SE302 – ПХ БҮТЭЭЛТ
Лекц 7

Defensive Programming

- Таны програмыг хамгаалах тухай биш
  - Програм сайн ажилладаг байх
  - Аюултай зүйл хийхэд
    - Гэмтэхгүй байх
  - Түүнийг ашиглаж байгаа хүмүүсийн алдаанаас өөрийгөө хамгаалсан, хариуцлагатай байх
  - Нэг функц буруу өгөгдөл дамжуулахад
    - Ямар нэгэн гэмтэл гарахгүй байх
    - Энэ бол өөр функцын алдаа

Protecting Your Program from Invalid Inputs

- Хог цуглуулах, хог гаргах (Garbage in, Garbage out)
  - Сайн програм хог гаргадаггүй байх
  - Сайн програмд хог оруулахад
    - Юу ч гаргадаггүй байх
    - Алдааны мэссэж гаргах
    - Хог оруулахыг зөвшөөрөхгүй байх
  - Оролтын хогыг үндсэндээ 3 янзын аргаар удирдана

Protecting Your Program from Invalid Inputs

- Гадаад эх үүсвэрээс ирж буй бүх өгөгдлийн утгыг шалгах
  - Өгөгдлийг унших (Өгөгдөл тохирсон хүрээнд байх)
    - Файл
    - Хэрэглэгчийн оруулсан өгөгдөл
    - Сүлжээ
    - Бусад гадаад интерфэйс
  - Жишээ: Санхүүгийн гүйлгээний ID, гэх мэт
  - Халдлага хийх сул тал болж болно
    - Buffer overflow
    - Injected SQL
    - Injected HTML
    - Integer overflow

Protecting Your Program from Invalid Inputs

- Функцын бүх параметрын утгыг шалгах
  - Гадаад эх үүсвэрээс ирсэн өгөгдлийг шалгах шиг
  - Өөр функцээс дамжуулсан утгууд
  - Ямар функцууд оролтын өгөгдлүүдээ шалгах шаардлагатай

Protecting Your Program from Invalid Inputs

- Буруу оролтын өгөгдлийг яахыг шийдэх
  - Нэг удаа оролтын буруу параметр илрүүллээ
    - Тэгвэл одоо юу хийх вэ? (Цаашид ярина)
  - Defensive coding
    - Алдаа оруулахгүй байх
    - Давтагдсан дизайн ашиглах
    - Кодлохоос өмнө псевпо код бичих
    - Кодлохоос өмнө тестын сорилоо бэлдэх
  - Жижиг харагдаж байгаа алдаа нь
    - Таны төсөөлсөнөөс илүү өөр байж болно

Protecting Your Program from Invalid Inputs

Assertions

- Хөгжүүлэлтийн явцад ашиглагддаг код
  - Ихэвчлэн функц эсвэл макро
  - Өөрийгөө ажиллаж байгаа эсэхийг шалгах
  - Баталгаа үнэн үед
    - Бүх зүйл хэвийн ажиллаж байна
  - Баталгаа худал үед
    - Кодонд гэнэтийн алдаа илэрсэн
  - Жишээ
    - Хэрэглэгчийн бүртгэлийг хэзээ ч 50,000-с илүү гарахгүй гэж үзвэл
      - 50,000-с бага үед ямар ч асуудалгүй
      - Алдаа илэрсэнийг програмд зарлана

Assertions

- Баталгаа нь том, төвөгтэй, өндөр нарийвчлалтай програмд ялангуяа ашигтай байдаг.
- Код өөрчлөгдөхөд үүсэх алдааг илүү хурдан засах боломж олгодог
- Баталгаа нь ихэвчлэн 2 аргумент авдаг
  - Boolean - төрлийн илэрхийлэл (Үнэн байх)
  - Үзүүлэх мэссэж

Java Example of an Assertion

```
assert denominator != 0 : "denominator is unexpectedly equal to 0.";
```

