プログラミング基礎

2023年度1Q　火曜日３,４時限(10:45~12:25)
金曜日1~４時限(　8:50~12:25)

工学院　情報通信系

中山実，渡辺義浩
伊藤泉，小杉哲
TA:小泊大輝，千脇彰悟

5/12(金) 8:50~12:25

- 第9回「再帰」

1. 再帰について
2. 階乗の計算
3. Fibonacci 数の計算
4. 指数関数の近似（テイラー展開）
5. 最大公約数
6. 型変換について（補足）

再帰について

- C言語では，関数のなかで，自分自身（自分と同じ関数）を呼び出すことができる.

- 利点
  - 複雑なアルゴリズムを明快に記述できる場合がある
- 欠点
  - 呼び出し回数が爆発的に増加して，スタックオーバーフローを起こす可能性がある
  - 計算が冗長になる可能性がある

再帰について

```
int main {
    function(n, ... );     //関数の最初の呼び出し
}

void function(n, ... ){
    if (脱出条件)  return; //再帰からの脱出
    function(n-1, ... );  //再帰呼び出し
}
```

5/12(金) 8:50~12:25

- 第9回「再帰」

1. 再帰について
2. 階乗の計算
3. Fibonacci 数の計算
4. 指数関数の近似（テイラー展開）
5. 最大公約数
6. 型変換について（補足）

階乗($n!$)の計算

$$n! = \prod_{k=1}^{n} k = n \times (n-1) \times \cdots \times 3 \times 2 \times 1$$

$$n! = \begin{cases} 1, & \text{if } n = 0 \\ n \times (n-1)!, & \text{if } n > 0 \end{cases}$$

階乗($n!$)の計算

```c
#include <stdio.h>

int factorial(int n); /*階乗*/

int main (void) {
    int i;

    for (i = 0; i < 11; i++){
        printf ("%d! = %d¥n", i, factorial(i));
    }
    return 0;
}

int factorial (int n){
    if (n == 0) return 1;
    else return n* factorial(n-1);
}
```

- プロトタイプ宣言

階乗($n!$)の計算

```c
#include <stdio.h>

int factorial(int n); /*階乗*/

int main (void) {
    int i;

    for (i = 0; i < 11; i++){
        printf ("%d! = %d¥n", i, factorial(i));
    }
    return 0;
}

int factorial (int n){
    if (n == 0) return 1;
    else return n* factorial(n-1);
}
```

- 結果の出力
- 関数の呼び出し

階乗($n!$)の計算

```c
#include <stdio.h>

int factorial(int n); /*階乗*/

int main (void) {
    int i;

    for (i = 0; i < 11; i++){
        printf ("%d! = %d¥n", i, factorial(i));
    }
    return 0;
}

int factorial (int n){
    if (n == 0) return 1;
    else return n* factorial(n-1);
}
```

- 再帰呼び出し

階乗($n!$)の計算

```c
#include <stdio.h>

int factorial(int n); /*階乗*/

int main (void) {
    int i;

    for (i = 0; i < 11; i++){
        printf ("%d! = %d¥n", i, factorial(i));
    }
    return 0;
}

int factorial (int n){
    if (n == 0) return 1;
    else return n* factorial(n-1);
}
```

- 再帰からの脱出

階乗($n!$)の計算

```
factorial (3)
→  3 x factorial (2)
→  3 x (2 x factorial (1) )
→  3 x (2 x (1 x factorial (0) )
→  3 x (2 x (1 x 1 ) )
```

```c
#include <stdio.h>

int factorial(int n); /*階乗*/

int main (void) {
    int i;

    for (i = 0; i < 11; i++){
        printf ("%d! = %d¥n", i, factorial(i));
    }
    return 0;
}

int factorial (int n){
    if (n == 0) return 1;
    else return n* factorial(n-1);
}
```

