CSC 215
Procedural Programming
Introduction and Course Logistics

Dr. Achraf El Allali

About the course

Instructor: Dr. Achraf El Allali aelallali@ksu.edu.sa

TA: Mr. Abdurahman Shamriri akalshememry@ksu.edu.sa

Office Hours: Sun-Tue-Wed 9-11 am room 2119
Tue 2pm -5pm

Course Time: Section 37635 Sun-Tue, 11 am to 11:50 am
Section 37633 Sun-Tue, 1 pm to 1:50 pm

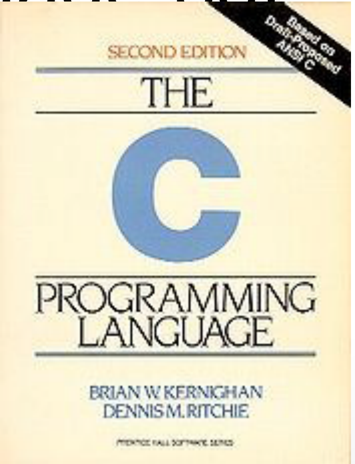
Course Website: http://fac.ksu.edu.sa/aelallali/node/44209

Grading Policy

POP Quizzes	10%
Labs:	15%
Lab Exam:	5%
Exam 1:	15%
Exam 2:	15%
Final Exam:	40%
Attendance:	Extra credit

Textbook

http://net.pku.edu.cn/~course/cs101/2008/resource/The_C_Programming_Language.pdf



Programming Languages

- Many programming languages exist, each with a specific purpose
- None is the best language
- Choose the right tool for the job based on:
  - problem scope,
  - target hardware/software,
  - memory and performance considerations,
  - portability,
  - concurrency.

Procedural programming

- The program is divided up into subroutines or procedures
- Allows code to become structured
- The programmer must think in terms of actions:
  - decide which procedures and data structures you want

Object Oriented programming

- Very useful to organize large software projects
- The data is broken into ‘objects’ and the sequence of commands becomes the interactions between objects:
    - Decide which classes you need, provide a full set of operations for each class, and make commonality explicit by using inheritance.

Procedural Languages

- Procedural languages include:
  - Fortran
  - BASIC
  - Pascal
  - C

Why C

- Provides low -level access to memory

- Provides language constructs that map efficiently to machine instructions

C Strengths

- **Efficiency**: intended for applications where assembly language had traditionally been used.
- **Portability**: hasn’t splintered into incompatible dialects; small and easily written
- **Power**: large collection of data types and operators
- **Flexibility**: not only for system but also for embedded system commercial data processing
- Standard library
- Integration with UNIX

C Weaknesses

- Error-prone:
  - Error detection left to the programmer
- Difficult to understand
  - Large programmes
  - Difficult to modify
- Memory management
  - Memory management is left to the programmer

Similarities with Java

- /* Comments */
- Variable declarations
- if / else statements
- for loops
- while loops
- function definitions (like methods)
- Main function starts program

Differences between C and Java

- C does not have objects
  - There are “struct”ures
- C is a procedural programming language
- C allows pointer manipulation
- Input / Output with C
  - Output with **printf** function
  - Input with **scanf** function
- C requires memory management

C vs. Java

C	Java
Procedural	Object Oriented
Compiled	Interpreted
No Memory Management	Memory Management
Pointers	References
Error Codes	Exceptions

Let’s multiply two number

```
a = 3;
b = 2;
```

Let’s multiply two number

```
a = 3;
b = 2;
c = a * b;
```

Let’s multiply two number

```
int a, b, c;
a = 3;
b = 2;
c = a * b;
```

Let’s multiply two number

```
int a, b, c;
a = 3;
b = 2;
c = a * b;
printf(“The product is %d”, c);
```

Let’s multiply two number

```
main()
{
  int a, b, c;
  a = 3;
  b = 2;
  c = a * b;
  printf(“The product is %d”, c);
}
```

Let’s multiply two number

```
#include<stdio.h>
main()
{
  int a, b, c;
  a = 3;
  b = 2;
  c = a * b;
  printf(“The product is %d”, c);
}
```

Let’s multiply two number

```
#include<stdio.h>  /*header file*/
main()
{
  int a, b, c; // variable declaration
  a = 3;
  b = 2;
  c = a * b;
  printf(“The product is %d”, c);
}
```

Compile and execute

- To compile “product.c”
    - gcc -o product product.c
        - “-o” place the output in file product
        - “product” is the executable file

