

1

2

3

1

2

1

前回の質問の解答

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.7

2進数を直接16進数に直すには?

➡「直接」とまではいかないけど...

- 2進数を右から4桁ずつに区切る
 - ➡ $(16)_{10} = (2^4)_{10} = (10000)_2$ なので、2進数4桁分(0000~1111)は、10進数で0~15になり、これは16進数だと0~F、つまり1桁分
- 4桁をそれぞれ10進数に直す
2. の10進数を16進数に直す(これが16進数に直した結果)

Ex. 2進数 $(0111011010)_2$ を16進数に直す

- 2進数を右から4桁ずつに区切る: $0111011010 \rightarrow 01 \mid 1101 \mid 1010$
- 4桁をそれぞれ10進数に直す: $(01)_2 \mid (1101)_2 \mid (1010)_2 = (1)_{10} \mid (13)_{10} \mid (10)_{10}$
2. の10進数を16進数に直す: $(1)_{10} \mid (13)_{10} \mid (10)_{10} = (1)_{16} \mid (D)_{16} \mid (A)_{16}$

$(0111011010)_2 = (1DA)_{16}$

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.8

コンピュータでの2進数の扱いは...

➡桁数を気にすることが一番重要!

➡コンピュータで扱うことができる数の範囲には限りがあり

コンピュータの一種の性能
= 無限に高性能なコンピュータは存在しない
= どんな計算をしても、範囲が広がることはない
(コンピュータの性能を超える計算はできない)

2進数を扱うときは、常に指定された桁数に合っているかを確認するくせをつけよう!

- X進数を2進数に直した(2進数が出てきた瞬間に桁数あわせ)
- 足し算・引き算・かけ算をした
- 0と1を反転させた
- etc.

何かした後は、そのつど、常に確認!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.9

追加の練習問題

➡授業の資料のページに追加の練習問題

➡後日、解答も掲載

1度やった問題も繰り返しやって、やり方をしっかり身に着けよう!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.10

Question!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.11

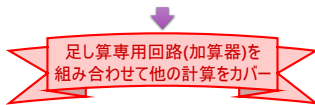
前回の復習

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.12

負の数の表現[1](p. 9)

- コンピュータでの計算は、全て足し算
 - コンピュータでの計算は、電気回路で実行
 - 電気回路: 電気が通る線を組み合わせて、様々な処理をするためのもの (コンピュータを構成する最も基本的な部品)
 - 足し算, 引き算, かけ算, 割り算をするには、それぞれのために専用の回路が必要
 - 足し算専用回路, 引き算専用回路, かけ算専用回路, 割り算専用回路

経済的に良くない



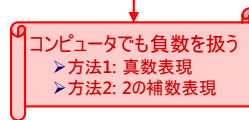
Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

13

負の数の表現[2](p. 9)

- 足し算の組み合わせで他の計算も行う

- 引き算: 「 $a-b$ 」を、「 $a+(-b)$ 」(b を負数と考える)
- かけ算: 足し算の繰り返しとして計算
- 割り算: 引き算の繰り返しとして計算



Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

14

真数表現(p. 9)

- 数を表す2進数に符号(+ or -)を表す1ビットを付加
 - 数の先頭のビットで符号を表す
 - 0が「+」、1が「-」を表す

「符号ビット」と呼ぶ

10進数	2進数
-3	1 0 1 1
-2	1 0 1 0
-1	1 0 0 1
0	1 0 0 0
+0	0 0 0 0
+1	0 0 0 1
+2	0 0 1 0
+3	0 0 1 1

数を表す部分

0が「+0」と「-0」の2種類できてしまう

具合が悪いので真数表現はあまり使われない

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

15

2の補数とは?

- そもそも「補数」とは: ある数Nに足したときに桁が1つ上がる数

- 2進数に限らず、どんなN進数であっても、「補数」は存在
- 2進数での補数が「2の補数」

5桁の数XXXXXに対して...

XXXXX + YYYYY = 100000

となるとき、「XXXXX」の補数が「YYYYY」

Ex. 2進数で「01001」だと...

01001 + 10111 = 100000 なので、「01001」の補数は「10111」

2の補数をマイナスの数として扱う...

