



Overview of C

Abdallah Karakra

Computer Science Department

Comp 132

Motivation

```
//C program for area of circle    Comment
#include <stdio.h> // standard header file (contains printf and scanf )
#define PI 3.141 //we use define for creating constant
int main()                // int, float , and return (reserved words)
{
    float r, a; // r, a are variables
    printf("Please enter the radius: "); //standard identifier
    scanf("%f", &r); //standard identifier
    a = PI * r * r; // = , *, {, } special symbols
    printf("%f\n", a); // standard identifier
    return 0;
}
```

preprocessor directives

- `#include`
 - gives a program access to a library
- `<stdio.h>`
 - standard header file
 - contains information about standard input and output functions such as `scanf` and `printf`

Preprocessor directives

- **#include <stdio.h>**
 - notify the preprocessor that some names used in the program are found in <stdio.h>
- **#define**
 - using only data values that never change should be given names

Preprocessor directives

- **Constant macro**

– a name that is replaced by a particular constant value

EX:

```
#define PI 3.141593
```

constant macro constant value

```
#define MAX_LENGTH 100
```

Comment

- Two types:
 - One-line comment `//`
 - Multiple-line comment `/* */`

Examples:

```
// This is a one-line comment
```

```
/* Hello, this is  
multiple-line comment*/
```

Variable declarations and data types

- Variable: a **name** associated with a **memory cell** whose value can change.

Examples:

sum , x , y , result,.....

Variable declarations and data types

1. A variable must consist only of **letters**, **digits**, and **underscores**.
2. A variable **cannot begin with a digit**.
3. A C reserved word cannot be used as a user variable.
4. A variable defined in a C **standard library should not be redefined**.

Reserved Words : A word that has special meaning in C
for example: int, float, double, char , return
,...etc

Variable declarations and data types

Syntax Display for Declarations:

Syntax :

- **int** *variable_list*;
- **float** *variable_list*;
- **double** *variable_list*;
- **char** *variable_list*;
- Examples :
 - **int** count, large;
 - **float** ans; or float ans=4.2;
 - **double** x, y, z; or double x=1.2,y=3.6,z=8.9;
 - **char** first_initial;

Variable declarations and data types

Data types:

- int (16 bit)
- float (32 bit)
- double (64 bit)

a real number has an **integral part** and a **fractional part** that are separated by a **decimal point**

Variable declarations and data types

Data types:

- char (8 bit)
 - represent an **individual character** value
 - include a **letter**, a **digit**, a **special symbol**
 - ex. **'A'** **'z'** **'2'** **'9'** **'*'** **':'** **'"** **'** **'**

Variable declarations and data types

Invalid variables names

Invalid identifier	Reason Invalid
1Letter	begins with a digit
double	reserved word
int	reserved word
TWO*FOUR	character * not allowed
joe's	character ' not allowed

To remove the ambiguity

Reserved Words	Standard Identifiers	User-Define Identifiers
int	printf	KMS_PER_MILE
void	scanf	miles
float		kms
double		sum
return		sum

NOTE: Sum, sum, SUM are viewed by the compiler as different identifiers

Placeholders in format strings

Placeholder	Variable Type	Function Use
%c	char	printf / scanf
%d	int	printf / scanf
%f	float	printf / scanf
%f	double	printf
%lf	double	scanf

Placeholders in format strings

```
int sum ;
```

```
float a, r ;
```

```
double num;
```

```
let sum=2
```

```
a=3.2 , r=5,2
```

```
num= 76.2232
```

- printf (“The area is %f, a);
- scanf(“ %f ”,&r);
- printf (“the result is %d”, sum);
- scanf (“%lf”,& num);
- printf (“the number is %f”, num)

Arithmetic expressions.

Arithmetic Operator	Meaning	Examples
+	addition	$5 + 2$ is 7
-	subtraction	$5 - 2$ is 3
*	multiplication	$5 * 2$ is 10
/	division	$5 / 2$ is 2
%	Remainder or Mod	$5 \% 2$ is 1

Arithmetic expressions.

