Computer Programming: Skills & Concepts (CP1)
Case Study: Coin change

18th October, 2010

Class rep Election

Volunteers?

Coin Change

We want to write a program that

- ask the user for an amount of money
- calculates the coins needed for this amount
- outputs the number of each coin

Type of Coins

Coins range from 1p to £2

```
const int
  C1 = 200, C2 = 100, C3 = 50, C4 = 20,
  C5 = 10,  C6 = 5,   C7 = 2,  C8 = 1;
```

Three functions

```
if ( ReadInput(&amount) != EXIT_SUCCESS) {
  printf("Failure in ReadInput\n");
  return EXIT_FAILURE;
}
if ( CalculateCoins(amount) != EXIT_SUCCESS) {
  printf("Failure in CalculateCoins\n");
  return EXIT_FAILURE;
}
if ( PrintResult(amount) != EXIT_SUCCESS) {
  printf("Failure in PrintResult\n");
  return EXIT_FAILURE;
}
```

Function structure of Program

```
const int C1 = 200, C2 = 100, C3 = 50, C4 =20,
      C5 = 10 C6 = 5, C7 = 2, C8 = 1;

int n1, n2,n3, n4, n5, n6, n7, n8;

int ReadInput(int* amount)  {

        "Take input from user"

}

int CalculateCoins(int amount) {

        "assign n1, ..., n8 approp."

}

int PrintResult(int amount) {

        "Print no. of each coins"

}

int main(void) {

 .....
 .....
}
```

Take Input from User

```
int input = 0 ;
printf("Enter the amount (in pence) to be returned
        to the user: ");
scanf("%d", &input);
```

Take Input from User (Error Tolerant)

```
int input = 0 ;

do {
  printf("Enter the amount (in pence) to be returned
          to the user: ");
  while (scanf("%d", &input) !=1) {
    scanf("%*s");
    printf("That wasn't a number - please try again: ");
  }

} while (input < 0 );
```

Take Input from User (Full Function)

```
int ReadInput(int *amount) {
  int input = 0 ;
  do {
    printf("Enter the amount (in pence) to be returned
            to the user: ");
    while (scanf("%d", &input) !=1) { scanf("%*s") ;
      printf("That wasn't a number - please try again: ");
    }
  } while (input < 0 ) ;
  *amount = input ;   // Set the value of amount to equal input
  return EXIT_SUCCESS ;
}
```

Take Input from User (Full Function)

```
int ReadInput(int *amount) {
  int input = 0 ;
  do {
    printf("Enter the amount (in pence) to be returned
            to the user: ");
    while (scanf("%d", &input) !=1) { scanf("%*s") ;
      printf("That wasn't a number - please try again: ");
    }
  } while (input < 0 ) ;
  *amount = input ;   // Set the value of amount to equal input
  return EXIT_SUCCESS ;
}
```

A "trick" is being used here. The fact that `input` is initialised to 0 is allowing us to check "success" by looking at the value of `input` (rather than testing the expression `scanf("%d", &input)` itself).

Take Input from User (Full Function)

```
int ReadInput(int *amount) {
  int input = 0 ;
  do {
    printf("Enter the amount (in pence) to be returned
            to the user: ");
    while (scanf("%d", &input) !=1) { scanf("%*s") ;
      printf("That wasn't a number - please try again: ");
    }
  } while (input < 0 ) ;
  *amount = input ;   // Set the value of amount to equal input
  return EXIT_SUCCESS ;
}
```

A "trick" is being used here. The fact that `input` is initialised to 0 is allowing us to check "success" by looking at the value of `input` (rather than testing the expression `scanf("%d", &input)` itself).

This would *not* work if 0 was to be an acceptable value for `input`.

Coin-changing: problem-solving

We make an *assumption*:

- Enough coins to change any value without running out.

Coin-changing: problem-solving

We make an *assumption*:

- Enough coins to change any value without running out.

We use a *Heuristic* (rule-of-thumb):

- Start with the largest coin possible.
  - Will need an `if` statement to compare values.

Coin-changing: problem-solving

We make an *assumption*:

- Enough coins to change any value without running out.

We use a *Heuristic* (rule-of-thumb):

- Start with the largest coin possible.
  - Will need an `if` statement to compare values.
- We do this *iteratively* (apply this rule many times).
  - Hence we will need a `for` or a `while`.

Calculate Coins

```
int pot = amount;  // Total value of coins so far selected.
while (pot > 0) {
  if (pot >= C1) {
    pot -= C1;  ++n1;
  } else if (pot >= C2) {
    pot -= C2;  ++n2;
  } else if (pot >= C3) {
    pot -= C3;  ++n3;
  } else if (pot >= C4) {
    pot -= C4;  ++n4;
  } else if (pot >= C5) {
    pot -= C5;  ++n5;
  } else if (pot >= C6) {
    pot -= C6;  ++n6;
  } else if (pot >= C7) {
    pot -= C7;  ++n7;
  } else {
    /*  pot >= C8.  (Why do we know this?) */
    pot -= C8;  ++n8;
  }
}
```

Catching Programming mistakes

```
assert(
  n1*C1 + n2*C2 + n3*C3 + n4*C4 + n5*C5 + n6*C6
  + n7*C7 + n8*C8 == pot && pot <= amount
  );
```

Catching Programming mistakes

```
assert(
  n1*C1 + n2*C2 + n3*C3 + n4*C4 + n5*C5 + n6*C6
  + n7*C7 + n8*C8 == pot && pot <= amount
  );
```

- Need to `#include` the `<assert.h>` header file
- The argument to `assert` must be a boolean condition
- If `assert(`*expression*`)` is false, the program stops with an error message.

Output to User

```
int PrintResult(int amount) {
 printf("%dp may be returned using the following
         combination of coins:\n", amount);
  if (n1) printf("%4d %dp coins,\n", n1, C1);
  if (n2) printf("%4d %dp coins,\n", n2, C2);
  if (n3) printf("%4d %dp coins,\n", n3, C3);
  if (n4) printf("%4d %dp coins,\n", n4, C4);
  if (n5) printf("%4d %dp coins,\n", n5, C5);
  if (n6) printf("%4d %dp coins,\n", n6, C6);
  if (n7) printf("%4d %dp coins,\n", n7, C7);
  if (n8) printf("%4d %dp coins.\n", n8, C8);
  return EXIT_SUCCESS;
}
```

Summary

Concepts revisited in this lecture:

- Functions
- `scanf` and error-checking
- *global* variables
- `if ... else` statement
- The `while` statement