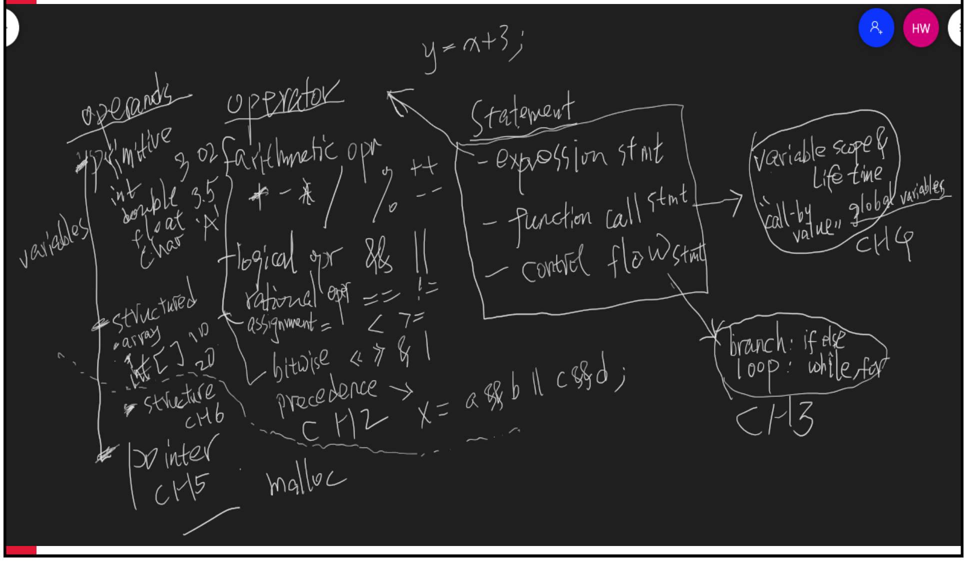
## Roadmap -- How the topics are related



38

## C basics

- **The first program – what it looks like**
- Compile and run C program
- Basic syntax
  - Comments
  - Variables
  - Functions
  - Basic IO functions
  - Expression
  - Statements
  - Preprocessing: #include, #define

39

39

## First C program -- what it looks like?

```c
#include <stdio.h>
/* import standard io header */

/* salute the world */

int main (int argc, char** argv)
{
   printf( "Hi, world\n" );
   return 0;
}
```

```java
import java.util.*;
/* import library functions */

public class Hello
{

 public static void main(String[] args)
 {
   // System.out.println("Hi World!");
    System.out.printf ("Hi, world\n");
 }
}
```

hello.c   first.c
any_name.c

Hello.java

40

40

## First C program -- what it looks like?

```c
#include <stdio.h>
/* import standard io header */

/* salute the world */

int main (int argc, char** argv)
{
   printf( "Hi, world\n" );
   return 0;
}
```

```java
import java.util.*;
/* import library functions */

public class Hello
{

 public static void main(String[] args)
 {
   // System.out.println("Hi World!");
    System.out.printf ("Hi, world\n");
 }
}
```

hello.c   first.c
any_name.c

Hello.java

41

41

## First C program -- what it looks like?

```
#include <stdio.h>
/* import standard io header */

/* salute the world */

 main ()
{
   printf( "Hi, world\n" );

}
```

```
import java.util.*;
/* import library functions */

public class Hello
{

 public static void main(String[] args)
 {
   // System.out.println("Hi World!");
    System.out.printf ("Hi, world\n");
 }
}
```

```
hello.c   first.c
any_name.c
```

```
Hello.java
```

42

42

## Another C program -- a function

cal.c

```
#include <stdio.h>


int sum (int i, int j) {
     int k;
     k = i + j;
     return k;    // return i+j;
}

/* main */
main()   {
   int x=2, y=3;
   int su = sum(x,y);
   printf("Sum: %d\n", su);
}
```

```
Sum: 5
```

Cal.java

```
import java.util.*;
public class Cal
{
   static int sum(int i, int j) {
      int k;
      k = i + j;
      return k;
   }


 public static void main(String[] args){
    int x=2, y=3;
    int su = sum(x,y);
    System.out.printf("Sum: %d\n", su);
    System.out.println("Sum: " + su);
 }
}
```

```
javac Cal.java
java Cal
```

43

43

## C basics

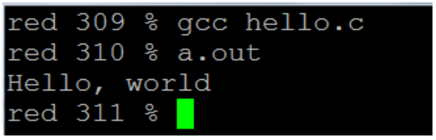
- The first program – what it looks like
- **Compile and run C program**
- Basic syntax
  - Comments
  - Variables
  - Functions
  - Basic IO functions
  - Expression
  - Statements
  - Preprocessing: # include, # define

44

44

## Compiling and running a C program (general)

- C programs (source code) are in files ending with `.c` e.g., **`hello.c`**
- To compile a C program, naturally in Unix. In our lab:
  `% gcc hello.c`
  - If no syntax error, complier returns silently and creates an executable program named **`a.out`** (in the current directory)
- To run
  `% ./a.out` or `a.out`

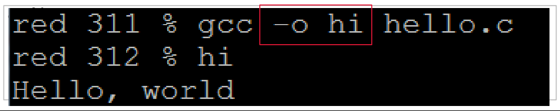


```
red 309 % gcc hello.c
red 310 % a.out
Hello, world
red 311 %
```

Only one a.out

`% gcc hello.c -o hi`

- create an executable program named `hi` (in the current directory)



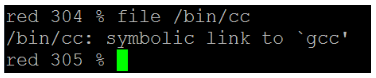
```
red 311 % gcc -o hi hello.c
red 312 % hi
Hello, world
```

Either before or after hello.c

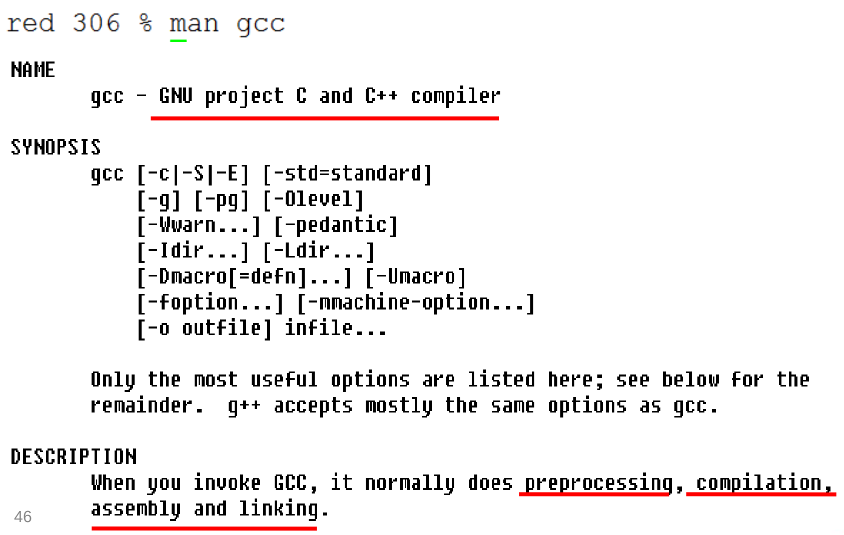
45

45

**cc hello.c** also works in our lab,
**cc** is a 'symbolic link' to **gcc** --- when you use **cc**, you are using **gcc**



```
red 304 % file /bin/cc
/bin/cc: symbolic link to `gcc'
red 305 %
```



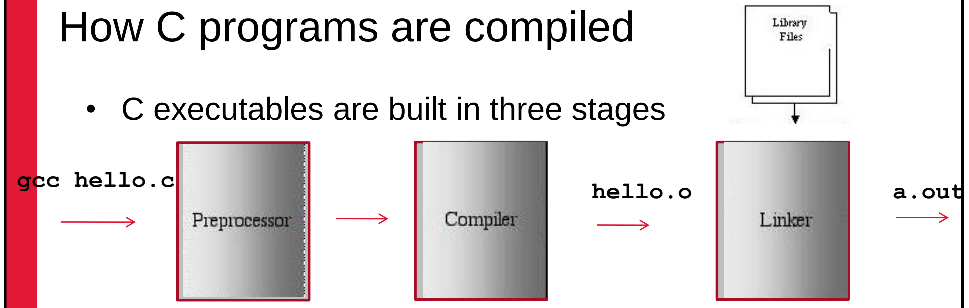
```
red 306 % man gcc

NAME
       gcc - GNU project C and C++ compiler

SYNOPSIS
       gcc [-c|-S|-E] [-std=standard]
           [-g] [-pg] [-Olevel]
           [-Wwarn...] [-pedantic]
           [-Idir...] [-Ldir...]
           [-Dmacro[=defn]...] [-Umacro]
           [-foption...] [-mmachine-option...]
           [-o outfile] infile...

       Only the most useful options are listed here; see below for the
       remainder.  g++ accepts mostly the same options as gcc.

DESCRIPTION
       When you invoke GCC, it normally does preprocessing, compilation,
       assembly and linking.
```