- To execute the program
    - ./product

C statements

- Variable declaration
  - int a;
  - int b, c;
- Assignment
  - a = b + 2;
  - a = b + c;
- Function call
  - printf(“CSC 215”);

Variables

- Hold values
- Must be declared before use
- Naming rules
    - Made up of letters (upper and lower case) and digits.
    - The underscore character ("_") is also permitted.
    - Must not begin with a digit
    - Must not be a special keyword
    - x = 1; /*x is a variable*/

Basic data types

- The int type
  - int a; /* Integer value like 1, 10 and -5 */
- The char type
  - char c; /* Character value like a, b, c, $ and \n */
- The float type
  - float f; /* Decimal fraction value like 0.1, 1.5 */
- The double type
  - double d; /* Decimal fraction value like 0.1, 1.5 */

int

- 4 Bytes (compiler dependent)
    - 2^32 values total
    - -2^31 to 2^31-1
- Variants
    - short int a; /* 2 bytes */
    - long int a; /* 8 bytes */
    - unsigned int a; /* Only positive numbers */
        - 0 to 2^32--1

char

- 1 byte
  - A total of 2^8 values
  - Example char x = ‘d’;
- ASCII representation
  - Ascii value of 'a' is 97
  - Ascii value of 'b' is 98
  - http://www.asciitable.com/

float

- 4 bytes
    - IEEE format
    - -3.4e ^ 38 to 3.4e ^ 38
- Example

```
float a;
a = 2.54;
```

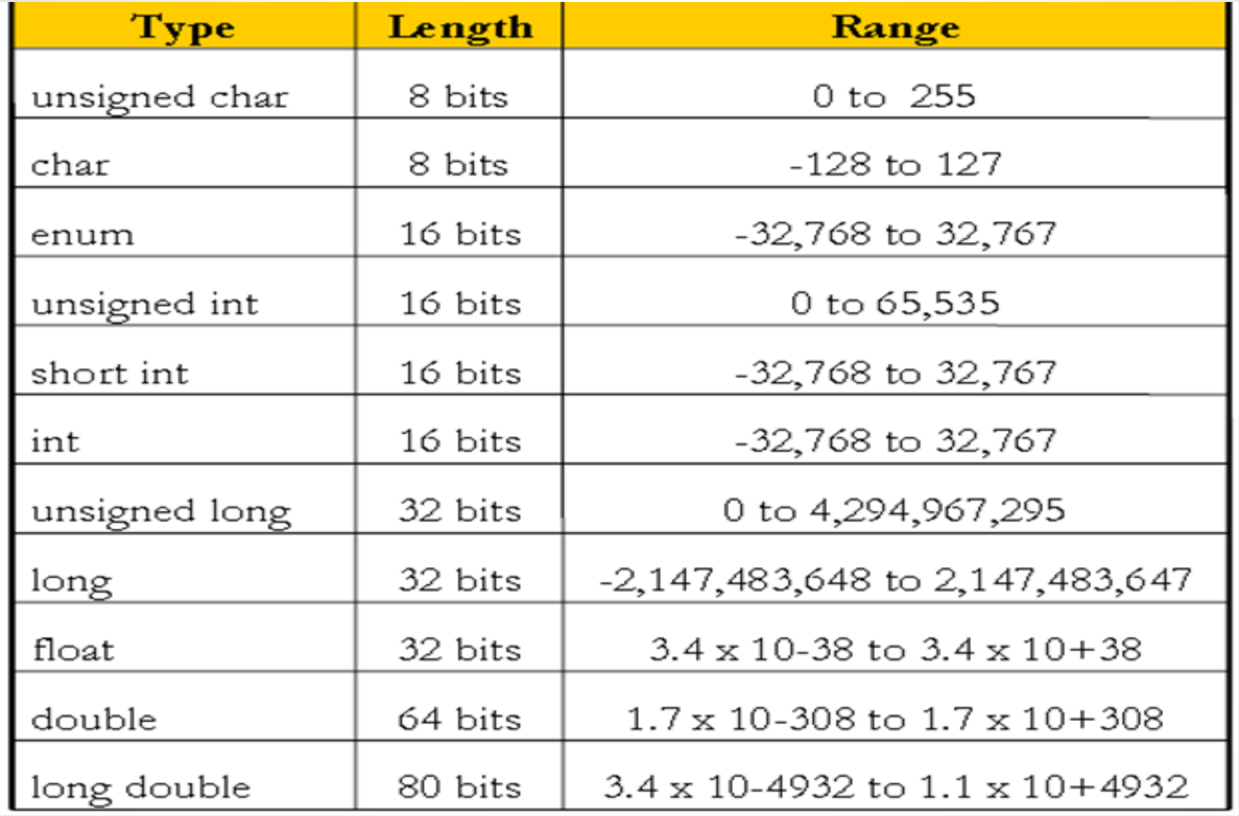
double

- Twice the memory as float
  - 8 bytes (generally)

What about the boolean type

?

