مدیریت استثناء

اصول و برنامه‌نویسی

پدرو مخیا الوارث و همکاران

ترجمهٔ

دکتر بابک مسعودی دکتر سیدحسن صادق‌زاده

امروز کتابخوانی و علم‌آموزی نه تنها یک وظیفه‌ی ملی، که یک واجب دینی است[1].

مقام معظم رهبری

در عصر حاضر یکی از شاخصه‌های ارزیابی رشد، توسعه و پیشرفت فرهنگی هر کشوری میزان تولید کتاب، مطالعه و کتاب‌خوانی مردم آن مرز و بوم است. ایران اسلامی نیز از دیرباز تاکنون با داشتن تمدنی چندهزارساله و مراکز متعدد علمی، فرهنگی، کتابخانه‌های معتبر، علما و دانشمندان بزرگ با آثار ارزشمند تاریخی، سرآمد دولت‌ها و ملت‌های دیگر بوده و در عرصه فرهنگ و تمدن جهانی به‌سان خورشیدی تابناک همچنان می‌درخشد و با فرزندان نیک‌نهاد خویش هنرنمایی می‌کند. چه کسی است که در دنیا با دانشمندان فرزانه و نام‌آور ایرانی همچون ابوعلی سینا، ابوریحان بیرونی، فارابی، خوارزمی و ... همچنین شاعران برجسته‌ای نظیر فردوسی، سعدی، مولوی، حافظ و ... آشنا نباشد و در مقابل عظمت آنها سر تعظیم فرود نیاورد. تمامی این افتخارات ارزشمند، برگرفته از میزان عشق و علاقه فراوان ملت ما به فراگیری علم و دانش از طریق خواندن و مطالعه منابع و کتاب‌های گوناگون است. به شکرانه الهی، تاریخ و گذشته ما، همیشه درخشان و پربار است. ولی اکنون در این زمینه در چه جایگاهی قرار داریم؟ آمار و ارقام ارائه‌شده از سوی مجامع و سازمان‌های فرهنگی در مورد سرانه مطالعه هر ایرانی، برایمان چندان امیدوارکننده نمی‌باشد.

کتاب، دروازه‌ای به سوی گستره دانش و معرفت است و کتاب خوب، یکی از بهترین ابزارهای کمال بشری است. همه دستاوردهای بشر در سراسر عمر جهان، تا آنجا که قابل کتابت بوده است، در میان دست‌نوشته‌هایی است که انسان‌ها پدید آورده و می‌آورند. در این مجموعه بی‌نظیر، تعالیم الهی، درس‌های پیامبران به بشر، و همچنین علوم مختلفی است که سعادت بشر بدون آگاهی از آن‌ها امکان‌پذیر نیست. کسی که با دنیای زیبا و زندگی‌بخش کتاب ارتباط ندارد بی‌شک از مهم‌ترین دستاورد انسانی و نیز از بیشترین معارف الهی و بشری محروم است. با این دیدگاه، به‌روشنی می‌توان ارزش و مفهوم رمزی عمیق در این حقیقت تاریخی را دریافت که اولین خطاب خداوند متعال به پیامبر گرامی اسلام(ص) این است که »بخوان!« و در اولین سوره‌ای که بر آن فرستاده عظیم‌الشأن خداوند، فرود آمده، نام «قلم» به تجلیل یاد

1. https://farsi.khamenei.ir/message-content?id=2696

شده‌است: «إقْرَأ وَ رَبُّکَ الْاَکْرَمُ. اَلَّذی عَلَّمَ بِالْقَلَم» در اهمیت عنصر کتاب برای تکامل جامعه انسانی، همین بس که تمامی ادیان آسمانی و رجال بزرگ تاریخ بشری، از طریق کتاب جاودانه مانده‌اند.