Results of / and % operations

$$2 / 15 = 0$$

$$16 / 3 = 5$$

$$4 / 0 \text{ undefined}$$

$$2 \% 5 = 2$$

$$5 \% 4 = 1$$

$$15 \% 0 \text{ undefined}$$

$$\text{int} / \text{int} = \text{int}$$

$$12/3= 4 , \quad 9/8=1$$

$$\text{int}/\text{float} = \text{float} , \quad \text{float}/\text{int} = \text{float}$$

$$\text{float}/\text{float} = \text{float}$$

$$9/8.0=1.125000$$

$$9.0/8=1.125000$$

$$9.0/8.0=1.125000$$

Arithmetic expressions.

- Example:

```
double k,m;
```

```
k= 9/6;
```

```
m=9/6.0;
```

```
printf("k=%f \nm= %f", k,m);
```

Output:

k=1.000000

m=1.500000

Arithmetic expressions.

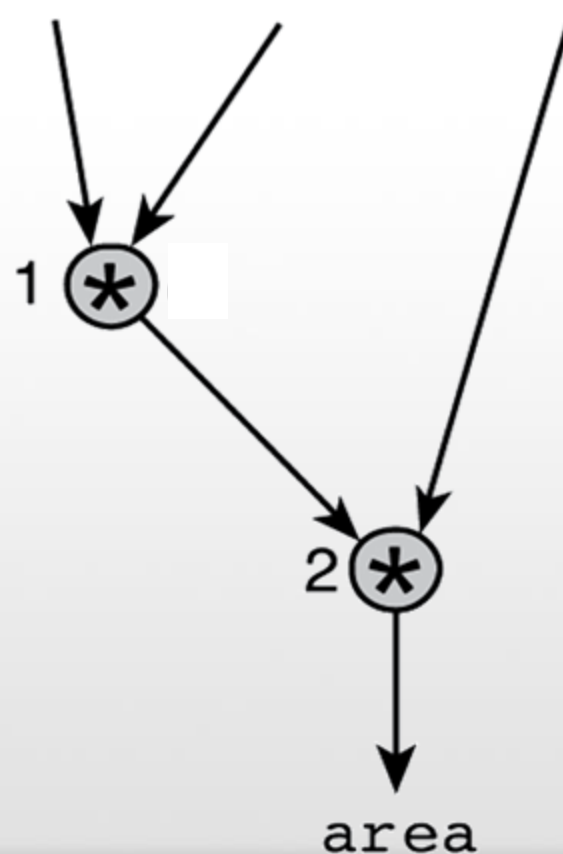
Precedence Rules:

- $()$
- $* / \%$
- $+ -$

Arithmetic expressions.

Example 1 : Evaluate $area = PI * radius * radius$

area = PI * radius * radius



Arithmetic expressions.