階乗($n!$)の計算

```c
#include <stdio.h>

int factorial(int n); /*階乗*/

int main (void) {
    int i;

    for (i = 0; i < 11; i++){
        printf ("%d! = %d¥n", i, factorial(i));
    }
    return 0;
}

int factorial (int n){
    if (n == 0) return 1;
    else return n* factorial(n-1);
}
```

実行結果

```
0! = 1
1! = 1
2! = 2
3! = 6
4! = 24
5! = 120
6! = 720
7! = 5040
8! = 40320
9! = 362880
10! = 3628800
```

5/12(金) 8:50~12:25

- 第9回「再帰」

1. 再帰について
2. 階乗の計算
3. Fibonacci 数の計算
4. 指数関数の近似（テイラー展開）
5. 最大公約数
6. 型変換について（補足）

Fibonacci数の計算

$$F(0) = 0,$$

$$F(1) = 1,$$

$$F(n) = F(n-1) + F(n-2)$$

Fibonacci数の計算

```c
#include <stdio.h>

int fibonacci (int n);

int main (void){
    int i, y;

    for (i = 0; i < 10; i++){
        y = fibonacci (i);
        printf ("fib(%d) = %d¥n", i, y);
    }
}

int fibonacci (int n) {
    if (n == 0 ) return 0;
    else if (n == 1) return 1;
    else return (fibonacci(n-2)+fibonacci(n-1));
}
```

- プロトタイプ宣言

Fibonacci数の計算

```c
#include <stdio.h>

int fibonacci (int n);

int main (void){
    int i, y;

    for (i = 0; i < 10; i++){
        y = fibonacci (i);
        printf ("fib(%d) = %d¥n", i, y);
    }
}

int fibonacci (int n) {
    if (n == 0 ) return 0;
    else if (n == 1) return 1;
    else return (fibonacci(n-2)+fibonacci(n-1));
}
```

- 関数の呼び出し

Fibonacci数の計算

```c
#include <stdio.h>

int fibonacci (int n);

int main (void){
    int i, y;

    for (i = 0; i < 10; i++){
        y = fibonacci (i);
        printf ("fib(%d) = %d¥n", i, y);
    }
}

int fibonacci (int n) {
    if (n == 0 ) return 0;
    else if (n == 1) return 1;
    else return (fibonacci(n-2)+fibonacci(n-1));
}
```

- 再帰呼び出し

Fibonacci数の計算

```c
#include <stdio.h>

int fibonacci (int n);

int main (void){
    int i, y;

    for (i = 0; i < 10; i++){
        y = fibonacci (i);
        printf ("fib(%d) = %d¥n", i, y);
    }
}

int fibonacci (int n) {
    if (n == 0 ) return 0;
    else if (n == 1) return 1;
    else return (fibonacci(n-2)+fibonacci(n-1));
}
```

- 再帰からの脱出

Fibonacci数の計算

```c
#include <stdio.h>

int fibonacci (int n);

int main (void){
    int i, y;

    for (i = 0; i < 10; i++){
        y = fibonacci (i);
        printf ("fib(%d) = %d¥n", i, y);
    }
}

int fibonacci (int n) {
    if (n == 0 ) return 0;
    else if (n == 1) return 1;
    else return (fibonacci(n-2)+fibonacci(n-1));
}
```

実行結果

```
fib(0) = 0
fib(1) = 1
fib(2) = 1
fib(3) = 2
fib(4) = 3
fib(5) = 5
fib(6) = 8
fib(7) = 13
fib(8) = 21
fib(9) = 34
```