دانشگاه پیام‌نور با گستره جغرافیایی ایران‌شمول خود با هدف آموزش برای همه، همه‌جا و همه‌وقت، به‌عنوان دانشگاهی کتاب‌محور در نظام آموزش عالی کشورمان، افتخار دارد جایگاه اندیشه‌سازی و خردورزی بخش عظیمی از جوانان جویای علم این مرز و بوم باشد. تلاش فراوانی در ایام طولانی فعالیت این دانشگاه انجام پذیرفته تا با بهره‌گیری از تجربه‌های گرانقدر استادان و صاحب‌نظران برجسته کشورمان، کتاب‌ها و منابع آموزشی درسی شاخص و خودآموز تولید شود. در آینده هم، این مهم با هدف ارتقای سطح علمی، روزآمدی و توجه بیشتر به نیازهای مخاطبان دانشگاه پیام‌نور با جدیت ادامه خواهد داشت. به‌طور قطع استفاده از نظرات استادان، صاحب‌نظران و دانشجویان محترم، ما را در انجام این وظیفه مهم و خطیر یاری‌رسان خواهد بود. پیشاپیش از تمامی عزیزانی که با نقد، تصحیح و پیشنهادهای خود ما را در انجام این وظیفه خطیر یاری می‌رسانند، سپاسگزاری می‌نماییم. لازم است از تمامی اندیشمندانی که تاکنون دانشگاه پیام‌نور را منزلگه اندیشه‌سازی خود دانسته و ما را در تولید کتاب و محتوای آموزشی درسی یاری نموده‌اند، صمیمانه قدردانی گردد. موفقیت و بهروزی تمامی دانشجویان و دانش‌پژوهان عزیز آرزوی همیشگی ما است.

دانشگاه پیام‌نور

فهرست مطالب

پیشگفتار

چه درحال ساخت یک برنامه ساده یا یک سیستم نرم‌افزاری پیچیده باشید، درهرصورت ممکن است رویدادی غیرمنتظره رخ دهد. این رویدادهای غیرمنتظره یا «استثناها» می‌توانند از مسائل ساده مانند پیدانشدن فایل تا خطاهای جدی که می‌توانند کل سیستم را خراب کنند، متفاوت باشند. اگر این استثناها به درستی مدیریت نشوند، می‌توانند منجر به رفتار نرم‌افزاری غیرقابل‌اعتماد و غیرقابل‌پیش‌بینی شوند. مدیریت استثناء، فرایند پاسخ به وقوع استثناها– شرایط غیرعادی که نیاز به پردازش ویژه دارند– در طول اجرای نرم‌افزار است. مدیریت استثناء در تولید نرم‌افزاری مطمئن که بتواند با موقعیت‌های غیرمنتظره مقابله کند، بسیار مهم است و باعث می‌شود که نرم‌افزار انعطاف‌پذیرتر، قابل نگهداری و کاربرپسندتر شود. در این کتاب، با مثال‌هایی که در C++ و پایتون نوشته شده‌اند، به‌طور جامع به موضوع مدیریت استثناء خواهیم پرداخت. با بررسی تاریخچه و سیر تکامل آن، جنبه‌های مختلف مدیریت استثناء، مانند نحوه نگارش، معناشناسی، چالش‌ها، بهترین شیوه‌ها و موارد دیگر را بررسی خواهیم کرد. شما تفاوت‌های ظریف بین خطاهای نحوی و معنایی را درک خواهید کرد، یاد خواهید گرفت که چگونه از بلوک‌های try-catch به‌طور مؤثر استفاده کنید، اهمیت ثبت وقایع استثناها را درک خواهید کرد و حتی در روش‌های پیشرفته مدیریت استثناء نیز مهارت پیدا خواهید کرد. فصل‌های زیر درک جامعی از این مفهوم مهم در توسعه نرم‌افزار را به شما ارائه می‌دهند: فصل ۱ مقدمه‌ای را ارائه می‌دهد که تاریخچه، تعاریف مختلف و چالش‌های مدیریت استثناء را پوشش می‌دهد. فصل ۲ به مبانی می‌پردازد و دیدگاه‌هایی را در مورد مفاهیم و روش‌های اساسی ارائه می‌دهد. فصل ۳ به بهترین شیوه‌ها برای مدیریت

