



C言語

菊地

注意

- 今日すごく長丁場になりそう、先にトイレ行ったり水持ってきたり、飯持ってきたりしていいよ。
- paizaの開け方みんなもうわかるよね？？？

お品書き

- 前回までのおさらい
- 配列の補足
- switch文の補足
- 関数プロトタイプ宣言について
- すべての知識を使った演習

第一回

printf

for文

While文

do…while文

第一回

printf

←標準出力関数、
第一引数を出力

for文 ←繰り返す回数がわかっている
ときに使う

While文 ←繰り返す回数がわからない
ときに使う

do…while文 ←1回以上繰り返したいと
きに使う

第二回

間違えやすい演算子まとめ

演算子	意味
$x=y$	x に y を代入
$x==y$	x と y は等しいか？
$x>=y$	x は y 以上
$x(\text{算術演算子})=y$	$x(\text{算術演算子})y$ の結果を x に代入

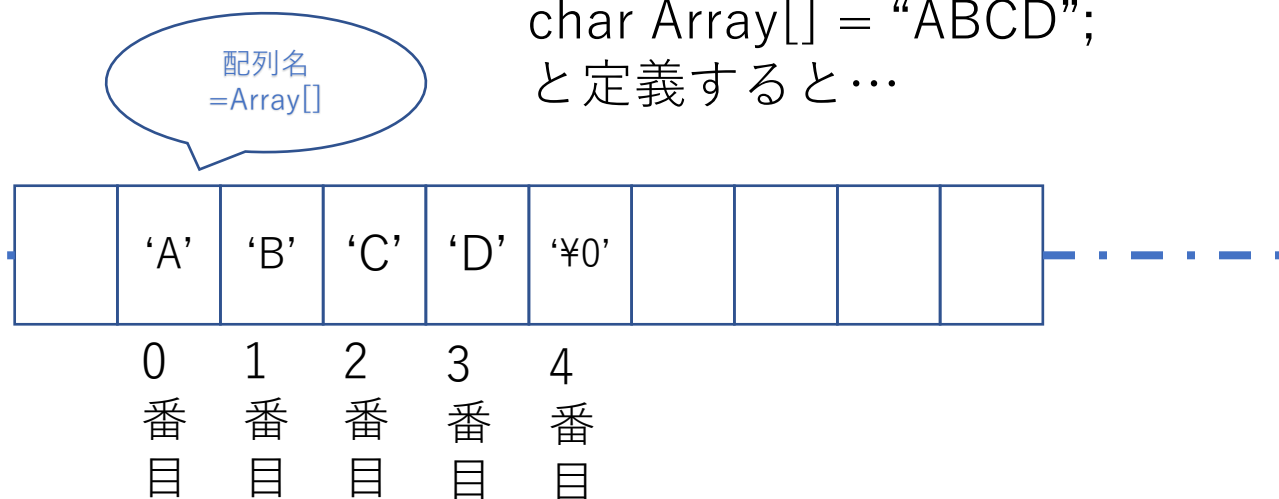
インクリメント・デクリメント演算子

演算子	意味
$++x$	(前置演算)xに 1足してxを評価
$x++$	(後置演算)xを 評価してからxに1足す
$--x$	(前置演算)xから 1引いてxを評価
$x--$	(後置演算)xを 評価してからxから1引く

第三回 配列

例えば

char Array[] = "ABCD";
と定義すると…



・ 定義

型名 配列名[配列の要素の数]

・ 要素の取り出し方

配列名[要素の先頭からの番号]

・ 注意点

添え字の始まりは0から。最後は要素の最大値-1までしかない。

**初期値は不定！
自分で入れないといけない！**

第三回 配列

- ・ 定義

型名 配列名[**配列の要素の数**]

第三回 配列

・定義

型名 配列名[**配列の要素の数**]

1.要素数を定義せず、代入する値によって定義する場合

(要素数は文字列の大きさによる)

文字配列の場合null文字は代入された文字列のすぐ後ろに来る

2.要素数を定義し、代入する文字列を後から決める場合。(一括代入ができない)

初期値は不定。

3.要素数を定義し、最初から文字列を入れる場合。(後から一括代入はできないが、定義時に一括代入が可能。)

第三回 配列

・定義

型名 配列名[**配列の要素の数**]

1.要素数を定義せず、代入する値によって定義する場合

(要素数は文字列の大きさによる)

文字配列の場合null文字は代入された文字列のすぐ後ろに来る

2.要素数を定義し、代入する文字列を後から決める場合。(一括代入ができない)

初期値は不定。

3.要素数を定義し、最初から文字列を入れる場合。(後から一括代入はできないが、定義時に一括代入が可能。)