Fibonacci数の比

$$\lim_{n\to\infty}\frac{F(n)}{F(n-1)} = \text{黄金比}$$

- 黄金比
  - 線分を a, b の長さで 2 つに分割するときに、
  - $a : b = b : (a + b)$
  - が成り立つように分割したときの比 $a : b$ のこと

$$1 : \frac{1+\sqrt{5}}{2} = 1 : 1.618033\ldots$$

黄金比

$$a : b = b : (a + b)$$

$$b^2 = a^2 + ab$$

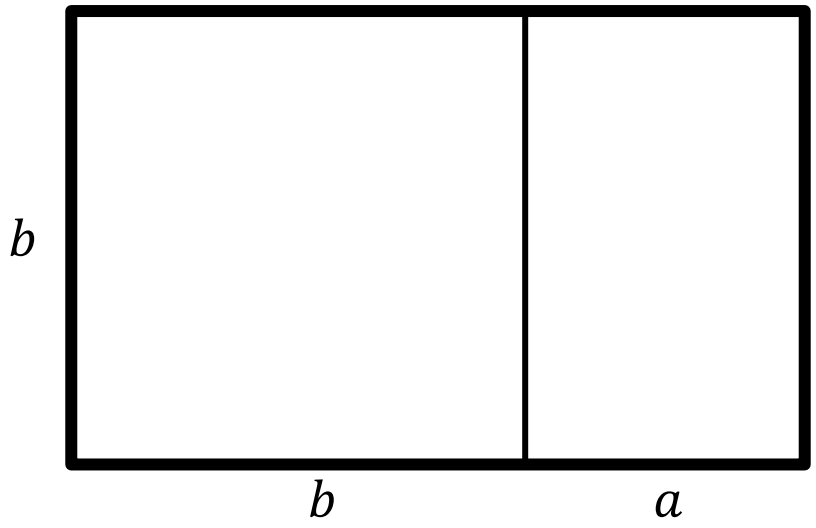
$$\left(\frac{b}{a}\right)^2 - \left(\frac{b}{a}\right) - 1 = 0$$

$$\boxed{\frac{b}{a} = \frac{1 + \sqrt{5}}{2}}$$

$$\frac{1 + \sqrt{5}}{2} = 1.618033 \ldots$$

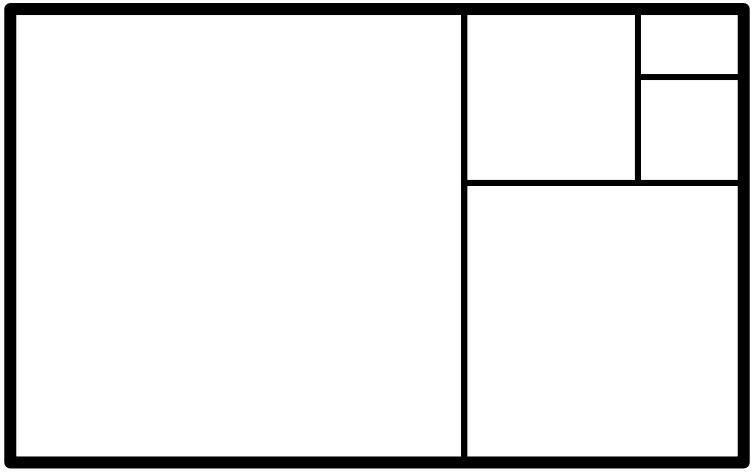
黄金比

- 縦横比が黄金比の矩形から最大正方形を切り落とした残りの矩形は、やはり黄金比の矩形となり、もとの矩形の相似になる (wikipedia)



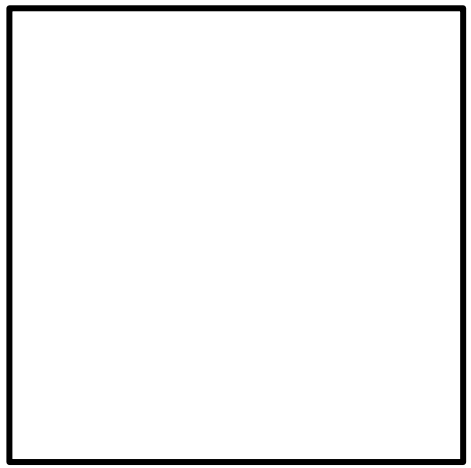
黄金比

- 縦横比が黄金比の矩形から最大正方形を切り落とした残りの矩形は、やはり黄金比の矩形となり、もとの矩形の相似になる (wikipedia)