استثناها، ازجمله تفاوت بین خطا و استثناء، استفاده از Assertionها، و نحوه ارائه پیام‌های خطای معنی‌دار اشاره می‌کند. فصل ۴ به بررسی روش‌های پیشرفته مدیریت استثناء می‌پردازد. در اینجا، الگوها، شرط‌های محافظ، مدیریت استثناء سلسله‌مراتبی و موارد دیگر را بررسی می‌کنیم. درنهایت، فصل ۵ بر پیچیدگی‌های مدیریت استثناء در سیستم‌های بلادرنگ و تعبیه‌شده تمرکز دارد. چه یک برنامه‌نویس باتجربه باشید که به‌دنبال بهبود و ارتقای درک خود از مدیریت استثناء هستید یا یک تازه‌کار مشتاق یادگیری اصول اولیه، این کتاب راهنمایی واضح، کامل و عملی برای تسلط بر این حوزهٔ مهم توسعهٔ نرم‌افزار را ارائه می‌دهد. با مطالعه کتاب، دانش و مهارت‌های لازم برای مدیریت مؤثر استثناها را به‌دست خواهید آورد و اطمینان حاصل می‌کنید که برنامه‌های نرم‌افزاری شما قوی‌تر، قابل اعتمادتر و در برابر خطاهای غیرمنتظره مقاوم‌تر هستند. به سفر جذاب تسلط بر مدیریت استثناء خوش آمدید!

پیشگفتار مترجمین

در دنیای پیچیده و دائماً درحال تحول برنامه‌نویسی، مدیریت استثناها به‌عنوان یکی از اجزای بنیادین کدنویسی مطمئن و قابل اتکا شناخته می‌شود. اغلب برنامه‌نویسان در سراسر جهان با مفهوم استثناها آشنا هستند، اما تعدادکمی از آن‌ها درکی عمیق از روش‌های پیشرفته برای مدیریت استثناء را دارند. اثر حاضر برگردان کتاب "Exception Handling, Fundamentals and Programming" است که با توجه به این نیاز ضروری، رویکردها و راهکارهای بهینه‌ای را ارائه می‌دهد که هر توسعه‌دهنده نرم‌افزاری را قادر می‌سازد استثناها را به شیوه‌ای کارآمد مدیریت کند.

امیدواریم حاصل تلاش ما دسترسی پژوهشگران، دانشجویان و علاقمندان به این حوزه را به اطلاعاتی دقیق و جامع فراهم آورد. تلاش برای حفظ دقت فنی متن ترجمه شده در کنار روانی آن از چالش‌های اصلی بود که در این مسیر با آن مواجه شدیم. امیدواریم با مطالعه این کتاب، ارزش افزوده‌ای را در کارهای روزمره و پروژه‌های برنامه‌نویسی خود حس نمایید.

در پایان از همه اساتید و دانشجویان محترم تقاضا داریم راهنمایی‌ها و پیشنهادات سازنده خود را ازطریق ایمیل زیر با ما در میان بگذارند.

