CS 6015: Software Engineering

Spring 2024

Lecture 12: Power of Variables

This Week

- Parsing cont. (Project related)
- MSDScript project overview
- Power of variables
- Libraries (Lab 5)

Next Week

- Test automation
- Design patterns

Considerations in code

```
double temp;
. . . . . .

// Compute roots of a quadratic equation.
temp = Sqrt( b*b – 4*a*c );
root[0] = ( –b + temp ) / ( 2 * a );
root[1] = ( –b – temp ) / ( 2 * a );

// swap the roots
temp = root[0];
root[0] = root[1];
root[1] = temp;
```

Considerations in code: Improvement

```
double newRoot;
. . . . . .

// Compute roots of a quadratic equation.
discriminant = Sqrt( b*b - 4*a*c );
root[0] = ( -b + discriminant ) / ( 2 * a );
root[1] = ( -b - discriminant ) / ( 2 * a );

// swap the roots
oldRoot = root[0];
root[0] = root[1];
root[1] = oldRoot;
```

Avoid Magic numbers in your code

```
for ( i = 0; i < 100; i++ ) {
    . . .
     }
```

```
const int MAX_ENTRIES=100;
for ( i = 0; i < MAX_ENTRIES; i++ ) {
    . . .
     }
```

Considerations in variable names

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

Considerations in variable names

```
balance = balance – lastPayment;

monthlyTotal = newPurchases + SalesTax( newPurchases );

balance = balance + LateFee( customerID, balance ) + monthlyTotal;

balance = balance + Interest( customerID, balance );
```

Variable names consideration

Purpose of Variable	Good Names, Good Descriptors	Bad Names, Poor Descriptors
Running total of checks written to date		
Velocity of a bullet train		
Current date		
Lines per page		

Variable names consideration

Purpose of Variable	Good Names, Good Descriptors	Bad Names, Poor Descriptors
Running total of checks written to date	runningTotal, checkTotal	written, ct, checks, CHKTTL, x, x1, x2
Velocity of a bullet train	velocity, trainVelocity, velocityInMph	velt, v, tv, x, x1, x2, train
Current date	currentDate, todaysDate	cd, current, c, x, x1, x2, date
Lines per page	linesPerPage	pp, lines, l, x, x1, x2

Variable length consideration

Length	Example
Too long	
Too short	
Convenient	

Variable length consideration

Length	Example
Too long	numberOfPeopleOnTheUsOlympicTeam numberOfSeatsInTheStadium maximumNumberOfPointsInModernOlympics
Too short	n, np, ntm n, ns, nsisd m, mp, max, points
Convenient	numTeamMembers, teamMemberCount, numSeatsInStadium, seatCount teamPointsMax, pointsRecord

Computed-Value Qualifiers in Variable Name

Qualifiers

- `Total, Sum, Average, Max, Min, Record, String, or Pointer`

Adding modifiers

- `revenue`**`Total,`** `expense`**`Total,`** `revenue`**`Average,`** and `expense`**`Average`**

Common opposites in Variable names

Common opposites in Variable names

- `begin/end`
- `first/last`
- `locked/unlocked`
- `min/max`
- `next/previous`
- `old/new`
- `opened/closed`

Naming loop indices

```
for ( i = 0; i < n; i++ ) {
    for ( j = 0; j < m; j++ ) {
        score[ i ][ j ] = 0;
    }
}
```

```
for ( teamIndex = 0; teamIndex < teamCount; teamIndex++ ) {
    for ( eventIndex = 0; eventIndex < eventCount; eventIndex++ ) {
        score[ teamIndex ][ eventIndex ] = 0;
    }
}
```

Naming Boolean Variables

Naming Boolean Variables

- `Done`
- `Found / isFound`
- `Success`
- `Ok`
- `Error`

Use positive Boolean variable names

Naming Constants

Bad names

- `FIVE`
- `THREE_POINT_ONE_FOUR`

Better names

- `CYCLES_NEEDED`
- `PI`

Naming conventions

Entity	Description
ClassName	Class names are in mixed uppercase and lowercase with an initial capital letter.
TypeName	Type definitions, including enumerated types and typedefs, use mixed uppercase and lowercase with an initial capital letter
localVariable	Local variables are in mixed uppercase and lowercase with an initial lowercase letter.
routineParameter	Routine parameters are formatted the same as local variables.
RoutineName()	
m_ClassVariable	Member variables that are available to multiple routines within a class
g_GlobalVariable	Global variables are prefixed with a g_.
CONSTANT	Named constants are in ALL_CAPS.

Kinds of Names to avoid

- Avoid names with similar meanings
- Avoid variables with different meanings but similar
- Avoid numerals in names
- Avoid the names of standard types, variables, and routines
- Avoid names containing hard-to-read characters

C++: namespaces

- Spaces/regions in a program
- Necessary if you want the two components with the same name.