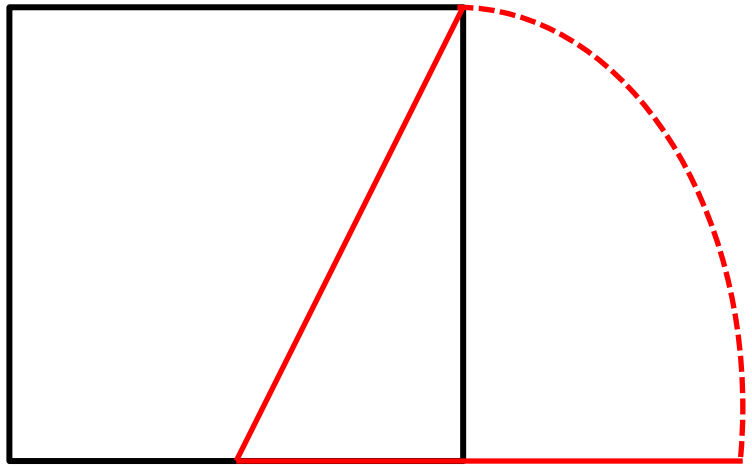
黄金比

- 作図
  - 正方形を描く



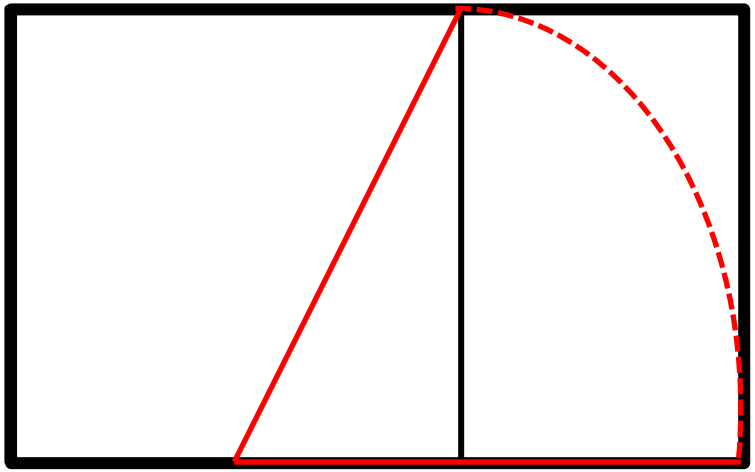
黄金比

- 作図
  - 中点を中心に円弧を描き、底辺の延長との交点を求める

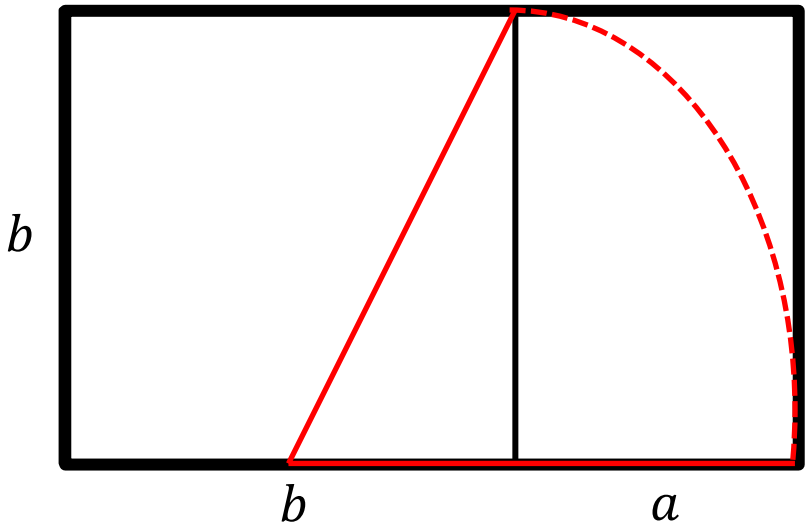


黄金比

- 作図
  - 長方形を描く



黄金比



$$\sqrt{\left(\frac{b}{2}\right)^2 + b^2} = \frac{b}{2} + a$$

$$\frac{\sqrt{5}}{2} b = \frac{b}{2} + a$$

$$a = \frac{\sqrt{5}-1}{2} b$$

$$\frac{b}{a} = \frac{2}{\sqrt{5}-1} = \frac{\sqrt{5}+1}{2}$$

Fibonacci数の比と黄金比

$$F(n+2) = F(n+1) + F(n)$$

$$\frac{F(n+2)}{F(n+1)} = 1 + \frac{F(n)}{F(n+1)}$$

$$G(n+1) = 1 + \frac{1}{G(n)} \qquad \left(G(n) = \frac{F(n+1)}{F(n)}\right)$$

$$k = 1 + \frac{1}{k} \qquad \left(\lim_{n\to\infty} G(n) = \lim_{n\to\infty} \frac{F(n+1)}{F(n)} = k\right)$$

$$k = \frac{1+\sqrt{5}}{2}$$

Fibonacci数の比

```c
#include <stdio.h>

int fibonacci (int n);

int main (void){
    int  i;
    double y;

    for (i = 2; i < 43; i++){
        y = (double) fibonacci(i) /fibonacci(i-1);
        printf ("fib(%d) /fib(%d) = %lf ¥n", i, i-1,y);
    }
}

int fibonacci (int n) {
    if (n == 0 ) return 0;
    else if (n == 1) return 1;
    else return (fibonacci(n-2)+fibonacci(n-1));
}
```

Fibonacci数の比

実行結果

```
fib(2) /fib(1) = 1.000000
fib(3) /fib(2) = 2.000000