با احترام

فصل اول

مقدمه‌ای بر مدیریت استثناء

مقدمه

مدیریت استثناء یک جنبه حیاتی در توسعه نرم‌افزار است که برنامه‌نویسان را قادر می‌سازد رویدادها و خطاهای غیرمنتظره را در طول اجرای برنامه مدیریت کنند. این فصل با نگاهی کلی به تاریخچه و سیر تکامل مدیریت استثناء آغاز می‌شود. سپس مفاهیم و اصول کلیدی مدیریت استثناء، ازجمله انواع استثناء، نحوه تعریف و ایجاد آن‌ها و رویکردهای مختلف برای مدیریت آن‌ها را شرح می‌دهیم. این بخش همچنین به تبیین معماری اشیاء استثناء، تمایز بین استثناهای بررسی‌شده[1] و بررسی‌نشده[2]، و استراتژی‌هایی برای مدیریت داخلی و خارجی می‌پردازد. علاوه‌بر این، فرایند اختصاص مدیریت‌کننده‌ها به استثناهای خاص موردتوجه قرار می‌گیرد و بر نقش محوری آن در مدیریت مؤثر استثناء تأکید می‌شود. رسیدگی به استثناها فقط یک مفهوم نظری نیست بلکه پیامدهای آن عمیق و دامنه آن گسترده است. در بخش بعدی، بر اهمیت فوق‌العاده مدیریت استثنای ماهرانه تأکید می‌شود. مدیریت استثناها با جلوگیری از بروز رفتارهای غیرمنتظره، تقویت استحکام، قابلیت نگهداری و افزایش امنیت، کیفیت نرم‌افزار را ارتقا می‌بخشد و تضمین می‌کند که کاربران قابلیت اطمینان[3] را در تعاملات خود با سیستم تجربه کنند. درنهایت، ما به انواع مختلف استثناها، ازجمله خطاهای نحوی و معنایی می‌پردازیم. در پایان فصل، خوانندگان درک جامعی از مفاهیم و اصول کلیدی مدیریت استثناء و ابزارها و چارچوب‌های موجود برای

1. Checked
2. Unchecked
3. Reliability

پیاده‌سازی مدیریت استثنای مؤثر در پروژه‌های نرم‌افزاری خود را خواهند داشت. این فصل پایه و اساس مفاهیم کتاب را بیان می‌کند و به تشریح موضوعات و تکنیک‌های خاص مربوط به مدیریت استثناء در توسعه نرم‌افزار می‌پردازد.

۱-۱ تاریخچه و سیر تکامل مدیریت استثناء

پیشینه مدیریت استثناها [۱] به نخستین روزهای پیدایش برنامه‌نویسی بازمی‌گردد، زمانی‌که زبان‌هایی مانند اسمبلی و فرترن ابزار اصلی کدنویسی بودند. با پیشرفت زبان‌های برنامه‌نویسی، نیاز به سازوکارهای مدیریت خطا و استثناء آشکار شد. این بخش مروری جامع بر سیر تکامل مدیریت استثناء را ارائه می‌دهد.

در دهه‌های ۱۹۵۰ و ۱۹۶۰، زبان اسمبلی و فرترن زبان‌های برنامه‌نویسی غالب بودند. این زبان‌ها عمدتاً بر سازوکارهای بنیادین بررسی خطا تکه داشتند و از دستورالعمل‌های انشعاب ساده برای مدیریت خطاها استفاده می‌کردند. درنتیجه، برنامه‌نویسان مجبور بودند کدنویسی گسترده‌ای انجام دهند تا انواع مختلف خطاها و استثناها را مدیریت کنند، امری که اغلب منجر به تولید کدهایی می‌شد که نگهداری آن‌ها دشوارتر و احتمال بروز خطا در آن‌ها بیشتر بود. در دهه ۱۹۶۰، یکی از نخستین زبان‌های برنامه‌نویسی که سازوکار مدیریت ساختاریافته استثناء را معرفی کرد، PL/I بود. در این زبان استثناهای مختلف می‌توانستند با کنترل‌کننده‌های خاصی مرتبط شوند. با این حال، این سیستم همچنان به مدیریت دستی انتشار خطا نیاز داشت، که آن را نسبت به سیستم‌های امروزی مدیریت استثناء کمتر انعطاف‌پذیر می‌کند. ظهور زبان‌های برنامه‌نویسی شی‌گرا مانند C++ و پایتون نقش بسزایی در محبوبیت ساختار try-catch-fnally ایفا کرد. این ساختار امکان تولید کدی ساختاریافته و با قابلیت نگهداری بهتر را برای مدیریت استثناها فراهم می‌کرد. همچنین در این سازوکار مدیریت استثناء، بلوک‌های کد مدیریت و بازیابی خطا از منطق اصلی برنامه مجزا بودند. همگام با تکامل زبان‌های برنامه‌نویسی، زبان‌های جدیدتر ساختارهای مدیریت استثنای پیشرفته‌تر و انعطاف‌پذیرتری را به‌کار گرفته‌اند. به‌عنوان مثال، زبان‌هایی مانند پایتون، C# و Ruby از ساختارهای مدیریت استثنایی مشابه آنچه در جاوا یافت می‌شود، پشتیبانی می‌کنند و بر افزایش خوانایی و قابلیت نگهداری کد تمرکز دارند. زبان‌های اسکریپتی مانند جاوا اسکریپت و PHP نیز سازوکارهای مدیریت استثنای ویژه خود را دارند. به‌عنوان مثال، جاوا اسکریپت بر ساختار try-catch-finally متکی است و مشابه جاوا و C++، PHP از try-catch با

