Fundamental Data Types

Data Types

- Programming language construct used to define a type of data objects, in terms of a range of
  - permissible values
  - a set of eligible operations.
- Data from different data types is stored and processed differently.
  - memory representation of data types is platform dependant.
  - memory representation and the quantity of bytes given for a data item determine the range of values that can be processed.

Data Types

- Numeric and text data
- Four fundamental data types:
  - int - for integer numbers;
  - char - for single characters;
  - float - for floating-point numbers;
  - double - for double-precision floating-point numbers.
- Qualifiers can be used to modify the range of data values for a basic type:
  - short
  - long
  - signed
  - unsigned

Integer Numbers Data Types

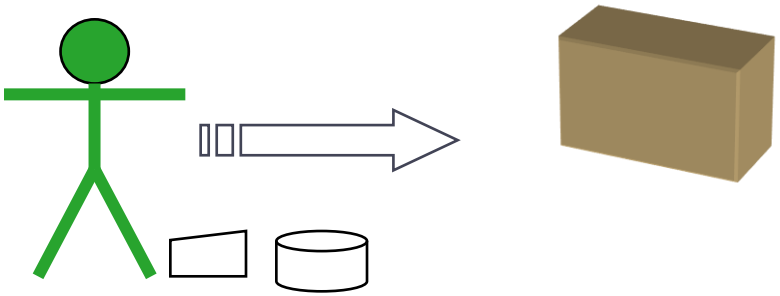
Data Type	*Bytes*	*Minimum Value*	*Maximum Value*
short	2	- 32768	+ 32767
int	machine word	- 32768	+ 32767
long	4	- 2147483648	+ 2147483647
unsigned	machine word	0	+ 65535
unsigned long	4	0	+ 4294967295

Real Numbers Data Types

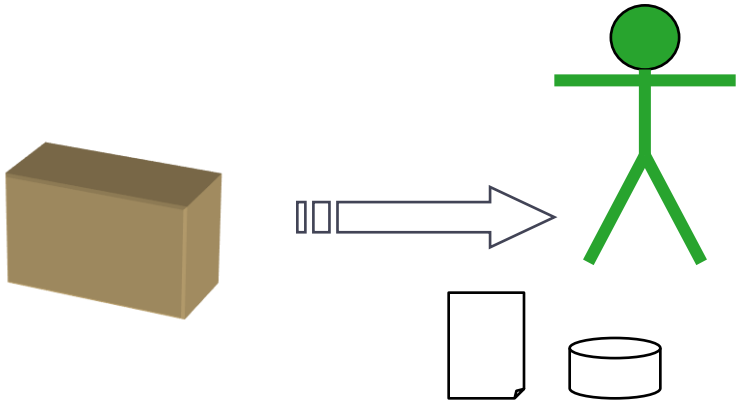
Floating Point	Maximum Exponent	Digits of Precision	Maximum Value
float	38	6	3.402823e+38
double	308	15	1.797693e+308
long double	4932	19	1.189731e+4932

Input and Output of Numeric Data

- Functions
  - scanf()



  - printf()



- Format
  - scanf(what, where_in_RAM);
  - printf(what, where_in_RAM);

Format

- What
  - place holder
  - format specifier
    - %<letter>
- Where_in_RAM
  - address for input - &<identifier>
  - name for output - <identifier>

Input and Output of Integer Data

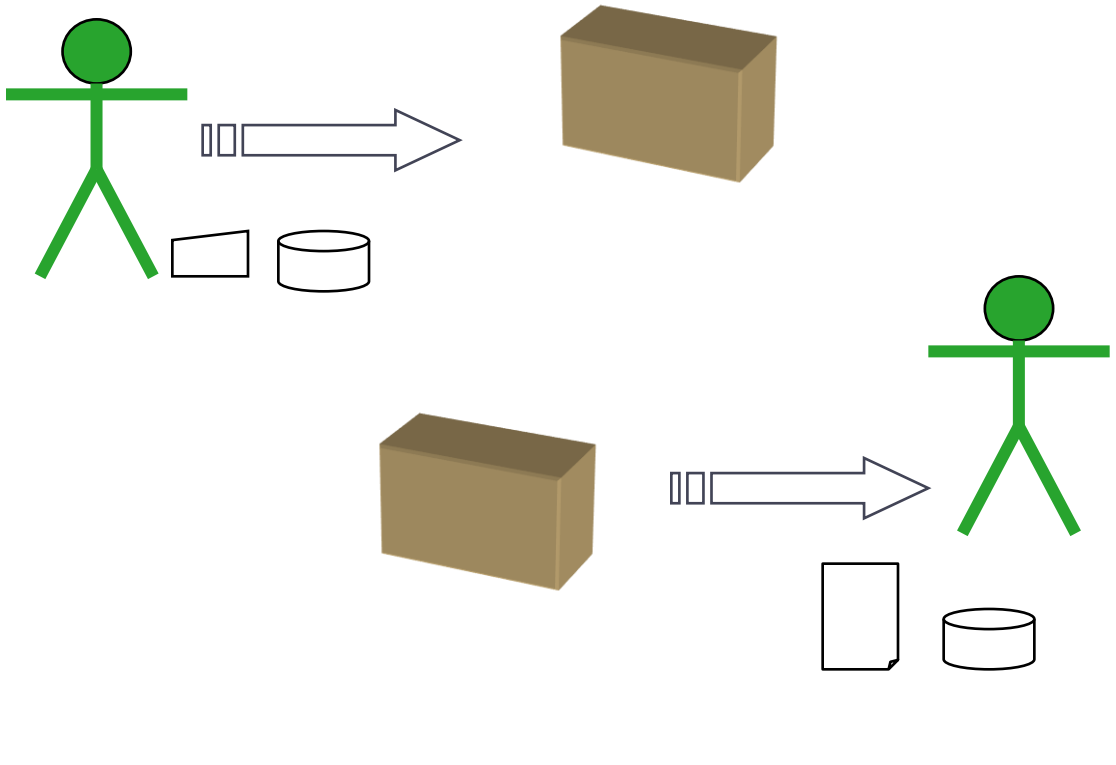
Variable Type	Output Type	Specifiers
short, int	int	%i, %d
int	short	%hi, %hd
long	long	%li, %ld
int	unsigned int	%u
int	unsigned short	%hu
long	unsigned long	%lu
int	octal int	%o
int	hexadecimal int	%x
int	decimal int	%d

