ECS 36A, April 12, 2023
Correction to Lecture 2

• Slide 6 said: (unsigned) \(-53 = 53\)

• WRONG!

• Here is \(-53\) represented as a 32-bit number:
  \[1111111111111111111111111001011\]

• But if you read it as unsigned, this represents a \textit{positive} number, here 4294967243

• Why? Because it is represented as \(2^{32} - 53\), not 53.
Problem with scanf.c

• Here is the problem code:
```c
if ((retval = scanf("%d %f", &xint, &yfloat)) == 2)
    printf("Got %d %f\n", xint, yfloat);
else if ((retval = scanf("%d xxx", &xint)) == 1)
    printf("Got %d\n", xint);
else
    printf("Didn't get anything\n");
```
• I typed “3 xxx” and got retval as 0, not 1 as I expected. Why?
  • First `scanf` read 3, but not a following float, so it returned 1
  • Do the else; `scanf` reads xxx, as 3 has already been read; it didn’t match %d, so that `scanf` read nothing, and returned 0.
  • So retval is 0 and this prints “Didn’t get anything\n” (from the last “else”)

April 12, 2023
ECS 36A, Spring Quarter 2023
3
Announcements

• Homework 1 and Extra Credit 1 are up and available on Canvas
  • The questions require some programs, and those are up too
  • Due date is April 24; you can turn it in late until April 29, but you will lose 10% for each day it is late (so if you turn it in on April 29, you will get only 50% credit)
• Gradescope is up for problems 1 and 2; it will be up for problems 3 and the extra credit by Friday
• No office hours today
Loops in C

• for loop
  • When you know where you will stop

• while loop

• do while loop
  • When termination depends on a condition being satisfied
for loop

for (initialization; condition; increment)

• Examples:
  for (i = 1; i < 10; i++)
  for ( ; j < 10; j += 3)
  for ( ; x < 10; )
  for ( ; ; )
while loop

while (condition)
   . . .

• Examples:
   while (i < 10)
      i = i + 1;
   while (j != 13)
      j = j - 1;
   while (1)
      ;

• Goes at top of loop; if condition is initially false, the loop is skipped
do while loop

do{

    ...  
}
while (condition)

• Examples:

  do{
      i = i + 1;
  } while (i != 13);
  do{
    
  } while (1)

• Goes at bottom of loop, which is always executed at least once