CS 354: The Other Room

Last Time:

Computer works:

CPU:
  - Fetch instruction
  - Decode (Figure out what instruction does)
  - Execute

Bits: 1's, 0's

Memory: simple byte-addressable

Towards C
  => Numbers (Integers)
    32-bits (4 bytes)
    C int (not a math integer)

=> Efficiency + Control
Variables

int x;

operators:
* / %
++

Assignment: = operator

x = expression;

x = 10;

y = x + 20

x = 10;

y = 20;

z = x + y;

address: 0xbff7c008

memory

4 bytes

C Instruction Pointer

C Abstract Machine
Comparisons

Boolean values: True, False
C represents: 1, 0 (non-zero)

(3 > 2) \Rightarrow 1

(\neg 10) \land (a < 20)

(\text{Boolean logic})

New operators: \text{logical operators}

binary $[$ AND $]$

$[$ OR $]$

unary $[$ NOT $]$

Truth Tables

\begin{array}{c|c|c|c|c|c}
X & Y & \text{AND} & \text{OR} \\
\hline
F & F & F & F \\
F & T & F & T \\
T & F & F & T \\
T & T & T & T \\
\end{array}
AND  $\equiv$  \[ \begin{cases} \end{cases} \\
OR  $\equiv$  \[ \begin{cases} \end{cases} \\
NOT  $\equiv$  \[ \begin{cases} \end{cases} 

Precedence: google C precedence

Make Decisions: if expression is true, exec. compound stmt

if ( expression )
  statement1;
  statement2;
  ...
}

if ( expression )
  statement1;
if (expr) ?

3 else ?

3

other:

switch stmt
(skip for now)

Short circuit:

if (x 3 3 y) ?

3

short circuit:
if x is false

=> (x 3 3 y) => false
don't evaluate y
Decisions: if

Repetition: Looping

while

for

\[ \text{while (expr)} \]
\[ s_1; \]
\[ s_2; \]
\[ \vdots \]

while expression is true

\Rightarrow \text{execute compound stmt}
Functions

int add(int x, int y) {
  return x + y;
}

int foo() {
  return 0;
}

calling a function:
  \[ \rightarrow \text{int a = 10;} \]
  \[ \rightarrow \text{int b = 11;} \]
  \[ \rightarrow \text{int c = add(a, b);} \]
  \[ \rightarrow \text{(c == 21) \Rightarrow true} \]
How to call a function and actually change variable value in caller

- Pointers (Addresses)
  - The Stack
  - Scope of variable

Pointers:

```
int x;

int* xp;
```

```
int main()
{
    x = 10;
    xp = &x;
    std::cout << *xp;  // prints the address of x
}
```

Diagram:

```
  +------------------+
  | memory           |
  +------------------+
    1000 1008
          |        |
          |        |
          |        |
          |        |
          +-------+
            xp
      +------+
      |      |
      |      |
      |      |
      |      |
      +-------+
        10
        x
        
pointer: address

address of x
```
```c
int x;
x = 10;
int *foo;

foo = &x;
*foo = 11;
*foo = 12;
```

Pointers, Stack, Scope
```c
int x = 10;

int* xp;

xp = &x;

new operator:
=> *

dereference

int z = (*xp) + 1;

reads
int

= 2 * *xp + 1;

= 2 * xp + 1;
```
Stack: C Runtime Stack

int main() {
    int x;
    int y;
    int* foo;
    ...
}

Stack allocated variables

allocated when function is entered

deadallocated when function returns

"automatic" variables
void inc (int value) {
    value = value + 1;
}

int main () {
    int x = 10;
    inc (x);
    return 0;
}
variable

Scope

\[\Rightarrow\text{ when name valid?}\]
void inc(int* x)
    {*x = *x + 1}
3

main()
    int f = 10;
    inc(3+f);
    Show(f); = 11
Array / Pointer "equivalence"

```c
int a[3];

int* xp;

xp = 3 * a[0];

*xp = 100;

xp = xp + 1;

pointer arith

*xp = 101;
```
\( a \in \mathbb{R} \)

\[ \iff * (a + 2) \]
More Types:

character

char x;

1 byte

numeric value

0,...,255

ASCII
Array

array of integers

int a[10];

new operator: [ ]

a[3] = 10;

weak type in C:

a[23] <= will "work"

10 ints (40 bytes)
string: new type
char array + a little bit