46

46

## How C programs are compiled

- C executables are built in three stages



Handles
#include
#define

Converts C code into binary processor instructions -- **hello.o**

- Puts multiple object files together,
- linking object files with necessary system libraries (e.g., code of **printf()**)
- creates an executable program e.g., **a.out**
- remove **hello.o**

look into this later

47

47

## C → K&R C → ANSI C (C89/90) → C99 → C11

`gcc` -- GNU C and C++ compiler, Only C compiler for Linux.
- Support different standards and more
- Default: C89/90 + <u>some</u> C99 features        Enough for the course

```
gcc hello.c
```

- A variable declared just before its first use.
  Mix variable declarations with uses   (C99 feature, incl.)
- /* */  (C89 feature)    //  comment also  (C99 feature, incl.)
- `int i; for (i=0; i<10;i++)`   (C89 feature)
- …..   `for (int i=0; i<10;i++)` ✗   (C99 feature, not inc.)

48

48

## C → K&R C → ANSI C (C89/90) → C99 → C11

`gcc` -- GNU C and C++ compiler, Only C compiler for Linux.
- Support different standards and more
- Default: C89/90 + <u>some</u> C99 features        Enough for the course

```
gcc hello.c
```

- A variable declared just before its first use.
  Mix variable declarations with uses   (C99 feature, incl.)
- /* */  (C89 feature)    //  comment also  (C99 feature, incl.)
- `int i; for (i=0; i<10;i++)`   (C89 feature)
- …..   `for (int i=0; i<10;i++)` ✗   (C99 feature, not inc.)

- To compile using ANSI (C89):

```
gcc -ansi     hello.c
gcc -std=c89  hello.c
```

- To compile using C99 :

```
gcc -std=c99 hello.c
```

```
for (int i=0; i<10;i++)
```
✓

49

For your information

49

## Compiling and running. C vs. Java

| | C | Java |
|---|---|---|
| Program | `hello.c:`<br><br>`#include <stdio.h>`<br><br>`int main(int argc, char**argv){`<br>`  printf("Hello, world\n");`<br>`  return 0;`<br>`}` | `Hello.java:`<br><br>`#import java.util.*;`<br><br>`public class Hello {`<br>`  public static void`<br>`    main(String[] args) {`<br>`      System.out.printf(`<br>`        "Hello, world\n");` |
| Compile | `% gcc hello.c`<br>`% ls`<br>`hello.c     a.out`<br>`%` | `% javac Hello.java`<br>`% ls`<br>`Hello.java    Hello.class`<br>`%` |
| Run | `% a.out   or   ./a.out`<br>`Hello, world`<br>`%` | `% java Hello`<br>`Hello, world`<br>`%` |

```
% gcc hello.c -o xyz
% xyz   or  ./xyz
```

YORK UNIVERSITÉ UNIVERSITY 50

50

## Compiling and running C

- In lab, coding using any text editor or IDE
  - JEditor, Neditor, Atom, Code:Blocks, Visual Studio Code…
  - In terminal, compile using command line **gcc**

Code::Blocks

- You may need some basic unix/linux command for your labs/labtests.

  **pwd  cd ./abc  cd ..  ls  cat  <  >**

  **rm  mkdir  cp  mv**

  - If you need to learn or recap, start off with the guided lab tour (CSE1020) and the UNIX tutorial posted on the course website.

- BTW, do you know the following?

  **grep  wc  chmod  sort  cut  find  cmp  uniq**

  - Don't worry for now. We will learn them later.

51

51

## Work off campus?

- Remotely: connect to the lab
  - Window: an ssh client
    - use PuTTY (and the like) to connect to server (red) and work online
      - Nano, vi, emacs as text-based editor
    - Graphical? Emulator for xterm (e.g., MobaXterm)
  - MAC:
    - ssh your_user_name@red.cse.yorku.ca
  - Window/MAC: Virtualbox instructions on course web

https://wiki.eecs.yorku.ca/dept/tdb/covid19:start /tdb/login:sshsupport

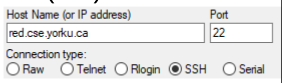
- An easier way to connect to the lab: (temporary)

52

## Work off campus?

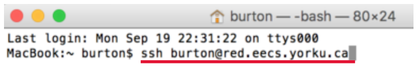
https://wiki.eecs.yorku.ca/dept/tdb/covid19:start /tdb/login:sshsupport

- Remotely: connect to the lab
  - Window: an ssh client
    - use PuTTY (and the like) to connect to server (red) and work online
      - ssh, port 22 red.cse.yorku.ca
      - pico, nano, vi, emacs as text-based editor



Host Name (or IP address): red.cse.yorku.ca
Port: 22
Connection type: ○ Raw ○ Telnet ○ Rlogin [✓ SSH] ○ Serial

    - Graphical? Emulator for xterm (e.g., MobaXterm)
  - MAC/Ubuntus:
    - Terminal: ssh your_user_name@red.cse.yorku.ca



```
burton — -bash — 80×24
Last login: Mon Sep 19 22:31:22 on ttys000
MacBook:~ burton$ ssh burton@red.eecs.yorku.ca
```

  - Window/MAC: Virtualbox instructions on course web
- An easier way to connect to the lab: (temporary)

53

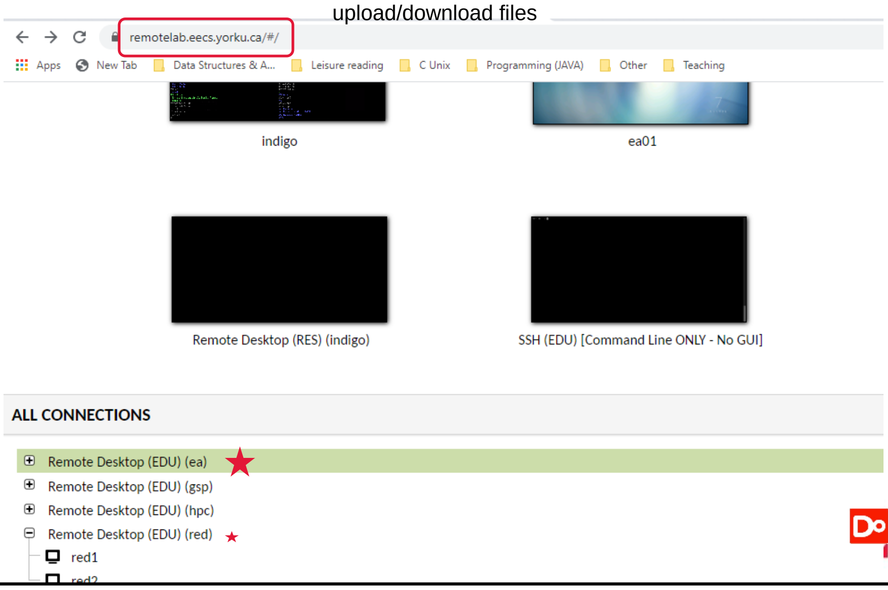
Let me know if you need help

53

## Connect to lab: new this term (temporary)

https://wiki.eecs.yorku.ca/dept/tdb/services:remotelab

Can use Guacamole menu (Ctrl+Alt+Shift) to upload/download files



ALL CONNECTIONS

- ⊞ Remote Desktop (EDU) (ea) ★
- ⊞ Remote Desktop (EDU) (gsp)
- ⊞ Remote Desktop (EDU) (hpc)
- ⊟ Remote Desktop (EDU) (red) ★
  - red1
  - red2

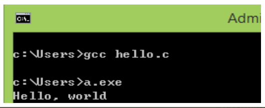
54

## Work off campus?

- Remotely: connect to the lab
  - Window: an ssh client
    - use PuTTY (and the like) to connect to server (red) and work online
    - Nano, vi, emacs as text-based editor
    - Graphical? Emulator for xterm (e.g., MobaXterm)
  - MAC:
    - ssh your_user_name@red.cse.yorku.ca
  - Window/MAC: Virtualbox instructions on course web

https://wiki.eecs.yorku.ca/dept/tdb/covid19:start /tdb/login:sshsupport

- Locally (@home, laptop) + transfer to lab
  - Windows: (not recommended) use a windows compiler for C codes
    - A good choice: GCC compiler called MinGW `cmd (dos) → gcc → a.exe`
    - IDE e.g., Code::Blocks (MinGW included)
  - For MAC
    - Xcode `./a.out`
  - Recommend: Ubuntu
    - different flavors
    - `./a.out`



```
c:\Users>gcc hello.c

c:\Users>a.exe
Hello, world
```

Let me know if you need help

55

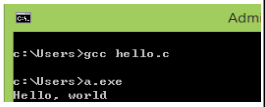
## Work off campus?