بلوک‌های finally اختیاری استفاده می‌کند. درنهایت، مدیریت استثناها از روزهای آغازین برنامه‌نویسی، راه درازی را پیموده است، به‌طوری که هر زبان برنامه‌نویسی رویکردها و ساختارهای جدیدی را برای مدیریت مؤثر خطاها و استثناها معرفی کرده است. بنابراین، بررسی تاریخچه و سیر تکامل مکانیسم‌های رسیدگی به استثناء، اطلاعات ارزشمندی را در مورد پیشرفت زبان‌های برنامه‌نویسی و فرایند توسعه نرم‌افزار، فراهم می‌آورد.

۱-۲ تعریف استثناها

استثناء را می‌توان یک رویداد یا وضعیت غیرمنتظره تعریف کرد که حین اجرای یک برنامه رخ می‌دهد و به رفتاری غیرعادی یا نامطلوب می‌انجامد [۲]. رخداد استثناء ممکن است پیامدهای مختلفی مانند خروجی‌های نادرست، ازکارافتادن سیستم یا آسیب‌دیدن داده‌ها شود. مدیریت استثناها فرایند مدیریت و حل استثناها است تا به برنامه اجازه دهد بدون ایجاد آسیب بیشتر به اجرای خود ادامه دهد یا خاتمه یابد. استثناها می‌توانند از منابع مختلفی ناشی شوند، مانند:

- ✓ خطا در کد یا منطق برنامه
- ✓ ورودی‌ها یا اقدامات نامعتبر کاربر
- ✓ محدودیت‌های منابع، مانند تمام شدن حافظه یا فضای دیسک
- ✓ خرابی یا نقص سخت‌افزار
- ✓ عوامل خارجی، مانند وقفه‌های شبکه یا قطع ارتباط با سرویس‌های راه دور

۱-۲-۱ مثال‌هایی از انواع استثناء

در این‌بخش نمونه‌هایی از انواع استثناء را به‌همراه پیاده‌سازی متناظر آن‌ها در C++، با ارجاع به منابعی که پیشتر ذکر شد، ارائه می‌کنیم:

- **تقسیم بر صفر:** این حالت زمانی رخ می‌دهد که تلاش شود عددی بر صفر تقسیم شود، این اقدام یک عملیات ریاضی تعریف نشده است و اغلب منجر به خطا یا استثنای زمان اجرا می‌شود.

```cpp
#include <iostream>
int main() {
    int numerator = 5;
    int denominator = 0;

    try {
        if (denominator == 0)
            throw std::runtime_error("Division by zero!");
        int result = numerator / denominator;
    } catch (const std::runtime_error& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

- **ارجاع به اشاره‌گر تهی**[1]: این وضعیت هنگام تلاش برای دسترسی یا مقداردهی یک اشاره‌گر تهی که به هیچ مکان معتبری در حافظه اشاره نمی‌کند، رخ می‌دهد و می‌تواند منجر به ازکارافتادن برنامه، خطاهای قطعه‌بندی[2] یا سایر خطاهای مرتبط به حافظه شود.