fib(4) /fib(3) = 1.500000
fib(5) /fib(4) = 1.666667
fib(6) /fib(5) = 1.600000
fib(7) /fib(6) = 1.625000
fib(8) /fib(7) = 1.615385
fib(9) /fib(8) = 1.619048
…
fib(36) /fib(35) = 1.618034
fib(37) /fib(36) = 1.618034
fib(38) /fib(37) = 1.618034
fib(39) /fib(38) = 1.618034
fib(40) /fib(39) = 1.618034
fib(41) /fib(40) = 1.618034
fib(42) /fib(41) = 1.618034
```

- $\frac{1+\sqrt{5}}{2} = 1.618033\ldots$
- nが大きくなるほど、再帰の回数が多くなる
- 結果として、計算時間が長くなる

プログラムにおける時間計測方法（参考）

- clock_t：long int (8 byte)
- clock()：実行中のプログラムが起動されてから、現在に至るまでのプロセッサ時間を取得
- 精度は10ms程度であるため、短い処理の場合は複数回分を計測

```
#include <stdio.h>
#include <time.h>

int main (void){
    int  i;
    double y;
    clock_t start, end;

        start = clock();
        計測対象の処理
        end = clock();

        printf("Elapsed %ld (ms) ¥n", end-start);
        printf("T(秒)= %f ¥n", (double)(end-start)/CLOCKS_PER_SEC);
```

5/12(金) 8:50~12:25

- 第9回「再帰」

1. 再帰について
2. 階乗の計算
3. Fibonacci 数の計算
4. 指数関数の近似（テイラー展開）
5. 最大公約数
6. 型変換について（補足）

指数関数の近似（テイラー展開）

- テイラー展開

$$f(x) = f(a) + \frac{f'(a)}{1!}(x-a) + \frac{f''(a)}{2!}(x-a)^2 + ..$$

- 指数関数e^xを、マクローリン展開($a = 0$としたときのテイラー展開)で近似

$$e^x \sim 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots + \frac{x^n}{n!}$$

