Computer systems

College of Saint Benedict & Saint John’s University

great insights of computer science[1]

Bacon, Leibniz, Boole, Turing, Shannon, & Morse

There are only **two nouns** that a computer has to deal with in order to represent “anything”: 0, 1.

[1] The great insights of computer science / CC BY-SA 3.0

- “anything”: there are some things computers cannot do — like determine if a program will ever finish.

great insights of computer science, cont’d

Turing

There are only **five verbs** that a computer has to perform in order to do “anything”:

1. move left one location;
2. move right one location;
3. read symbol at current location;
4. print 0 at current location;
5. print 1 at current location.

great insights of computer science, cont’d

Boehm and Jacopini

There are only **three grammar rules** needed to combine these verbs (into more complex ones) that are needed in order for a computer to do ”anything”:

1. *sequence*: first do this, then do that;
2. *selection*: IF such-and-such is the case, THEN do this, ELSE do that;
3. *repetition*: WHILE such-and-such is the case DO this.

great insights of computer science, cont’d

1. two nouns
2. five verbs
3. three grammar rules

<	move left one location
>	move right one location
0	print 0 at current location
1	print 1 at current location
[	if current location is 0, then go to instruction after matching]
]	go to matching [instruction

```
1>1>0>1>0<<<<[0>]1
```

- *sequence*: start at left-most instruction and progress a single instruction to the right
- *selection* and *repetition:* [...] provide both — repetition is just fancy selection

comparison

Java	C
object-oriented	procedural
interpreted	compiled
`String`	`char` array
condition (`boolean`)	condition (`int`)
garbage-collected	no memory management
references	pointers
exceptions	error codes

- in Java, everything is a method that is called on an object
- in C, everything is a function

- in Java, source code is compiled to byte code, which is then interpreted by Java VM
- in C, source code is compiled into binary machine code

- in Java, String is a class
- in C, a string is just an array of `char` values which ends with the `char '\0'`

- in Java, the Java VM takes care of deallocating memory used
- in C, any memory you allocate, you must also deallocate

hello, world

```c
/* file: helloworld.c */

#include <stdio.h>

int main() {
  printf("hello, world\n");
  return 0;
}
```

```
$ gcc -o helloworld helloworld.c
$ ./helloworld
hello, world
```

- The tradition of using the phrase ”Hello, world!” as a test message was influenced by an example program in the seminal book *The C Programming Language*

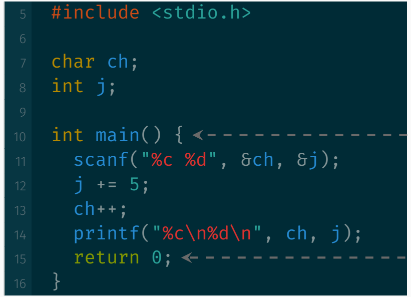
global variables

```c
// file: figure2-4.c
// Stan Warford
// A nonsense program to illustrate global variables

#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

```
$ gcc -o figure2-4 figure2-4.c
$ ./figure2-4
M 419
N
424
```

- What would you expect for input ’Z -3’?
- What would you expect for input ’9 a’?
- What would you expect for input ’~ 2147483643’?

program breakdown

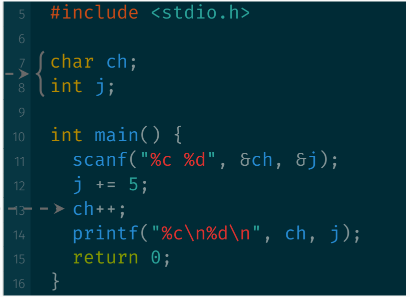


```c
#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

C programs ALWAYS start execution with the **`main`** function

returning from **`main`** ends the program

program breakdown

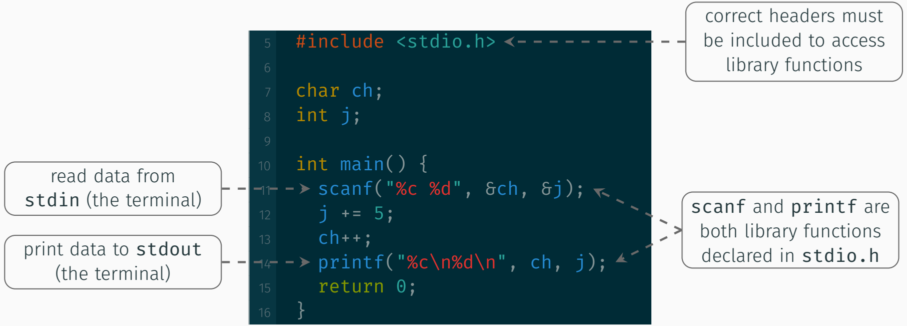


```c
#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

global variables are declared here — outside of any function

characters in C are treated internally like signed integers

program breakdown



```c
#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

correct headers must be included to access library functions

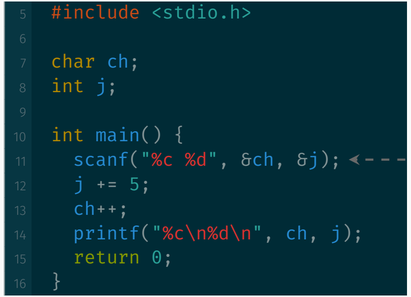
read data from `stdin` (the terminal)

print data to `stdout` (the terminal)

`scanf` and `printf` are both library functions declared in `stdio.h`

- C has no “built-in” functions; however, it does have a standard library that includes many useful utility functions.

program breakdown



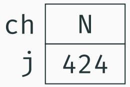
```c
#include <stdio.h>

char ch;
int j;

int main() {
  scanf("%c %d", &ch, &j);
  j += 5;
  ch++;
  printf("%c\n%d\n", ch, j);
  return 0;
}
```

& is the address of operator — `scanf` expects the address of the variables where the data will be stored

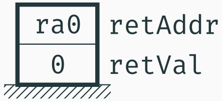
memory model — part i

global variables
declared outside of any function and remain in place throughout the execution of the entire program. they are stored at a fixed location in memory.

local variables
declared within a function and come into existence when the function is called and cease to exist when the function terminates. they are stored on the run-time stack.


ch N
j 424

(a) Fixed location.


ra0 retAddr
0 retVal

(b) Run-time stack.

- I will be using graphical notation consistent with that of the book.
- In this case, (a) and (b) represent the state of relevant memory for the previous program just before it terminates, i.e., in the process of executing line 15.
- How would the previous program behave had it declared `ch` and `j` as local variables instead of global variables?
- What would the memory model look like given the above?

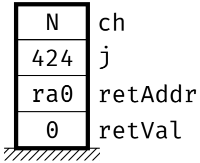
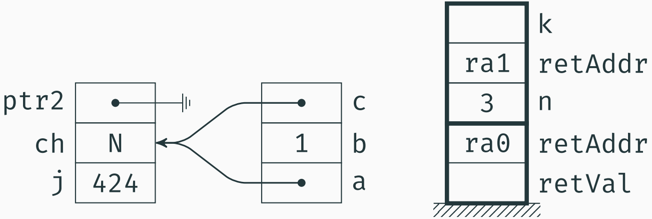

N ch
424 j
ra0 retAddr
0 retVal

Figure 2: Run-time stack.

FIXME — tikzstack testing



(a) Fixed location.

(b) Run-time stack.

conditions

- under what conditions will each of the following be execute?

```
if (x) {
  /* ??? */
}
if (x-y) {
  /* ??? */
}
if (x=y) {
  /* ??? */
}
```

- x != 0
- x != y
- y != 0