ALevel CS C16 Data representation

Data types

Build-in data types: programming language defines **the range of possible values** and **the operations available for manipulating**

User-defined data types: the programmer has included the definition in the program.

1. create a new datatype
2. allow data types not available in a programming language

```
TYPE <TypeIdentifier>
   DECLARE <field identifier> : <data type>
ENDTYPE
DECLARE <variable identifier> : <record type>
```

Non-composite data types: does **not involve a reference** to another data type.

Enumerated data type: an example of a **user-defined non-composite** data type.

```
TYPE
TDirections = (North, East, South, West)
DECLARE Direction1 : TDirections
Direction1 ← North
```

Composite user-defined data types: **reference to at least one other type**.

record data type and class

```
DECLARE
TYPE <type_name> IS RECORD
(
   <column1> <datatype>,
   ...
   ...
)
```

Point data type: **reference** to a memory location,

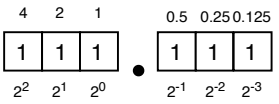
```
TYPE TIntegerPointer ← ^Integer
DECLARE MyIntegerPointer : TIntegerPointer
DECLARE Number1, Number2 : INTEGER
Number1 ← 100
MyIntegerPointer ← @Number1
Number2 ← MyIntegerPointer^ * 2
```

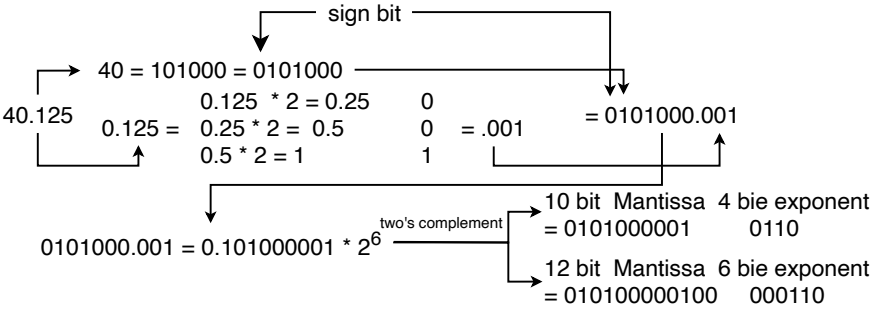
Set data type:

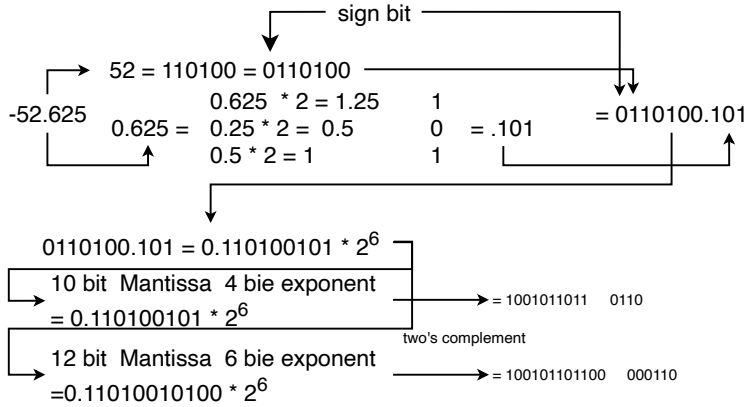
a collection of data items that lacks any structure; contains **no duplicates** and has a number of defined operations that can be performed on it

Convert real number to binary number

Size of mantissa decreased - precision is reduced
Size of exponent increased - range is increased







Precision

real number in program language precision error :
1. 0.1+0.1+0.1 != 0.3
2. round(2.5) = 2

reason:

1. real number cannot be represented exactly in binary
2. the manipulate will increase the precision.
3. difference after the calculation is significant enough to be seen

File Organisation

Binary file: a file designed for storing data to be used by a computer program

Record: a **collection of fields** containing data values

text file: contains data stored according to a character code of the type

binary file: stores data in its internal representation

Serial files: contains records that have **not been organised in any defined order**. A database that stores values as a series of items, one after the other

Sequential files: records that are **ordered**

Direct-access files: only that the access can be to any record in the file **without sequential reading** of the file

1. use a sequential search to look for a vacant address following the calculated one
2. keep a number of overflow addresses at the end of the file
3. have a linked list accessible from each address.

Convert binary number to real number

Mantissa 00101000 0.0101000 0010.1000 2.5
Exponent 00000011 3

-1 1's c shift 1
Mantissa 1001 1.001 ~ 1.000 ~ 0.111 ~ 01.11 ~ -1.75
Exponent 0001 1

Mantissa 10110000 -1 1's complement shift 2 places
1.0110000 ~ 1.0101111 ~ 0.1010000 ~ 0.001010000 ~ -5/32

Exponent 1110 1101 ~ 0010 ~ -2

Normalization

For a positive number, the bits in the mantissa are shifted left until the most significant bits are 0 followed by 1. For each shift left the value of the exponent is reduced by 1. The same process of shifting is used for a negative number until the most significant bits are 1 followed by 0. In this case, no attention is paid to the fact that bits are falling off the most significant end of the mantissa.

Mantissa 0000000111 0.000000111 ~ 0.111 (6)
Exponent 100111 100111 ~ 100110 ~ 011001 ~ -25 - 6
= -31~ 0111101~0111110~100001

Mantissa 110100 1.10100 ~ 1.0100 (1)
Exponent 0010 0010 ~ 2 - 1 = 1 ~ 0001

problem not normalization:

1. **precision lost**
2. Redundant leading zeros in the mantissa
3. **Bits lost off right hand end / least significant end**
4. **Multiple representations of a single number**

Overflow

Overflow - Overflow occurs when calculations produce results exceeding the capacity of the result.

Underflow - A calculation resulting in a number so small that the negative number used for the exponent is beyond the number of bits used for exponents is called underflow

Description	BinaryCode	Denary equivalent
Largest positive value	0 111 0111	$0.875 * 2^7 = 112$
Smallest positive value	0 001 1000	$0.125 * 2^{-8} = 1/2048$
Smallest magnitude negative value	1 111 1000	$-0.125 * 2^{-8} = -1/2048$
Largest magnitude negative value	1 000 0111	$-1 * 2^7 = -128$

Problem

1. the conversion of a real value in denary to a binary representation almost guarantees a degree of approximation.
2. There is also a restriction of the number of bits used to store the mantissa.