MISRA C++ Nedir?

Misra c ve c++ için hazırlanmış bir kodlama standardıdır. Özellikle otomotiv, raylı sistemler gibi alanlarda güvenli ve güvenilir yazılımlar yazmak için uyulması gereken bir dizi kurallardan oluşur. Bu kurallara uyulmasa da yazılım çalışabilir ancak güvenilir ve anlaması kolay, belirli standartları olan bir program yazmak için Misra standartlarına uymak büyük önem taşımaktadır.

Not: Bu doküman ile ilgili örneklerin bulunduğu github repositorysine "https://github.com/BerkeCanGoktas/Staj/tree/master/MISRAC%2B%2B" adresinden erişilebilir.

Kurallar

Kurallar konu ve derece bakımından sınıflandırılabilirler. Derece bakımından kurallar required (R) (mutlaka uygulanması gereken), advisory (A) (uyulması gereken ancak uyulmasının mümkün olmadığı durumlarda çiğnenebilen), document (D) (kod içerisinde kullanılması gerekirse zorunlu) olarak üçe ayrılır. Konu bakımından kurallar alt başlıklar şeklinde sınıflandırılacaktır.

Not: Github'ta örneği bulunan projeler (Ö) ile işaretlenmiştir.

1- Dil ile İlgili Hatalar

1.1-Gereksiz Yapılar

1- Bir proje asla ulaşılamaz kod içermemelidir. (R)
2- Bir proje asla infeasible kod içermemelidir. (Infeasible kod: hiçbir olası input ile test edilemeyen bir kod bloğu) (R)
3- Bir proje asla kullanılmamış değişken barındırmamalıdır. (R)
4- Bir proje asla bir kere kullanılmış, volatile olmayan, POD (C++'a özgü olmayan olarak özetlenebilir) değişkenler içermemelidir. (R) (Ö)
5- Bir proje asla kullanılmamış bir tip tanımlaması içermemelidir. (R)
6- Bir proje asla daha sonra kullanılmayan değer verilmiş, volatile olmayan değişken içermemelidir. (Loop kontrolü için kullanılan değişkenler bu kuraldan muaftır) (R) (Ö)
7- Void olmayan ve overload edilmemiş bir fonksiyon'dan return edilen value her zaman kullanılmalıdır. (R)
8- Void bir fonksiyon her zaman programa etki edecek şekilde yazılmalıdır. (Bir dosyadan okumak, local olmayan bir değişkeni değiştirmek vb) (R)
9- Ölü kod bulunmamalıdır. (Ölü kod: çıkartılması outputa etki etmeyen kod) (R)
10- Tanımlanan her fonksiyon en az bir defa çağrılmalıdır. (R)
11- Sanal olmayan fonksiyonlarda kullanılmamış parametre olmamalıdır. (R)
12- Sanal fonksiyon ve bu sanal fonksiyonu override eden diğer tüm fonksiyonlarda kullanılmamış parametre olmamalıdır. (R)

1.2-Depolama

1- Bir obje üst üste binen başka bir objeye atanmamalıdır. (R) (Ö)

1.3-Çalışma Zamanı Hataları

1- Statik, dinamik analizler veya açık kod kontrollerinden en az biri ile çalışma zamanı hataları en aza indirilmelidir. (D)
2- Eğer ki bir fonksiyon hata üretiyorsa, o hata test edilmelidir. (R)

1.4-Aritmetik

1- Scaled-integer ve fixed-point aritmetiği kullanıldığında dökümante edilmelidir. (D)
2- Floating-point aritmetiği kullanıldığında dökümante edilmelidir. (D)

3- Floating-point uygulamaları tanımlı bir floating-point standardına uymalıdır. (D)

2-Genel Hatalar

2.1-Dil

1- Tüm kodlar ISO/IEC 14882:2003 “The C++ Standard Incorporating Technical Corrigendum 1”e uymalıdır. (R)
2- Birden fazla compiler sadece ortak ve tanımlı bir arayüze sahiplerse kullanılabilir. (D)
3- Seçili compilerdaki integer bölme işlemleri kararlaştırılmalı ve dökümante edilmelidir. (D)