Assertions

- Дараах зүйлсийг шалгах зорилгоор баталгаа ашигладаг
  - Параметрын утга заасан хүрээнд байгаа эсэх
  - Функц ажиллахад file эсвэл stream нээлттэй эсэх
  - Уншиад бичихэд Read-only эсвэл write-only байгаа эсэх
  - Оролтын параметрын утгыг функцээр өөрчлөөгүй байх
  - Заагч нь null биш байх
  - Array болон бусад бүтэц хамгийн багадаа X элементтэй байх
  - Агуулагч хоосон биш байх (array, list, stack)

Assertions

- Баталгаа нь хөгжүүлэлтийн процессийн үед хэрэглэгчддэг
  - Алдааны мэссэж нь хэрэглэгчид үзэгдэхгүй
- Бүтээгдэхүүн гаргахад баталгааг кодноос ялгаж хөрвүүлэлт хийдэг

- Баталгаа нь систеийн гүйцэтгэлийг үнэлэхгүй

Building Your Own Assertion Mechanism

- Ихэнх програмчлалын хэлүүд баталгааг дэмждэг
  - C++, Java, MS Visual Basic, C#
  - C# - Debug.Assert()

C++ Example of an Assertion Macro

```
#define ASSERT( condition, message ) {          \
   if ( !(condition) ) {                        \
       LogError( "Assertion failed: ",          \
           #condition, message );               \
       exit( EXIT_FAILURE );                    \
    }                                           \
}
```

Guidelines for Using Assertions

- Гарч болох нөхцөлд error-handle ашиглах
  - Буруу оролтын өгөгдлийг шалгах
  - Тохирох хариу үр дүнг үзүүлэх
- Хэзээ ч гарахгүй нөхцөлд баталгаа ашиглах
  - Кодын алдааг шалгах (bug)
  - Програмын кодонд өөрчлөлт оруулах
  - Дахин хөрвүүлэлт хийх
  - Програмын шинэ хувилбар гаргах

Guidelines for Using Assertions

- Баталгаанд гүйцэтгэх код бичихээс зайлсхийх
  - Бүтээгдэхүүн гаргахад баталгааг орхиж хөрвүүлэлт хийдэг

Visual Basic Example of a Dangerous Use of an Assertion

```
Debug.Assert( PerformAction() ) ' Couldn't perform action
```

Visual Basic Example of a Safe Use of an Assertion

```
actionPerformed = PerformAction()
Debug.Assert( actionPerformed ) ' Couldn't perform action
```

Guidelines for Using Assertions

- Өмнөх нөхцөл болон дараах нөхцөл байдлаар баталгааг ашиглах
  - Гэрээгээр загварчлах (Design by Contract)
    - ПХ хөгжүүлэлтийн хандлага
    - Precondition - объект эсвэл функц дуудагдахаас өмнө бүх зүйл хэвийн байх
      - Түүний гэрээгээр хүлээсэн үүрэг
    - Postcondition - Гүйцэтгэж дууссаны дараа бүх зүйл хэвийн байх
      - Түүний гэрээгээр хүлээсэн үүрэг

Guidelines for Using Assertions

Visual Basic Example of Using Assertions to Document Preconditions and Postconditions

```
Private Function Velocity ( _
   ByVal latitude As Single, _
   ByVal longitude As Single, _
   ByVal elevation As Single _
   ) As Single

   ' Preconditions
   Debug.Assert ( -90 <= latitude And latitude <= 90 )
   Debug.Assert ( 0 <= longitude And longitude < 360 )
   Debug.Assert ( -500 <= elevation And elevation <= 75000 )

   ...

   ' Postconditions
   Debug.Assert ( 0 <= returnVelocity And returnVelocity <= 600 )

   ' return value
   Velocity = returnVelocity
End Function
```

Guidelines for Using Assertions

- Баталгаа ба error-handling code
  - Функцэд баталгаа эсвэл error-handling code ашиглаж болно
    - Зарим судлаачид аль нэг л хэрэгтэй гэж зөвлөдөг
  - Бодит амьдрал дээр
    - 5-10 жил хөгжүүлэлт
    - Систем дизайнерууд нь цаг хугацаа, орон зайгаар тусгаарлагдсан
    - Ялгаатай хэсгүүдэд ялгаатай технологи

