知能情報工学演習I
第8回（後半第2回）

岩村雅一

masa@cs.osakafu-u.ac.jp

後半の予定

7. 5月31日　プログラミング環境（テキスト1,2章）
8. 6月7日*　変数とデータ型（3章）、演算子（4章）
9. 6月14日　コンソール入出力（6章）、配列（3章）
10. 6月21日*　制御文1（テキスト5章）
11. 6月28日*　制御文2（テキスト5章）
12. 7月12日　関数1（テキスト7章）
13. 7月19日　配列（3章）、応用プログラム

本日のメニュー

- 変数のデータ型
  - データ型
  - ビットとバイト
- 演算子
  - 算術演算子
  - インクリメント演算子、デクリメント演算子
  - キャスト演算子

変数のデータ型

- データ型を決める要因
  - 整数か小数か
  - 符号の有無（マイナスを扱えるか）
  - 精度

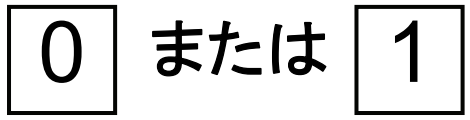
主なデータ型（テキストP.51）

	データ型	バイト	表現範囲
整数（符号の有無）	char	1	-128～127
	unsigned char	1	0～255
	short int	2	-32768 ～ 32767
	unsigned short int	2	0～65535
	int	4	-2147483648～2147483647
	unsigned int	4	0 ～ 4294967295
	long int	4	-2147483648～2147483647
	unsigned long int	4	0 ～ 4294967295
小数	float	4	有効6桁
	double	8	有効15桁

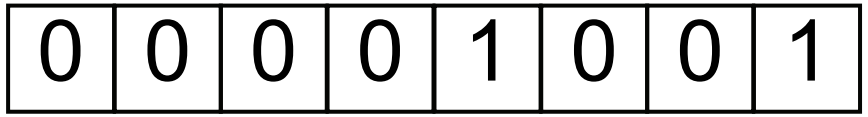
精度

ビットとバイト

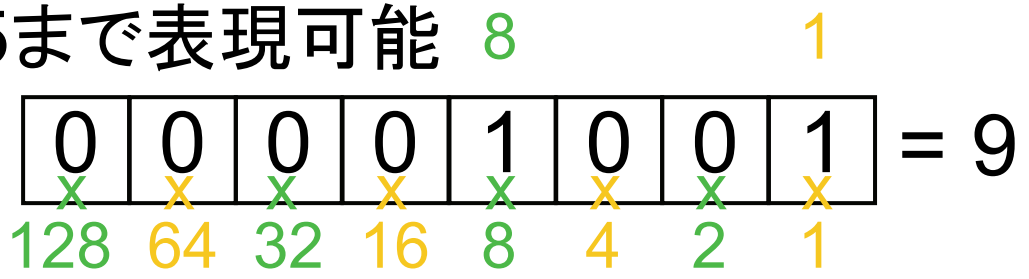
- ビット
  - 0か1の2通り



- バイト
  - 8ビット

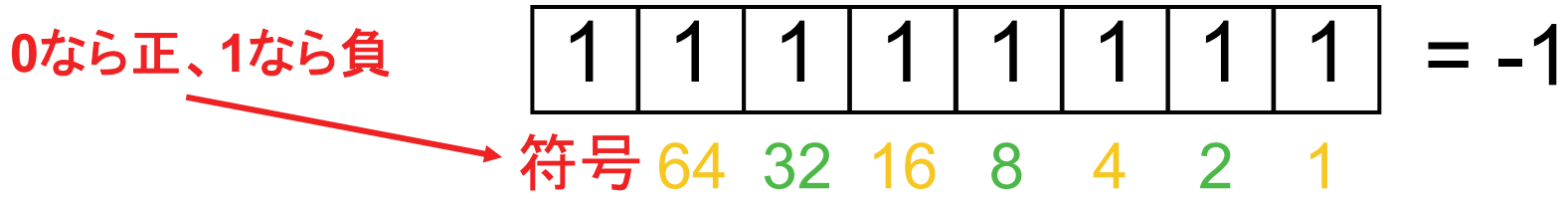


  - 符号なし：0から255まで表現可能



  - 符号あり：-128～127まで表現可能

詳細は省略



参考：符号つき2進数と符号なし2進数

- 符号なし2進数
  - 11111111 = 255
  - 11111110 = 254
  - …
  - 10000001 = 128
  - 01111111 = 127
  - …
  - 00000100 = 4
  - 00000011 = 3
  - 00000010 = 2
  - 00000001 = 1
  - 00000000 = 0
- 符号つき2進数
  - 01111111 = 127
  - …
  - 00000010 = 2
  - 00000001 = 1
  - 00000000 = 0
  - 11111111 = -1
  - 11111110 = -2
  - 11111101 = -3
  - 11111100 = -4
  - …
  - 10000000 = -128