Example 1 : Evaluate $area = PI * radius * radius$

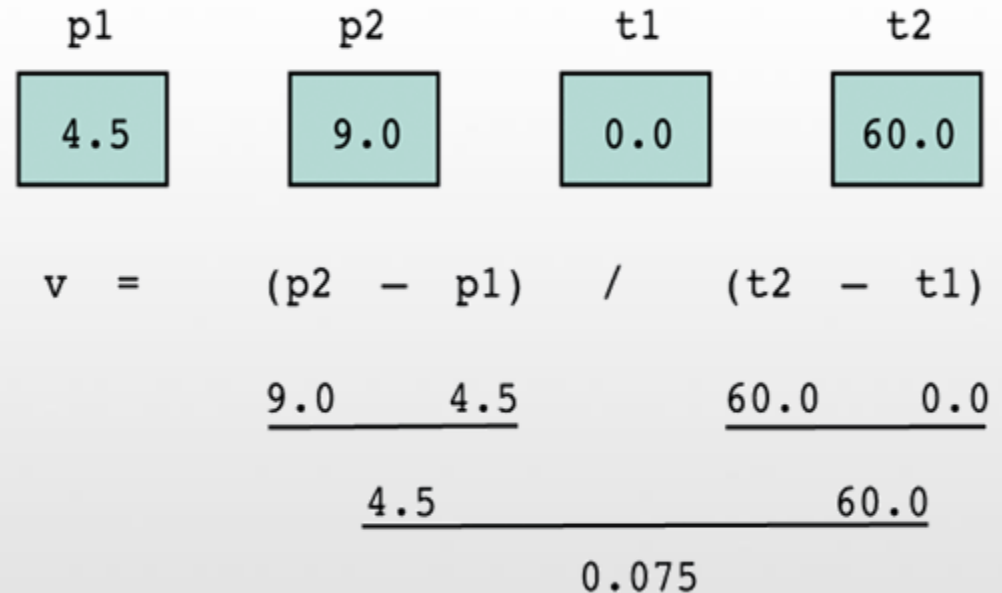
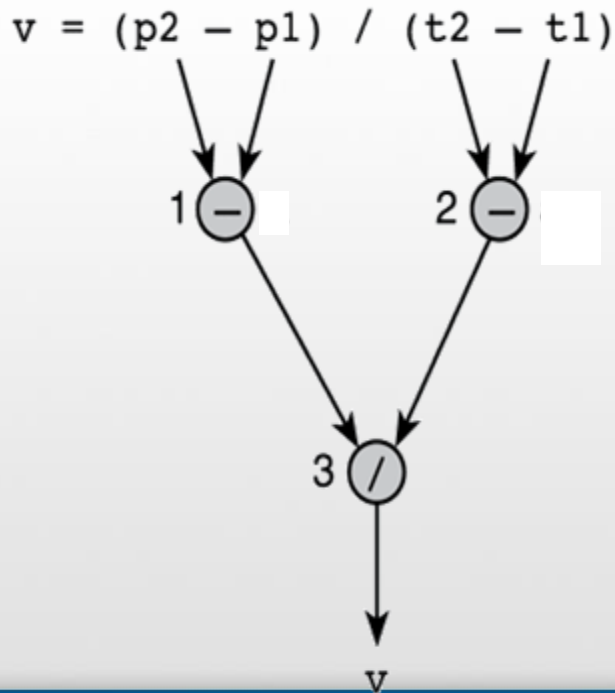
Let $PI = 3.14159$, $radius = 2.0$

$$\begin{array}{rcccccccc} area & = & & PI & * & radius & * & radius \\ & & & 3.14159 & & 2.0 & & 2.0 \\ & & & \hline & & & 6.28318 & & & & \\ & & & & & & & \hline & & & & & & & 12.56636 \end{array}$$

Arithmetic expressions.

