Example: Exchange Kiosk

```c
/* Exchange kiosk */
#include <stdio.h>

void main()
{
    float dollars_for_mark; /* exchange rate */
    int commission; /* commission in dollars */
    float marks, /* how many marks */, dollars;
    dollars_for_mark = 0.666;
    commission = 3;
    marks = 100;
    dollars = marks * dollars_for_mark - commission;
    printf("For %6.2f marks you will get %6.2f dollars!
", marks, dollars);
}
```
What we can learn: Style

- Make program more readable
  - Indentation (use Tabs!)
  - Empty lines
  - Blanks inside expressions

- Make programs more understandable
  - Comments
  - Meaningful names for variables

Bad Code for Exchange Kiosk

```c
#include <stdio.h>

void main() { float n; int k; float m, r;
n = 0.666;
k = 3; m = 100; r = m * n - k;
printf("For %6.2f marks you will get %6.2f dollars!\n", m, r);
}
```
What We Can Learn: Programming

- Programming is problem solving
- Understand problem
- Design solution
- Implement solution (yes, coding!)
- Test/debug solution
- And often go over

The first step in solving any problem is understanding the problem. We call this specifying the problem.

You can think of a programming task as a word problem.

- What information is given? This is the starting point.
- What is the desired result?
- How do you get there?
Designing the Solution

- What information is needed?
- What steps need to be performed?

First design the solution, then implement the program

Remember, we should be able to design the solution without regard to the actual programming language.

Design for Exchange Kiosk

- Required data:
  - exchange rate, commission, how many DM

- Expected result:
  - how many USD

- Design:
  - Get data
  - Calculate USD
  - Print result
# include <stdio.h>

void main ( ) {
    float dollars_for_mark; /* exchange rate */
    int commission; /* commission in dollars */
    float marks; /* marks given */
    float dollars; /* dollars returned */

    /* get data */
    dollars_for_mark = 0.666;
    commission = 3;
    marks = 100;

    /* calculate USD */
    dollars = marks * dollars_for_mark - commission;

    /* print result */
    printf( "For %6.2f marks you will get %6.2f dollars!\n", marks, dollars );
}
Exchange Kiosk: Assignment

```c
void main()
{
    float dollars_for_mark; /* exchange rate */
    int commission; /* commission in dollars */
    float marks; /* marks given */
    float dollars; /* dollars returned */

    /* get data */
    dollars_for_mark = 0.666;
    commission = 3;
    marks = 100;

    /* calculate USD */
    dollars = marks * dollars_for_mark - commission;

    /* print result */
    printf("For %6.2f marks you will get %6.2f dollars!
", marks, dollars);
}
```

Exchange Kiosk: Initialization

```c
void main()
{
    float dollars_for_mark = 0.666; /* exchange rate */
    int commission = 3; /* commission in dollars */
    float marks = 100; /* marks given */
    float dollars; /* dollars returned */

    /* calculate USD */
    dollars = marks * dollars_for_mark - commission;

    /* print result */
    printf("For %6.2f marks you will get %6.2f dollars!
", marks, dollars);
}
```
Exchange Kiosk: # define

# define DOLLARs_FOR_MARK 0.666
# define COMMISSION 3
# define MARKS 100

void main(
{
float dollars; /* dollars returned */
/* calculate USD */
dollars = MARKS * DOLLARs_FOR_MARK - COMMISSION;
/* print result */
printf("For %6.2f marks you will get %6.2f dollars!\n"), MARKS, dollars);

Defined Constants

- # define is a preprocessor directive for defining constants
- Example: # define COMMISSION 3
  COMMISSION is a defined constant – Now every appearance of commission is literally replaced by 3 – It is not a variable, it cannot be assigned a value
Why to Use Defined Constants?

- Remember Temperature Converter?

```c
#include <stdio.h>

void main()
{
  printf( "100 Fahrenheit = %.2f Celsius \n", (5.0/9.0) * (100 - 32) );
}
```

- Need to replace 100 to 150 in two places (in real programs in dozens places!)