- Remotely: connect to the lab

putty.exe

MobaXterm

- Locally (@home, laptop) + transfer to lab
  - For MAC
    - Xcode `./a.out`



```
c:\Users>gcc hello.c

c:\Users>a.exe
Hello, world
```

  - Windows: (not recommended) use a windows compiler for C codes
    - A good choice: GCC compiler called MinGW

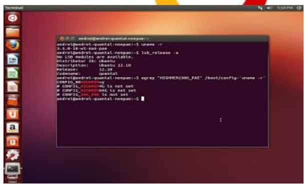
```
cmd (dos)  →  gcc  →  a.exe
```

    - IDE e.g., Code::Blocks (MinGW included)

Code::Blocks

ubuntu

  - Windows: (recommend): Ubuntu
    - different flavors
    - Windows 10? Linux subsystem (WLS)
    - `./a.out`



56 Let me know if you need help

56

## Work locally (on your own machine)?

MobaXterm

- OK, but need to submit from lab environment
  - Need to somehow transfer files to lab
    - An FTP client, e.g., WinSCP, FileZilla
    - MobaXterm can also do the transfer.
    - Guacamole menu of Remote lab
  - Web-submission is mostly not working
  - Instruction on course web

WinSCP

FileZilla

Let me know if you need help

- All submitted work
  - **must compile in our lab!!!** **default: (C89 + some C99)**

```
gcc hello.c
```

  - are welcomed to follow ANSI-C (C89) but not required

```
gcc -ansi     hello.c
gcc -std=c89  hello.c
```

- Kind suggestion for working mostly on your machine:
  - 57 Wrap up and compile in our lab for final deliverable version
  - Gets you better prepared for the term tests and unix

57

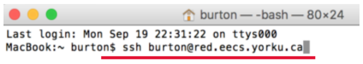
## Coding environment -- summary

https://wiki.eecs.yorku.ca/dept/tdb/covid19:start /tdb/login:sshsupport

- Work on lab directly (work + submit)
  - Remote lab -- an easier way to connect to the lab: (temporary)
    - create folder Desktop download directly can save
    - submit 2031A labX fileY.c
  - MAC: ssh pico Windows: Putty, MobaXterm, cmd terminal, pico
  - Window/MAC: Virtualbox instructions on course web

- Locally (work + transfer + submit)
  - MAC:
    - work: terminal.
    - transfer: FileZilla MAC, remote lab Guacamole Menu (Ctrl+Alt+Shift), scp command
    - submit: remote lab, ssh

```
scp Point2.java burton@red.eecs.yorku.ca:submit
scp SpiroUtil.java burton@red.eecs.yorku.ca:submit
```



```
burton — -bash — 80×24
Last login: Mon Sep 19 22:31:22 on ttys000
MacBook:~ burton$ ssh burton@red.eecs.yorku.ca
```

  - Window:
    - work: GCC compiler called MinGW (not recommended)
    - transfer: FileZilla, WinSCP, MobaXterm, remote lab Guacamole Menu (Ctrl+Alt+Shift)
    - submit: remote lab, Putty, MobaXterm

**must compile in our lab!!!**

- Ubuntus on Windows: (recommended)
  - ❖ Dual boot, install gcc graphic
  - ❖ WLS need to install gcc, no graphic. need command line for two file systems.

***cp hi.c /mnt/C/User/Desktop***

Let me know if you need help

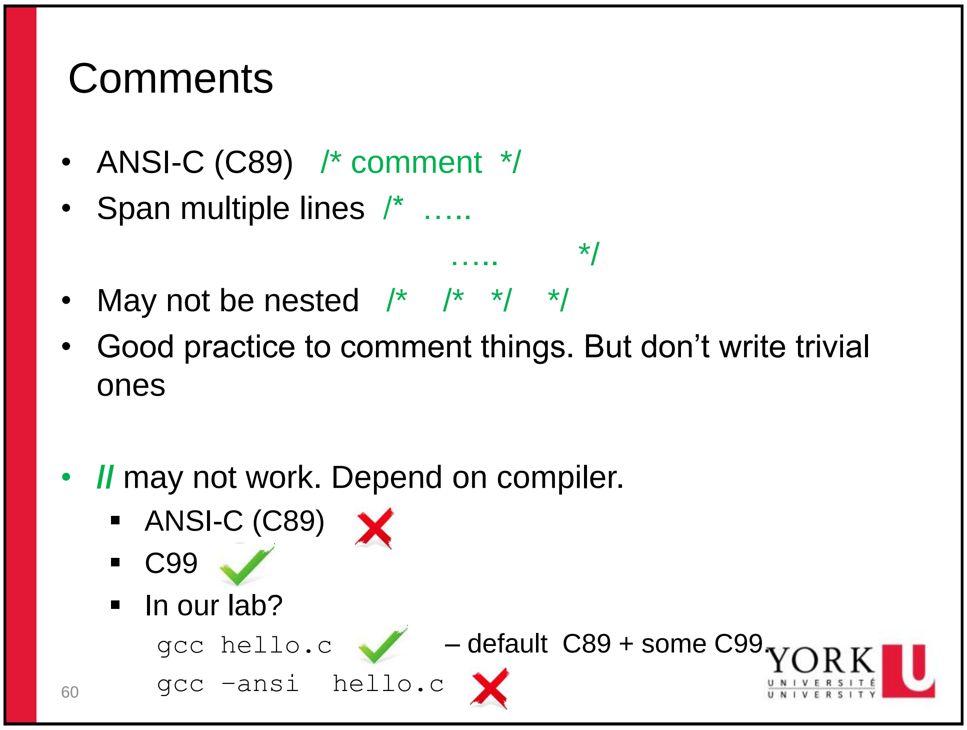
58

## C basics

- Compile and run C program
- Basic syntax
  - **Comments**
  - Variables
  - Functions
  - Basic IO functions
  - Expression
  - Statements
  - Preprocessing: # include, # define

59

59



## Comments

- ANSI-C (C89) /* comment */
- Span multiple lines /* …..

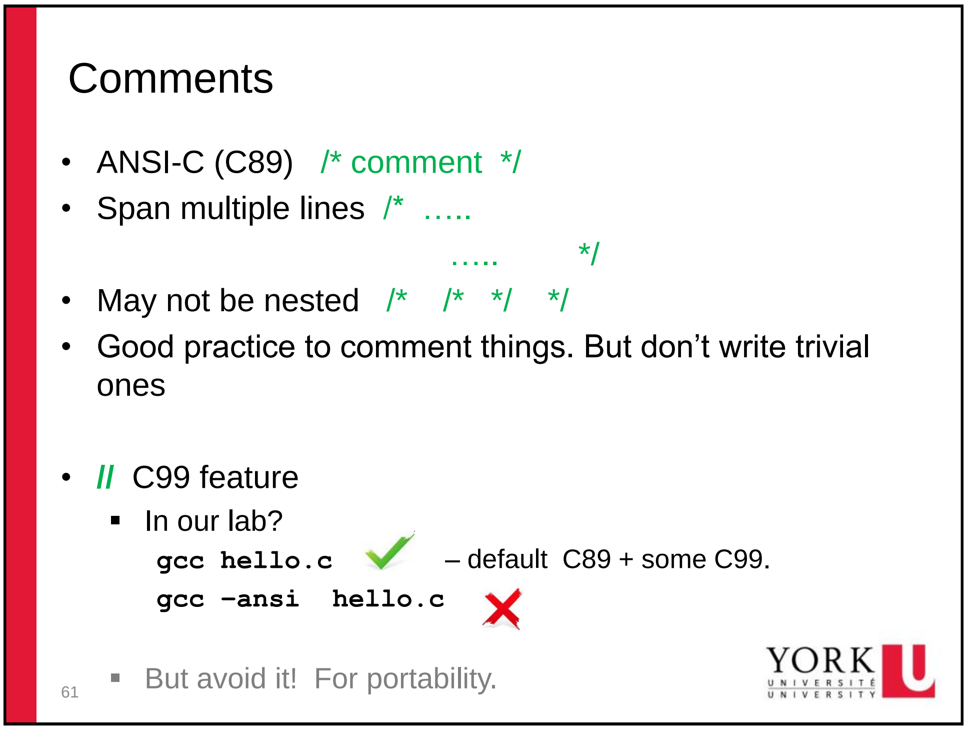
  ….. */
- May not be nested /* /* */ */
- Good practice to comment things. But don't write trivial ones

- // may not work. Depend on compiler.
  - ANSI-C (C89) ✗
  - C99 ✓
  - In our lab?

```
gcc hello.c          ✓   – default C89 + some C99.
gcc -ansi  hello.c   ✗
```

60

60



## Comments

- ANSI-C (C89) /* comment */
- Span multiple lines /* …..

