فصل چهارم

پردازش استثنا ها

مقدمه

- قابلیت توسعه C++ می تواند انواع و تعداد خطاهایی را که اتفاق می افتد افزایش دهد. این ویژگی باعث می شوند برنامه های قدرتمند عاری از خطا وتحمل کننده عیب ها نوشته شوند.
- به کمک پردازش استثنا هنگامی که خطایی در برنامه رخ دهد برنامه به طور خودکار روال کنترل خطا رافراخوانی می کند.
- کد کنترل خطا به ماهیت و اندازه سیستم های نرم افزاری بستگی دارد.در بعضی از سیستم ها مثل سیستم های تجاری باید کد کنترل خطای زیادی وجود داشته باشد.

مقدمه

- ابزارهای زیادی برای کنترل خطا وجود دارد. معمولا کدهای کنترل خطا در سراسر کد سیستم پراکنده اند. خطاها در جاهایی که اتفاق می افتند تحت کنترل قرار می گیرند.
- مزیت
  - هنگام خواندن کد، کنترل خطا در نزدیکی خود کد مشاهده می شوند و متوجه می شود که آیا کنترل خطا به طور مناسب پیاده سازی شده است یا خیر.
- عیب
  - کد سیستم با پردازش خطا در یکجا هستند که موجب می شود خواندن برنامه به منظور پی بردن به صحت کامل آن مشکل شود. همچنین درک و نگهداری کد دشوار خواهد شد.

مقدمه

- بعضی از استثناهای متداول عبارتند از: عدم موفقیت عملگر new جهت تخصیص حافظه، خروج از حد آرایه، سر ریز در محاسبات، تقسیم بر صفر و پارامترهای نامعتبر تابع.
- ویژگی های جدید C++ در پردازش استثنا برنامه نویس را قادر می سازد تا کد کنترل خطا را از متن برنامه جدا کند و به این ترتیب به خوانایی برنامه بیافزاید.
- با ویژگی های پردازش استثنا در C++ می توان تمام استثناهای مربوط به یک نوع خاص یا تمام استثناهای انواع مرتبط را تشخیص داد. به این ترتیب احتمال اینکه برنامه نتواند خطایی را تشخیص دهد کاهش می یابد.

مقدمه

- پردازش استثنا موجب می شوند تا برنامه نویس خطاها را تشخیص دهد و آنها را تحت کنترل در بیاورد تا موجب اشکالاتی در برنامه نشوند. اگر برنامه نویس نتواند خطاهای مهم را پردازش کند برنامه خاتمه می یابد.
- پردازش استثنا برای کنترل خطاهای همگامی مثل تقسیم بر صفر طراحی شده است. با پردازش استثنا قبل ازاینکه برنامه عمل تقسیم را انجام دهد مقسوم علیه را کنترل می کند و در صورتی که مقسوم علیه صفر باشد استثنایی را صادر می نماید.

مقدمه

- پردازش استثنا در وضعیت هایی مورد استفاده قرار می گیرد که اگر خطایی موجب استثنا شد سیستم بتواند ترمیم شود. روند ترمیم را **پردازش استثنا** گویند. به خصوص در مواردی که برنامه نتواند ترمیم شود ولی لازم است به طرز خوبی خاتمه یابد پردازش استثنا ابزار مفیدی است.

اصول پردازش استثناها

- پردازش استثناها در C++ با سه کلمه کلیدی `try`, `catch`, `throw` انجام می شوند. دستورالعمل هایی که ممکن است تولید خطا کنند و موجب بروز استثنا شوند در بلوکی به نام `try` قرار می گیرند. پس از هر بلوک `try` یک یا چند بلوک `catch` قرار می گیرد.
- هر بلوک `catch` حاوی یک پردازشگر استثنا است. اگر استثنا با یکی از پارامترهای موجود در بلوک های `catch` مطابقت داشته باشد کد مربوط به ان بلوک `catch` اجرا می شود.
- بنابراین ممکن است بیش از یک بلوک `catch` برای هر بلوک `try` وجود داشته باشد ونوع استثنا مشخص می کند که کدام بلوک `catch` باید اجرا شود. اگر استثنایی به وجود نیاید و یا خطایی در بلوک `try` رخ ندهد هیچ بلوک `catch` اجرا نمی شود.