2.2- Söz Dizimsel Düzenler

1- Kullanılan karakter seti ve encoding dökümante edilmelidir. (D)
2- Trigraphler kullanılmamalıdır. (Trigraph, bir karaktere karşılık gelen üç karakterlik diziliştir.) (R)
3- Digraphler kullanılmamalıdır. (Digraph, bir karaktere karşılık gelen iki karakterlik diziliştir.) (A)
4- C-style yorumlar içinde /* kullanılmamalıdır. (R) (C-style yorum: /**/)
5- Kod blokları c-style yorum ile yoruma çevrilmemelidir. (R)
6- Kod parçaları C++ yorumları (//) ile yoruma çevrilmemelidir. (R)
7- Farklı tanımlayıcılar tipografik olarak açık olmaldır. (Örn: Identifier, ldentifier (ilk kelime büyük i ile ikincisi küçük L ile yazılmıştır.) (R)
8- İç kapsamdaki bir tanımlayıcı, daha geniş bir kapsamdaki tanımlayıcıyla aynı ada sahip olmamalıdır. (R)
9- typedef isimleri eşsiz olmalıdır. (R)
10- Sınıf, union, enum isimleri eşsiz olmalıdır. (R)
11- Üye olmayan, statik depolama süresine sahip bir obje veya fonksiyonun tanımlayıcısı tekrar kullanılmamalıdır. (A)
12- Bir tipin tanımlayıcısı aynı kapsam içinde bir obje veya fonksiyonu tanımlamamalıdır. (R)
13- Yalnızca ISO/IEC 14882:2003’te tanımlı escape sequence’lar kullanılmalıdır. (Örn: \n,\t) (R)
14- Octal sabitler ve octal escape sequence’lar kullanılmamalıdır. (Örn: int x = 012) (R)
15- “U” son eki tüm octal ve hexadecimal işaretsiz integer sabitlere eklenmelidir. (R)
16- Sabit son ekleri büyük harfle yazılmalıdır. (R)
17- Narrow ve wide stringler birleştirilmemelidir. (R)

3-Genel Konseptler

3.1-Statement ve Tanımlamalar

1- Header dosyalarını birden fazla translation unit’e dahil etmek, tek tanım kuralını çiğnemeden mümkün olmalıdır. (R)
2- Fonksiyonlar blok içinde tanımlanmamalıdır. (R)
3- Bir array tanımlanırken, boyutu açık veya örtük olarak belirtilmelidir. (R)

3.2-Tek Tanım Kuralı

1- Tüm obje ve fonksiyon tanımlamaları uygun tipe sahip olmalıdır. (R)
2- Tek tanım kuralı çiğnenmemelidir. (R)
3- Birden fazla translation unit’te kullanılan tip, obje veya fonksiyonlar sadece tek bir dosyada tanımlanmalıdır. (R)
4- Dış bağlantısı olan bir tanımlayıcı sadece bir tanımlamaya sahip olmalıdır. (R)

3.3- Tanımlayıcı Bölge ve Kapsamlar

1- Dış bağlantısı olan obje ve fonksiyonlar bir header dosyasında tanımlanmalıdır. (R)
2- Eğer bir fonksiyonun iç bağlantısı varsa tüm yeniden tanımlamalar static ön ekini almalıdır. (R)

3.4-İsim Lookup

1- Obje veya tip tanımlamaları, erişilebilirliğini minimize edecek şekilde yapılmalıdır. (Örn: Sadece bir fonksiyon içinde kullanılacak bir değişken o fonksiyonun içinde tanımlanmalıdır.) (R)

3.5-Tipler

1- Bir obje, fonksiyon parametresi veya fonksiyonun return tipinin tanımlamaları ve yeniden tanımlamaları özdeş olmalıdır. (R)
2- Temel nümerik tipler yerine işaret ve boyut belirten typedefler kullanılmalıdır. (A)
3- Floating-point değerlerin bit karşılıkları kullanılmamalıdır. (R)