```cpp
#include <iostream>
int main() {
    int* ptr = nullptr;

    try {
        if (!ptr)
            throw std::runtime_error("Null pointer dereference
    !");
        *ptr = 10;
    } catch (const std::runtime_error& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

- **دسترسی خارج از محدودهٔ آرایه**[3]: این مورد زمانی رخ می‌دهد که بخواهیم با استفاده از اندیسی که از اندازهٔ آرایه بزرگ‌تر است یا خارج از محدوده تعریف‌شده آن قرار دارد، به یک عنصر آرایه دسترسی داشته باشیم و می‌تواند باعث رفتار غیرقابل پیش‌بینی، آسیب دیدن حافظه یا ازکارافتادن برنامه شود.

---

1. Null pointer dereference
2. Segmentation faults
3. Out-of-bounds array access

```cpp
#include <iostream>
int main() {
    int arr[5] = {1, 2, 3, 4, 5};

    try {
        int index = 10; // out-of-bounds
        if (index < 0 || index >= sizeof(arr) / sizeof(arr[0])
    )
            throw std::out_of_range("Out-of-bounds array
    access!");
        int val = arr[index];
    } catch (const std::out_of_range& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

- **شکست در تخصیص حافظه**[1]: در سناریوهایی که از تخصیص حافظه پویا استفاده می‌شود (به‌عنوان مثال، استفاده از new در C++ ممکن است تخصیص حافظه به‌دلایل مختلفی مانند حافظه ناکافی یا تکه‌تکه‌شدن[2] با شکست مواجه شود. این وضعیت منجر به بروز یک استثنای std:: bad_alloc می‌شود.

```cpp
#include <iostream>
#include <new>
int main() {
    try {
        int* largeArray = new int[std::numeric_limits<std::
    size_t>::max()];
        delete[] largeArray;
    } catch (const std::bad_alloc& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

- **پیدانشدن فایل یا غیرقابل‌دسترس‌بودن آن**: این وضعیت هنگام تلاش برای دسترسی به فایلی که وجود ندارد یا به‌دلیل مجوزها، مشکلات شبکه یا دلایل دیگر قابل‌دسترسی نیست، رخ می‌دهد و منجر به خطا یا استثناء در زمان اجرا می‌شود.
- **تبدیل نامعتبر نوع داده**: این سناریو هنگام انجام عملیات تبدیل نامعتبر نوع داده، مانند تلاش برای تفسیر مجدد مقداری از یک نوع به یک نوع ناسازگار دیگر، رخ می‌دهد و می‌تواند منجر به رفتار تعریف‌نشده، خرابی داده‌ها یا خطاهای مربوط به نوع شود.

---

1. Memory allocation failure
2. Fragmentation

این مثال‌ها سناریوهای رایجی از رخداد استثناها در برنامه‌نویسی را نشان می‌دهند. مکانیسم‌های رسیدگی به استثناء به توسعه‌دهندگان اجازه می‌دهد تا این استثناها را به‌خوبی شناسایی و مدیریت کنند، و امکان کنترل بهتر بر رفتار برنامه و ارائه استراتژی‌های بازیابی خطا را فراهم کنند.

```cpp
// File not found or inaccessible in C+
#include <iostream>
#include <fstream>
int main() {
    std::ifstream file("non_existent_file.txt");

    try {
        if (!file.is_open())
            throw std::runtime_error("File not found or
    inaccessible!");
    } catch (const std::runtime_error& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

