C Types and Variables

How C keeps track of data

Problem

- All computer data consists of strings of 0's and 1's
- How does the computer know what the string means?

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	12	12	11	10	9	8	7	6	5	4	3	2	1	0
1	1	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Float: -2.75

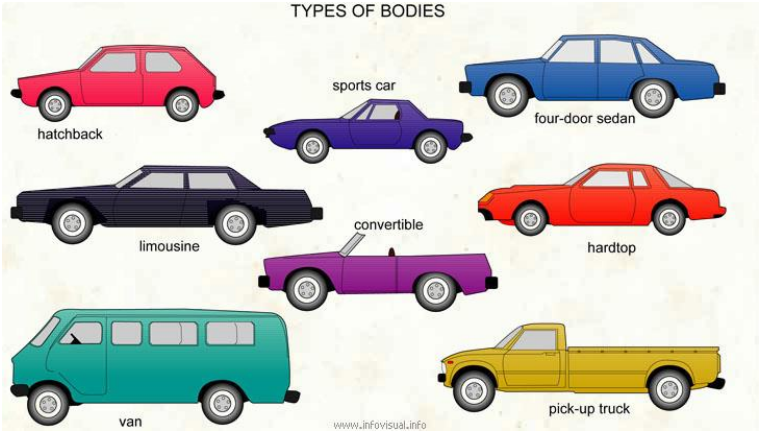
Char: ␣ØØØØØØ

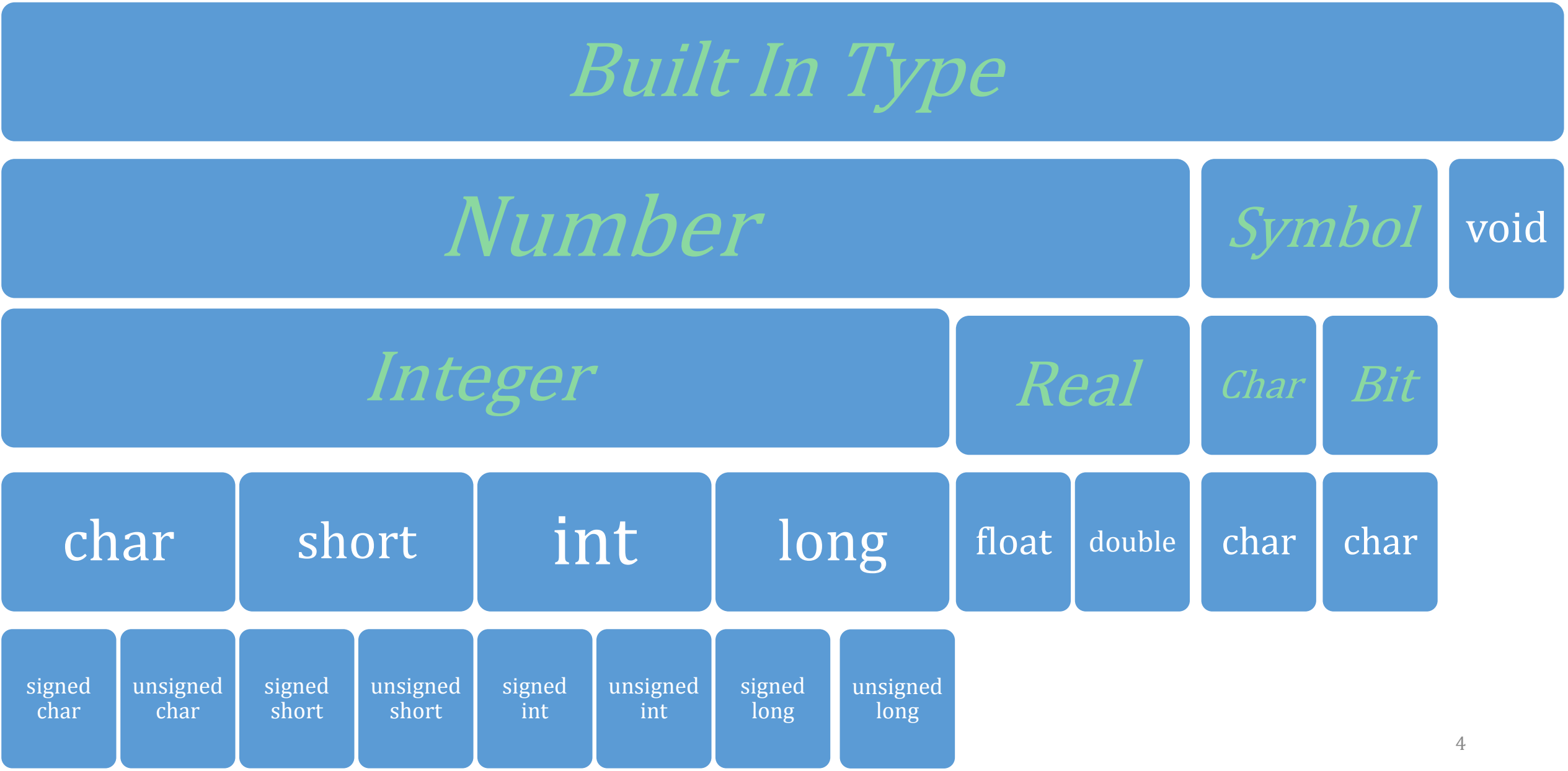
Unsigned Int: 3,244,371,200

Int: -1,070,596,096

Solution

- Tag each piece of data / location in memory with a “type”
- Type tells compiler how many bits to use
- Type tells compiler how to interpret those bits
- Enables automatic conversion from one type to another
- Enables compiler to check to make sure data is used correctly




Built In Type
Number
Symbol
void
Integer
Real
Char
Bit
char
short
int
long
float
double
char
char
signed char
unsigned char
signed short
unsigned short
signed int
unsigned int
signed long
unsigned long

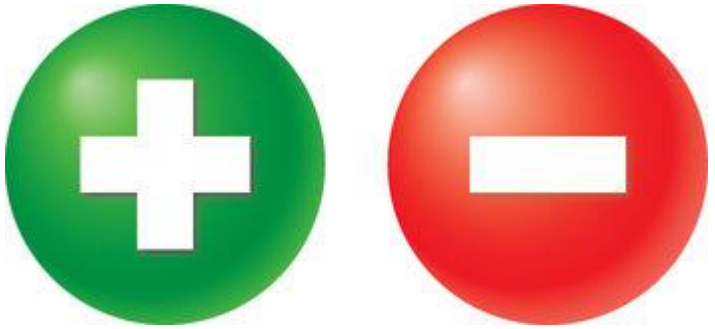
Q: Why four flavors of Integers?

- A: Allows programmer to choose size

Type	# Bits[1]	Min	Max	U Max