5桁の計算の例: $(01001)_2 + (10111)_2 = (100000)_2 \rightarrow (00000)_2$ (オーバーフロー)

✓ $(01001)_2 = (9)_{10}$, $(10111)_2 = (-9)_{10}$ なので、 $(9)_{10} + (-9)_{10} = (0)_{10}$ なので、計算がきちんとできている!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

16

2の補数表現[1](p. 9)

- 負の数Nを、正の数N(2進数)の0と1を反転させて1を加えた数で表現する方法

- 0と正の整数(自然数)は、そのまま表現(この計算はしない)

Ex.

$(-10)_{10}$

「10」を2進数にして「-」をつけたもの

$= (-01010)_2$

$\rightarrow (10101 + 1)_2 = (10110)_2$

「-01010」の「-」をとって「1」と「0」を逆にしたもの

$(-10)_{10}$ の2進数(2の補数表現)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

17

2の補数表現[2](p. 9)

- 2の補数 = 負の数を2進数で表現したもの(コンピュータの世界では)

- 計算方法(例: -20を10桁の2進数に直す)

- 2の補数に直したい10進数のマイナスを取り除く

$\rightarrow (-20)_{10} \rightarrow (20)_{10}$

1. の結果を2進数に直す

(この時点で桁数あわせ!! 後で桁数あわせをしないと、合わせ方を間違えやすい)

$\rightarrow (20)_{10} = (0000010100)_2$

2. の結果の0と1を逆にする(0の桁を1、1の桁を0にする)

0 0 0 0 1 0 1 0 0

1 1 1 1 1 0 1 0 1 1

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

18

2の補数表現[2](p. 9)

- ➡ 2の補数 = 負の数を2進数で表現したもの(コンピュータの世界では)
- ➡ 計算方法(例: -20を10桁の2進数に直す)
- 4. 3.の結果に1を足し算する

$$\begin{array}{r} 1111101011 \\ + 1 \\ \hline 1111101100 \end{array}$$

-20を2進数に直した結果
(2の補数 = 2進数での負の数の表現)

2進数での負の数の表現では、
「-」の記号はつけない

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

19

2の補数表現の利点(p. 10)

- ➡ 引き算(符号付きの足し算)をそのまま足し算として処理できる

Ex.

$$(10 + 3)_{10} = (1010 + 0011)_2 = (01101)_2 = (13)_{10}$$

$$(6 - 3)_{10} = (110 + 101)_2 = (1011)_2 \rightarrow (011)_2 \text{ (オーバーフロー)} = (3)_{10}$$

「-3」の2の補数表現

真数表現: 符号付きの足し算を処理するには、別の回路が必要
(単純に足すことはできない)

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

20

2の補数を10進数に変換[1]

- ➡ 2の補数から1を引き、0と1を反転させて10進数になおして「-」をつける
- ➡ 負の数を2の補数に変換するときの逆
- ➡ この計算は、負の数だけ

Ex.

$$\begin{array}{l} (110110)_2 \\ - 1 \\ \hline (110110 - 1)_2 = (110101)_2 \\ - \\ (-001010)_2 \\ \hline (110110)_2 \text{ (2の補数)の10進数} \\ = (-10)_{10} \end{array}$$

2の補数から1を
引いたもの

「110101」の0と1を
逆にしたもの

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

21

2の補数を10進数に変換[2]

- ➡ 計算方法(例: 1111101100を10進数に直す)

- 2の補数から1を引き算する

$$\begin{array}{r} 1111101100 \\ - 1 \\ \hline 1111101011 \end{array}$$

1. 結果の0と1を逆にする(0の桁を1、1の桁を0にする)

$$\begin{array}{r} 1111101011 \\ \downarrow \\ 0000010100 \end{array}$$

※2の補数→10進数の方法は、10進数→2の補数の逆

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

22

2の補数を10進数に変換[2]

- ➡ 計算方法(例: 1111101100を10進数に直す)

2. の結果を10進数に直す

$$(0000010100)_2 = (20)_{10}$$

3. の結果に-(マイナス)をつける

$$(20)_{10} \rightarrow (-20)_{10}$$

$$\begin{array}{r} 1111101100 \\ \rightarrow 10 \text{ 進数に直した数} \end{array}$$

※2の補数→10進数の方法は、10進数→2の補数の逆

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

23

2進数の引き算[1]

- ➡ 10進数の引き算だと...