  ….. */
- May not be nested /* /* */ */
- Good practice to comment things. But don't write trivial ones

- // C99 feature
  - In our lab?

```
gcc hello.c          ✓   – default C89 + some C99.
gcc -ansi  hello.c   ✗
```

  - But avoid it! For portability.

61

61

## C variables

- Compile and running Comments
- Basic syntax
  - Comments
  - **Variables**
  - Functions
  - Basic IO functions
  - Expression
  - Statements
  - Preprocessing: # include, # define

62

62

## C variables

- Store data, whose value can change.
  - Declaration and initialization.
    - `int x;   x = 5;`
    - `int x =5; x = 9;`

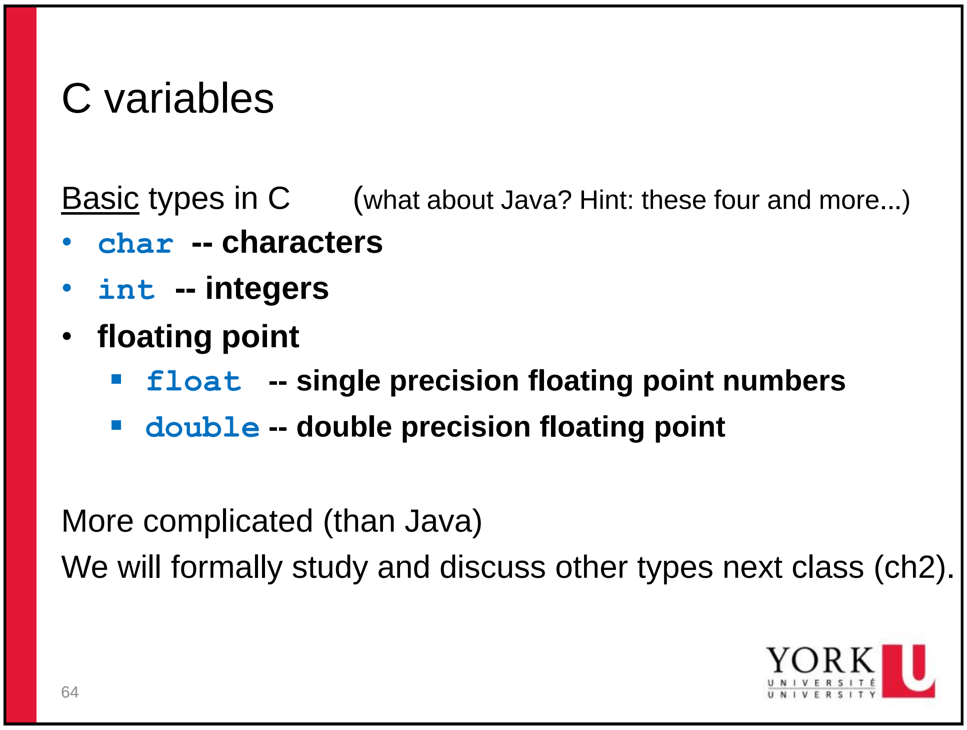
Same in JS, Java, C++

- Variable names
  - combinations of letters (including underscore character _), and numbers.
  - that do not start with a number; avoid starting with _;
  - are not a keyword.
  - upper and lower case letters are distinct (**x** ≠ **X**).

- Examples: Identify valid and invalid variable names
  - **`abc, aBc, abc5, aA3_ , my_index`** ✓
  - **`5sda, _360degrees, _temp, char, struct, while`** ✗

63

63



## C variables

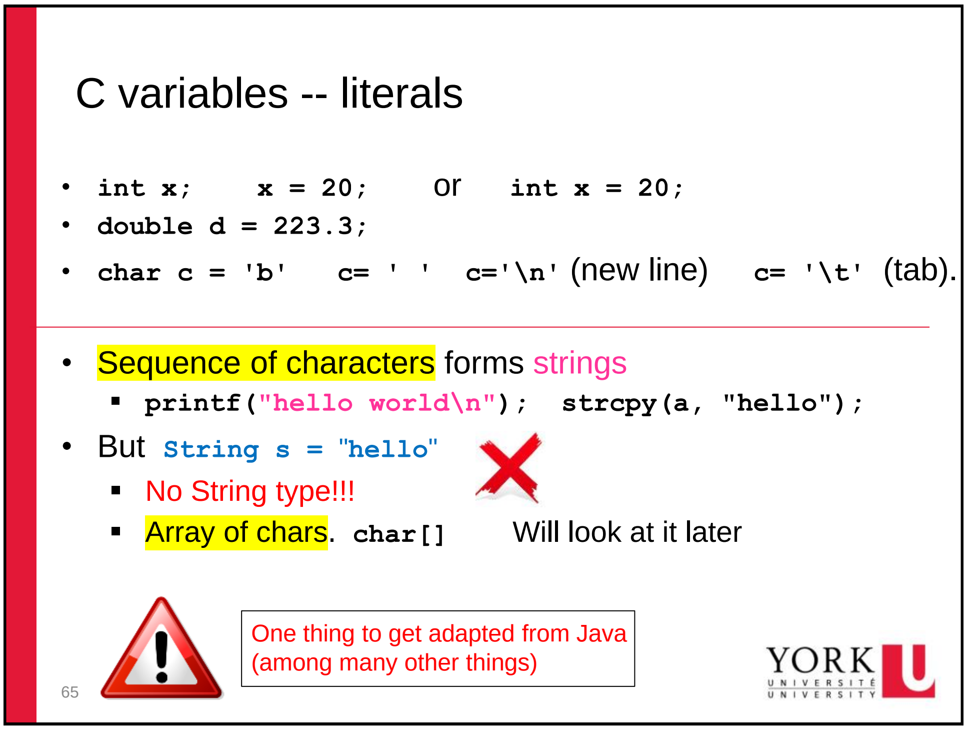
Basic types in C (what about Java? Hint: these four and more...)

- `char` **-- characters**
- `int` **-- integers**
- **floating point**
  - `float` **-- single precision floating point numbers**
  - `double` **-- double precision floating point**

More complicated (than Java)
We will formally study and discuss other types next class (ch2).

64

64



## C variables -- literals

- `int x;     x = 20;` or `int x = 20;`
- `double d = 223.3;`
- `char c = 'b'` `c= ' '` `c='\n'` (new line) `c= '\t'` (tab).