```cpp
// Invalid data type conversion or casting in C++
#include <iostream>
int main() {
    int num = 5;
    void* voidPtr = &num;

    try {
        char* charPtr = reinterpret_cast<char*>(voidPtr);
        if (!charPtr)
            throw std::runtime_error("Invalid data type
    conversion or casting!");
    } catch (const std::runtime_error& e) {
        std::cerr << "Error: " << e.what() << std::endl;
    }

    return 0;
}
```

۱-۲-۲ اشیاء و ساختارهای استثناء

در بیشتر زبان‌های برنامه‌نویسی، استثناها به شکل اشیاء یا ساختارهایی نمود پیدا می‌کنند که حاوی اطلاعاتی درباره ماهیت استثناء، مکان وقوع آن در کد و هرگونه داده اضافی دیگری هستند، و این اطلاعات ممکن است برای تشخیص یا رفع مشکل مفید باشند [۲]. مدیریت‌کننده‌های استثناء می‌توانند از این اطلاعات برای تعیین رویکرد مناسب استفاده کنند، اقداماتی مانند نمایش یک پیام خطای معنی‌دار به کاربر، نمایش خطا با هدف اشکال‌زدایی، یا تلاش برای بازیابی و از سرگیری اجرای عادی برنامه. سازوکارهای مدیریت

استثناء در زبان‌های برنامه‌نویسی مانند C++ و پایتون توانایی شناسایی اشیاء استثناء را فراهم می‌کنند. اشیاء استثناء اطلاعات مربوط به وضعیت استثناء را که در طول اجرای برنامه رخ داده است را در بردارند. در این بخش نحوه استفاده از اشیاء و ساختارهای استثناء در C++ و پایتون را بررسی خواهیم کرد.

۱. زبان برنامه‌نویسی C++: در C++، استثناها به‌وسیله اشیایی معرفی می‌شوند که از کلاس پایه std::exception مشتق شده‌اند. توسعه‌دهندگان می‌توانند کلاس‌های استثنای سفارشی را ازطریق ارث‌بری از std::exception یا کلاس‌های مشتق‌شده از آن ایجاد کنند. این کلاس‌های استثناء می‌توانند شامل متغیرها و متدهای عضو اضافی برای ارائه اطلاعات بیشتر درباره استثناء باشند. هنگامی‌که یک شرط استثناء در C++ رخ می‌دهد، دستورthrow برای پرتاب‌کردن[1] یک شیء استثناء استفاده می‌شود. سپس این شیء استثناء توسط بلوک‌های try-catch کنترل می‌شود. بلوک catch نوع استثناء را مشخص می‌کند و امکان مدیریت خاص براساس نوع استثناء را فراهم می‌آورد. در اینجا مثالی از استفاده از اشیاء استثناء در C++ آورده شده است:

```cpp
#include <iostream>
#include <exception>
class MyException : public std::exception {
public:
    const char* what() const throw() {
        return "My custom exception";
    }
};
int main() {
    try {
        throw MyException();
    } catch(const std::exception& ex) {
        std::cerr << "Exception caught: " << ex.what() << std
    ::endl;
    }

    return 0;
}
```

۲. زبان برنامه‌نویسی پایتون: در پایتون، استثناها بوسیله اشیایی معرفی می‌شوند که نمونه‌هایی از کلاس‌های مشتق‌شده از کلاس پایه Exception هستند. پایتون یک سلسله‌مراتب داخلی از کلاس‌های استثناء را ارائه می‌دهد که شرایط استثنای مختلف را پوشش می‌دهد. توسعه‌دهندگان همچنین می‌توانند با مشتق گرفتن از Exception یا

1. Raise

کلاس‌های مشتق‌شده از آن، کلاس‌های استثنای سفارشی خود را ایجاد کنند. دستور raise برای پرتاب‌کردن یک استثناء در پایتون استفاده می‌شود و شیء پرتاب‌شده باید نمونه‌ای از یک کلاس استثناء باشد. مدیریت استثناء در پایتون با استفاده از بلوک‌های try-except انجام می‌شود. بلوک except نوع استثنایی که باید گرفته شود را مشخص می‌کند و به‌این‌ترتیب، امکان مدیریت ویژه براساس نوع استثناء فراهم می‌شود. در اینجا مثالی از کاربرد اشیاء استثناء در پایتون آورده شده است.