4-Standart Dönüşümler

1- Bool tipi ifadeler, (=, &&, ||, !, !=, ==, & operatörleri hariç) dahili operatörlere işlenmesi için verilmemelidir. (R)
2- Enum tipi ifadeler, ([], =, ==, !=, <, <=, >, >=, & operatörleri hariç) dahili operatörlere işlenmesi için verilmemelidir. (R)
3- Char ve wchar_t tipi ifadeler, (=, ==, !=, & operatörleri hariç) dahili operatörlere işlenmesi için verilmemelidir. (R)
4- NULL bir integer olarak kullanılmamalıdır. (R)
5- 0 (sıfır) null-pointer-constant olarak kullanılmamalıdır. (R)

5-İfadeler

5.1-Genel

1- Standartların izin verdiği ölçüde, bir ifadenin her değerlendirmesi aynı değeri vermelidir. (R) (Ö)
2- C++ operatör üstünlükleri (() operatörü ile sağlanır, işlem önceliği gibidir) mümkün olduğunca az kullanılmalıdır. (Örn: x = 7 + 2 uygun, x = 7 + (2) gereksiz) (A)
3- Bir cvalue ifadesi örtük olarak başka bir underlying türe dönüştürülmemelidir. (R)
4- Örtük bir integral dönüşüm underlying türün işaretliliğini değiştirmemelidir. (R)
5- Örtük floating-integral dönüşümleri olmamalıdır. (R)
6- Örtük integral veya floating-point dönüşümleri underlying türün boyutunu düşürmemelidir. (R)
7- Cvalue ifadelerden açık olarak floating-integrale dönüşüm yapılmamalıdır. (R)
8- Açık integral veya floating-type dönüşümleri, cvaluenun underlying türünün boyutunu büyütmemelidir. (R)
9- Açık integral dönüşümü, cvaluenun underlying türünde işaret değişimine yol açmamalıdır. (R)
10- Unsigned char veya unsigned short underlying türüne sahip bir işlenen, ~ veya << operatörü ile işlenirse sonuç anında o işlenenin underlying türüne cast edilmelidir. (R)
11- Char tipi sadece karakter değerlerinin depolanması ve kullanımı için kullanılmalıdır. (R)
12- Signed char ve unsigned char tipleri sadece nümerik karakterlerin depolanması ve kullanımı için kullanılmalıdır. (R)
13- Bir if-statement veya iteration-statementın koşulu bool türünde olmalıdır. (R)
14- Bir koşul operatörünün ilk işleneni bool türünde olmalıdır. (R)

15- Pointer aritmetiği sadece array indexing formunda olmalıdır. (R)
16- İşlenen bir pointer ve bu işleneni kullanan bir pointer aritmetiği işleminin sonucu olan herhangi bir pointer aynı arrayin elemanlarına işaret etmelidir. (R)
17- Pointerlar arası çıkarma işlemi sadece aynı arrayin elementlerine işaret eden pointerlar arası uygulanmalıdır. (R)
18- Aynı arrayi işaret ettikleri durumlar hariç; <, <=, >, >= operatörleri pointer objelerini karşılaştırmak için kullanılmamalıdır. (R)
19- Obje tanımlamaları, ikiden fazla pointer yönlendirmesi içermemelidir. (R)
20- Binary bit operatörleri tarafından işlenen sabit olmayan değerler aynı underlying türe sahip olmalıdır. (R)
21- Bit operatörleri sadece unsigned underlying türden işlenenlere uygulanmalıdır. (R)