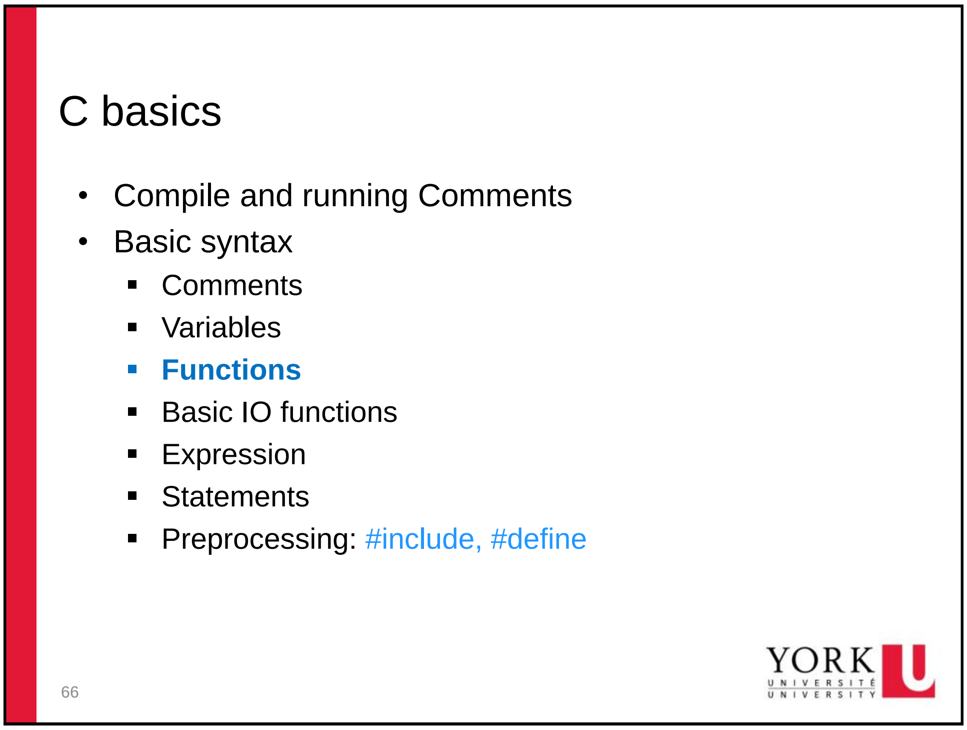
---

- Sequence of characters forms strings
  - `printf("hello world\n");` `strcpy(a, "hello");`
- But `String s = "hello"`
  - No String type!!!
  - Array of chars. `char[]` Will look at it later

One thing to get adapted from Java
(among many other things)

65

65

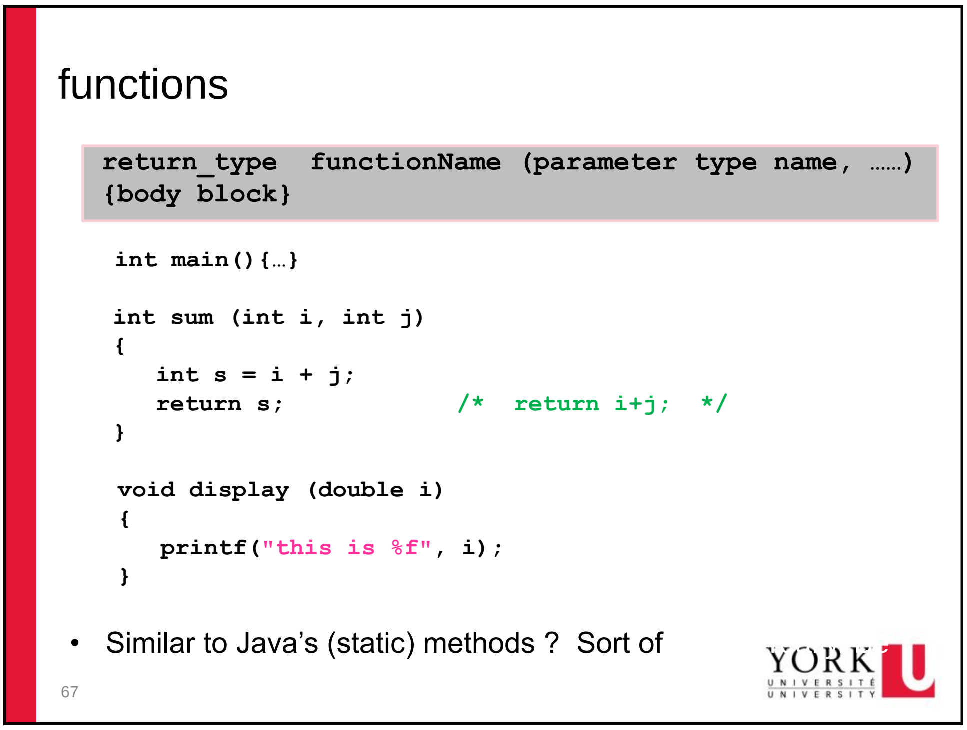


## C basics

- Compile and running Comments
- Basic syntax
  - Comments
  - Variables
  - **Functions**
  - Basic IO functions
  - Expression
  - Statements
  - Preprocessing: #include, #define

66

66



## functions

```
return_type  functionName (parameter type name, ……)
{body block}
```

```
int main(){…}

int sum (int i, int j)
{
   int s = i + j;
   return s;            /*  return i+j;  */
}

void display (double i)
{
   printf("this is %f", i);
}
```

- Similar to Java's (static) methods ? Sort of

67

67

## functions

One thing to get adapted from Java (among many other things)

- ***Must be declared or defined before point of the (first) call !***
  - *Otherwise compiling error* *C89, C99* *– different from Java*

- Declaration (prototype) – describe arguments and return type, but no implementation

```
int sum (int i, int j);        or    int sum(int, int);

void display(double i);        or    void display(double);
```

- Definition – describe arguments and return value, and gives the code

```
int sum (int i, int j){
   return i+j;        /*  int s = i + j; return s; */
}

void display (double i)
{
   printf("this is %f", i);
}
```

68

## functions

/* Contains declaration (prototype) of printf() */

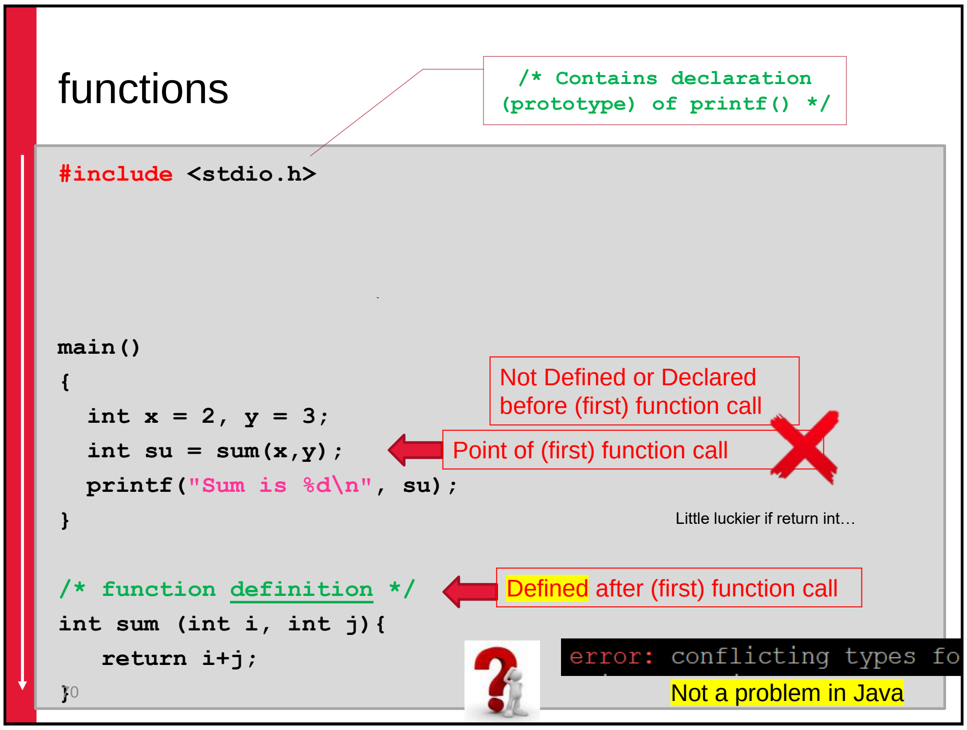
```
#include <stdio.h>

/* function definition */
int sum (int i, int j){
   return i+j;
}

main()
{
  int x = 2, y = 3;
  int su  = sum(x , y);
  printf("Sum is %d\n", su);
}
```

Defined before (first) function call

Point of (first) function call

69

69



## functions

/* Contains declaration (prototype) of printf() */

```
#include <stdio.h>




main()
{
  int x = 2, y = 3;
  int su = sum(x,y);
  printf("Sum is %d\n", su);
}

/* function definition */
int sum (int i, int j){
   return i+j;
}
```