```python
class MyException(Exception):
    def __str__(self):
        return "My custom exception"

try:
    raise MyException()
except Exception as ex:
    print(f"Exception caught: {ex}")
```

کد ۱. *مدیریت استثناء در پایتون*

اشیاء و ساختارهای استثناء امکان کپسوله‌سازی و انتقال اطلاعات مربوط به وضعیت‌های استثنائی در یک برنامه را فراهم می‌آورند. با استفاده از اشیاء استثناء، توسعه‌دهندگان می‌توانند پیام‌های خطای دقیق، ردیابی‌های پشته و سایر اطلاعات مرتبطی را ارائه دهند که می‌تواند به اشکال‌زدایی و مدیریت وضعیت‌های استثناء کمک کند. جزئیات پیاده‌سازی و نحو ممکن است در زبان‌های برنامه‌نویسی مختلف، متفاوت باشد، اما مفهوم استفاده از اشیاء استثناء ثابت باقی می‌ماند. ایده استفاده از استثناها در زبان‌های مختلف به توسعه‌دهندگان امکان می‌دهد کدهای مقاومی در برابر خطا بنویسند.

```cpp
// Checked exceptions in C++
#include <iostream>

void readFile() {
    throw std::runtime_error("File not found");
}
int main() {
    try {
        readFile();
    } catch (const std::exception& e) {
        // Handle the exception
    }

    return 0;
}
```

کد ۱-۲. *استثنای بررسی‌شده (checked) در C++*

۱-۲-۳ استثناهای بررسی‌شده[1] و بررسی‌نشده[2]

در مدیریت استثناء در زبان‌های برنامه‌نویسی معمولاً استثناها به دودسته تقسیم می‌شوند: استثناهای بررسی‌شده و بررسی‌نشده. در این بخش مفاهیم استثناهای بررسی‌شده و بررسی‌نشده و نحوه مدیریت آن‌ها در زبان‌های برنامه‌نویسی مختلف را موردبحث قرار خواهیم داد [۳]. استثناهای بررسی‌شده استثناهایی هستند که باید در زمان کامپایل اعلان یا گرفته[3] شوند. استثناهای بررسی‌نشده، که به‌عنوان استثناهای زمان اجرا نیز شناخته می‌شوند، استثناهایی هستند که نیازی به اعلان یا گرفته‌شدن در زمان کامپایل ندارند. در زبان‌هایی مانند C++ و پایتون، استثناها به‌طور پیش‌فرض بررسی‌نشده درنظر گرفته می‌شوند. برنامه‌نویسان این امکان را دارند که استثناها را با استفاده از بلوک‌های try-catch یا try-except مدیریت کنند، اما در این‌کار الزامی نیست. در ادامه، نمونه‌هایی از استثناهای بررسی‌شده در زبان‌های C++ و پایتون را خواهیم دید.

۱. زبان برنامه‌نویسی C++: در C++، استثناها معمولاً بررسی‌نشده هستند، که این امر رویکردی سبک‌تر و انعطاف‌پذیرتر را برای مدیریت خطا فراهم می‌کند. برنامه‌نویسان می‌توانند درصورت نیاز، استثناها را گرفته و مدیریت کنند، بدون اینکه ملزم به اعلان صریح[4] استثناها باشند. در C++، می‌توانید یک استثناء داخلی یا سفارشی را با استفاده از بلوک try-catchبگیرید، همان‌طور که در مثال بالا توضیح داده شد.

۲. زبان برنامه‌نویسی پایتون: در پایتون، همه استثناها بررسی‌نشده هستند و این امر به توسعه‌دهندگان این اختیار را می‌دهد که خود انتخاب کنند چگونه و چه زمانی استثناها را مدیریت کنند. استفاده از بلوک‌های try-except امکان مدیریت هدفمند استثناها را فراهم می‌کند و نوشتن کدی مختصر و خوانا را آسان‌تر می‌سازد. در پایتون، می‌توانید یک استثناء داخلی را با استفاده از بلوک try-except ایجادکرده یا سفارشی کنید، مانند کد زیر.

1. Checked
2. Unchecked
3. Caught
4. Explicit exception declarations