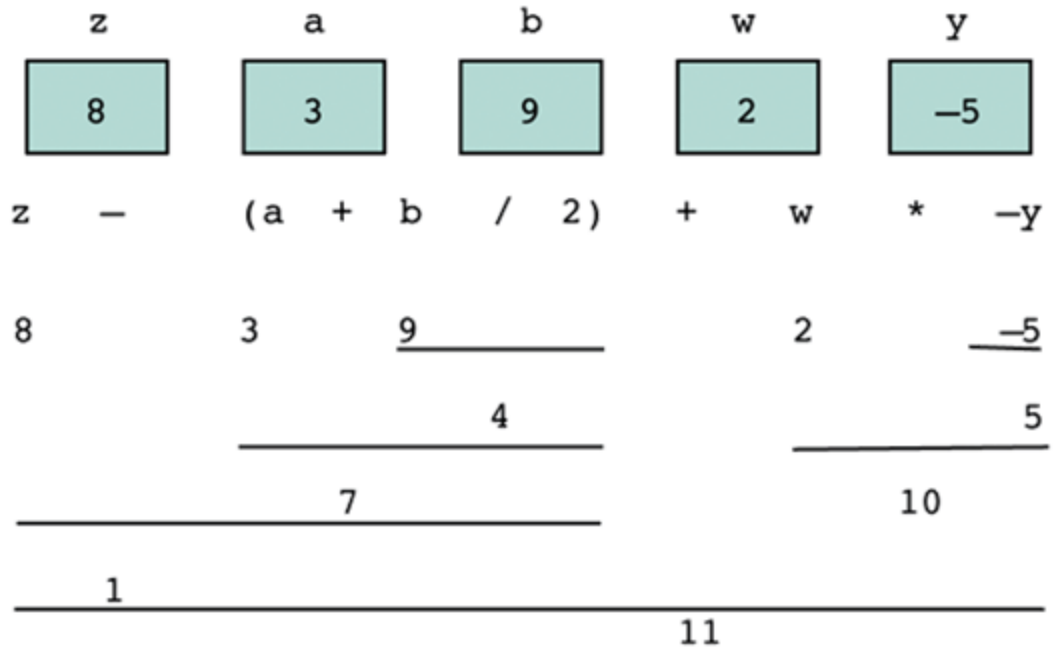
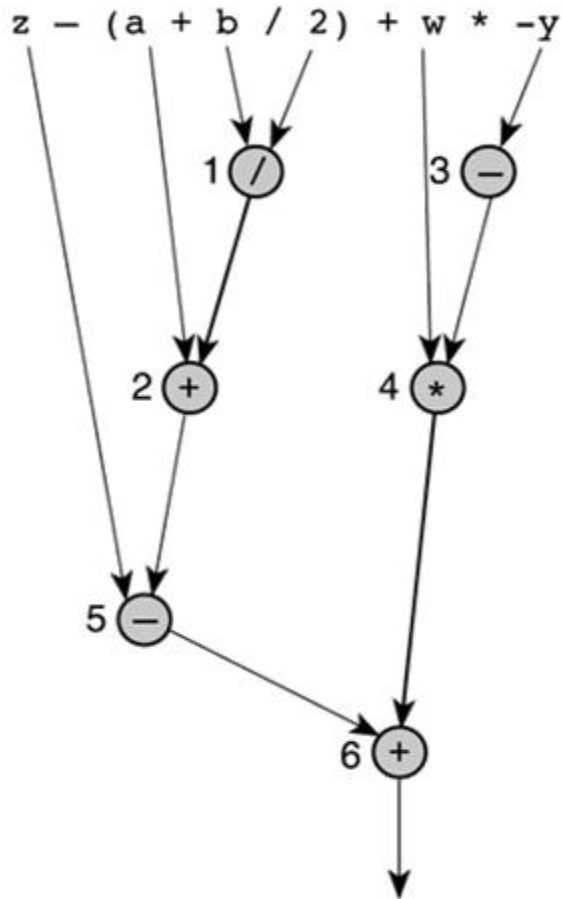
Example 1 : Evaluate $v = \frac{p2-p1}{t2-t1}$

let $P1=4.5$, $P2=9.0$, $t1=0.0$, $t2=60.0$



Arithmetic expressions.

Example 1 : Evaluate $z - (a + b / 2) + w * -y$



Arithmetic expressions.

Mathematical Formula as C Expression

Mathematical Formula	C Expression
$b^2 - 4ac$	<code>b * b - 4 * a * c</code>
$a + b - c$	<code>a + b - c</code>
$\frac{a+b}{c+d}$	<code>(a + b) / (c + d)</code>
$\frac{1}{1+x^2}$	<code>1 / (1 + x * x)</code>
$a \times -(b + c)$	<code>a * -(b + c)</code>

Arithmetic expressions.

