

# What Does It Take to Be a Good Software Developer?

Software developers are, above all,  
*systematic* problem solvers

a “logical mind”

a willingness to approach problems methodically

break problems up into sub-problems

don't panic at the size of a problem;  
it may seem overwhelming but it is  
within your grasp to solve it *if*

you approach it methodically

# The Software Development Method

1. *Problem specification*
2. *Analysis*
3. *Design*
4. *Test plan*
5. *Implementation or coding*
6. *Testing*

# The Software Development Method

1. *Problem specification:* State the problem and gain a clear understanding of what is required for its solution. This sounds easy, but it can be the most critical part of problem solving. A good problem solver must be able to recognize and define the problem precisely. If the problem is not totally defined, you must study the problem carefully, eliminating the aspects that are unimportant and zeroing in on the root problem.
2. *Analysis:* Identify problem inputs, desired outputs, and any additional requirements or constraints on the solution. Identify what information is supplied as problem data and what results should be computed and displayed. Also, determine the required form and units in which the results should be displayed (for example, as a table with specific column headings).

3. *Design*: Develop a list of steps (called an *algorithm*) to solve the problem and verify that the algorithm solves the problem as intended. Writing the algorithm is often the most difficult part of the problem-solving process. Once you have the algorithm, you should verify that it is correct before proceeding further.
  
4. *Test plan*: Develop a strategy for proving to yourself and to others that your algorithm will get the proper results. It is highly advisable to write a plan for testing the program you will write, even before you have written it. Which test cases will you choose? What are the special cases that must be tested? Pretend you are a potential purchaser of the program and ask, “Which tests would I require to be convinced that this program behaves as advertised?”

5. *Implementation or coding*: Implement the algorithm as a program. This requires knowledge of a particular programming language. Each algorithm step must be converted into a statement in that programming language.
6. *Testing*: Run the completed program, testing it with the test cases specified in the test plan.

## Example: Average Speed on Trip

### *Problem Specification*

Next week, you're driving to Long Island to visit your family. When you drive, you'd like to know what your average end-to-end speed, in miles per hour, was for the trip.

You're not very good at arithmetic. Luckily, you'll have your trusty laptop with you, so you can develop a little program that will compute your average speed from the distance you've driven and the number of hours the trip took.

Unfortunately, your laptop is a bit lame-brained, so it's able to caculate only with "whole numbers" (integers).

## *Analysis*

To solve this problem, we must be given the number of miles (to the nearest mile) and the number of hours (to the nearest hour). The average speed (to the nearest MPH) will be the quotient of the number of miles divided by the number of hours.

## *Data Requirements and Formulas*

### *Problem Inputs:*

miles - the number of miles, an integer

hours - the number of hours, an integer

### *Problem Outputs:*

mph - the average speed, to the nearest integer value

### *Relevant Formulas*

$\text{mph} = \text{miles} \div \text{hours}$

## *Initial Algorithm*

1. Prompt the user and read in the number of miles.
2. Prompt the user and read in the number of hours.
3. Compute the average speed.
4. Display the average speed.

## *Algorithm Refinements*

This is a very simple algorithm and needs no refinements.

## *Test Plan*

Let's develop one right here!

## *Implementation*

*Start with the template - it's unnecessary to start with a blank page*

```
csstud@hobbes:3: cp programs49/ProjectTemplate.c .
csstud@hobbes:4: cat ProjectTemplate.c
/*-----
| ProjectTemplate.c
| You can use this as a starting point for your projects.
| Author: <your name>, The George Washington University
| Last Modified: January 2, 2006
|-----*/
#include <stdio.h>
int main()
{
    /* Put your own program statements here! */
    return 0; /* indicate that program ended successfully */
}
csstud@hobbes:5:
```

*Declare the variables and copy the main algorithm steps as comments. This is a framework file.*

```
/*-----  
| AverageSpeed.c  
| Finds average speed on a trip  
| Author: M.B. Feldman, The George Washington University  
| Last Modified: January 10, 2006  
|-----*/  
#include <stdio.h>  
int main()  
{  
  
    int miles;          /* input   - number of miles driven */  
    int hours;         /* input   - number of hours driven */  
    int mph;           /* output - miles per hour          */  
  
    /* Prompt the user and read in the number of miles */  
    /* Prompt the user and read in the number of hours */  
    /* Compute the average speed                        */  
    /* Display the average speed                       */  
  
    return 0;  
  
}
```

*Compile the framework; fix any errors. Then fill in the rest of the program statements.*

```
/*-----  
| AverageSpeed.c  
| Finds average speed on a trip  
| Author: M.B. Feldman, The George Washington University  
| Last Modified: January 10, 2006  
|-----*/  
#include <stdio.h>  
int main()  
{  
  
    int miles;           /* input - number of miles driven */  
    int hours;          /* input - number of hours driven */  
    int mph;            /* output - miles per hour */  
  
}
```

```
/* Prompt the user and read in the number of miles */
printf
    ("How many miles did you drive (to the nearest mile)? ");
scanf("%d", &miles);

/* Prompt the user and read in the number of hours */
printf
    ("How many hours did you drive (to the nearest hour)? ");
scanf("%d", &hours);

/* Compute the average speed */
mph = miles / hours;

/* Display the average speed */
printf
    ("Your average speed was %d miles per hour\n", mph);

return 0;

}
```

## CSci 53, Objectives for Lecture 3

- Quiz: In reference to software development, what is a "framework file"?
- Another software development example: average age of professors

## Example: Average Age of Professors

### *Problem Specification*

You're a very curious person. One thing you're curious about is the ages of three of your professors. But how can you find out how old they are? Some people are offended if you ask their age.

