



**زیربرنامه:**

Find(T address, type tname)

|  |  |  |
| --- | --- | --- |
| **توسعه دهندگان** | مرتضی نامور |  |
| **تهیه کنندگان مستند** | مرتضی نامور | |
| **تاییدکنندگان** | مرتضی نامور | |
| **تاریخ تنظیم سند** | 22/02/1394 | |
| **شناسه سند** | **MC2F003F1** | |
| **زبان برنامه‌نویسی** | **Fortran 90** | |

1. وظایف

همانطور که قبلاً در مورد ساختار داده ای platform توضیح داده شد ، به منظور سهولت در فرآیند به روزرسانی اطلاعات شبکه ، تعدادی از اطلاعات را در اشیاء شبکه ذخیره نمی کنیم و برای دستیابی به آن ها باید در میان اطلاعات موجود جستجو کرده و آن ها را بیابیم.

در این platform این وظیفه بر عهده متد find است که برای کاربر ، اطلاعاتی را که به طور مستقیم قابل دسترسی نیستند ، پیدا کرده و بر اساس نوع آن ها در اختیار کاربر قرار دهد.

اطلاعاتی که در این platform به صورت مستقیم قابل دسترسی نیستند عبارتند از :

• گره های یک المان (زیرا هر المان فقط اضلاع خود را ذخیره می کند)

• المان های همسایه یک ضلع

• المان های همسایه یک المان

این سه گروه اطلاعات فوق را متد find می تواند برای کاربر پیدا کند.

1. توضیحات و تئوری­ها

با توجه به اینکه متد find باید بتواند انواع مختلفی از اطلاعات و متغیر ها را به عنوان ورودی بپذیرد و به عنوان خروجی برگرداند لذا به مانند متد A2N باید از template استفاده بشود.

متد find دارای دو ورودی می باشد . ورودی اول آدرس یا به عبارتی اشاره گر (pointer) یک شئ ( ضلع یا المان) و ورودی دوم متغیری از نوع TYPE (پارامتر TYPE یک متغیر از نوع enum است که در ابتدای کلاس MESH قرار گرفته است) می باشد که مشخص می کند کاربر چه چیزی را از متد find به عنوان خروجی می خواهد. به طور مثال اگر ورودی آدرس یک المان باشد ، کاربر باید مشخص کند که آیا گره های آن المان را می خواهد یا المان های همسایه آن المان را یا اگر ورودی آدرس یک ضلع باشد طبیعتاً کاربر به دنبال پیدا کردن المان های همسایه آن ضلع است. نکته ای که در مورد این متد وجود دارد این است که این متد اطلاعاتی را که پیدا می کند ، با توجه به نوع آن ها درون متغیر های global به نام های Point ، Edge و Cell قرار می دهد. خروجی خود متد از نوع int بوده که مشخص کننده تعداد اشیائی است که در هر حالت پیدا کرده است.

1. بخش­های زیربرنامه
2. تعریف متغیرهای اولیه

بدون توضیح

1. ذخیره آدرس ورودی به متد

در این بخش آدرس ورودی در قالب reinterperet\_cast در متغیری به نام add از کلاس union ذخیره می شود. سپس یک آرایه به نام s تعریف می شود که درایه های آن تفاضل آدرس خانه اول حافظه اشغال شده توسط اشیا و آدرس اشغال شده در متغیر add توسط اشیاء متناظر آن می باشد. به ترتیب درایه اول مربوط به المان ها ، درایه دوم مربوط به اضلاع و درایه سوم مربوط به گره هاست.

1. شروع فرآیند پیدا کردن گره های یک المان

در این بخش این شرط بررسی می شود که اگر درایه اول آرایه s بزرگتر از 0 و کوچکتر مساوی تعداد المان های شبکه باشد و همچنین متغیر type ورودی برابر با point باشد آن گاه متد شروع به پیدا کردن گره های المان ورودی به متد می کند.