Example:

Write a complete C program that prompts the user to enter the radius of a circle and displays the circumference. **Circumference=2 πr**

```
#include <stdio.h>
#define PI 3.14159
int main(void)
{
    double radius, circum;
    printf("Please enter radius of circle> ");
    scanf("%lf", &radius);
    circum = 2 * PI * radius;
    printf("The circumference is %.2f.\n", circum);
    return 0;
}
```

Formatting output

```
int x= 4678, y=3 , z=19
```

1. `printf (“%d %d %d”, x,y,z)`

Output

4678 3 19

2. `printf (“%7d %5d %6d”, x,y,z)`

Output

4678 3 19

Formatting output

- `float x=56.2757 y=2.3849 z=114.2 ;`
`printf ("%8.3f%-7.2f%7.4f",x,y,z);`





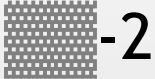
```
 56.276 2.38 114.2000
```

- `double a= 38.56, b= 201.117;`
`printf("Is it%6.1f%9.4f", a, b);`

```
Is it 38.6 201.1170
```

- `float x=333.256;`
`printf("%0.2f",x);` `333.26`

Formatting output (Practice)

Value	Format	Displayed Output		Value	Format	Displayed Output
234	%4d	 234		-234	%4d	-234
234	%5d	 234		-234	%5d	 -234
234	%6d	 234		-234	%6d	 -234
234	%1d	234		-234	%2d	-234

Formatting output (**Practice**)

Value	Format	Displayed Output	Value	Format	Displayed Output
3.14159	%5.2f	█ 3.14	3.14159	%4.2f	3.14
3.14159	%3.2f	3.14	3.14159	%5.1f	█ 3.1
3.14159	%5.3f	3.142	3.14159	%8.5f	█ 3.14159
.1234	%4.2f	0.12	-.006	%4.2f	-0.01
-.006	%8.3f	█ -0.006	-.006	%8.5f	-0.00600
-.006	%.3f	-0.006	-3.14159	%.4f	-3.1416

Extra Exercises

1. Which of the following identifiers are (a) C reserved words, (b) standard identifiers, (c) conventionally used as constant macro names, (d) other valid identifiers, and (e) invalid identifiers?

void	MAX_ENTRIES	double	time	G	Sue's
return	printf	xyz123	part#2	"char"	#insert
this_is_a_long_one					

2. Do a step-by-step evaluation of the expressions that follow if the value of celsius is 38.1 and salary is 38450.00 .

- $1.8 * \text{Celsius} + 32.0$
- $(\text{salary} - 5000.00) * 0.20 + 1425.00$

Extra Exercises

1. Which of the following identifiers are (a) C reserved words, (b) standard identifiers, (c) conventionally used as constant macro names, (d) other valid identifiers, and (e) invalid identifiers?

void	MAX_ENTRIES	double	time	G	Sue's
return	printf	xyz123	part#2	"char"	#insert
this_is_a_long_one					

2. Do a step-by-step evaluation of the expressions that follow if the value of celsius is 38.1 and salary is 38450.00 .

- $1.8 * \text{Celsius} + 32.0$
- $(\text{salary} - 5000.00) * 0.20 + 1425.00$

Example

- Write a program to reverse any two digits number? ([check slide 15](#), Algorithm lecture)

```
#include <stdio.h>
int main()
{
    int num;
    int rem;
    int rev;
    int tens;
    printf("Please enter two digits number");
    scanf ("%d", &num);
    tens= num / 10;
    rem=num % 10;
    rev= rem * 10;
    rev= rev+ tens;
    printf ("the result is %d", rev);
    return 0;
}
```


Common programming errors

- **Syntax Errors**
 - is a mistake in the syntax.

Ex:

- missing semicolon
- undeclared variable
- last comment is not closed because of blank in `*/` close-comment sequence

Common programming errors

- Logic Errors
 - an error caused by following an incorrect algorithm.

Ex:

sum = x-y (minus instead of plus)

Common programming errors

- Run-Time Errors
 - an attempt to perform an invalid operation, detected during program execution.