- 上式右辺は、以下の関数$f_x(n)$を再帰的に呼び出すことで計算可能である

$$f_x(n) = f_x(n-1) + \frac{x^n}{n!},\ f_x(0) = 1$$

5/12(金) 8:50~12:25

- 第9回「再帰」

1. 再帰について
2. 階乗の計算
3. Fibonacci 数の計算
4. 指数関数の近似（テイラー展開）
5. 最大公約数
6. 型変換について（補足）

最大公約数

- 最大公約数：複数の数があるとき，それらがすべて割り切れる数の最大値
  - 数：45, 90, 120
  - 公約数：1, 3, 5, 15
  - 最大公約数：15

最大公約数

- 二つの整数の最大公約数 (gcd: greatest common divisor) を求める関数は以下で示される

```
int gcd ( int a, int b ) {
    int i;
    for ( i=a;  i>0;  i--) {
        if ( a % i == 0 & b % i == 0 )  break;
    }
   return i;
}
```

最大公約数

- n 個の数の最大公約数を求めるには，「n 番目の数」と「$1 \sim n-1$ 番目の数の最大公約数」の二つの数の最大公約数を求めればいい

```
gcd( num[n], multi_gcd(n-1) )
```

型変換について

- 文字型：char型
  - 英数字1文字，1 byte (-128~127)
- 整数型：int型
  - 整数，1 byte
  - (short int (2 byte, -32768~32767))
  - (unsigned int, (4 byte, 0~65535))
- 実数型（単精度）：float型
  - 実数，4 byte ($\pm 3.4 \times 10^{-38} \sim \pm 3.4 \times 10^{38}$)
- 実数型（倍精度）：double型
  - 実数，8 byte ($\pm 1.7 \times 10^{-308} \sim \pm 1.7 \times 10^{308}$)

型変換について

- 異なるデータ型の混合計算の場合，精度の高い方へ型変換がされる
  - short int → int → float → double → long double

```
int a=2;
float b=3;
float c;

c = a/b;
printf("%f", c);  → 結果：0.666667
```

型変換について

- 同じ型の場合は計算結果も同じ型のまま

```
int a=2, b=3;
float c;

c = a/b;
printf("%f", c);  → 結果：0.000000
```

型変換について

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    short si1, si2, si3;
    int i3;

    si1 = 100, si2 = 32767, si3 = 33000, i3 = 33000;
    printf("si1= %d, si2=%d, si3=%d, i3=%d¥n", si1, si2, si3, i3);

    return 0;
}
```

```
si1= 100, si2=32767, si3=-32536, i3=33000
```

型の範囲を超えるとおかしな値になる

型変換について

```c
#include <stdio.h>
#include <stdlib.h>

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

    i1 = 100.25, f1 = 100.25;
    i2 = f1, f2 = i1;
    printf("i1=%d, f1= %f, i2=%d, f2=%f¥n", i1, f1, i2, f2);

    return 0;
}
```

```
i1=100, f1= 100.250000, i2=100, f2=100.000000
```

より小さな型に代入すると一部切捨てになる

型変換について

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
    float f1, f2, f3;
    int i1, i2, i3, i4;

    i1 = 2, i2 = 3, i3 = i1 * i2;
    printf("i1=%d, i2=%d, i3=%d ¥n", i1, i2, i3);
    i4 = i1 / i2, f1 = i1 / i2, f2 = (float)i1 / i2, f3 = i1 / (float)i2;
    printf("i4=%d, f1=%f, f2=%f, f2=%f ¥n", i4, f1, f2, f3);

    return 0;
}
```

```
i1=2, i2=3, i3=6
i4=0, f1=0.000000, f2=0.666667, f2=0.666667
```

同じ型の計算の場合，答えも同じ型

型変換について

```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{

    int i1, i2;
    char c1, c2;

    c1 = 'a', c2 = 'b', i1 = 'a', i2 = 'b';
    printf("c1= %c, c2= %c, c1int= %d, c2int= %d¥n", c1, c2, i1, i2);

    return 0;
}
```

```
c1= a, c2= b, c1int= 97, c2int= 98
```