Not Defined or Declared before (first) function call

Point of (first) function call

Little luckier if return int…

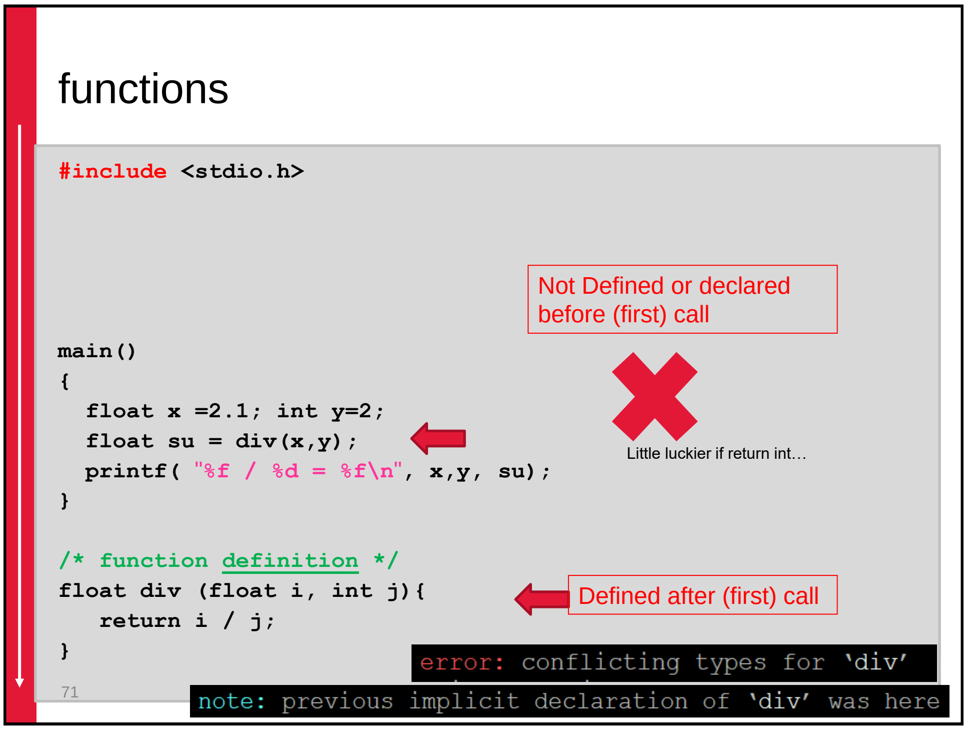
Defined after (first) function call

```
error: conflicting types fo
```

Not a problem in Java

70

70



## functions

```
#include <stdio.h>




main()
{
  float x =2.1; int y=2;
  float su = div(x,y);
  printf( "%f / %d = %f\n", x,y, su);
}

/* function definition */
float div (float i, int j){
   return i / j;
}
```

Not Defined or declared before (first) call

Little luckier if return int…

Defined after (first) call

```
error: conflicting types for 'div'
note: previous implicit declaration of 'div' was here
```

71

71

## functions

/* Contains declaration (prototype) of printf() */

```
#include <stdio.h>

/* function declaration */
int sum(int, int);     /* int sum(int a, int b) */

main()
{
  int x = 2, y = 3;
  int su = sum(x,y);
  printf("Sum is %d\n", su);
}

/* function definition */
int sum (int i, int j){
   return i+j;
}
```

Declared before (first) function call

Point of (first) function call ✓

Defined after (first) function call

Even other file, What about printf()? Declared or defined?

72

72

## C basics

- Compile and running Comments
- Basic syntax
  - Comments
  - Variables
  - Functions
  - **Basic IO functions**
  - Expression
  - Statements
  - Preprocessing: # include, # define

73

73

## Basic I/O functions <stdio.h>

- Every program has a Standard Input: keyboard
- Every program has a Standard Output: screen
- Can be redirected in Unix `< inputFile` `> outputFile`

- **int printf (char *format, arg1, …. );**
  - Formats and prints arguments on standard output (screen or > outputFile)
  - **printf("This is a test %d \n", x)**
- **int scanf (char *format, arg1, …. );**
  - Formatted input from standard input (keyboard or < inputFile)
  - **scanf("%d %d", &x, &y)**

Others (for today?)
- **int getchar();**
  - Reads and returns the next char on standard input (keyboard or < inputFile)
- **int putchar(int c)**
  - Writes the character c on standard output (screen or > outputFile)

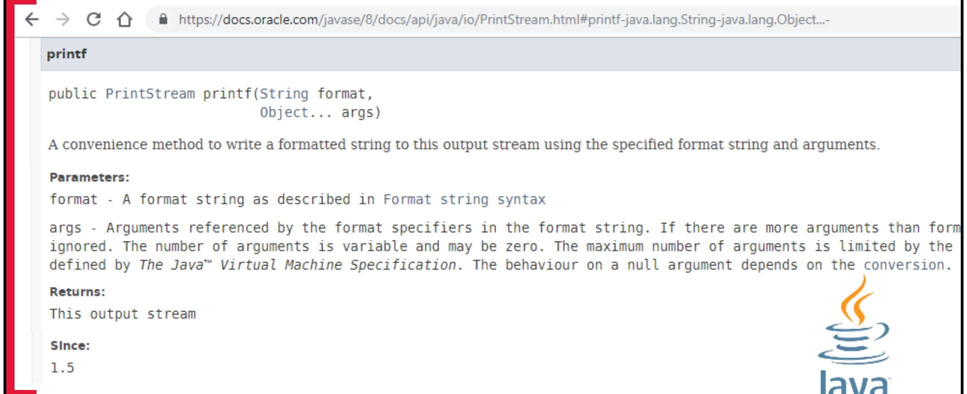
74

74

format string

/* conversion specification */

- `printf("This is day %d of Sep\n", x)`
  - Formats and prints arguments on standard output (screen or > outputFile)
  - added in Java 1.5 (year 2005)



https://docs.oracle.com/javase/8/docs/api/java/io/PrintStream.html#printf-java.lang.String-java.lang.Object...-

**printf**

```
public PrintStream printf(String format,
                          Object... args)
```

A convenience method to write a formatted string to this output stream using the specified format string and arguments.

**Parameters:**

format - A format string as described in Format string syntax

args - Arguments referenced by the format specifiers in the format string. If there are more arguments than form ignored. The number of arguments is variable and may be zero. The maximum number of arguments is limited by the defined by *The Java™ Virtual Machine Specification*. The behaviour on a null argument depends on the conversion.

**Returns:**

This output stream

**Since:**

1.5

```
System.out.printf("This is test %d today\n", x)
```

- also introduced in Java 1.5

```
String s = String.format("This is test %d today\n", x);
```

75

75

format string

/* conversion specification */

- `printf("This is day %d of Sep\n", x)`
  - Formats and prints arguments on standard output (screen or > outputFile)
  - Returns number of chars printed (often discarded)

- Format string contains: 1) regular chars 2) conversion specifications
  - **%d to be replaced/filled with an integer – decimal**
  - **%c to be replaced/filled with a character**
  - **%f to be replaced/filled with a floating point number (float, double)**
  - **%s to be replaced/filled with a "string" (array of chars)**
  - **…**

```
printf("Hello World\n");            Hello World
printf("%s\n", "Hello World");      Hello World
printf("%s World\n", "Hello");      Hello World

int a = 15; int b = 3;
printf("This is day " + a + " of Jan.\n");       This is day 15 of Jan.

printf("This is day " + a + ", week " + b + "of Jan.\n");
                                                 This is day 15, week 3 of Jan.
```

76

format string

/* conversion specification */

- `printf("This is day %d of Sep\n", x)`
  - Formats and prints arguments on standard output (screen or > outputFile)
  - Returns number of chars printed (often discarded)

- Format string contains: 1) regular chars 2) conversion specifications
  - **%d to be replaced/filled with an integer – decimal**
  - **%c to be replaced/filled with a character**
  - **%f to be replaced/filled with a floating point number (float, double)**
  - **%s to be replaced/filled with a "string" (array of chars)**
  - **…**

```
printf("Hello World\n");            Hello World
printf("%s\n", "Hello World");      Hello World
printf("%s World\n", "Hello");      Hello World
```

```
int a = 15; int b = 3;
printf("This is day %d of Jan.\n", a);        This is day 15 of Jan.

printf("This is day %d, week %d of Jan.\n", a, b);
                                                 This is day 15, week 3 of Jan.
```

replace in order

77

format string

`/* conversion specification */`

- `printf("This is day %d of Sep\n", x)`
  - Formats and prints arguments on standard output (screen or > outputFile)
  - Returns number of chars printed (often discarded)

- Format string contains: 1) regular chars 2) conversion specifications
  - **%d to be replaced/filled with an integer – decimal**
  - **%c to be replaced/filled with a character**
  - **%f to be replaced/filled with a floating point number (float, double)**
  - **%s to be replaced/filled with a "string" (array of chars)**
  - **…**

```
int a =2;
float b = 3.5;
printf("This is week %d of Sep. b's value is %f\n", a, b);
```