5.2-Postfix İfadeler

1- && veya || operatörleri tarafından işlenen her değer bir postfix ifade olmalıdır. (R)
2- Sanal bir base sınıfı işaret eden bir pointer, dynamic_cast ile yalnızca cast edilmiş sınıflarına ait pointerlara atanmalıdır. (R)
3- Base sınıftan türetilmiş sınıflara cast etme işlemleri, polymorphic türler üstünden yapılmamalıdır. (A)
4- C-style castler (void castler hariç) ve functional notation castler (açık constructor çağırmaları hariç) kullanılmamalıdır. (R)
5- Bir cast, bir pointer veya referansın const veya volatile özelliğini kaldırmamalıdır. (R)
6- Bir cast, bir fonksiyonu işaret eden pointerları başka herhangi bir türden pointera dönüştürmemelidir (başka fonksiyonu işaret eden pointerlar dahil). (R)
7- Pointer türü olan bir obje, ilişkisiz başka bir pointer türüne doğrudan veya dolaylı olarak atanmamalıdır. (R)
8- Integer veya pointer to void türü bir obje, pointer türü bir objeye dönüştürülmemelidir. (R)
9- Bir cast pointer türünü integral türüne dönüştürmemelidir. (A)
10- Arttırma (++) ve azaltma (--) operatörleri ifade içerisinde diğer operatörlerle beraber kullanılmamalıdır. (A)
11- Virgül (,), &&, || operatörleri overload edilmemelidir. (R)
12- Array tipi bir tanımlayıcı bir fonksiyona argüman olarak, pointer ile verilmemelidir. (R)

5.3-Unary İfadeler

1- !, &&, || operatörlerine verilen her işlenen bool türü olmalıdır. (R)
2- Unary eksi operatörü underlying türü unsigned olan bir ifadeye uygulanmamalıdır. (R)
3- Unary & operatörü overload edilmemelidir. (R)
4- Sizeof operatörü kullanılırken başka yan işlemler yapılmamalıdır. (R)

5.4-Shift Operatörleri

1- Bir shift operatörünün sağ tarafındaki işlenen 0 ile sol taraftaki işlenenin underlying türünün bitinin 1 eksiği arasında olmalıdır. (R)

5.5-Mantıksal AND Operatörü

1- && ve || operatörlerinin sağ tarafındaki işlenen başka yan işlemler içermemelidir. (R)

5.6-Atama Operatörleri

1- Bir binary operatör ile bu operatörün atama operatörü arasındaki anlamsal ilişki korunmalıdır. (Örn: + ve +=) (R)

5.7-Virgül Operatörü

1- Virgül operatörü kullanılmamalıdır. (R)

5.8-Sabit İfadeler

1- Constant unsigned integer ifadelerin değerlendirilmesi wrap-arounda sebep olmamalıdır. (A)

6-Statements

6.1-Expression Statements

1- Atama operatörleri alt ifadelerde kullanılmamalıdır. (R)
2- Floating-point ifadeler direkt veya dolaylı olarak eşitlik/eşitsizlik açısından test edilmemelidir. (R)
3- Ön işleme öncesi, null bir statement bir satırda yalnızca tek başına bulunabilir. Bir white-space sonrasında yorum eklenebilir. (R)

6.2-Compound Statements

1- Switch, while, do… while, for statementların gövdeleri compound (birleşik) olmalıdır. (Yani mutlaka süslü parantez ({}) içinde olmalıdır.) (R)

6.3-Selection Statements

1- Bir if yapısının ardından compund statement (süslü parantezle belirtilmiş kod bloğu) gelmelidir. Else keywordünden sonra bir compound statement veya if yapısı gelmelidir. (R)
2- Tüm if… else if yapıları bir else yapısı ile sonlanmalıdır. (R)
3- Bir switch yapısı well-formed switch statement olmalıdır. (R)
4- Bir switch-label (case 1 vb.) sadece switch yapısı ile aynı derecede kullanılmalıdır. (Yani case içinde case kullanılmamalıdır. (R)
5- Bir koşulsuz throw veya break statement her boş olmayan switch yapısında bulunmalıdır. (R)
6- Bir switch yapısının sonu default-clause olmalıdır. (R)
7- Bir switch ifadesinin koşulu bool türü olmamalıdır. (R)
8- Her switch yapısı en az bir case içermelidir. (R)

