Types

CMPSC 461
Programming Language Concepts
Penn State University
Fall 2016

Values and Types

Values are type-less in hardware

0100 0000 0101 1000 0000 0000 0000 0000

Floating point number: 3.375
32-bit integer : 1,079,508,992

Two 16-bit integer: 16472 and 0
Four ASCII characters: @ X NUL NUL

Operations expect certain values

- `add` take integers, `fadd` take floats

How to ensure operators get right ***type*** of values?

What Can Go Wrong?

Operation on wrong values will produce garbage

- float addition on two integers
- assign a string to a float
- pass an int to a function expecting a string

All of these errors are ***type errors***

Type

An abstraction of a set of values, and legal operations on these values

- `int`: -2^{31} to $2^{31}-1$, with operations +,-,*,/, …

Meaning of type

- Denotational: a set of value
- Constructive: set of primitive types, type constructors
- Abstraction: an interface (set of operations)

Why Types?

- Identify/Prevent type errors
- Program organization/abstraction
- Documentation
- Support optimization

Kinds of Types

Primitive

Constructed

- Products
- Unions
- Arrays
- Lists

User-Defined

Type System

A method or specification for associating types with variables, expressions, etc.

Type equivalence: are two types equivalent

Type compatibility: can int be used as float?

Type safety: absence of type errors

Type Checking

Strongly typed: all type errors caught by type checking

Weakly typed: type checking may miss type errors

Static typing: type checking happens at compile time

Dynamic typing: type checking happens at run time

Type inference: the type-checker infers types for variables

```
union U {int a; float p} u;
float x = 1.0;
u.a = 1;
x = x + u.p;
```

C is weakly typed, at compile time

Basic Types

Numeric

- byte, integers, floats

Booleans

Characters

Data types available on contemporary machines

Integers

Length depends on language and compiler

Representation: two's complement format

n	0	binary representation of n
15	0	000 0000 0000 0000 0000 0000 0000 1111
-n	1	flip all bits of binary representation of n, and add 1
-5	1	111 1111 1111 1111 1111 1111 1111 1011
15-5?	0	000 0000 0000 0000 0000 0000 0000 1010

Just add binaries of 15 and -5

Floats

Single precision (float): 32 bits

Double precision (double): 64bits

Due to the limited space, floats are *estimations* of the number they present

```
float z = 1.345+1.123;
printf("%d\n", z==2.468);
```

Floats

IEEE 754 Standard

Representation of floating point numbers in IEEE 754 standard:

single precision

	1	8	23
sign	**S**	**E**	**M**

exponent:
bias 127
binary integer

mantissa:
sign + magnitude, normalized binary significand w/ hidden integer bit: 1.M

actual exponent is
$e = E - 127$

$0 < E < 255$

$$N = (-1)^{S}\ 2^{E-127}\ (1.M)$$

Boolean

Most languages: true or false

C: 0 means false, all other values mean true

In most implementations, a boolean value occupies more than one bit in memory (word is the basic unit of load/store)

Character

All languages support ASCII code (7-bit)

Most modern language support Unicode (e.g., Java `char` uses UTF-16, a 16-bit char set)

Enumeration Types

Provide names to a sequence of integral values

C/C++

```
enum day {Monday, Tuesday, Wednesday,
          Thursday, Friday, Saturday, Sunday};
enum day myDay = Friday;
```

Enumeration type improves readability

Enumeration Types

Provide names to a sequence of integral values

C/C++

```
enum day {Monday, Tuesday, Wednesday,
          Thursday, Friday, Saturday, Sunday};
enum day myDay = Friday;
```

Java

```
enum Day {Monday, Tuesday, Wednesday,
          Thursday, Friday, Saturday, Sunday};
for (Day d: Day.values()) {
    System.out.println(d);
}
```