Summary of Data types



Type	Length	Range
unsigned char	8 bits	0 to 255
char	8 bits	-128 to 127
enum	16 bits	-32,768 to 32,767
unsigned int	16 bits	0 to 65,535
short int	16 bits	-32,768 to 32,767
int	16 bits	-32,768 to 32,767
unsigned long	32 bits	0 to 4,294,967,295
long	32 bits	-2,147,483,648 to 2,147,483,647
float	32 bits	3.4 x 10-38 to 3.4 x 10+38
double	64 bits	1.7 x 10-308 to 1.7 x 10+308
long double	80 bits	3.4 x 10-4932 to 1.1 x 10+4932

sizeof

```
#include<stdio.h> /*Header file*/
main() /* The main function */
{
   int x = 7; /*Variable Declaration*/
   printf ("x is %d bytes", sizeof(x));
}
```

Casting

- Cast a variable to a different type than its actual type

```
int x;
float y;
x = 3;
y = (float) x; /* Explicit casting */
y = x; /* Implicit casting */
```

Function printf

printf(control_string, arg1, arg2, …);

- control_string is the control string or conversion specification.
- Consists of the character % followed by optional minimum width and precision as well as a required conversion control character

Example

```
printf(”The product of %d and %d is %d”, a,b,c);
```

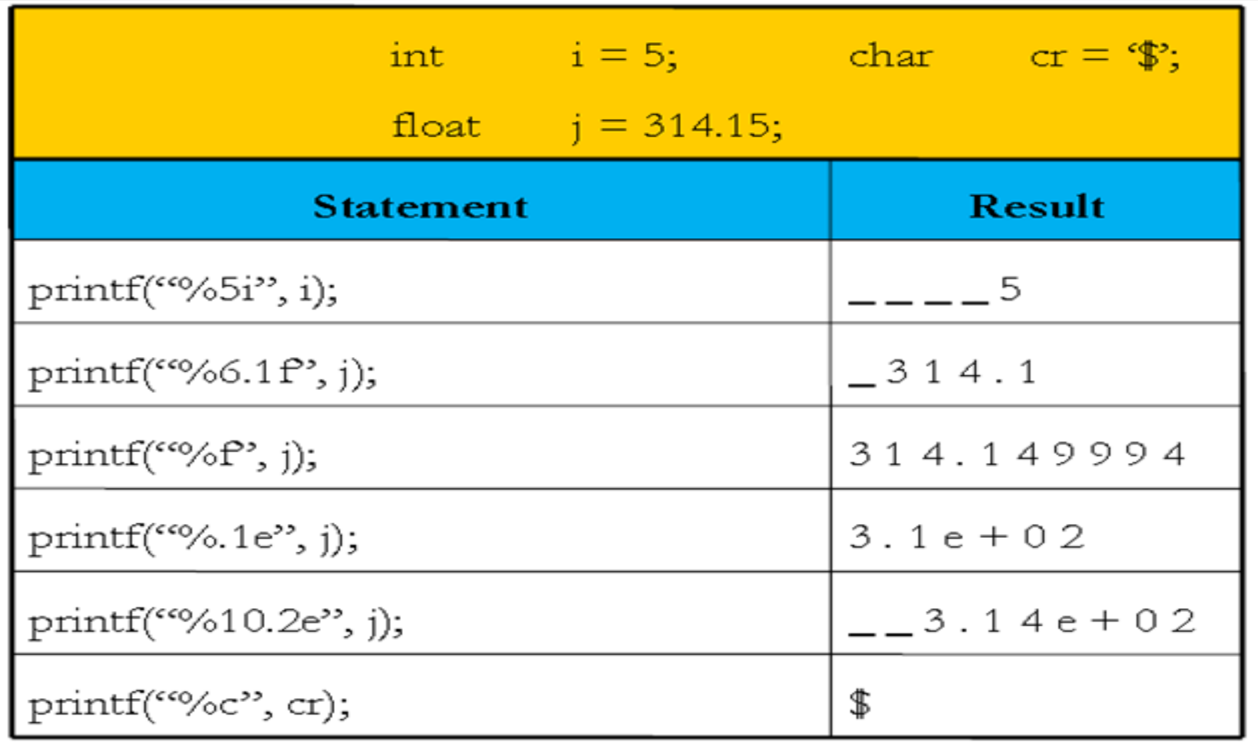
- Ouput

```
The product of 3 and 2 is 6
```