Guidelines for Using Assertions

```
Private Function Velocity ( _
   ByRef latitude As Single, _
   ByRef longitude As Single, _
   ByRef elevation As Single _
   ) As Single

   ' Preconditions
   Debug.Assert ( -90 <= latitude And latitude <= 90 )
   Debug.Assert ( 0 <= longitude And longitude < 360 )
   Debug.Assert ( -500 <= elevation And elevation <= 75000 )
   ...

   ' Sanitize input data. Values should be within the ranges asserted above,
   ' but if a value is not within its valid range, it will be changed to the
   ' closest legal value
   If ( latitude < -90 ) Then
      latitude = -90
   ElseIf ( latitude > 90 ) Then
      latitude = 90
   End If
   If ( longitude < 0 ) Then
      longitude = 0
   ElseIf ( longitude > 360 ) Then
   ...
```

Error-Handling Techniques

- Баталгаа хэзээ ч үүсэх ёсгүй алдааг зүгшрүүлэхэд ашиглагддаг
- Алдаа гарвал хэрхэн зүгшрүүлэх вэ?
  - Боловсруулаагүй утга буцаах (тодорхой нөхцөлд)
  - Дараагийн зөв өгөгдлийг ашиглах
  - Өмнөхтэй ижил утга буцаах
  - Зөвшөөрөгдсөн утгатай ойролцоо утга буцаах
  - Анхааруулга Лог руу бичих
  - Алдааны код буцаах
  - Алдаа боловсруулах функц дуудах (object)
  - Алдааны мэссэж үзүүлэх
  - Унтраах

Error-Handling Techniques

- Боловсруулаагүй утга буцаах (тодорхой нөхцөлд)
  - Заримдаа буруу өгөгдөлд хоргүй утга буцаах нь хамгийн сайн байдаг
    - Тоон тооцоололд - 0
    - Тэмдэгт мөрийн үйлдэлт - хоосон тэмдэгт мөр
    - Зурах үйлдэлд - анхдагч background, foreground color

Error-Handling Techniques

- Дараагийн зөв өгөгдлийг ашиглах
  - Өгөгдлийн сангаас бичлэг уншиж байхад
    - Эвдрэлтэй бичлэг тулгарах
    - Зөв бичлэгийг олох хүртлээ магадгүй үргэлжлүүлж болох юм
  - Халуун хэмжигчийн мэдээллийг секундэнд 100 удаа уншдаг
    - Нэг удаа зөв өгөгдлийг уншиж чадахгүй байж болно
    - 1/100 секунд хүлээгээд дараагийн өгөгдлийг унших

Error-Handling Techniques

- Өмнөхтэй ижил мэдээллийг буцаах
  - Халуун хэмжигч програм 1 удаа зөв мэдээллээ уншиж чадахгүй үед
    - Энгийнээр хамгийн сүүлд уншсан утгаа буцааж болно
  - Видео тоглоомонд
    - Дэлгэцийн тодорхой хэсгийг зурах өнгө нь буруу гэдгийг илрүүлсэн
  - Мөнгөний машины хувьд
    - Гүйлгээ баталгаажуулах үед
    - Энэ аргыг ашиглаж болох уу?

Error-Handling Techniques

- Зөвшөөрөгдсөн утгатай ойролцоо утга буцаах
  - Халуун хэмжигч магадгүй 0-100 цельс тохируулагдсан байдаг гэвэл
    - 0-с бага утга илэрвэл 0-ийг орлуулах
    - 100-с их байвал 100-г орлуулах
  - Тэмдэгт мөрийн хувьд урт нь
    - 0-с бага бол 0 гэж үзэх
  - Машин ухрах үед хурд хэмжигч нь
    - Сөрөг утга заадаггүй
    - Багадаа 0 байна