- 1.要素数を定義せず、代入する値によって定義する場合

- (要素数は文字列の大きさによる)
- 文字型配列の場合null文字は代入された文字列のすぐ後ろに来る

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      → /*配列の要素数を決めていない*/
8      → /*代入した値によって配列の大きさが定義される*/
9      → char s[] = "Hello!";
10     → int d[] = {1,2,3};
11
12     → /*main関数の戻り値*/
13     → return 0;
14 }
```

• 1.要素数を定義せず、代入する値によって定義する場合

- (要素数は文字列の大きさによる)
- 文字型配列の場合null文字は代入された文字列のすぐ後ろに来る

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めていない*/
8      /*代入した値によって配列の大きさが定義される*/
9      char s[] = "Hello!";
10     int d[] = {1,2,3};
11
12     /*要素数も値も定義しないのはエラーになる!*/
13     char s1[];
14
15     /*main関数の戻り値*/
16     return 0;
17 }
```

• 1.要素数を定義せず、代入する値によって定義する場合

- (要素数は文字列の大きさによる)
- 文字型配列の場合null文字は代入された文字列のすぐ後ろに来る

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めていない*/
8      /*代入した値によって配列の大きさが定義される*/
9      char s[] = "Hello!";
10     int d[] = {1,2,3};
11     int i;
12
13     /*s[6] = "Hello!"*/
14     printf("%s\n",s);
15
16     /*null文字まで表示させてみる。*/
17     for (i = 0; i <= 6; i++)
18     |   printf("s[%d] = %c\n",i,s[i]);
19
20     /*d[3] = {1,2,3}*/
21     /*null文字の概念はないけれどそこまでいかせてみる。*/
22     for (i = 0; i <= 3; i++)
23     |   printf("d[%d] = %d\n",i,d[i]);
24     /*main関数の戻り値*/
25     return 0;
26 }
```

1.要素数を定義せず、代入する値によって定義する場合

(要素数は文字列の大きさによる)

文字型配列の場合null文字は代入された文字列のすぐ後ろに来る

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めていない*/
8      /*代入した値によって配列の大きさが定義される*/
9      char s[] = "Hello!";
10     int d[] = {1,2,3};
11     int i;
12
13     /*s[6] = "Hello!"*/
14     printf("%s\n",s);
15
16     /*null文字まで表示させてみる。*/
17     for (i = 0; i <= 6; i++)
18     {
19         printf("s[%d] = %c\n",i, s[i]);
20     }
21     /*d[3] = {1,2,3}*/
22     /*null文字の概念はないけれどそこまでいかせてみる。*/
23     for (i = 0; i <= 3; i++)
24     {
25         printf("d[%d] = %d\n",i, d[i]);
26     }
27     /*main関数の戻り値*/
28     return 0;
29 }
```

-
- 1.要素数を定義せず、代入する値によって定義する場合
 - (要素数は文字列の大きさによる)
 - 文字配列の場合null文字は代入された文字列のすぐ後ろに来る

Hello!

s[0] = H

s[1] = e

s[2] = l

s[3] = l

s[4] = o

s[5] = !

s[6] =

d[0] = 1

d[1] = 2

d[2] = 3

d[3] = 1818576896

.

1.要素数を定義せず、代入する値によって定義する場合

(要素数は文字列の大きさによる)

文字配列の場合null文字は代入された文字列のすぐ後ろに来る

Hello!

s[0] = H

s[1] = e

s[2] = l

s[3] = l

s[4] = o

s[5] = !

s[6] =

d[0] = 1

d[1] = 2

d[2] = 3

d[3] = 1818576896

.....

第三回 配列

・定義

型名 配列名[**配列の要素の数**]

1.要素数を定義せず、代入する値によって定義する場合

(要素数は文字列の大きさによる)

文字配列の場合null文字は代入された文字列のすぐ後ろに来る

2.要素数を定義し、代入する文字列を後から決める場合。(一括代入ができない)

3.要素数を定義し、最初から文字列を入れる場合。(後から一括代入はできないが、定義時に一括代入が可能。)

- 2.要素数を定義し、代入する文字列を後から決める場合。

- (一括代入ができない)

- 初期値は不定。

- 文字型配列の場合は、値を入れるとそのすぐ後ろに¥0が代入される。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      → /*配列の要素数を決めた。*/
8      → /*この要素数を後から変更することはできない。*/
9      → char s[6];
10     → int d[3];
11
12     → /*main関数の戻り値*/
13     → return 0;
14 }
```

-
- 2.要素数を定義し、代入する文字列を後から決める場合。
 - (一括代入ができない)
 - 初期値は不定。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めた。*/
8      /*この要素数を後から変更することはできない。*/
9      char s[6];
10     int d[3];
11
12     s = "Hello!";
13     d = {1, 2, 3};
14     /*main関数の戻り値*/
15     return 0;
16 }
```

- 2.要素数を定義し、代入する文字列を後から決める場合。

- (一括代入ができない)