Records and Structures

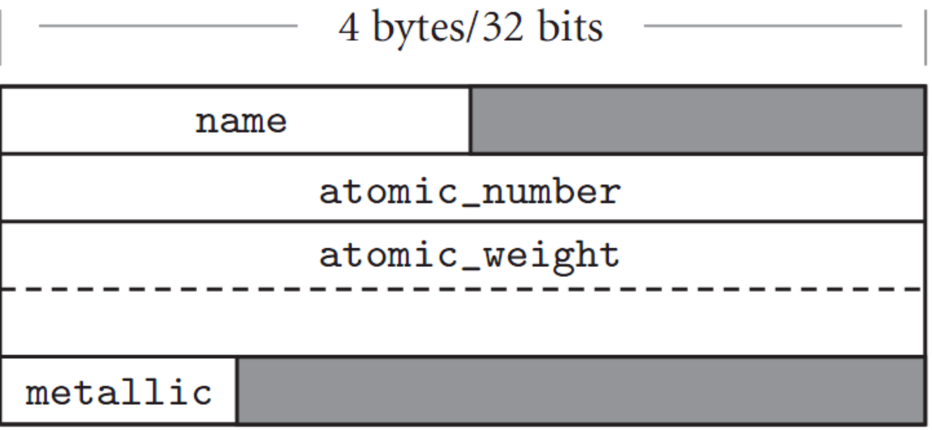
Usually laid out contiguously

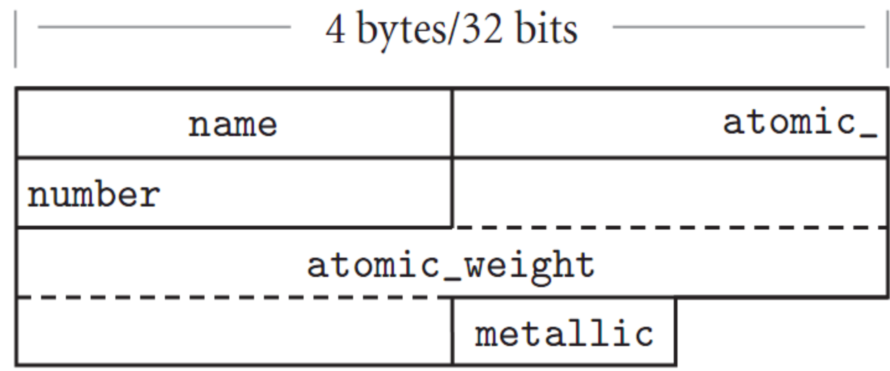
Possible holes for alignment

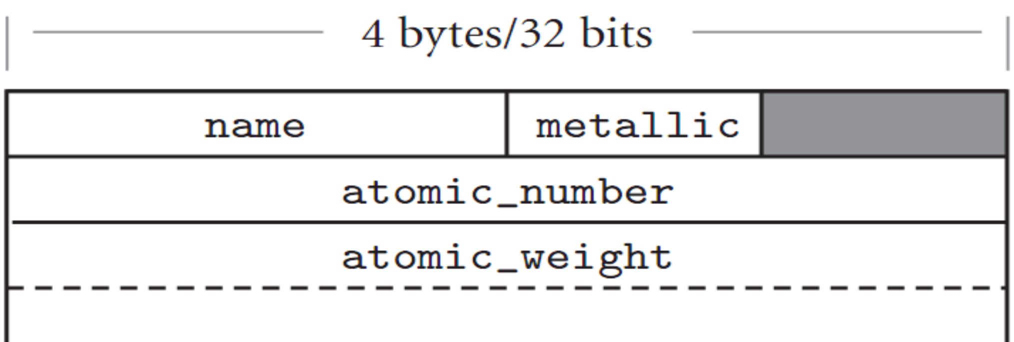
```
struct id {
  int i;
  double d;};
struct id x;
x.i, x.d
```

Compilers may re-arrange fields to minimize holes

```
struct element {
    char name[2];
    int atomic_number;
    double atomic_weight;
    _Bool metallic;}
```







Memory Layout