Writing a Program

Introduction

This is an example of how to write a program. The method is called *top-down programming* because you start with the goal (the top), and break it down into a series of steps (computations) that will achieve that goal.

Step 1: Goal and General Algorithm Idea

The goal is a statement of what the program is to do.

Goal: write a program to determine how many quarters, dimes, nickels, and pennies make up a given amount of change

Next, we say what the inputs and outputs of the program are:

Inputs: User enters an amount as an integer

Outputs: Program prints number of quarters, dimes, nickels, and pennies that make up the given amount

Finally, we give a high-level approach for doing all this:

High-level design:
1. read in amount
2. figure out how may quarters are in the amount
3. determine how much is left over from this
4. figure out how many dimes are in what's left over
5. determine how much is left over from this
6. figure out how many nickels are in what's left over
7. what's left is the number of pennies

Step 2: Data Representation and Program Structure

First, figure out how we want to represent the data on the computer:

Data Representation: Represent the amount as an integer

Now work out the structure of the program; in what follows, note how the steps in the structure correlate to the high-level design:

Program Structure:
1. Read in the input and save it
2. Divide by 25 to get the number of quarters
3. Get the remainder
4. Divide by 10 to get the number of dimes
5. Get the remainder
6. Divide by 5 to get the number of nickels
7. Get the remainder
8. Print the original input, number of quarters, dimes, nickels, and pennies

Now, we refine the structure further, and continue to do so until we are comfortable that there is enough *explicit* detail to allow us to write the program from the refinement:

Refine Algorithm:
1. ask the user for the amount, read it and store it in integer variables `amount` and `leftover`
2. divide `leftover` by 25 to get the number of quarters `numquarters`
3. take the remainder of `leftover` when divided by 25 to get the new integer `leftover`
4. divide `leftover` by 10 to get the number of dimes `numdimes`
5. take the remainder of `leftover` when divided by 10 to get the new integer `leftover`
6. divide `leftover` by 5 to get the number of nickels `numnickels`
7. take the remainder of `leftover` when divided by 5 to get the new integer `leftover`
8. this is the number of pennies `numpennies`

9. print `amount` "cents is" `numquarters` "quarters," `numdimes` "dimes," `numnickels` "nickels, and" `numpennies` "pennies"

There is one very important item missing — the error checking. We will add that near the end, as it requires an **`if`** statement, and we have not covered those yet.

Translate This Into C Statements

```
scanf("%d", &amount); leftover = amount;
numquarters = leftover / 25;
leftover = leftover % 25;
numdimes = leftover / 10;
leftover = leftover % 10;
numnickels = leftover / 5;
leftover = leftover % 5;
numpennies = leftover;
printf("%d cents is %d quarters, %d dimes, %d nickels, and %d pennies\n",
                        amount, numquarters, numdimes, numnickels, numpennies);
```

Turn This Into a Program

I omitted the header comments. You can look on Canvas for the real program and see the comments.

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

int main(void)
{
        int amount;        /* how much change the user has */
        int leftover;      /* used to hold amount as it decreases */
        int numquarters;   /* how many quarters */
        int numdimes;      /* how many dimes */
        int numnickels;    /* how many nickels */
        int numpennies;    /* how many pennies */

        /* read in the amount and save it */
        printf("Enter amount of change: ");
        scanf("%d", &amount); leftover = amount;

        /* computer the number of quarters and how much is left over */
        numquarters = leftover / 25;
        leftover = leftover % 25;

        /* computer the number of dimes and how much is left over */
        numdimes = leftover / 10;
        leftover = leftover % 10;

        /* computer the number of nickels and how much is left over */
        numnickels = leftover / 5;
        leftover = leftover % 5;

        /* whatever is left over is the number of pennies */
        numpennies = leftover;

        /* print the result */
        printf("%d cents is %d quarters, %d dimes, %d nickels, and %d pennies\n",
                                amount, numquarters, numdimes, numnickels, numpennies);
```

```
        /* thank you, and goodnight! */
        return(EXIT_SUCCESS);
}
```

Add Error Checking

What happens if the user types "a dollar" to the input? That's not a number. So, what happens is *scanf* doesn't find an integer, and so it leaves the value of `amount` alone. When I ran the program, it happened to contain 4195472, so my output was:

```
4195472 cents is 167818 quarters, 2 dimes, 0 nickels, and 2 pennies
```

The problem is that `amount` could have contained anything; it was not set to an initial value. So we need to know when *scanf* fails.

That function returns the number of items read, so here *scanf* should return 1. If it returns anything else, an error occurred. So let's use an **if** statement to check for that. Change the line containing *scanf* to:

```
if (scanf("%d", &amount) != 1){
        /* oops ... bad input! */
        printf("Amount must be nonnegative integer\n");
        return(EXIT_FAILURE);
}
leftover = amount;
```

Now when I typed the same input, I got:

```
Amount must be nonnegative integer
```

Error checking is an integral part of any program. Normally, figure out how to do it when you write a statement that might cause a problem (here, the *scanf* line). In fact, we haven't done a thorough job of error checking (for example, what happens if the user enters a negative integer or a floating point number), but this is a good start.

User Interface

Also, we need to tell the user what to enter, because what is required may not be clear from the name of the program. So, above the *scanf* line, add this:

```
/* prompt the user for input */
printf("Amount of change to convert to coins: ");
```

Finally, we should change the output to handle units of 1 coin (make the coin name singular) and 0 coins (omit the number of coins and coin name entirely), but that's a good homework or extra credit question!