- ➡ ある桁の引かれる数が引く数より小さければ、1つ大きな桁から10を借りる
- ➡ 10を借りる: 貸した桁から1を引き、借りた桁に10を足す

$$\begin{array}{r} 10 \text{ を借りる} \\ 100 \\ - 1 \\ \hline 099 \end{array} \quad \rightarrow \quad \begin{array}{r} 10 \text{ を借りる} \\ 0100 \\ - 1 \\ \hline 0099 \end{array} \quad \rightarrow \quad \begin{array}{r} 0910 \\ - 0099 \\ \hline 0099 \end{array}$$

引き算の答え: 99

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

24

2進数の引き算[2]

2進数の引き算だと...

- ある桁の引かれる数が引く数より小さければ、1つ大きな桁から(10)₂(10進数で2)を借りる
- 2を借りる: 貸した桁から1を引き、借りた桁に2を足す
- 要は、10進数の「10」が、2進数では「2」になるだけ!!

$$\begin{array}{r}
 100 \\
 -001 \\
 \hline
 \end{array}
 \rightarrow
 \begin{array}{r}
 020 \\
 -001 \\
 \hline
 \end{array}
 \rightarrow
 \begin{array}{r}
 012 \\
 -001 \\
 \hline
 011
 \end{array}$$

引き算の答え: 011

※コンピュータ的には引き算はしないので、人間が2の補数→10進数の計算をするための引き算

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

25

正の数と負の数の見分け方[1]

大前提: 数を表す2進数の桁数は決まっている

- 普通のコンピュータで32桁(or 64桁)

ということは...例えば(10)₁₀は、コンピュータ的には...

0000...00001010
28個の「0」と考えている

※授業のスライド中では32桁分も書けないので、そのときどきで適当なところで割愛

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

26

正の数と負の数の見分け方[2]

負の数(2の補数)の計算方法: 負の数Nを、正の数N(2進数)の0と1を反転させて1を加える

コンピュータ的には32桁で数を表すので...

$$\begin{aligned}
 (-10)_{10} &\rightarrow (10)_{10} \\
 &= (0000...00001010)_2 \\
 &\rightarrow (1111...1111)101 + 1 = (1111...11110110)_2
 \end{aligned}$$

28個の「0」も全て「1」に反転される

- 負の数は結果的に一番大きな桁(一番左の桁)が「1」になる

- 一番大きな桁(一番左の桁)が「0」であれば正の数、「1」であれば負の数として扱う

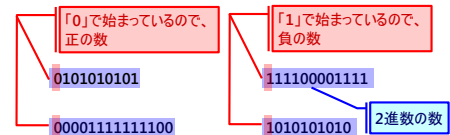
Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

27

正の数と負の数の見分け方[3]

2進数を見たときに...(2の補数を考える場合)

- 「2の補数を考える」という場合は、先頭の桁を見て、正の数か負の数かを判断
- 「2の補数を考える」と書かれていない場合は、負の数を考えなくてOK



Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

28

正の数と負の数の見分け方[3]

2進数で表された数は、一番大きな桁(一番左の桁)が「0」であれば正の数、「1」であれば負の数

10進数で表された数は、普通に正の数、負の数として計算

- 正の数であれば、割り算だけで2進数に変換
 - Ex. 「+5」または「5」と書かれていれば、割り算だけで2進数に変換
- 負の数であれば、2の補数の方法で2進数に変換
 - Ex. 「-5」と書かれていれば、2の補数の方法で2進数に変換

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

29

桁あふれ(オーバーフロー)(p. 11)

2の補数に関係した桁あふれ(オーバーフロー)が起こりうる

Ex. 2進数5桁の計算(10進数で14+5の計算)

$$\begin{array}{r}
 01110 \\
 +00101 \\
 \hline
 10011
 \end{array}$$

先頭の桁が1になってしまった

- 先頭の桁が1の場合は、2進数で負の数として扱う

$$\begin{array}{r}
 10011 \\
 \hline
 \end{array}$$

負の数を表す

計算結果: (-13)₁₀(負の数)

2の補数に関係した桁あふれ(オーバーフロー)

Copyright (C) Junko Shirasane, Tokyo Woman's Christian University, 2016. All rights reserved.

30

負数込みの計算の考え方

→ 計算結果を見て...(オーバーフローがどーと考えるのではなく!)

