Defensive Programming

Introduction to Defensive Programming

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Defensive Programming

An approach to improve software and source code, in terms of:

- Reducing the number of software bugs and problems
- Making the source code comprehensible
  - Easy to be approved in a code audit
  - Easy to be reused
- Making the software behave in a predictable manner despite unexpected inputs or user actions

Reducing Software Bug

- Testing at different levels
- Achieving good code coverage
- Test automation

Will be covered in detail after the mid-semester examination

Writing Comprehensible Code

- Use meaningful variable names
- Write high-quality routines
- Write comments wherever meaningful
- ...

Example: Poor and Good Variable Names

Poor variable names

```
x = x - xx;
xxx = fido + SalesTax( fido );
x = x + LateFee( x1, x ) + xxx;
x = x + Interest( x1, x );
```

Good variable names

```
balance = balance - lastPayment;
monthlyTotal = newPurchases + SalesTax( newPurchases );
balance = balance + LateFee( customerID, balance ) + monthlyTotal;
balance = balance + Interest( customerID, balance );
```

Example: Low Quality Routine

Low Quality Routine

```
void HandleStuff( CORP_DATA & inputRec, int crntQtr, EMP_DATA empRec,
double & estimRevenue, double ytdRevenue, int screenX, int screenY,
COLOR_TYPE & newColor, COLOR_TYPE & prevColor, StatusType & status,
int expenseType )
{
  int i;
  for ( i = 0; i < 100; i++ ) {
    inputRec.revenue[i] = 0;
    inputRec.expense[i] = corpExpense[ crntQtr ][ i ];
    }
  UpdateCorpDatabase( empRec );
  estimRevenue = ytdRevenue * 4.0 / (double) crntQtr;
  newColor = prevColor;
  status = SUCCESS;
  if ( expenseType == 1 ) {
    for ( i = 0; i < 12; i++ )
      profit[i] = revenue[i] - expense.type1[i];
      }
  else if ( expenseType == 2 ) {
    profit[i] = revenue[i] - expense.type2[i];
    }
  else if ( expenseType == 3 )
    profit[i] = revenue[i] - expense.type3[i];
    }
```

Example: Low Quality Routine

- Bad name
- Not documented
- Bad layout
- Input variable is changed
- Uses global variables
- Doesn't have a single purpose
- Does not defend against bad data
- Uses magic numbers
- Too many parameters
- Poor use of parameters

Defensive Programming

Introduction to Defensive Programming

Indranil Saha

Department of Computer Science and Engineering
Indian Institute of Technology Kanpur