اصول پردازش استثناها

- اگر در بلوک try خطایی اتفاق افتد استثنایی توسط دستور throw صادر می شود. این دستور به صورت زیر به کار می رود :

```
throw operand;
```

- این دستور استثنایی را که در جلوی آن با عملوند(Operand) مشخص شده است صادر می کند. استثنا باید در بلوک try یا تابعی که در داخل بلوک try اجرا می شود صادر گردد. عملوند throw می تواند از هر نوعی باشد. اگر یک شی باشد آن را شی استثنا می نامیم . به جای شی می توان یک عبارت شرطی را ارسال کرد.

بلوک try

- شکل کلی استفاده از بلوک `try` به صورت زیر است :

```
try {
   بدنه بلوک
}
```

- بدنه بلوک حاوی دستورالعمل هایی است که ممکن است در اثر بروز خطا استثنایی را صادر کند.دقت داشته باشید که بلوک `try` می تواند حاوی چند دستو اندک در داخل یک تابع یا کل دستورالعمل های موجود در تابع `main()` باشد.

بلوک catch

- پس از هر بلوک try باید یک یا چند بلوک catch قرار داشته باشد تا استثناها را پردازش کنند .به عبارت دیگر پردازشگر استثناها در بلوک catch قرار دارد. شکل کلی بلوک catch به صورت زیر است :

```
catch(پارامتر نوع ۱)  {
   بدنه بلوک
}
catch(پارامتر نوع ۲)  {
   بدنه بلوک
}
...
catch(n  پارامتر نوع)  {
   بدنه بلوک
}
```

بلوک catch

- بلوک `catch` در آرگومان خود نوع شیئی را که باید پردازش استثنا در آن اجرا شود مشخص می کند. پارامتر موجود در پردازشگر `catch` می تواند فاقد نام باشد. اگر پارامتر دارای نام باشد می توان در داخل بلوک `catch` به آن مراجعه کرد. اگر پارامتر فاقد نام باشد مثلا فقط نوع پارامتر مشخص شده باشد تا تطابق آن با نوع شیئی که پرتاب شد انجام شود فقط کنترل از نقطه پرتاب به پردازشگر منتقل می شود. این حالت برای بسیاری از استثناها قابل قبول است.

بلوک catch

- استثنایی که شی پرتاب شده آن با نوع آرگومان در بلوک `catch` یکسان باشد پردازشگر استثنای موجود در آن بلوک `catch` اجرا می شود. بلوک `catch` به صورت زیر نیز قابل استفاده است:

```
catch(...)  {
   بدنه بلوک
}
```

- مفهوم این بلوک این است که تمام استثناها را می پذیرد. توجه داشته باشید که اگر چند بلوک `catch` را به این شیوه به کار می برید این شیوه را باید به عنوان اخرین بلوک در نظر بگیرید. اشکالی که این روش کاربرد دارد این است که نمی توانید مطمئن شوید که استثنا از چه نوعی است. ممکن است چند پردازشگر استثنا با نوع استثنایی که پرتاب شد مطابقت داشته باشند. در این حالت اولین آنها اجرا می شود.

بلوک catch

- اگر چند پردازشگر استثنا با استثنای پرتاب شده مطابقت داشته باشند و هر کدام از آنها استثنا را به روش خاصی پردازش کند ترتیب پردازشگرهای استثنا، شیوه پردازش آن را تحت تاثیر قرار می دهد.
- نوع پارامتر پردازشگر `catch` در موارد زیر با شی پرتاب شده تطابق پیدا می کند:
  - وقتی از یک نوع باشند.
  - پارامتر پردازشگر `catch` کلاس پایه عمومی از کلاس شی پرتاب شده باشد.
  - پارامتر پردازشگر نوع اشاره گر پایه یا مرجع و شی پرتاب شده نوع اشاره گر یا مرجع کلاس مشتق باشد.
  - پردازشگر `catch` به صورت `catch (...)` باشد.