1. 計算結果が指定の桁数を超えていれば、超えた分の桁を削除(桁数あわせ)

4桁の2進数の計算結果: $(100110)_2$ → 桁数を超えた部分 = 削除 → 計算結果: $(0110)_2$

2. 計算結果の先頭の桁が0か1かで、正か負を判断

- 正の数(先頭の桁が0)であれば、割り算だけで10進数に直す
- 負の数(先頭の桁が1)であれば、2の補数の方法で10進数に直す

4桁の2進数の計算結果: $(0101)_2$ → 正の数と判断 → 計算結果: $(5)_{10}$

4桁の2進数の計算結果: $(1010)_2$ → 負の数と判断 → 計算結果: $(-6)_{10}$

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

31

桁あふれ[まとめ][1]

→ 桁あふれの分類(その1)

- 足し算等の何らかの計算の結果、コンピュータが扱うことのできる数の桁数の限界を超えてしまう場合

→ Ex. 4桁の数「0110+0110+0110」の計算

→ 本来の計算結果は「10010」で5桁になってしまうので、5桁目が無視されてコンピュータが出す結果は「0010」

→ コンピュータが出す結果と本来の結果が違ってしまう現象

ある意味、コンピュータの性能の限界を超えてしまうということで、その結果として、本来の計算結果とは違う結果がでる現象

※ 人間やコンピュータがミスをした、という現象ではない

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

32

桁あふれ[まとめ][2]

→ 桁あふれの分類(その2)

- 足し算等の何らかの計算の結果、数の正と負が違ってしまう場合

→ Ex. 4桁の数「0110+0110」の計算

- 本来の計算結果は「1100」で、1桁目が1なので、コンピュータは計算結果を負の数(-4)として取り扱い
- 本来の計算結果は正の数(12)
- コンピュータが出す結果と本来の結果が違ってしまう現象

本来の計算結果は正(負)の数なのに、計算結果が負(正)の数になってしまう現象

※ 人間やコンピュータがミスをした、という現象ではない

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

33

やってみよう!

- $(+10) + (+8)$ を5桁の2の補数として計算し、10進数として表現

- $(-10) + (+8)$ を5桁の2の補数として計算し、10進数として表現

オーバーフローも考慮すること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

34

小数の表現方法

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

35

整数以外の数表現するには?

→ コンピュータが表現できる数: 整数, 小数

→ 整数以外の数

- 小数
- 分数
- n乗根(平方根, 立方根, etc)
- π (円周率)
- etc.

全て小数として表現

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

36

2進数の小数を10進数に直す[1]

1. 10進数の整数部分は、通常の方法で2進数に直す
2. 10進数の小数部分に2をかけ算する
3. 2.の結果、整数部分を小数点第1桁にする
4. 3.の結果、小数部分をまた10進数の数とする

- 小数部分が0になるまで繰り返す
- ✓ 無限小数になることも(10進数でけりい数も2進数では無限小数になりえる)
- 2.の整数部分を小数点第2桁、第3桁、...と置いていく
- 1.の整数部分と2.の整数部分を並べたものが2進数での小数になる

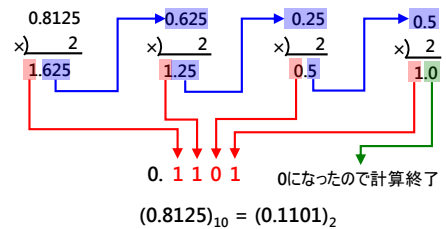
Copyright (C) Junko Shiranaga, Tokyu Women's Christian University 2016. All rights reserved.

37

2進数の小数を10進数に直す[2]

➡ 10進数の小数を2進数に直すには?(例1)

➡ 10進数の0.8125を2進数に直す



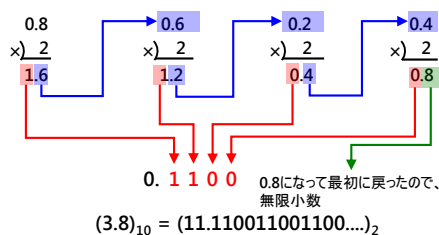
Copyright (C) Junko Shiranaga, Tokyu Women's Christian University 2016. All rights reserved.

38

2進数の小数を10進数に直す[3]

➡ 10進数の小数部分を2進数に直すには?(例2)

➡ 10進数の3.8を2進数に直す(整数部分の3は2進数で11)



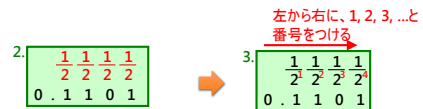
Copyright (C) Junko Shiranaga, Tokyu Women's Christian University 2016. All rights reserved.