char	8	-128	127	255
short	16	-32,768	32,767	65,535
int	32	$\sim -2.15 \times 10^9$	$\sim 2.15 \times 10^9$	$\sim 4.3 \times 10^9$
long	64	$\sim -9 \times 10^{18}$	$\sim 9 \times 10^{18}$	$\sim 18.5 \times 10^{18}$

[1]Number of bits may be different on different machines!

Q: Why signed vs. unsigned?



- A: Unsigned holds 2x value
- A: Some data can never go negative – e.g. “width”
  - Compiler checks

- Signed is the default (what you get if you ask for “int”)
- Avoid unsigned data… can get you into trouble!

Q: What is char?

- "char" is ambiguous… it can be used for small integers, characters, or bit flags
- It's up to the programmer to determine how to use char data!

C Variables

- A variable is a named piece of data
- Variables in C have…
  - A name (specified by the programmer)
  - A value (may be unassigned/unknown)
  - A location in memory (determined by the compiler)
  - A type (size and interpretation)
  - … (more to come… scope/ storage class/ etc.)
- Variables must be declared before they are used!

Variable Declaration Statement

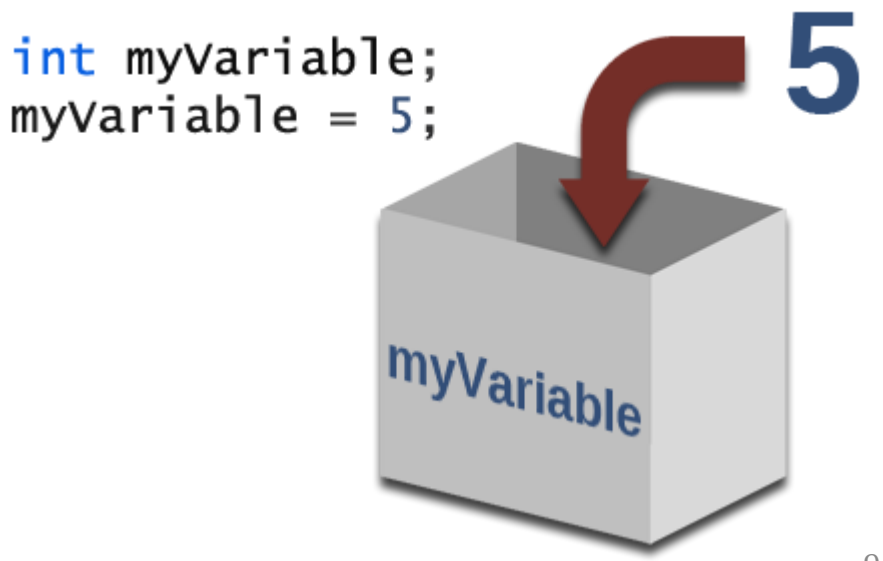
<type> <name>;