Luckily, the GW Undergraduate Bulletin has a list of (most of) the professors in the back of the book. Each professor's listing gives his/her degrees, showing the college or university and year of each degree.

If we assume the typical student is 21 when (s)he earns a bachelors degree, we can estimate a professor's age just by knowing the current calendar year and that professor's graduation year.

You're also curious about the mean (average) age of the profs.

## *Analysis*

To solve this problem, we must be given the current year, and the graduation years of the three profs. The estimated age (to the nearest year) of a prof can be computed from these values. The mean age will be the sum of the ages divided by the number of profs (3).

## *Data Requirements and Formulas*

### *Problem Inputs:*

thisYear - the current calendar year, an integer

year - the graduation year of each prof in turn, an integer

### *Problem Outputs:*

age1, age2, age3 - the ages of the three professors (integers)

meanAge - the mean age of the professors

### *Relevant Formulas*

Estimated age = current year - graduation year + 21

## *Initial Algorithm*

1. Prompt the user and read in the current year.
2. Prompt the user and read in each professor's graduation year, then calculate and display that professor's estimated age.
3. Compute the mean age.
4. Display the mean age.

## *Algorithm Refinements*

Step 2 can be refined:

- 2.1 Prompt the user and read in 1st professor's graduation year, then calculate and display that professor's estimated age.
- 2.2 Prompt the user and read in 2nd professor's graduation year, then calculate and display that professor's estimated age.
- 2.3 Prompt the user and read in 3rd professor's graduation year, then calculate and display that professor's estimated age.

## *Test Plan*

Let's develop one right here!

## *Implementation*

*Start with the template - it's unnecessary to start with a blank page*

```
csstud@hobbes:3: cp programs49/ProjectTemplate.c .
csstud@hobbes:4: cat ProjectTemplate.c
/*-----
| ProjectTemplate.c
| You can use this as a starting point for your projects.
| Author: <your name>, The George Washington University
| Last Modified: January 2, 2006
|-----*/
#include <stdio.h>
int main()
{
    /* Put your own program statements here! */
    return 0; /* indicate that program ended successfully */
}
csstud@hobbes:5:
```

*Declare the variables and copy the main algorithm steps as comments. That is, make a framework file.*

```
/*-----  
| MeanAge.c  
| Finds mean estimated age of three professors  
| Author: M.B. Feldman, The George Washington University  
| Last Modified: January 10, 2006  
-----*/  
#include <stdio.h>  
int main()  
{  
  
    int thisYear;    /* input - the current calendar year        */  
    int year;        /* input - year of a prof's bachelors deg.*/  
    int age1;        /* output - 1st prof's estimated age    */  
    int age2;        /* output - 1st prof's estimated age    */  
    int age3;        /* output - 1st prof's estimated age    */  
    int meanAge;     /* output - mean of 3 ages              */  
}
```

```
/* Prompt the user and read in the current calendar year */
/* Prompt the user and read in 1st prof's degree year */
/* and compute and display that prof's estimated age */
/* Prompt the user and read in 2nd prof's degree year */
/* and compute and display that prof's estimated age */
/* Prompt the user and read in 3rd prof's degree year */
/* and compute and display that prof's estimated age */
/* Compute the mean age */
/* Display the mean age */
return 0;
}
```

*Compile the framework; fix any errors. Then fill in the rest of the program statements.*

```
/*-----  
| MeanAge.c  
| Finds mean estimated age of three professors  
| Author: M.B. Feldman, The George Washington University  
| Last Modified: January 10, 2006  
-----*/  
#include <stdio.h>  
int main()  
{  
  
    int thisYear;    /* input - the current calendar year        */  
    int year;        /* input - year of a prof's bachelors deg.*/  
    int age1;        /* output - 1st prof's estimated age    */  
    int age2;        /* output - 2nd prof's estimated age    */  
    int age3;        /* output - 3rd prof's estimated age    */  
    int meanAge;     /* output - mean of 3 ages              */  
}
```

```

/* Prompt the user and read in the current calendar year */
printf ("What is the present calendar year (e.g. 2002)? ");
scanf ("%d", &thisYear);

/* Prompt the user and read in 1st prof's degree year      */
/* and compute and display that prof's estimated age      */
printf ("Enter year of 1st professor's bachelors degree >");
scanf ("%d", &year);
age1 = (thisYear - year) + 21;
printf ("That prof is about %d years old.\n", age1);

/* Prompt the user and read in 2nd prof's degree year      */
/* and compute and display that prof's estimated age      */
printf ("Enter year of 2nd professor's bachelors degree >");
scanf ("%d", &year);
age2 = (thisYear - year) + 21;
printf ("That prof is about %d years old.\n", age2);

/* Prompt the user and read in 3rd prof's degree year      */
/* and compute and display that prof's estimated age      */
printf ("Enter year of 3rd professor's bachelors degree >");
scanf ("%d", &year);
age3 = (thisYear - year) + 21;
printf ("That prof is about %d years old.\n", age3);

```

```
/* Compute the mean age */
meanAge = (age1 + age2 + age3) / 3;

/* Display the mean age */
printf
    ("The average age of the 3 profs is %d years.\n", meanAge);

return 0;

}
```