文字型も中身は数値（ASCIIコードなど）

ASCII文字コード

文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進	文字	10進	16進
NUL	0	00	DLE	16	10	SP	32	20	0	48	30	@	64	40	P	80	50	`	96	60	p	112	70
SOH	1	01	DC1	17	11	!	33	21	1	49	31	A	65	41	Q	81	51	a	97	61	q	113	71
STX	2	02	DC2	18	12	"	34	22	2	50	32	B	66	42	R	82	52	b	98	62	r	114	72
ETX	3	03	DC3	19	13	#	35	23	3	51	33	C	67	43	S	83	53	c	99	63	s	115	73
EOT	4	04	DC4	20	14	$	36	24	4	52	34	D	68	44	T	84	54	d	100	64	t	116	74
ENQ	5	05	NAK	21	15	%	37	25	5	53	35	E	69	45	U	85	55	e	101	65	u	117	75
ACK	6	06	SYN	22	16	&	38	26	6	54	36	F	70	46	V	86	56	f	102	66	v	118	76
BEL	7	07	ETB	23	17	'	39	27	7	55	37	G	71	47	W	87	57	g	103	67	w	119	77
BS	8	08	CAN	24	18	(	40	28	8	56	38	H	72	48	X	88	58	h	104	68	x	120	78
HT	9	09	EM	25	19	)	41	29	9	57	39	I	73	49	Y	89	59	i	105	69	y	121	79
LF*	10	0a	SUB	26	1a	*	42	2a	:	58	3a	J	74	4a	Z	90	5a	j	106	6a	z	122	7a
VT	11	0b	ESC	27	1b	+	43	2b	;	59	3b	K	75	4b	[	91	5b	k	107	6b	{	123	7b
FF*	12	0c	FS	28	1c	,	44	2c	<	60	3c	L	76	4c	\ ¥	92	5c	l	108	6c	|	124	7c
CR	13	0d	GS	29	1d	-	45	2d	=	61	3d	M	77	4d				m	109	6d	}	125	7d
SO	14	0e	RS	30	1e	.	46	2e	>	62	3e	N	78	4e	]	93	5d	n	110	6e	~	126	7e
SI	15	0f	US	31	1f	/	47	2f	?	63	3f	O	79	4f	^	94	5e	o	111	6f	DEL	127	7f
															_	95	5f						

型変換について

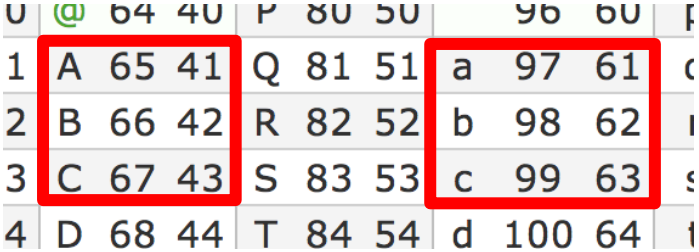
```c
#include <stdio.h>
#include <stdlib.h>

int main(void)
{

    char c3, c4, c5, c6, c7, c8;

    c3 = 'a' + 1, c4 = 'a' + 2, c5 = 98, c6 = 99, c7 = 'b' - 32, c8 =
'c' - 32;
    printf("c3=%c, c4=%c, c5=%c, c6=%c, c7=%c, c8=%c¥n", c3, c4, c5,
c6, c7, c8);

    return 0;
}
```



0	@	64	40	P	80	50		96	60
1	A	65	41	Q	81	51	a	97	61
2	B	66	42	R	82	52	b	98	62
3	C	67	43	S	83	53	c	99	63
4	D	68	44	T	84	54	d	100	64

```
c3=b, c4=c, c5=b, c6=c, c7=B, c8=C
```

文字型変数の数値的演算