<type> : One of the built-in types or a derived type

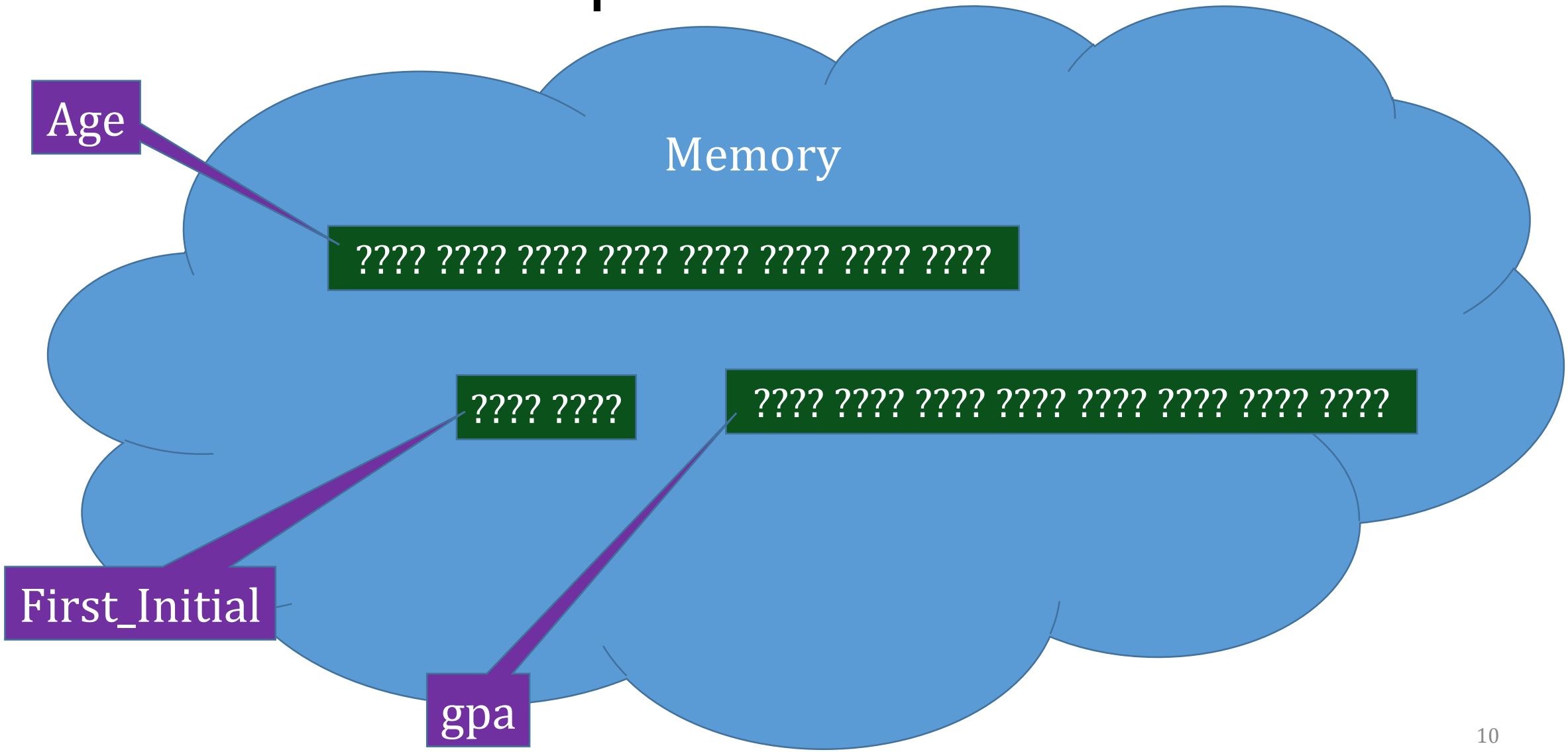
<name> : Any valid identifier

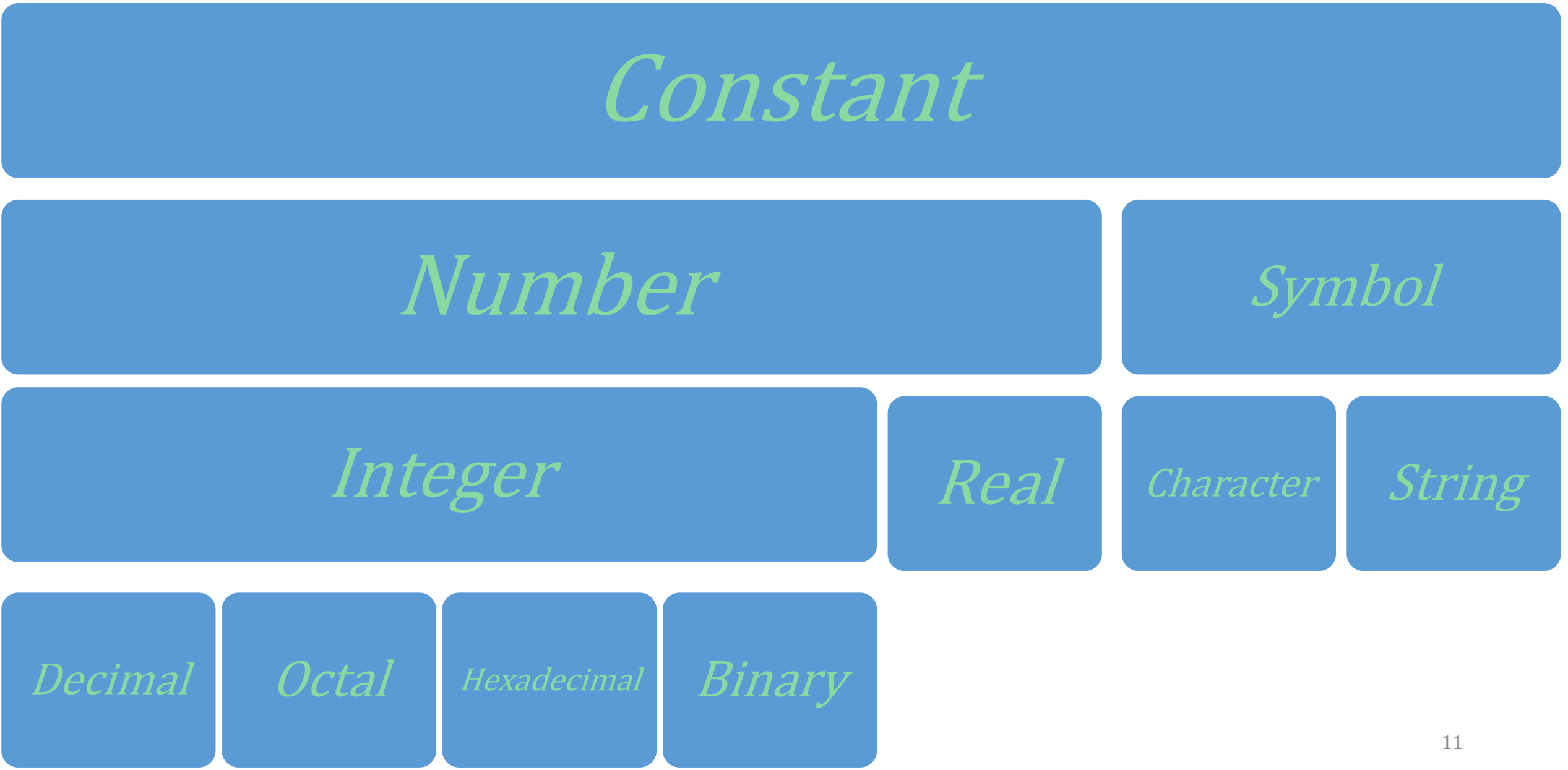
Examples:

```
int age;
char First_Initial;
float gpa;
```



Variable Concept




Constant
Number
Symbol
Integer
Real
Character
String
Decimal
Octal
Hexadecimal
Binary

Specifying Constants

Flavor	Format	Example
Decimal	<0-9>*	379
Octal	0<0-7>*	0573
Hexadecimal	x<0-F>*	x017B
Binary	0b<0-1>*	0b01 0111 1011
Real	<0-9>*.<0-9>*	379.0
	<0-9>*.<0-9>*e<0-9>*	3.79e2
Character	‘<letter>’	‘a’
String	“<letter>*”	“hi there”

Variable Declaration Statement (w/ init.)

<type> <name> = <const>;

<type> : One of the built-in types or a derived type
<name> : Any valid identifier
<const> : Constant specification (type should match)

Examples:

```
int age = 17;
char First_Initial = 'a';
float gpa = 3.985;
```

Variables in Functions

- Variable Declaration within function (usually at top)
- "Automatic" storage class by default
- New variable each time function starts
- Initialized when function starts (if initializer specified)
- No longer available when function ends

Parameters: Special "Variables"

- Like Variables, have
  - Name
  - Type
  - Value
  - Location in Memory
- Created when a function is invoked
- Initialized with argument value (as specified in the invocation)
- Destroyed after function returns
  - Implication: changing a parameter value DOES NOT change the argument!

Next: What can we do with Variables?

Resources

- Programming in C, Chapter 3
- WikiPedia: Type Theory (https://en.wikipedia.org/wiki/Type_theory)
- WikiPedia: C Data Types (https://en.wikipedia.org/wiki/C_data_types)