39

10進数の小数を2進数に直す[1]

➡ 例: 101.1101 (整数部分101は10進数で5)

1. 2進数の整数部分は、通常の方法で10進数に直す
 2. 10進数の小数部分各桁の上に「1/2」を書く
 3. 2. で書いた「1/2」の「2」の右肩に左から1, 2, 3, ...と書いていく
- ➡ $1/2^0, 1/2^1, 1/2^2, \dots$ ができていく

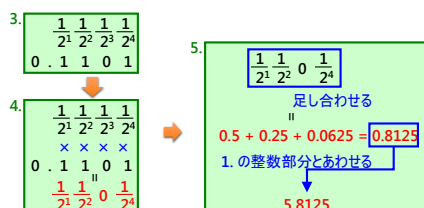


Copyright (C) Junko Shiranaga, Tokyu Women's Christian University 2016. All rights reserved.

40

10進数の小数を2進数に直す[2]

4. 各桁の上の「1/2^n」と、それぞれの桁の数をかけあわせる
5. 4.の結果を足し合わせ、1.の整数部分とあわせる



Copyright (C) Junko Shiranaga, Tokyu Women's Christian University 2016. All rights reserved.

41

やってみよう!

- ➡ 2進数1.101を10進数に変換 (2010年度ITパスポート春季試験問題)
- ➡ 2進数に変換した時、有限小数で表現できる10進数は、以下のうちどれか
- ➡ 0.1
 - ➡ 0.2
 - ➡ 0.4
 - ➡ 0.5
- (2012年度ITパスポート秋季試験問題)

Copyright (C) Junko Shiranaga, Tokyu Women's Christian University 2016. All rights reserved.

42

小数の表現方式

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

43

小数を表現する方法(p. 10)

- ➡ 固定小数点方式
- ➡ 浮動小数点方式

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

44

固定小数点方式[1](p. 10)

- ➡ 小数部分の桁数がnの場合: 2進数では、小数部分が $1/2^n$ 刻みで表現

↓
nを大きくすると、それだけ小数部分を細かく表現可能

※ただし、実際コンピュータは小数も2進数で考えているが、
人間が考えときの便宜上、10進数で考えることが多い

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

45

固定小数点方式[2](p. 10)

- ➡ 小数を表す桁数が決まっていると...

Ex. 右から2桁を小数部分とすると...
 $(2 \div 1000)_{10} = (0.002)_{10} = (0.00)_{10}$
 小数を正確に表現できない

何桁分の小数部分を持っているかは数値によって異なる
 = 固定小数点方式で小数を表せる場合は少ない

↓
浮動小数点方式

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

46

浮動小数点方式[1](p. 10)

- ➡ 小数: $D \times 10^n$ と表現できる

- Ex. 1: $0.5 = 5 \times 10^{-1}$
- Ex. 2: $-0.0625 = -6.25 \times 10^{-2}$
- Ex. 3: $0.0000000084 = 8.4 \times 10^{-9}$

どの数でも「 $\times 10^n$ 」の「10」の部分は同じ(実際のコンピュータでは「 $\times 2^n$ 」)

↓
小数を「 $D \times 10^n$ 」の形と考え、「D」と「n」だけ記憶しておく

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

47

浮動小数点方式[2](p. 10)

- ➡ 浮動小数点方式:
 小数を「 $D \times 10^n$ 」と考え、「D」と「n」を記憶することで小数を表す方式

Ex.
 D = 6.25, n = -3の場合: 0.00625
 D = 6.25, n = -2の場合: 0.0625
 D = 6.25, n = -1の場合: 0.625
 D = 6.25, n = 0の場合: 6.25

↓
D: 仮数部
 n: 指数部
 と呼ぶ

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

48

浮動小数点方式[3](p. 10)

- ➡ 仮数部
 - ➡ 符号は「0」が「+」、「1」が「-」
 - ➡ 固定小数点方式
- ➡ 指数部
 - ➡ 符号は「0」が「+」、「1」が「-」(ただし、2の補数表現とは別の特殊な形で表現される)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

大きな数の表現(p. 10)

- ➡ 浮動小数点方式を利用して表現

Ex.
 $2000000000000 = 2 \times 10^{12}$
 $-42500000000000000 = -4.25 \times 10^{17}$

指数部が「+」の数になる

コンピュータは「2」と「+12」、「-4.25」と「+17」を記憶しておく
 (実際には、「 $\times 10^n$ 」ではなく「 $\times 2^n$ 」で表現)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

大きな数の表現[例](p. 10)

- ➡ Windowsの電卓のあるモード(オーバーフローをなかなかしないモード)で...