6.4-Iteration Statements

1- Bir for döngüsü, floating-type olmayan, yalnız bir adet loop-counter içermelidir. (R)
2- Bir loop-counter, -- veya ++ operatörleri ile kullanılmıyorsa o loop-counter yalnızca <=, <, >, >= operatörlerinden biri ile işlenmelidir. (R)
3- Loop-counter, koşul veya statement içerisinde güncellenmemelidir. (R)
4- Loop-counterlar --, ++, -=n, +=n (n döngü boyunca sabit kalmalıdır) operatörlerinin biri ile güncellenmelidir. (R)
5- Loop-counter hariç bir loop-control değişkeni condition veya expression içinde güncellenmemelidir. (R)
6- Statement içerisinde güncellenen bir loop-control değişkeni (loop-counter hariç) bool türünde olmalıdır. (R)

6.5-Jump Statements

1- Goto statement ile referans verilen her etiket, goto statement ile aynı blok veya daha iç bir blokta tanımlı olmalıdır. (R)
2- Bir goto statement kendisinden sonra tanımlanmış bir etikete atlamalıdır. (R)
3- Continue statement sadece well-formed for döngülerinde kullanılmalıdır. (R)
4- Hiçbir iteration statement birden fazla break veya goto statement içermemelidir. (R)
5- Bir fonksiyonun sonunda bir çıkış olmalıdır. (R)

7-Declarations

7.1-Specifiers

1- Değiştirilmeyen bir değişkene const ibaresi eklenmelidir. (R)
2- Eğer ki bir fonksiyona parametre olarak verilen bir pointer veya reference, const bir değişkeni gösteriyorsa const ibaresi eklenmelidir. (R)

7.2-Enumeration Declarations

1- Enum underlying türü olan bir ifade sadece enumeratorün belirttiği değerleri alabilir. (R)

7.3-Namespaces

1- Global namespace sadece main, namespace declarations ve extern “C” declerations içermelidir. (R)
2- Main kelimesi global main fonksiyonu hariç bir fonksiyonu tanımlamak için kullanılmamalıdır. (R)
3- Header file içinde isimsiz namespaceler bulunmamalıdır. (R)
4- Using-directives kullanılmamalıdır. (R)
5- Aynı namespace içerisinde bir tanımlayıcının birden fazla declarationı, o tanımlayıcının using-declarationı içinde üst üste binmemelidir. (R)
6- Using-directives ve using-declarations (class ve fonksiyon içi using-declarationlar hariç) header file içinde kullanılmamalıdır. (R)

7.4-Asm Declaration

1- Tüm assembler kullanımları dökümante edilmelidir. (D)
2- Assembler yönergeleri sadece asm declaration kullanımı ile tanımlanmalıdır. (R)
3- Assembly dili enkapsüle ve izole edilmelidir. (R)

7.5-Linkage Specifications

1- Bir fonksiyon, içinde tanımlanmış bir değişkene ait (parametreler dahil) referans veya pointerı döndürmemelidir. (R)
2- Otomatik depolamalı bir objenin adresi, o obje yok olduktan sonra da varlığını sürdürebilecek başka bir objeye atanmamalıdır. (R)
3- Bir fonksiyon, referans veya const referans ile verilen bir parametrenin referans veya pointerını geri döndürmemelidir. (R)
4- Fonksiyonlar doğrudan veya dolaylı olarak kendilerinni çağırmamalıdır. (A)

8-Declarations

8.1-Genel

1- Bir init-declarator-list veya member-declarator-list tek tür init-declarator veya member-declarator içermelidr. (R)

8.2-Declaratorların Anlamı

1- Başka bir fonksiyonu override eden virtual fonksiyonlar, ya override ettikleri fonksiyonla aynı default argümanları kullanmalıdırlar ya da hiç default argument tanımlamamalıdırlar. (R)