چند نکته

- پردازشگر استثنا نمی تواند به اشیای اتوماتیکی که در بلوک `try` تعریف شدند دسترسی داشته باشد زیرا وقتی استثنا اتفاق می افتد بلوک `try` خاتمه می یابد و تمام اشیای اتوماتیک موجود در بلوک `try` قبل از اجرای پردازشگر استثنا از بین می روند.
- اگر استثنایی در پردازشگر استثنا رخ دهد استثنای اول در پردازشگر استثنا پردازش می شود. لذا استثناهایی که در پردازشگر استثنا اتفاق می افتند باید در خارج از بلوکی که استثنای اول پرتاب شد پردازش شوند.

چند نکته

- با استفاده از دستور `return` در پردازشگر `catch` نمی توان به نقطه پرتاب استثنا برگشت. دستور `return` موجب می شود تا کنترل به تابعی برگردد که تابع حاوی بلوک `catch` را فراخوانی کرده است.
- برای هر بلوک `try` می توان بیش از یک بلوک `catch` در نظر گرفت.
- هنگامی که می خواهید هم استثناهای نوع کلاس پایه و هم استثناهای کلاس مشتق را پردازش کنید ابتدا بلوک های `catch` مربوط به کلاس مشتق را قرار دهید. در غیر این صورت بلوک `catch` مربوط به کلاس پایه می تواند با تمام کلاس های مشتق تطابق داشته باشد.

چند نکته

- استثناهایی را که توابع می توانند پرتاب کنند می توان محدود کرد. به عبارت دیگر می توان کاری کرد که توابع هر نوع استثنایی را پرتاب نکنند. برای اعمال این محدودیت باید یک بلوک `throw` به تعریف تابع اضافه کنید. شکل کلی آن به صورت زیر است :

```
نوع تابع  نام تابع (لیست آرگومان ها) throw (لیست انواع)
{
   بدنه بلوک
}
```

چند نکته

- به این ترتیب فقط استثناهایی که انواع آنها در جلوی `throw` در پرانتز آمده اند توسط تابع می توانند پرتاب شوند. اگر استثناهای دیگری پرتاب شوند برنامه خاتمه می یابد. اگر نخواهید تابعی استثنایی را صادر کند لیست انواع را ذکر نکنید.
- اگر سعی شود استثنایی صادر گردد که توسط هیچ تابعی پشتیبانی نمی شود تابع کتابخانه ای `unexpected()` اجرا می شود. این تابع به نوبه خود تابع `abort( )` را فراخوانی می کند و برنامه را خاتمه می دهد.

مثال

- برنامه ای که استثنای تقسیم بر صفر را پردازش می کند.

```
Class  divexcp {
  public :
   divexcp():message("attempt  to  divide  by  zero") {}
   const  char  *what () const  { return  message ; }
  private :
   const  char  *message ;
};
Double  quote (int  numerator , int  denum )
{
   if (denum == 0 )
      throw  divexcp() ;
   return  (double) (numerator) / denum ;
}
```

مثال

```
int  main()  {
   int  num1 , num2 ;
   double  result ;
   cout  << "Enter  two  integer : " ;
   while (cin >> num1 >> num2 ) {
      try {
           result = quote (num 1 , num 2 );
           cout << "The  quotient  is :" << result << endl ;
        }
      catch  (divexcp  ex)  {
        cout  <<  "Exception  occurred :" << ex.what() << "\n";
        }
        cout << "\n Enter  tow  integers :" ;
    }
    cout  << endl;
    return  0 ;
}
```

```
Enter  tow  integers : 100 7
The  quotient  is :  14.2857
Enter  tow  integers  : 100 0
Exception  occurred : attemted to divide by zero
Enter  tow  integers :
```