CSE 143

Error Handling

Error Handling

Need good documentation about module behavior and expectations

- Helps with understanding program
- Helps with debugging and maintenance

Assertions

- An **assertion** describes an assumption about the current state of a program, typically:
  - At function entry and exit
  - At the beginning of each loop iteration
- Assertions may appear as
  - Comments
  - Executable statements
- Help to reason about program and provide error-checking at runtime

Preconditions and Postconditions

- Precondition: A condition assumed true before executing some operation (often a function call)
- Postcondition: A condition guaranteed true when an operation terminates
- Example: sqrt(double x) function
  - precondition: `x >= 0`
  - postcondition: `returns r such that r*r is equal to x`

Checking Preconditions

- Example: Average a list of numbers

```
double Average(int nums[], int len);
// PRE: len > 0
// POST: Returns average of
//       nums[0]..nums[len-1]
```

- What happens if `len <= 0`?

Option: Assume input OK

```
// PRE: len > 0
// POST: Returns average of
//       nums[0]..nums[len-1]
double Average(int nums[], int len)
{
  int sum = 0;
  for (int j = 0; j < len; j++)
    sum = sum + nums[j];
  return ((double) sum / (double) len);
}
```

Option: Return “Safe” Value

- If input bad, return a “safe” value
- Actually modifying spec to include bad inputs

```
double Average(int nums[], int len)
{
   if (len <= 0) return 0; // 0 is “safe”

   int sum = 0;
   for (int j = 0; j < len; j++)
      sum = sum + nums[j];
   return ((double) sum / (double) len);
}
```


Option: Check and Exit

```
#include <stdlib.h>
#include <iostream.h>

double Average(int nums[], int len)
{
  if (len <= 0) {
     cerr << “Average: len <= 0” << endl;
     exit(1);  // exit with error status
  }

  int sum = 0;
  for (int j=0; j<len; j++)
    sum = sum + nums[j];
  return ((double) sum / (double) len);
}
```


Option: Status Flag

```
double Average(int nums[], int len, bool &error)
{
 if (len <= 0) {
    error = true;
    return 0;
 }
 error = false;
 int sum = 0;
 for (int j = 0; j < len; j++)
    sum = sum + nums[j];
 return ((double) sum / (double) len)
}
```

- Client must test for an error after a call

Option: `assert` macro

```
#include <assert.h>

double Average(int nums[], int len)
{
  assert(len > 0);
  int sum = 0;
  for (int j = 0; j < len; j++)
     sum = sum + nums[j];
    return ((double) sum / (double) len);
}
```

- If an error occurs, program exits, printing:

```
Assertion failed:  len > 0
file main.cpp, line 23
```


Option: Exceptions

- **Exceptions** allow an implementation to signal errors to clients
  - Separate error detection from handling
  - Client doesn’t check for errors at each call, but writes a function that handles the error
  - Module determines when an exception should be **raised** (error in input, resource exhaustion, etc.)
- Exceptions are found in many other programming languages
  - heavily used in Java
- Probably won’t be covered in 143

Error Handling Summary

- Assertions are useful for making explicit any assumptions made in code
  - Assertions can be in comments or code
  - Use `assert` macro from `#include <assert.h>` for explicit checking
  - Use comments in `.h` file to help clients do the right thing
- Explicit error-checking can be cumbersome
  - But if you leave it out, you have to trust callers...

Use *assert*() to aid debugging

- Use *assert* liberally in the programming projects
  - Test preconditions especially, in as much detail as practical
  - Test invariants and postconditions when reasonable
- Don't worry too much about the overhead
  - Think of your programs as still in debug mode, even when turned in.
  - It is possible to disable assertion checking in "production" code.

Assert vs. Error Checking

- Use *asserts* to catch programming errors
- Asserts are for fatal errors
- Use explicit error checking to catch bad inputs from user.
- User input should never cause an assert to fail.
- The program should always recover from bad input (even if to exit)