8.3-Fonksiyon Declarations

1- Üç nokta kullanarak fonksiyon tanımlamaları yapılmamalıdır. (R)
2- Bir fonksiyon yeniden tanımlanırsa, parametrelerin tanımlayıcıları ilk tanımlamadaki ile aynı kullanılmalıdır. (R)
3- Void olmayan tüm fonksiyonların sonu bir return statement içermelidir. (R)
4- Bir fonksiyon tanımlayıcısı ya o fonksiyonu çağırmak için ya da & ile kullanılabilir. (R)

8.4-Initializers

1- Tüm değişkenlerin kullanılmadan önce bir değeri olmalıdır. (R)
2- Süslü parantezler, array ve structureları değer ile başlatmak için kullanılmalıdır. (R)
3- Bir enumerator listesinde ilk eleman hariç açık bir şekilde = ile başlatma yapılmamalıdır. (Tüm elemanlar = ile başlatılabilir.) (R)

9-Sınıflar

9.1-Üye Fonksiyonlar

1- Const üye fonksiyonlar class-dataya const olmayan pointer veya referans geri döndürmemelidir. (R)
2- Üye fonksiyonlar class-dataya const olmayan handlelar geri döndürmemelidir. (R)
3- Eğer ki bir üye fonksiyonu static yapılabiliyorsa static, const yapılabiliyorsa const yapılmalıdır. (R)

9.2-Unions

1. Unionlar kullanılmamalıdır. (R)

9.3-Bit-fields

1. Bir bit-fieldı temsil eden bitlerin mutlak pozisyonlarına ihtiyaç duyulduğunda, bit-fieldların davranış ve paketlemeleri dökümante edilmelidir. (D)
2. Bit-fieldlar bool, explicitly unsigned veya signed integral türü olmalıdır. (R)
3. Bit-fieldların enum türü olmamalıdır. (R)
4. İsimlendirilmiş, signed integer türü bit-fieldlar bir bitten uzun olmalıdır. (R)

10-Derived Sınıflar

10.1-Çoklu Base Sınıflar

1- Sınıflar virtual baselerden türetilmemelidir. (A)
2- Bir base sınıf sadece diamond hiyerarşisi içinde ise virtual olarak tanımlanmalıdır. (R)
3- Ulaşılabilir bir base sınıf aynı hiyerarşi içinde hem virtual hem de non-virtual olmamalıdır. (R)

10.2-Member Name Lookup

1- Aynı çoklu kalıtım hiyerarşisi içindeki tüm erişilebilir varlıklar eşsiz olmalıdır. (A)

10.3-Sanal Fonksiyonlar

1- Kalıtım hiyerarşisinin her yolunda, sanal fonksiyonun en fazla bir tanımı olmalıdır. (R)
2- Override eden tüm samal fonksiyonlar virtual keywordünü almalıdır. (R)
3- Bir sanal fonksiyon pure virtual olarak tanımlanmışsa yalnızca bir püre virtual fonksiyon tarafından override edilebilir. (R)

11-Member Access Control

11.1-Genel

1- POD olmayan (destructor, constructor, sanal fonksiyon içermeyen) sınıfların member dataları private olmalıdır. (R)

12-Special Member Functions

12.1-Constructors

1- Bir objenin dinamik türü, constructor veya destructorın gövdesinde kullanılmamalıdır. (R)
2- Tüm constructorlar, açık olarak base sınıflarının constructorlarını çağırmalıdır. (A)
3- Tek argümanla çağrılan tüm constructorlar explicit keywrrodünü almalıdır. (R)

12.2-Sınıf Objelerini Kopyalama

1- Bir copy constructor sadece kendi base sınıfları ve üyesi olduğu sınıfın statik olmayan üyelerini başlatmalıdır. (R)
2- Kopya atama operatörü bir soyut sınıf içinde protected veya private olarak tanımlanmalıdır. (R)