Ex:

`result= x / 0 (undefined)`

Common programming errors (Practice)

A Program with a syntax errors

```
221 /* Converts distances from miles to kilometers. */
222
223 #include <stdio.h>          /* printf, scanf definitions */
266 #define KMS_PER_MILE 1.609 /* conversion constant */
267
268 int
269 main(void)
270 {
271     double kms
272
273     /* Get the distance in miles. */
274     printf("Enter the distance in miles> ");
***** Semicolon added at the end of the previous source line
275     scanf("%lf", &miles);
***** Identifier "miles" is not declared within this scope
***** Invalid operand of address-of operator
276
277     /* Convert the distance to kilometers. */
278     kms = KMS_PER_MILE * miles;
***** Identifier "miles" is not declared within this scope
279
280     /* Display the distance in kilometers. */
281     printf("That equals %f kilometers.\n", kms);
282
283     return (0);
284 }
***** Unexpected end-of-file encountered in a comment
***** ")" inserted before end-of-file
```

Common programming errors (Practice)

A Program with a run-time error

```
111 #include <stdio.h>
262
263 int
264 main(void)
265 {
266     int    first, second;
267     double temp, ans;
268
269     printf("Enter two integers> ");
270     scanf("%d%d", &first, &second);
271     temp = second / first;
272     ans = first / temp;
273     printf("The result is %.3f\n", ans);
274
275     return (0);
276 }
```

Enter two integers> 14 3

Arithmetic fault, divide by zero at line 272 of routine main

Files

- Declare a **file pointer** variable
 - `FILE *ftp_in , /* pointer to input file */`
 - `FILE *ftp_out; /* pointer to output file */`
- The calls to function `fopen`
 - `ftp_in = fopen("distance.dat", "r");`
 - `ftp_out = fopen("distance.out", "w");`
- Use of the functions
 - `fscanf(ftp_in, "%lf", &miles);`
 - `fprintf(ftp_out, "The distance in miles is %.2f. \n", miles);`
- End of use
 - `fclose(ftp_in);`
 - `fclose(ftp_out);`

Files (Examples)

1. Write a program to read two integers from a file (input.txt), find the sum of them and save the result into another file (output.txt).
2. Repeat the above example, but print the result on the screen instead of saving the result of the file.

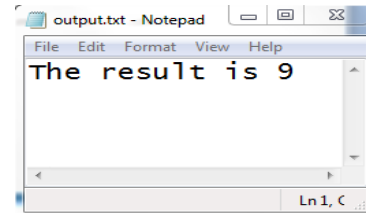
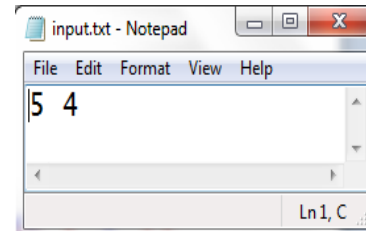
Files (Example 1 solution)

```
#include <stdio.h>
int main()
{
    FILE *fpt_in, *fpt_out;
    int num1, num2;
    int sum;

    fpt_in = fopen ("input.txt", "r");
    fpt_out = fopen ("output.txt", "w");

    fscanf (fpt_in, "%d%d", &num1, &num2);
    sum=num1+num2;

    fprintf(fpt_out, "The result is %d", sum);
    fclose(fpt_in);
    fclose(fpt_out);
    return 0;
}
```



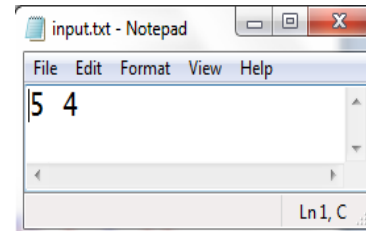
Files (Example 2 solution)

```
int main ()
{
    FILE *fpt_in;
    int num1,num2;
    int sum;

    fpt_in = fopen ("input.txt", "r");

    fscanf (fpt_in, "%d%d", &num1, &num2);
    sum=num1+num2;

    printf("The result is %d", sum);
    fclose(fpt_in);
    return 0;
}
```



```
The result is 9
Process returned 0 (0x0)   execution time : 0.009 s
Press any key to continue.
```

Type conversion through casts

- type cast
 - converting an expression to a different type by writing the desired type in parentheses in front of the expression
- Example 1: `n = (int)(9 * 0.5);`
The value of n is 4