Error-Handling Techniques

- Анхааруулах мэссэжийг файл руу бүртгэх (Log)
  - Буруу өгөгдөл илрэх үед файл руу анхааруулга бичээд
    - Програмыг цааш үргэлжлүүлэх
    - Энэ арга нь өөр техникүүдийг давхар хэрэглэнэ
      - Ойролцоох утга ашиглах
      - Дараагийн зөв өгөгдлийг ашиглах
  - Хэрэгв Лог ашиглаж байвал аюулгүйгээр зохион байгуулах
    - Бүгдэд нээлттэй байхаас хамгаалах
    - Шифрлэх

Error-Handling Techniques

- Алдааны код буцаах
  - Системийн тодорхой хэсэгт алдаа зүгшрүүлэх үед
  - Бусад хэсгүүд зөвхөн дотооддоо зүгшрүүлдэггүй байх
  - Алдаа гарахад систем дээгүүр мэдэгдэх тогтсон механизм хэрэгтэй
    - Төлөвийн хувьсагчид утга олгох
    - Функцын буцах утга шиг төлөвийн утга буцаах
    - Онцгой тохиолдол шидэх (exception)

Error-Handling Techniques

- Унтраах
  - Алдаа илрэхэд системийг унтраах
  - Энэ хандлага аюулгүй талаасаа хэрэглэгддэг.
  - Цацраг идэвхитэй туяаг хянадаг удирдах програм
    - Хорт хавдартай өвчтөнгүүдийг эмнэх
    - Цацрагын туяаны доз хэтрэх
    - Аль арга нь хамгийн тохиромжтой вэ?
      - Ойролцоо утга
      - Хамгийн сүүлд ашигласан утга

Exceptions

- Онцгой үйл явдалд алдаа гарах
- Кодонд гэнэтийн үйл явдал болох
  - Үүнийг яаж шийдэхээ мэдэхгүй байх
  - Програмист үүнийг мэдэх учиртай
    - Массивын индэкс хэтрэх
    - 0 хуваах оролдлого хийх
    - Файлын төгсгөлд хүрэх

Exceptions

- Хайхрахгүй байж болохгүй алдааны талаар програмын бусад хэсэгт мэдэгдэх
  - Алдааг мэдээллэх чадвар сайн
    - Хэрэгсэхгүй байж болохгүй
- Онцгой тохиолдолыг дамжуулахгүй байх
  - Алдааг дотооддоо шийдэх боломжтой бол дотооддооо шидэх
  - Гадагшаа ОТ болгож шийдэх шаардлага байхгүй
- Байгуулагч болон устгагч функц дотор ОТ шидэхээс зайлсхийх
  - Байгуулах явцад ОТ шидвэл устгагч функц дуудагдахгүй (Санах ойн сул ашиглалт)

Exceptions

- Хийсвэрлэлийн зөв түвшинд ОТ шидэх
  - Функц ОТ шиднэ гэдэг нь тухайн функцын интерфэйсийн нэг хэсэг
  - Доод түвшний ОТ EOFException
  - Encapsulation алдагдаж байна

```
class Employee {
    ...
    public TaxId GetTaxId() throws EOFException {
        ...
    }
    ...
}
```

Exceptions

- ОТ-ын мэссэж нь ОТ-ын бүхий л мэдээллийг агуулсан байх
  - ОТ бүр тодорхой нөхцөл байдалд үүснэ
  - Энэ мэдээлэл нь ОТ уншиж буй хүнд үнэлж баршгүй ач тустай
  - Мэссэж нь яагаад ОТ үүссэн гэдгийг ойлгохуйц байна
    - Array index error
      - Дээшээ эсвэл доошоо, буруу индэксэд хандсан

Exceptions

- Хоосон catch блок ашиглахгүй байх
  - Хийсвэрлэлийн түвшинд ОТ дүрсэлж чадахгүй бол
    - Лог бүртгэх хэрэгтэй

Good Java Example of Ignoring an Exception

```
try {
    ...
    // lots of code
   ...
} catch ( AnException exception ) {
    LogError( "Unexpected exception" );
}
```