Placeholders

- %d - int (same as %i)
- %ld - long int (same as %li)
- %f - float
- %lf - double
- %c - char
- %s - string
- %x - hexadecimal

Precision

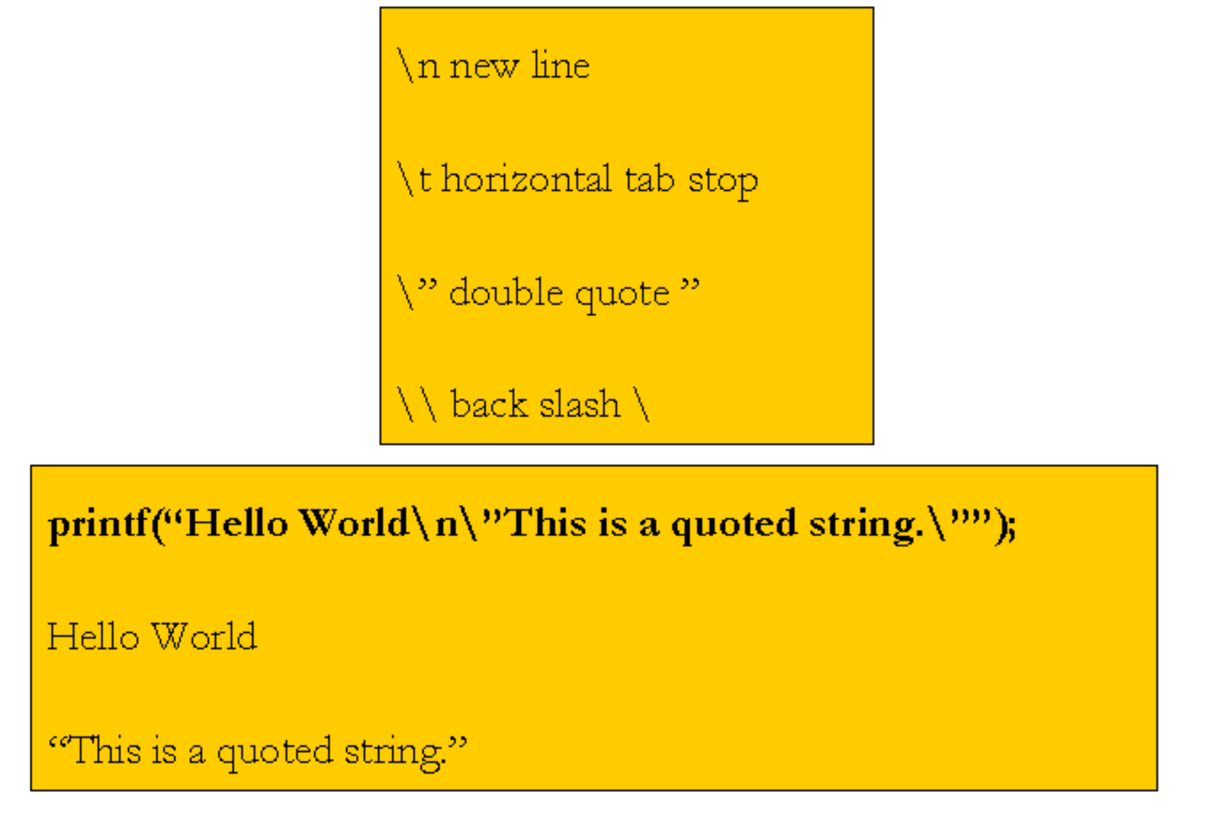


int i = 5; char cr = ‘$’;

float j = 314.15;

Statement	Result
printf(““%5i””, i);	____5
printf(““%6.1f””, j);	_314.1
printf(““%f””, j);	314.149994
printf(““%.1e””, j);	3.1e+02
printf(““%10.2e””, j);	__3.14e+02
printf(““%c””, cr);	$

New line, tabs and escape character



\n new line

\t horizontal tab stop

\” double quote ”