Type conversion through casts

Using Cast

```
#include <stdio.h>
int main()
{
    int sum = 17, count = 5;
    double mean;
    mean = (double) sum / count;
    printf("Value of mean : %f\n", mean );
    return 0;
}
```

Value of mean : 3.400000

Without Cast

integer division would cause the loss of the fractional part of the mean

```
#include <stdio.h>
int main()
{
    int sum = 17, count = 5;
    double mean;
    mean = sum / count;
    printf("Value of mean : %f\n", mean );
    return 0;
}
```

Value of mean : 3.000000

Find more examples: http://www.tutorialspoint.com/cprogramming/c_type_casting.htm

Escape sequences

Escape Sequence causes the program to escape from the normal interpretation of a string, so that the next character is recognized as having **a special meaning**. The back slash “\” character is called the “**Escape Character**”.

The escape sequence includes the following:

`\n` => new line

`\t` => tab

`\r` => carriage return

`\"` => double quotations

`\\` => back slash etc.

Extra Exercises

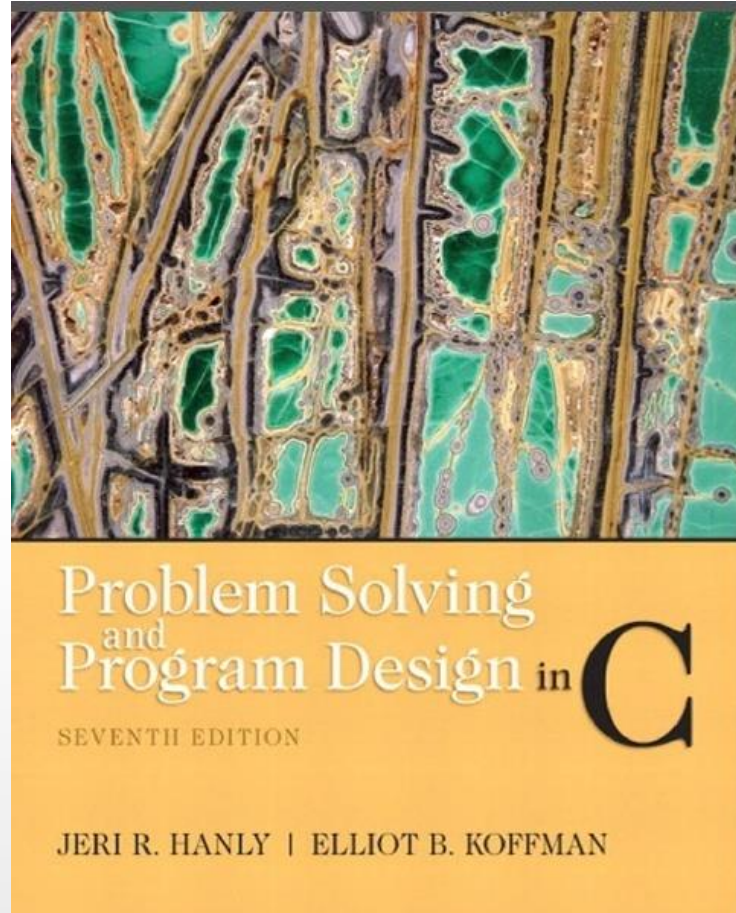
- 1) What will be the output of the printf statement `printf("hello\ryou");`
- 2) Evaluate the following formulas:
 - $7 - 15 / 4$
 - $6 * 5 / 10$
 - $2 - 4 * 3 + 26 / 2$
- 3) Find the value of x after applying the casting
 - $x = (\text{double}) (r/t)$, $r=10, t=3$
 - $x = (\text{double}) r/t$, $r=10, t=3$
 - $x = r/(\text{double})t$, $r=10, t=3$

Question?



“Success is the sum of small efforts, repeated day in and day out.”

Robert Collier



Reference: *Problem Solving & Program Design in C*