算術演算子

- 四則演算と、余りの5種類
  - 可算・・・+
  - 減算・・・-
  - 乗算・・・*
  - 除算・・・/
  - 剰余・・・%
    - 100%30 → 10
    - 7%3 → 1

演算の優先順位

- 演算の優先順位は数学と同じ
  1. 括弧で囲む : (,)
  2. かけ算、割り算、余り : *, /, %
  3. 足し算、引き算 : +, -
- 優先順位が同じならば左から処理される
  - 例 : 1+2*3-4

    → 1+6-4

    → 7-4

    → 3

インクリメント、デクリメント演算子

- インクリメント演算子
  - 変数の値を1加算する
  - 記述例：　　a++; または ++a;

    a = a + 1;と等価
- デクリメント演算子
  - 変数の値を1減算する
  - 記述例：　　a--;または --a;

    a = a - 1;と等価

インクリメント、デクリメント演算子

- a++; と ++a; の違いは？
  - いつ加算するかが違う
  - b=a++;

    → b=a; a++;
  - b=++a;

    → a++; b=a;

サンプルプログラム1
（インクリメント演算子）

```
#include <stdio.h>

int main(void){
  int i, j;

  i = 1;
  j = 10;
  printf("1: i=%d, j=%d¥n", i,j);
  i++;
  printf("2: i=%d, j=%d¥n", i,j);
  ++i;
  printf("3: i=%d, j=%d¥n", i,j);
  j = j + i++;
  printf("4: i=%d, j=%d¥n", i,j);
  j = j + ++i;
  printf("5: i=%d, j=%d¥n", i,j);

  return(0);
}
```

i = 1, j = 10

i = 2, j = 10

i = 3, j = 10

i = 4, j = 13

i = 5, j = 18

j=j+iを実行してから
i++を実行

++iを実行してから
j=j+iを実行

整数の演算と小数の演算

- 割り算は注意が必要
  - 例えば、10/3はいくつになる?

    3.33333333…

    になるような気がするけど、実際は？

サンプルプログラム２
（出力されるデータ型）

```
#include <stdio.h>

int main(void){
    int i1, i2;
    float f1, f2;

    i1 = 10;
    i2 = 3;
    f1 = 10;
    f2 = 3;

    printf("i1/i2=%d¥n", i1/i2);
    printf("f1/f2=%f¥n", f1/f2);

    return(0);
}
```

整数

小数

printfの書式

- %d は整数を出力する
- %f は小数を出力する

出力されるデータ型

- 整数の場合

ソース

```
int i1, i2;
i1 = 10;
i2 = 3;                          int / int
printf("i1/i2=%d¥n", i1/i2);
```

出力

```
i1/i2=3
```

← 整数

つまり、 整数 / 整数 = 整数

出力されるデータ型

- 小数の場合

ソース

```
float f1, f2;
f1 = 10;
f2 = 3;                    float / float
printf("f1/f2=%f¥n", f1/f2);
```

出力

```
f1/f2=3.333333
```

← 小数

つまり、 小数 / 小数 = 小数

データ型の自動変換

- 表現力の大きいほうに自動的に変数の型を合わせる

整数	/	整数	=	整数
整数	/	小数	=	小数
小数	/	整数	=	小数
小数	/	小数	=	小数

では、 整数 / 整数 = 小数 にしたいときは？

整数	/	整数	=	整数
整数	/	小数	=	小数
小数	/	整数	=	小数
小数	/	小数	=	小数

整数を小数に変換すればいいんじゃない？

サンプルプログラム３（キャスト）

```
#include <stdio.h>

int main(void){
    int i1, i2;
    i1 = 10;
    i2 = 3;
                                int / int
    printf("i1/i2=%d¥n", i1/i2);          float / float
    printf("(float)i1/(float)i2=%f¥n", (float)i1/(float)i2);

    return(0);
}
```

キャスト演算子

- データの型を変換する
  - 書式:(変換する型)変数
  - 例:

```
int i;
float f;

i = 10;
f = (float)i;
```

  - 整数を小数にキャストするときは問題なし
  - 小数を整数にキャストするときは、小数点以下切り捨て