13-Templates

13.1-Template Declarations

1- Üye olmayan bir generic fonksiyon sadece associated olmayan bir namespace içinde tanımlanabilir. (R)
2- Generic ve tek bir parametresi olan bir template constructor var ise bir copy constructor tanımlanmalıdır. (R)
3- Generic parametresi olan bir template assignment operatörü var ise bir copy assignment operatörü tanımlanmalıdır. (R)

13.2-Name Resolution

1- Bağlı bir basei olan bir template sınıf içindeki, bağlı base içinde bulunabilecek her isim bir qualified-id ile veya this-> ile gösterilmelidir. (R)
2- Overload resolution ile seçilen bir fonksiyon, çeviri ünitesinde daha önceden tanımlanmış bir fonksiyona çözümlenmelidir. (R)

13.3-Template Instantiation and Specialization

1- Tüm sınıf templateleri, fonksiyon templateleri, sınıf templatelerinin üye fonksiyonları ve statik üyeleri en az bir kere başlatılmalıdır. (R)
2- Hiçbir template specialization için, specializationda kullanılan template-argümanlarına sahip templatein açık başlatmaları programı ill-formed yapmamalıdır. (R)
3- Bir template için yapılan tüm parçalı ve açık specializationlar, ana template ile aynı file içerisinde olmalıdır. (R)

13.4-Function Template Specialization

1- Overload edilmiş fonksiyon templateleri açık olarak specialize edilmemelidir. (R)
2- Bir fonksiyon çağrısı için geçerli fonksiyon seti ya hiç fonksiyon specialization içermemelidir ya da sadece fonksiyon specialization içermelidir. (A)

14-Exception Handling

14.1-Genel

1- Exceptionlar sadece error handling için kullanılmalıdır. (D)
2- Bir exception objesinin türü pointer olmamalıdır. (A)
3- Goto veya switch statement ile try catch bloğuna geçiş yapılmamalıdır. (R)

14.2-Throwing an Exception

1- Bir throw statementın atama ifadesi başka bir exceptionın fırlatılmasına sebep olmamalıdır. (R)
2- NULL açıkça fırlatılmamalıdır. (R)
3- Boş bir throw (throw;) sadece bir compund-statementın catchinde kullanılmalıdır. (R)

14.3-Handling an Exception

1- Exceptionlar sadece programın başlangıcından sonra ve bitişinden önce kaldırılmalıdır. (R)
2- Başka türlü catch edilemeyen her exception için en az bir exception handler bulunmalıdır. (A)
3- Bir sınıf constructor veya destructorının function-try-block uygulamasının handlerları bu sınıfın veya baseinin static olmayan üyelerine referans göstermemelidir. (R)
4- Açıkça fırlatılan her exception için kodun her yolunda uygun türde bir handler bulunmalıdır. (R)
5- Bir sınıf türü exceptionı her zaman bir referans tarafından yakalanmalıdır. (R)
6- Bir derived sınıf ve onun tüm veya bazı baseleri için, bir try-catch statement veya function-try-block birden fazla handler içeriyor ise handlerlar derivedtan base sınıfa doğru sıralanmalıdır. (R)
7- Bir try-catch statement veya function-try-block içinde birden fazla handler var ise, ellipsis (hepsini yakala) handler en son sırada olmalıdır. (R)

14.4-Exception Specifiactions

1- Bir fonksiyon exception-specification ile tanımlanmışsa, bu fonksiyonun tüm tanımlamaları (diğer çeviri ünitelerinde) aynı type-id seti ile yapılmalıdır. (R)

14.5-Special Functions

1- Bir sınıf destructorı exception ile sonlanmamalıdır. (R)
2- Bir fonksiyonun tanımlanması, bir exception-specification içeriyorsa bu fonksiyon sadece belirtilmiş türlerden exceptionlar fırlatabilmelidir. (R)
3- terminate() fonksiyonu kapalı olarak çağırılmamalıdır. (R)