\\ back slash \

```
printf(“Hello World\n\”This is a quoted string.\””);
```

```
Hello World
“This is a quoted string.”
```

scanf

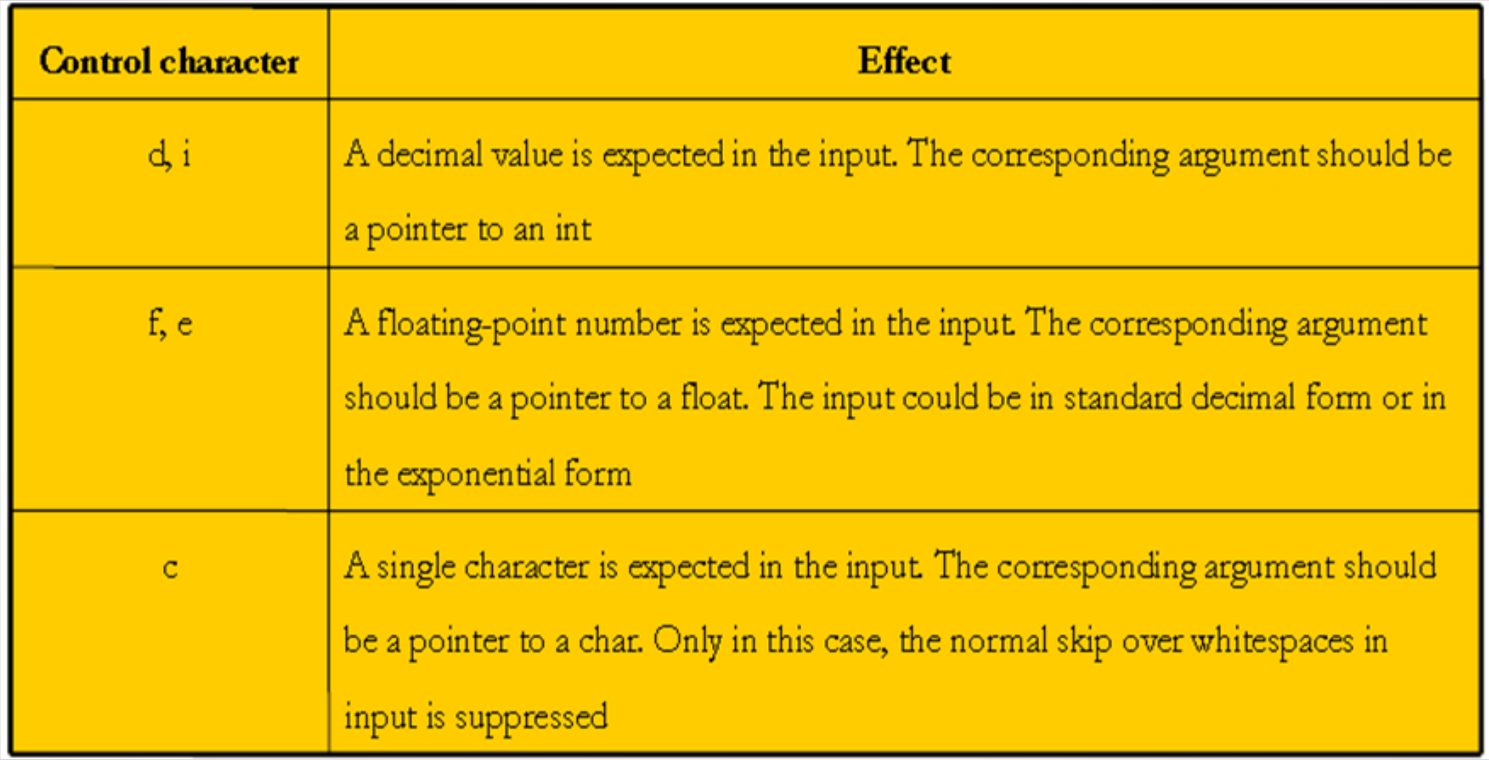
```
scanf(control_string, arg1, arg2, …);
```

- Control_string governs the conversion, formatting, and printing of the arguments
- Each of the arguments must be a pointer to the variable in which the result is stored.
- So:

`scanf(“%d”, &var);` is a correct one, while

`scanf(“%d”, var);` is not correct

Spaceholders



Control character	Effect
d, i	A decimal value is expected in the input. The corresponding argument should be a pointer to an int
f, e	A floating-point number is expected in the input. The corresponding argument should be a pointer to a float. The input could be in standard decimal form or in the exponential form
c	A single character is expected in the input. The corresponding argument should be a pointer to a char. Only in this case, the normal skip over whitespaces in input is suppressed

Example

```
#include<stdio.h> /*Header file*/
main() /* The main function */
{
  int a, b, c; /*Variable Declaration*/
  printf(”Enter a:”);
  scanf(”%d”, &a); /* Wait for input */
  printf (”Enter b:”);
  scanf (”%d”, &b); /* Wait for input */
  c = a * b;
  printf ("The product is %d", c);
}
```