Example: Temperature Converter

```c
#include <stdio.h>

#define FAHRR 100.0 /* the temperature to be converted */

void main()
{
  printf( "%.2f Fahrenheit = %.2f Celsius \n", FAHRR, (5.0 / 9.0) * (FAHRR - 32) );
}
```
More About Preprocessor

- Preprocessor is a separate part of a compiler, which differs from the rest of it.
- Makes file insertions:

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

- System files:

```c
#include "myfile.h"
```

- Resolves definitions:

```c
#define NUM 100.0
```

- Removes comments:

```c
/* this is simply a comment */
```

Example after preprocessing...

```
... content of the file stdio.h ...
```
# include <stdio.h>
#define FAH 100.0 /* the temperature to be converted */

void main()
{
    printf("%6.2f Fahrenheit = %6.2f Celsius \n", FAH, (5.0 / 9.0) * (FAH - 32));
    FAH = 80;
    printf("%6.2f Fahrenheit = %6.2f Celsius \n", FAH, (5.0 / 9.0) * (FAH - 32));
}

... after preprocessing
... content of the file

void main()
{
    printf("%6.2f Fahrenheit = %6.2f Celsius \n", 100.0, (5.0 / 9.0) * (100.0 - 32));
    100.0 = 80;
    printf("%6.2f Fahrenheit = %6.2f Celsius \n", 100.0, (5.0 / 9.0) * (100.0 - 32));
}
# include <stdio.h>

void main ( ) {
    float fah;
    /* the temperature to be converted */
    /* data input */
    printf ( "Enter temperature in Fahrenheit: " ) ; /* prompt */
    scanf ( " %f " , &fah ) ; /* input */
    /* calculation and printing */
    printf ( "%6.2f Fahrenheit = %6.2f Celsius \n" , fah , ( 5.0 / 9.0 ) * ( fah - 32 ) ) ;
}
Getting Change Inside

- Defined constant
- Initialization of variables
- Assignment
- Input data

`scanf( "%d", &marks );`

- The computer will read the symbols from standard input, consider it as an integer (%d tells about it) and assign to the given variable marks.
- Treat & as a magic sign for now

More About Input

- Input specification is important!

`scanf( "%f", &fahr );`
`scanf( "%d", &marks );`

- The computer will read the symbols from standard input, considering it according to the given input specification (i.e., %f for %d) and assign to the given variable.
- Several values could be entered

`scanf( "%d %f", &marks, &fahr );`
Example: Better Exchange Kiosk

```c
#include <stdio.h>

#define DOLLARS_FOR_MARK 0.666
	/* exchange rate */

#define COMMISSION 3
	/* commission in dollars */

void main()
{
    int marks;
    float dollars;
    /* get data */
    printf("Marks to exchange? :"); /* prompt */
    scanf("%d", &marks); /* input */
    /* calculate USD */
    dollars = marks * DOLLARS_FOR_MARK - COMMISSION;
    /* print results */
    printf("For %d marks you will get %.6f dollars!
", marks, dollars);
}
```

Example: Capital Growth

```c
#include <stdio.h>

void main()
{
    float interest_rate; /* interest rate in percent */
    int capital; /* starting capital in dollars */
    float annual_interest; /* annual interest in dollars */
    printf("Capital (\$ \$) and interest rate (\% \% x x . x) : ");
    scanf("%d \%f", &capital, &interest_rate); /* input */
    annual_interest = capital * interest_rate / 100;
    printf("Interest %.6f; Total %.9f \n", annual_interest, capital + annual_interest);
}
```
A Simple Calculation Program

- Analyze the problem
  - What we need to calculate
  - What data do we need to calculate

- Define variables and constants
  - Declare target values as variables
  - Define known data as initialized variables or symbolic constants
  - Declare unknown data as variables and request the values from the user via input

- Calculate and print result

Before Next Lecture:

- Do reading assignment (quiz!)
  - Perry: Chapter 7; Chapter 8

- Run Classroom Examples

- Check yourself by working with Knowledge Tree and WAD in system

- Exercise: Do Celsius to Fahrenheit conversion

- Suggest a simple calculation problem and write a program with input to solve it