سپس متغیرهای لازم در ادامه این بخش تعریف می شود. به طور مثال m تعداد گره های پیدا شده است. n تعداد اضلاع المان است. و آرایه P هم آرایه ایست که تمامی گره ها در آن ذخیره می شود.

1. ذخیره تمام گره های اضلاع المان ورودی

در این بخش با حرکت بر روی اضلاع المان ورودی، تک تک گره های المان ورودی را در یک آرایه با 8 درایه (حداکثر تعداد گره در المان های دوبعدی) ذخیره می کنیم. توجه شود که در این جا تعدادی گره تکراری درون آرایه قرار گرفته است.

1. پر کردن دو درایه اول از آرایه Point که global است

با توجه به اینکه دو گره اول مربوط به اولین ضلع المان ورودی بوده اند لذا مطمئنیم این دو گره یکسان نیستند. پس این دو گره را به آرایه گلوبال Point که نتایج متد find در آن ذخیره می شود می فرستیم.

1. پیدا کردن گره سوم

در این بخش از درایه سوم آرایه P شروع به جستجو می کنیم و هر گره را با دو گره آرایه Point مقایسه می کنیم و در صورتیکه برابر نباشند آن را به آرایه Point منتقل می کنیم و از حلقه خارج می شویم.

1. مقایسه تعداد گره های پیدا شده و تعداد اضلاع المان

در اینجا تعداد اضلاع المان را با گره های پیدا شده تا کنون مقایسه کرده و در صورتیکه برابر باشند از متد خارج می شویم ( المان مثلثی بوده است) . ولی اگر برابر نباشد آنگاه به بخش بعد می رویم.

1. پیدا کردن گره چهارم

در صورتیکه برنامه وارد این بخش بشود بدان معنیست که المان ورودی چهار گرهی بوده و ما تا کنون سه گره پیدا کردیم. اکنون بایستی گره آخر را نیز بیابیم. بدین منظور مطابق بخش 6 عمل می کنیم ولی این بار گره درون آرایه P را به سه گره درون آرایه Point مقایسه می کنیم.

1. مقایسه تعداد گره های پیدا شده و تعداد اضلاع المان

مانند بخش 7 عمل می کنیم.

1. شروع فرآیند پیدا کردن المان های همسایه یک ضلع

در این بخش می خواهیم المان های متصل به یک ضلع را بیابیم. در ابتدا متغیر های اولیه مورد نیاز این بخش را تعریف می کنیم

1. جستجو در بین المان های متصل به یکی از گره های ضلع

به منظور پیدا کردن المان های متصل به ضلع مورد نظر فقط کافیست در بین المان های متصل به یکی از گره های آن مثل گره اول آن ضلع جستجو کنیم و هر المانی که یکی از اضلاع آن ، همان ضلع ورودی باشد را بیابیم. مطمئناً تعداد این المان ها بیشتر از 2 نخواهد بود.

1. شروع فرآیند پیدا کردن المان های همسایه یک المان

در این بخش می خواهیم المان های متصل به یک المان را بیابیم. ابتدا متغیرهای اولیه مربوط به این بخش را تعریف می کنیم.

1. جستجو در بین المان های متصل به اضلاع المان ورودی

در این بخش ابتدا المان های همسایه اضلاع المان ورودی را به کمک متد Find می یابیم . در صورتیکه نتیجه متد Find دارای دو المان خروجی باشد آن دو المان را در یک آرایه ذخیره می کنیم. توجه شود اگر نتیجه متد Find در این بخش برابر 1 باشد بدان معنیست که یکی از اضلاع المان ورودی مرزی بوده و فقط یک المان همسایه داشته است که طبیعتا آن المان همان المان ورودی است که با این تفسیر نیازی به ذخیره آن نیست.

1. پیدا کردن المان های غیر از المان ورودی

در انتها آن المان هایی را در آرایه Cell ذخیره می کنیم که مخالف همان المان ورودی باشند.