This is week 2 of Sep. variable b is 3.500000

```
printf("Adding %d to %.f gets %.3f\n", a, b, 5.5);
```

replace in order

Adding 2 to 3.500000 gets 5.500

78

format string

`/* conversion specification */`

- `printf("This is day %d of Sep\n", x)`
  - Formats and prints arguments on standard output (screen or > outputFile)
  - Returns number of chars printed (often discarded)

- Format string contains: 1) regular chars 2) conversion specifications
  - **%d to be replaced/filled with an integer – decimal**
  - **%c to be replaced/filled with a character**
  - **%f to be replaced/filled with a floating point number (float, double)**
  - **%s to be replaced/filled with a "string" (array of chars)**
  - **…**

Format should match
If does not match, error may occur

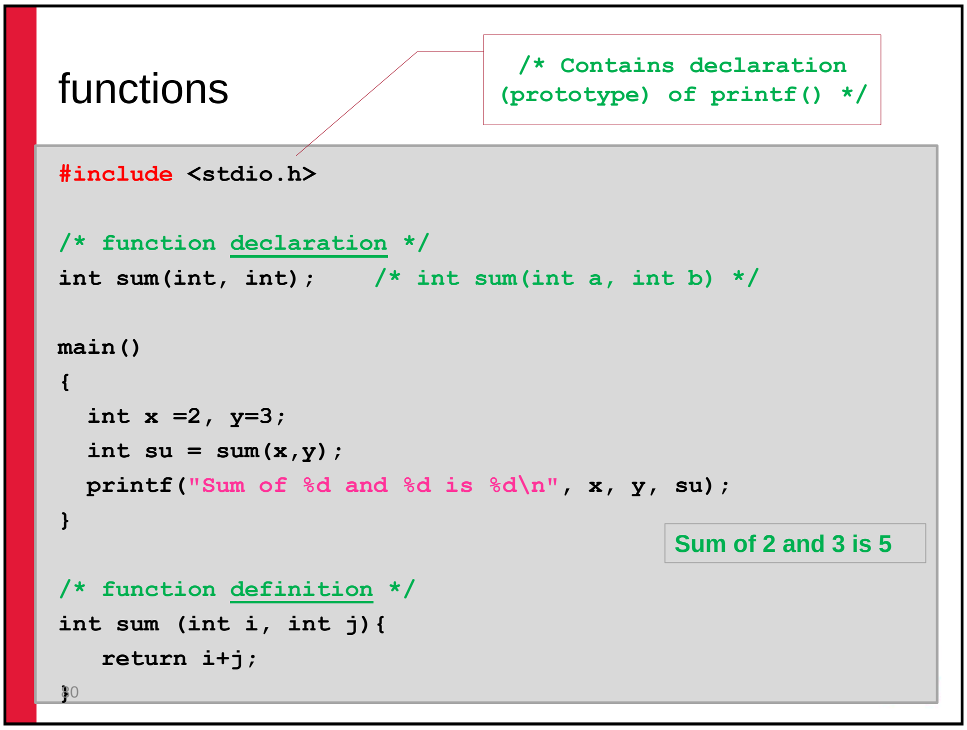
```
int a = 2;
printf("Value of a is %f\n", a);
```

Value of a is 0.000000

```
float b = 3.5;
printf("Value of b is %d\n", b);
```

Value of b is 2147343639

79



## functions

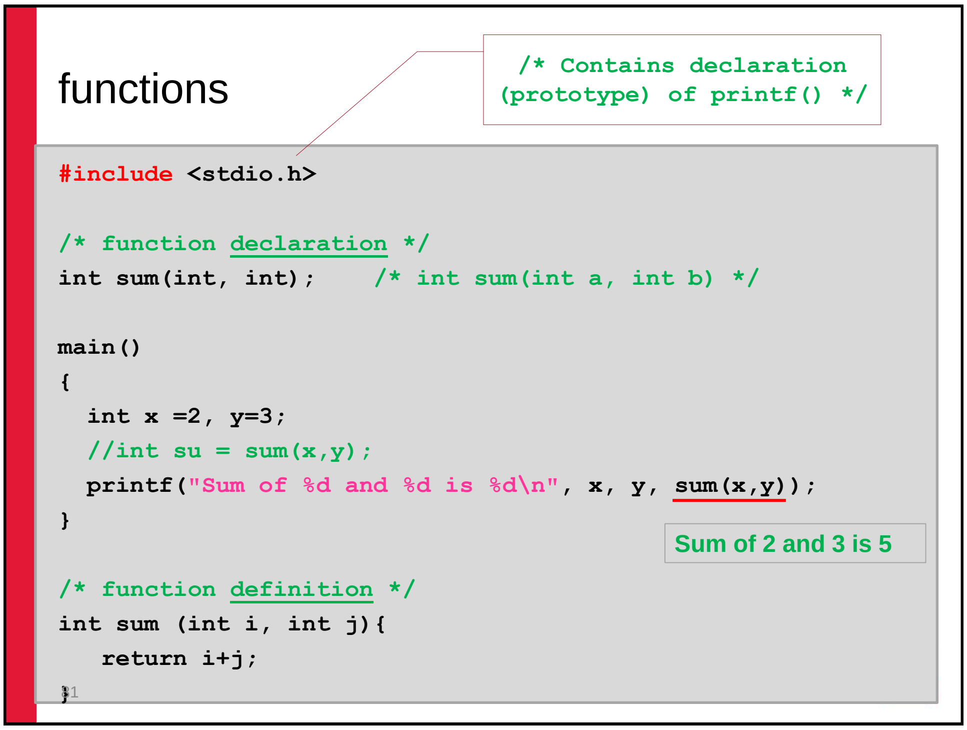
/* Contains declaration (prototype) of printf() */

```
#include <stdio.h>

/* function declaration */
int sum(int, int);    /* int sum(int a, int b) */

main()
{
  int x =2, y=3;
  int su = sum(x,y);
  printf("Sum of %d and %d is %d\n", x, y, su);
}

/* function definition */
int sum (int i, int j){
   return i+j;
}
```

Sum of 2 and 3 is 5

80

80



## functions

/* Contains declaration (prototype) of printf() */

```
#include <stdio.h>

/* function declaration */
int sum(int, int);    /* int sum(int a, int b) */

main()
{
  int x =2, y=3;
  //int su = sum(x,y);
  printf("Sum of %d and %d is %d\n", x, y, sum(x,y));
}

/* function definition */
int sum (int i, int j){
   return i+j;
}
```

Sum of 2 and 3 is 5

81

81

## scanf()

- `int x;`
- `scanf("%d", &x)`

  - opposite to `printf()`
  - Formatted input from standard input (keyboard or < inputFile)
  - Return number of successful scans/conversions (usually discarded) or EOF
  - Wait for standard input), then converts input to int, and assign value to `x`

- Format string contains: 1) regular chars 2) conversion specifications
  - %d **convert input to an integer – decimal**
  - %c **convert input to a character**
  - %f **convert input to a floating point number (float. %lf for double)**
  - %s **convert input to a "string"**

Format expectation from users.
If does not match, error may occur

- `&x` → memory address of `x`.
  - Details later. Take as it is for now.

82

82

```
#include <stdio.h>


main()
{
  int a; int b;
  printf("Please enter the number: " );

  scanf( "%d",  &a);   /* assign value to a */

  b = a * 2;
  printf("double of input %d is %d\n", a, b);
}
```

No \n ?

