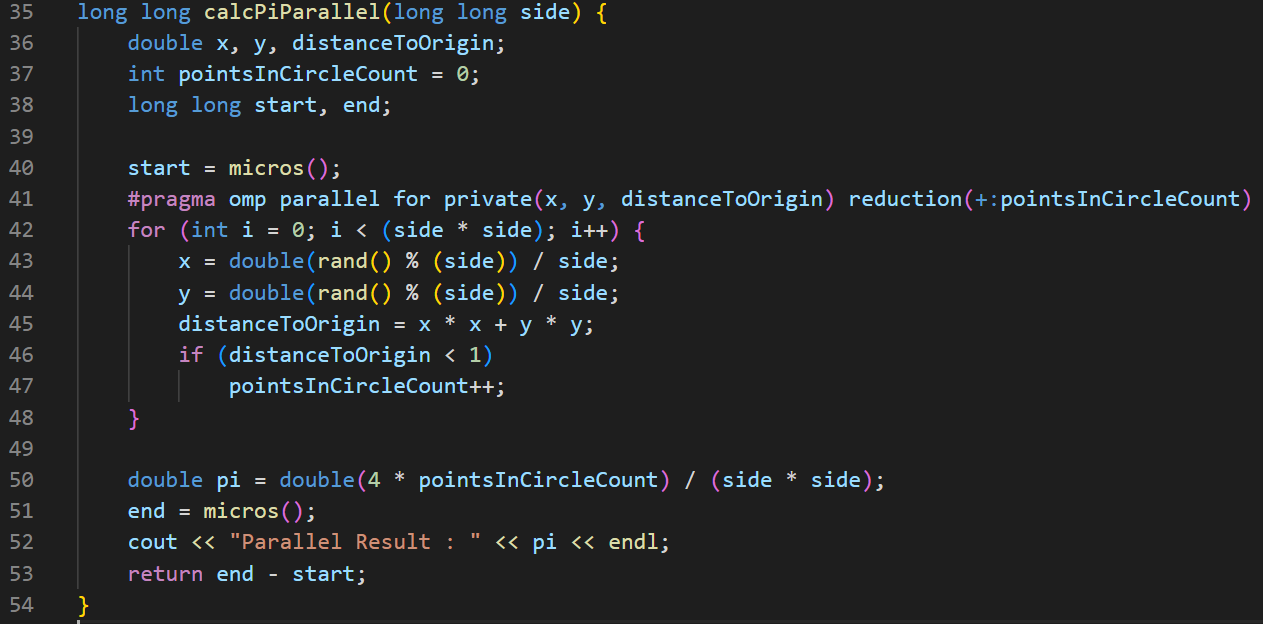


سپس برای محاسبه عدد پی کافی است حاصل چهار برابر تعداد نقاط داخل دایره را تقسیم بر مساحت مربع کنیم .

هرچه تعداد تکرار این کار بیشتر باشد خروجی دقیق تر خواهد بود.

برای موازی سازی این قطعه کد ، بخشی از برنامه که بیشترین عملیات را انجام میدهد حلقه فور است که میتواند به صورت موازی اجرا شود.

با استفاده از کتابخانه openMP ، قطعه کد موازی به شکل زیر خواهد بود:



عبارت زیر به قطعه کد اضافه شده است :

این عبارت تکرار های حلقه را بین ترد ها تقسیم میکند . متغیر های x , y و فاصله از مرکز private هستند تا هربار سربار ایجاد آن ها در داخل حلقه را نداشته باشیم .

تعداد نقاط داخل دایره نیز برای هر ترد جداگانه محاسبه میشود . سپس بعد از اتمام کار همه ترد ها این مقدار با هم جمع میشود .



خروجی برای حدود 50 هزار بار تکرار حلقه و برای حالات موازی و سریال به صورت زیر خواهد بود که حدود 7 برابر speed up میدهد.