Input and Output of Real Data

Variable Type	Output Type	Specifiers
float	double	%f, %e, %E, %g, %G
double	double	%f, %e, %E, %g, %G
long double	long double	%Lf, %Le, %LE, %Lg, %LG

Input and Output of Char Data

- More functions
  - getchar()
  - getch()
  - getche()
  - gets()
  - putchar()



Input and Output

```
/* Volume and surface area of a planet */
#include <stdio.h>
#define PI 3.14159265
main()
{
    float radius, diameter;      // dimensions
    double volume, area;         // size
    char planet[10];

    printf("Please, enter the name of the planet: ");
    gets(planet);
    printf("The value of the diameter: ");
    scanf("%f", &diameter);
    radius = diameter / 2;
    volume = 4/3 * PI * radius * radius * radius;
    area = 4 * PI * radius * radius;
    printf("The volume of planet %s is %f km3\n"
    "The %s surface area is %f km2\n", planet, volume, planet, area);
}
```

Data Processing

```
/* Volume and surface area of a planet */
#include <stdio.h>
#define PI 3.14159265
main()
{
    float radius, diameter;      // dimensions
    double volume, area;         // size
    char planet[10];

    printf("Please, enter the name of the planet: ");
    gets(planet);
    printf("The value of the diameter: ");
    scanf("%f", &diameter);
    radius = diameter / 2;
    volume = 4/3 * PI * radius * radius * radius;
    area = 4 * PI * radius * radius;
    printf("The volume of planet %s is %f km3\n"
    "The %s surface area is %f km2\n", planet, volume, planet, area);
}
```

Expressions

- Components
  - operands
    - one - unary
    - two - binary
  - operators
  - result
- Types
  - Arithmetic
  - Logical
  - String

Unary Arithmetic Operations

Operation	Operator	Examples
Positive	+	+5
Negative	-	-b
Increment	++	i++, ++i
Decrement	--	i--, --i
	() []	a[5]

Binary Arithmetic Operations

Operation	**Operator**	Examples
Addition	+	a + b
Subtraction	-	a - b
Multiplication	*	a * b
Division	/	a / b
Modulus	%	a % b
Assignments	= += -= *= /= %=	a = b a += b a -= b a *= b a /= b a %= b

Priority of Operators

- The order in which the operators are performed
- Precedence
  - levels of priority for the operators - standard algebraic precedence:
    - operators within parentheses are always evaluated first;
    - if the parentheses are nested, the innermost operators have the highest priority;
    - unary operators are evaluated before binary operators;
    - *, / and % have higher priority than + and -.
- Associativity
  - when the expression contains more than one operators of same priority, the operands are grouped (or associated) with the operators in a specific order

Priority of Operators

Precedence	Operators	Associativity
1	( )	Innermost first
2	++ (postfix) --_(postfix)	left to right
3	+ (unary) - (unary) ++ (prefix) -- (prefix)	right to left
4	(binary) * / %	left to right
5	(binary) + -	left to right
6	(assignment) = += -= *= /= etc.	right to left

Logical Expressions

- Result is 0 or 1
  - true or false
- Implementation
  - Relational operators
  - Logical operators

Relational Operators

Operator	Meaning	Examples
<	is less than	a < b
<=	is less than or equal to	a <= b
>	is greater than	a > b
>=	is greater than or equal to	a >= b
==	is equal to	a == b
!=	is not equal to	a != b

Logical Operators

Operator	Meaning
!	not
&&	and
||	or

A	B	!A	A && B	A || B
False	False	True	False	False
False	True	True	False	True
True	False	False	False	True
True	True	False	True	True

“true” - 1

“false” - 0

Priority of Logical Operators

- 1 < <= > >=
- 2 == !=
- 3 &&
- 4 ||