Expect a single int

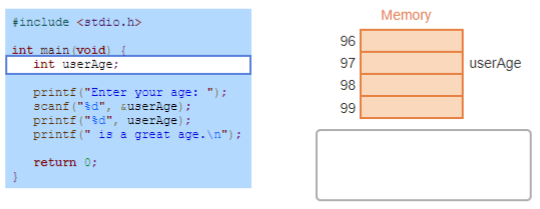
- `&a` → memory address of `a`. Details later. Take as it is for now.

```
red 314 % gcc scanf.c
red 315 % a.out
Please enter the number: 34
double of input 34 is 68
red 316 %
```

If not int, no guarantee

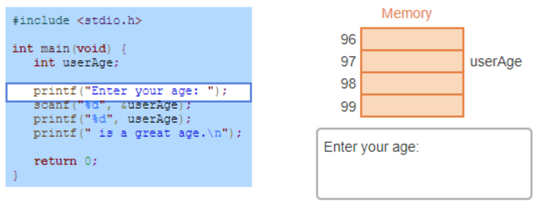
83

83



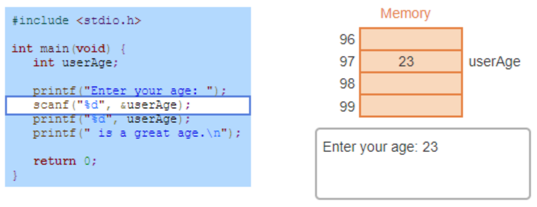
```
#include <stdio.h>

int main(void) {
   int userAge;

   printf("Enter your age: ");
   scanf("%d", &userAge);
   printf("%d", userAge);
   printf(" is a great age.\n");

   return 0;
}
```

Compiler allocates a memory location for *userAge*, in this case location 97.



```
#include <stdio.h>

int main(void) {
   int userAge;

   printf("Enter your age: ");
   scanf("%d", &userAge);
   printf("%d", userAge);
   printf(" is a great age.\n");

   return 0;
}
```

After printing, wait for user input



```
#include <stdio.h>

int main(void) {
   int userAge;

   printf("Enter your age: ");
   scanf("%d", &userAge);
   printf("%d", userAge);
   printf(" is a great age.\n");

   return 0;
}
```

User types 23, scanf() assigns *userAge* with 23, by writing into location 97 represented by *&userAge*

84

## Java version?

```
import java.util.Scanner;  // or use bufferedReader, Console
public class Scan {

 public static void main(String[] args) {

   Scanner scan = new Scanner(System.in);

   System.out.print("Please enter the number: ");

   int a = scan.nextInt();
   int b = a * 2;

   System.out.printf("double of input %d is %d\n",a, b);
  //System.out.println("double of input "+ a + " is " +  b);

 }
}
```

85

```
javac Scan.java
java Scan
```

85

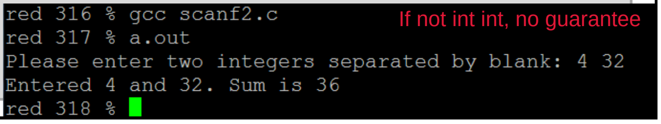
## Read two ints

Java?

```
#include <stdio.h>

int sum (int, int); /* function declaration (prototype) */

main()
{
  int a, b;
  printf("Please enter two integers separated by blank: " );

  scanf( "%d %d",  &a, &b);     /* assign value to a b  */

  printf("Entered %d and %d. Sum is %d\n", a, b, sum(a,b));
 }

 int sum (int i, int j)
{
  return i+j;
}
```

Expect two ints by blank

86



```
red 316 % gcc scanf2.c
red 317 % a.out
Please enter two integers separated by blank: 4 32
Entered 4 and 32. Sum is 36
red 318 %
```

If not int int, no guarantee

86

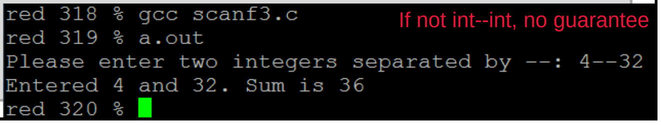
## Read two ints

Java?

```
#include <stdio.h>

int sum (int, int); /* function declaration (prototype) */

main()
{
  int a, b;
  printf("Please enter two integers separated by --: " );

  scanf( "%d--%d",  &a, &b);    /* assign value to a b  */

  printf("Entered %d and %d. Sum is %d\n", a, b, sum(a,b));
 }

 int sum (int i, int j)
{
  return i+j;
}
```

Expect two ints by - -

87



```
red 318 % gcc scanf3.c
red 319 % a.out
Please enter two integers separated by --: 4--32
Entered 4 and 32. Sum is 36
red 320 %
```

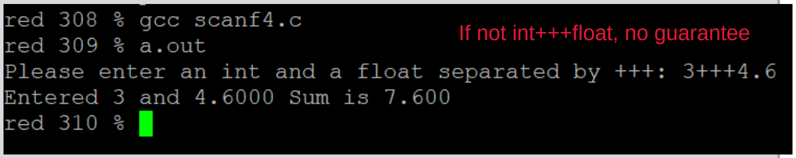
If not int--int, no guarantee

87

**Read two ints**

Java?

```
#include <stdio.h>



main()
{
  int a; float b;
  printf("Please enter an int and a float separated by +++: ");

  scanf( "%d+++%f",   &a, &b);     /* assign value to a b  */

  printf("Entered %d and %.4f  Sum is %.3f\n", a, b, a+b );
}
```



```
red 308 % gcc scanf4.c
red 309 % a.out
Please enter an int and a float separated by +++: 3+++4.6
Entered 3 and 4.6000 Sum is 7.600
red 310 %
```

If not int+++float, no guarantee

88

stop here

88

## getchar, putchar (Ch 1.5)

- **`int getchar(void)`**
  - To read one character at a time from the *standard input*
  - Returns the next input char each time it is called;
  - Returns EOF when it encounters end of file.
    - end of file;
      - Using < : end of input file
      - keyboard: Ctrl-d (Unix) or Ctrl-z (Windows).    *"Keyboard is a file"*
    - EOF: an **`int`** defined in <**`stdio.h`**>, value is -1.

- **`int putchar(int c)`**
  - Puts the character c on the *standard output*
  - Returns the character written (usually ignored);
  - Like **`printf("%c", c);`**

89

89

## getchar, putchar (Ch 1.5)

- **countChar.c**

```
#include <stdio.h> // define EOF

main(){
 int c;
 int count = 0;

 c = getchar();
 while(c != EOF) /* no end of file*/
 {
   count++; //include spaces and '\n'
   c = getchar(); /* read next */
 }
 printf("# of chars: %d\n",count);
}
```

```
red 309 % a.out
hello ↵
how are you ↵
i am good ↵
^D
red 310 %
# of chars: 28
```

Redirected from file

```
red 312 % cat greeting.txt
hello ↵
how are you ↵
i am good ↵
red 313%

red 314 % a.out < greeting.txt
 # of chars: 28

red 315 % a.out < greeting.txt > out.txt

red 316 % cat out.txt
# of chars: 28
```

redirect input from a file

redirect output to a file

YORK U UNIVERSITÉ UNIVERSITY

90

90

## getchar + putchar (Ch 1.5)

Redirected from file

- **copy.c**

```
#include <stdio.h>

main(){
 int c;
 c = getchar();
 while(c != EOF)
 {
   putchar(c);

   c = getchar(); /*read next*/
 }
}
```

```
red 309 % a.out
hello ↵
hello
how are you ↵
how are you
^D
red 310 %
```

Actually get/put chars line by line!

```
red 314 % a.out < greeting.txt
hello
how are you
i am good
see you

red 315 % a.out < greeting.txt > out.txt

red 316 % cat out.txt
hello
how are you
i am good
see you
```

redirect input from a file

redirect output to a file

ATTENTION

Not
hheellllloo
Buffer until '\n' or EOF

91

91

## getchar, putchar

- **copy + char, line counting**

```
#include <stdio.h>

main(){
 int c, cC, lC;
 cC = lC = 0;

 c = getchar();   /* read 1 char */
 while(c != EOF)
 {
   putchar(c);

   cC ++;

   if (c == '\n') /*a newline char*/
     lC ++;

   c = getchar(); /* read again */
 }
 printf("char:%d line:%d\n",cC,lC);
}
```

Compare directly

```
indigo 337 % a.out
hello   ↵
hello
how are you   ↵
how are you
i am good   ↵
i am good
^D
char:28  line:3
```

```
indigo 337 % cat greeting.txt
hello
how are you
i am good

indigo 338 % a.out < greeting.txt
hello
how are you
i am good
char:28  line:3
```

92

char 'a' 'b' compared directly. Strings not "a"== "b" ✗ Will elaborate soon.

92

## Summary and plan for next few lectures

- Intro. C basics  how to compile/run, basic structure
  - Variables
  - Functions: declaration vs. definition
  - Basic IO functions
    - `scanf & printf,`
    - `getchar & putchar` (self-study)

today

- Next few lectures:
  - Finishes C basic syntax
  - C data, type, operators (Ch 2)
  - C control flow (Ch 3) self-study, slides posted
- Lab0: watch the videos etc and determine your coding environment
  - Contact me if need help
- Lab1: will be released soon
  - due in about a week
  - I and/or TAs in zoom lab on Wed and Thursday to help out
    - 19:00~21:00  20:00~22:00
  - please start early, don't need to wait until lab hour

YORK UNIVERSITÉ UNIVERSITY

93

93