Ex. $1000000000000000 \times 1000000000000000$
 $= 1.e+30$

1.0×10^{30} という意味
 (浮動小数点方式での表現)

※オーバーフローしにくいモードは、大きな数を浮動小数点方式で表現する
 = それだけたくさんの桁がある数を表現できるので、オーバーフローしにくい

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

桁落ち(1)

- ➡ 小数部分が無限のものを扱えるわけではない
 - ➡ 例えば割り算で割り切れない数や円周率
 - ➡ 小数部分を適当なところで切り捨てる(四捨五入ではない)
- 例えば... $1 \div 6$:
 コンピュータは「0.16666...666」と考える
 本来はこの後も無限に続く

コンピュータが扱える小数の桁数:
 「有効桁数」と呼ぶ

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

桁落ち(2)

- ➡ 小数部分が無限のものは適当なところまでで切り捨てられる
 本来の数よりも、小数の桁数が小さくなってしまう現象

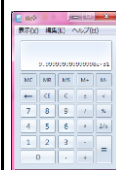
桁落ち

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

桁落ちの例

- ➡ Windowsの電卓のあるモード(オーバーフローをなかなかしないモード)で...

Ex. 電卓での計算: $999999999999999 \times 999999999999999$
 $\rightarrow 9.999999999999998e+31 = 9.999999999999998 \times 10^{31}$
 $= 9999999999999980000...00000$



でも実際に計算すると...

999999999999999
 $\times 999999999999999$

$\dots 91$
 $\times \dots 91$
 $\dots 01$

つまり本当の計算では...
 $999999999999999 \times 999999999999999$
 $= 9.999999999999998e+31$

桁落ち

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

やってみよう!

- ➡ スマートフォンの電卓で、大きな数のかけ算をしてみよう!
 - ➡ 結果が浮動小数点表示になるか?
 - ➡ 桁落ちしているか?
- を確認しよう!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

57

桁落ちが起こると...

- ➡ 数が本来の数よりも小さくなってしまう
 - ➡ 微妙な数値が必要な場合には要注意
- ➡ 桁落ちをした数に大きな数をかけると、本来の数に大きな数をかけたときとの差が大きくなる

例えば...有効桁数が小数点第3位とすると、 $1 \div 6 \times 100000$ の計算は...

- 「 $1 \div 6$ 」の時点で0.166に桁落ちし、それに100000をかけて、16600
- 1×100000 を7で割ると(計算の順序を変えると)、16666.666
- 本来の $1 \div 6$ に100000をかけると、16666.66666666...

- コンピュータで計算をするときは、計算の順番に注意
(割り算はなるべく後にすること)
- 例えば「 $1 \div 6 \times 100000$ 」の計算は、「 1×100000 」をしてから6で割る

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

58

やってみよう!

- 0.0000055を浮動小数点方式で表現
- 0.000000001234を浮動小数点方式で表現
- 456000000000000を浮動小数点方式で表現
※3つとも仮数部は小数点第2位の小数とすること
- $(10 \div 7) \times 10000$ を計算
 - ➡ 小数点第2位までが有効桁数
 - ➡ 桁落ちを考えて計算すること
- $10 \div 7 \times 10000$ を計算
 - ➡ 小数点第2位までが有効桁数
 - ➡ 桁落ちの影響がなるべく少ないように計算すること

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

59

Question!

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

60

文字の表現

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

61

文字の符号化(p. 13)

- ➡ 文字: コンピュータは整数に置き換えて扱う(番号をつけて扱う)
 - ➡ 文字を2進数で表現する(「符号化」と呼ぶ)
- ➡ 2進数で表現される文字集合
 - ➡ 半角英数文字
 - ➡ 図形文字
 - ➡ 制御文字
 - ➡ 多バイト文字
 - ➡ 図形文字

※文字集合: 文字の集まり

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

62

図形文字と制御文字(p. 13)

- ➡ **図形文字**: 通常、画面に表示される文字
 - ➡ 人間が明示的に書いたり読んだりする文字
 - ➡ アルファベット, 数字, ひらがな, 漢字, 記号, etc.
- ➡ **制御文字**: 通常、画面に表示されない文字
 - ➡ コンピュータに何らかの制御をするための文字
 - ➡ 改行, TAB, ESC, etc.