15-Preprocessing Directives

15.1-Genel

1- Bir file içerisindeki #include direktiflerinden önce sadece diğer preprocessor direktifleri veya yorumlar gelebilir. (R)
2- Macrolar sadece global namespace içinde #define veya #undef edilmelidir. (R)
3- #undef kullanılmamalıdır. (R)

4- Fonskiyonvari (isim(parametre) şeklinde) macrolar tanımlanmamalıdır. (R)
5- Fonksiyonvari macroların argümanları preprocessing direktif gibi gözüken tokenlar almamalıdır. (R)
6- Fonskiyonvari bir macronun tanımlanmasında, parametreler her kullanımda parantez içerisine alınmalıdır (# veya ## operatörleri ile kullanılmıyorlarsa). (R)
7- Tanımlanmamış macro belirteçleri #if ve #elif preprocessor direktifleri içinde kullanılmamalıdır (tanımlanmış bir operatöre işlenen olarak verildikleri durumlar hariç). (R)
8- # tokeni bir satırdaki ilk token ise hemen bir preprocessing tokeni tarafından takip edilmelidir. (R)

15.2-Conditional Inclusion

1- defined preprocessorü yalnızca iki standart formdan (defined (identifier) veya defined idenfitifer) birinde kullanılmalıdır. (R)
2- Tüm #else ve #endif preprocessor direktifleri ilişkili oldukları #if veya #ifdef direktifleri ile aynı fileda bulunmalıdır. (R)

15.3-Source File Inclusion

1- Preprocessor sadece file inclusion ve include guardlar için kullanılmalıdır. (R)
2- C++ macroları yalnızca include guardlar, tür niteleyicileri ve depolama sınıfı belirleyiciler için kullanılmalıdır. (R)
3- Include guardlar sağlanmalıdır. (R)
4- ‘, “, /*, // header fileın isminde olmamalıdır. (R)
5- \ karakteri header file isminde olmamalıdır. (A)
6- #include direktifi <filename> veya “filename” ile takip edilmelidir. (R)

15.4-Macro Replacement

1- Bir macro tanımlamasında en fazla bir # ve/veya ## olmalıdır. (R)
2- # ve ## operatörleri kullanılmamalıdır. (A)

15.5-Pragma Directive

1- Tüm #pragma kullanımı dökümante edilmelidir. (D)

16-Library Introduction

16.1-Genel

1- Standard librarydeki tanımlama, macro ve fonksiyonlar, define, undefine veya redifine edilmemelidir. (R)
2- Standard librarydeki macro ve obje isimleri tekrar kullanılmamalıdır. (R)
3- Standard librarydeki fonksiyon isimleri override edilmemelidir. (R)
4- Tüm library kodları MISRA C++’a uygun olmalıdır. (D)
5- setjmp macrosu ve longjmp fonksiyonu kullanılmamalıdır. (R)

17-Language Support Library

17.1-Genel

1- C library kullanılmamalıdır. (R)
2- <cstdlib> kütüphanesinin atof, atoi, atol fonksiyonları kullanılmamalıdır. (R)
3- <cstdlib> kütüphanesinin abort, exit, getenv ve system fonksiyonları kullanılmamalıdır. (R)

4- <ctime> kütüphanesinin time handling fonksiyonları kullanılmamalıdır. (R)
5- <cstring> kütüphanesinin unbounded fonksiyonları kullanılmamalıdır. (R)

17.2-Implementation Properties

1- Offsetof macrosu kullanılmamalıdır. (R)

17.3-Dynamic Memory Management

1- Dynamic heap memory allocation kullanılmamalıdır. (R)

17.4-Other Runtime Support

1- <csignal> kütüphanesinin sinyal handling olanakları kullanılmamalıdır. (R)

18-Diagnostics Library

18.1-Error Numbers

1- Hata belirteci errno kullanılmamalıdır. (R)

19-Input/Output Library

19.1-Genel

1- <cstdio> kütüphanesi kullanılmamalıdır. (R)

Berke Can Göktaş

10.11.2021