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

63

ASCII文字

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

64

ASCII文字(p. 13)

- ➡ **ASCII**: American Standard Code for Information Interchange
 - ➡ 半角文字を表す文字集合
 - ➡ アルファベット大文字(26文字)
 - ➡ アルファベット小文字(26文字)
 - ➡ 数字(10文字)
 - ➡ 記号(スペース, 「,」, 「.」, etc.)
- 1文字を表すために、最低限7ビット必要
(6ビット: 64種類の情報, 7ビット: 128種類の情報)

※1文字を表す2進数の桁数(ビット数)は、どの文字でも同じ(つまり7ビット)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

65

図形文字(p. 13)

- ➡ **ASCII**: 情報量が7ビットで収まるように、扱う文字を取り決めた文字集合
 - ➡ アルファベット(大文字・小文字): 52文字
 - ➡ 数字: 10文字
 - ➡ 記号(スペースを含む): 33文字
- 図形文字: 合計95文字

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

66

番号例(p. 14)

番号	文字	番号	文字	番号	文字
47	0	65	A	97	a
48	1	67	B	98	b
49	2	68	C	99	c
50	3	69	D	100	d
51	4	70	E	101	e
52	5	71	F	102	f
53	6	72	G	103	g
54	7	73	H	104	h
55	8	74	I	105	i
56	9	75	J	106	j

➡ 「数」としての0~9ではなく、「文字」としての0~9

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

67

番号の決まり(p. 14)

- ➡ 7ビットで1文字を表現 = 128文字表現可能
 - ➡ 図形文字: 95文字
 - ➡ 32番~126番までが図形文字
 - ➡ 残り33文字: 制御文字を表現
 - ➡ 0番~31番と127番が制御文字
 - ※32番のスペースを、制御文字と考えることもある
- ➡ アルファベットの大文字と小文字の番号に規則
 - ➡ 小文字の番号は、大文字の番号に32(=2⁵)₁₀=(100000)₂を足したもの
 - ➡ Ex. A: (65)₁₀ = (01000001)₂, a: (97)₁₀ = (01100001)₂
 - ➡ 大文字⇄小文字の変換はやりやすい

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University 2016. All rights reserved.

68

制御文字例(p. 14)

- ➡ BS(8番): Back Space
- ➡ HT(9番): TAB
- ➡ LF(10番): UNIX系OSでの改行
- ➡ CR(13番): Mac OSでの改行
 - Windowsでの改行は、13番と10番を合わせて「CRLF」の2つの制御文字で表現される
 - ✓ つまり、Windowsでは1つの改行が2文字分(2ビット)
- ➡ DEL(127番): Delete

※OS: Operating System(オペレーティングシステム)

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

69

ビット数[1](p. 14)

- ➡ コンピュータでは8ビットを1つの単位として扱うことが多い
 - ASCII文字も8ビットで表現すると扱いやすい
- ➡ 8ビットのうち、7ビット分(2進数で7桁目まで)で文字を表現する
- ➡ 残りビット(2進数で8桁目)に常に0を入れておく
 - ASCII文字としては無駄なビット
 - 日本語を表現するときに利用

例えば...
 A: 65番(10進数)
 = 1000001番(2進数)
 = 01000001番(2進数, コンピュータ内での表現)
 ↓ ASCII的には無駄な(何も利用していない)ビット

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

70

ビット数[2](p. 14)

- ➡ 8ビットで1文字を表現 = 1バイトで1文字を表現

「1バイト文字」と呼ばれる

Ex. 「Hello, my name is John.」

- アルファベット: 17文字
 - 記号: 2文字
 - スペース: 4文字
- } 23文字 = 23バイト

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

71

ちなみに...

- ➡ アスキーアートも文字コードのASCIIから(ASCII art)
 - アスキーアート: 文字だけで作った絵
 - 感情を表す「(^_^)」のような単純なものから、人や動物に見えるものまで様々

- ➡ アスキーアートの例

- <http://ja.wikipedia.org/wiki/%E3%82%A2%E3%82%B9%E3%82%AD%E3%83%BC%E3%82%A2%E3%83%BC%E3%83%88>
- <http://bhdaa.sakura.ne.jp/zukan/>

Copyright (C) Junko Shirogane, Tokyo Woman's Christian University, 2016. All rights reserved.

72