→代替案

- 初期値は不定。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めた。*/
8      /*この要素数を後から変更することはできない。*/
9      char s[6];
10     int d[3];
11
12     // s = "Hello!";
13     // d = {1,2,3};
14     s[0] = 'H';
15     s[1] = 'e';
16     s[2] = 'l';
17     s[3] = 'l';
18     s[4] = 'o';
19     s[5] = '!';
20     s[6] = '\0'; /*自分でnull文字を代入する必要がある!*/
21     /*値の確認*/
22     printf("%s\n", s);
23
24     /*dの値は連続してるのでこんな風に入れることもできる!*/
25     for (int i=1; i<4; i++)
26     {
27         d[i-1] = i;
28         printf("d[%d] = %d\n", i-1, d[i-1]);
29     }
30     /*main関数の戻り値*/
31     return 0;
32 }
```

2.要素数を定義し、代入する文字列を後から決める場合。

(一括代入ができない)

→代替案

自分でnull文字を入れる必要があることに注意！

初期値は不定。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めた。*/
8      /*この要素数を後から変更することはできない。*/
9      char s[6];
10     int d[3];
11
12     // s = "Hello!";
13     // d = {1,2,3};
14     s[0] = 'H';
15     s[1] = 'e';
16     s[2] = 'l';
17     s[3] = 'l';
18     s[4] = 'o';
19     s[5] = '!';
20     s[6] = '\0'; /*自分でnull文字を代入する必要がある！*/
21     /*値の確認*/
22     printf("%s\n", s);
23
24     /*dの値は連続してるのでこんな風に入れることもできる！*/
25     for (int i=1; i<4; i++)
26     {
27         d[i-1] = i;
28         printf("d[%d] = %d\n", i-1, d[i-1]);
29     }
30     /*main関数の戻り値*/
31     return 0;
32 }
```

第三回 配列

・定義

型名 配列名[**配列の要素の数**]

1.要素数を定義せず、代入する値によって定義する場合

(要素数は文字列の大きさによる)

文字配列の場合null文字は代入された文字列のすぐ後ろに来る

2.要素数を定義し、代入する文字列を後から決める場合。(一括代入ができない)

初期値は不定。

3.要素数を定義し、最初から文字列を入れる場合。(後から一括代入はできないが、定義時に一括代入が可能。)

- 3.要素数を定義し、最初から文字列を入れる場合。

- (後から一括代入はできないが、定義時に一括代入が可能。)

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めた*/
8      /*代入した値によらず定義した要素数の範囲は値を入れることができる。*/
9      char s[10] = "Hello!";
10     int d[20] = {1,2,3};
11
12     /*main関数の戻り値*/
13     return 0;
14 }
```

- 3.要素数を定義し、最初から文字列を入れる場合。

- (後から一括代入はできないが、定義時に一括代入が可能。)

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めた*/
8      /*代入した値によらず定義した要素数の範囲は値を入れることができる。*/
9      char s[10] = "Hello!";
10     int d[20] = {1,2,3};
11
12     /*一括代入はできない!*/
13     s = "ABCDE";
14     d = {4,5,6};
15
16     /*main関数の戻り値*/
17     return 0;
18 }
```

3.要素数を定義し、最初から文字列を入れる場合。

(後から一括代入はできないが、定義時に一括代入が可能。)

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めた*/
8      /*代入した値によらず定義した要素数の範囲は値を入れることができる。*/
9      char s[10] = "Hello!";
10     int d[20] = {1,2,3};
11     int i;
12
13     /*一括代入はできない!*/
14     // s = "ABCDE";
15     // d = {4,5,6};
16
17     /*値を書き換えるときは一つずつ*/
18     d[0] = 4;
19     d[1] = 5;
20     d[2] = 6;
21     /*文字配列でも連続ならこうやって書き換えることができる!*/
22     for (i = 0; i < 5; i++)
23     {
24         s[i] = 'A' + i;
25     }
26     /*null文字の代入*/
27     s[i] = '\0';
28
29     /*値の確認*/
30     printf("%s\n", s);
31     /*main関数の戻り値*/
32     return 0;
33 }
```

3.要素数を定義し、最初から文字列を入れる場合。

(後から一括代入はできないが、定義時に一括代入が可能。)

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*配列の要素数を決めた*/
8      /*代入した値によらず定義した要素数の範囲は値を入れることができる。*/
9      char s[10] = "Hello!";
10     int d[20] = {1,2,3};
11     int i;
12
13     /*一括代入はできない!*/
14     // s = "ABCDE";
15     // d = {4,5,6};
16
17     /*値を書き換えるときは一つずつ*/
18     d[0] = 4;
19     d[1] = 5;
20     d[2] = 6;
21     /*文字配列でも連続ならこうやって書き換えることができる!*/
22     for (i = 0; i < 5; i++)
23     {
24         s[i] = 'A' + i;
25     }
26     /*null文字の代入*/
27     s[i] = '\0';
28
29     /*値の確認*/
30     printf("%s\n", s);
31     /*main関数の戻り値*/
32     return 0;
33 }
```

Asciiってなーに？

- 半角英数字、記号、制御記号（空白等）等の**ASCII**コードで表現される**文字**です。 ひらがなや漢字などの日本語（**2バイト文字**）は含まれません。
- Asciiコード表(https://www.k-cube.co.jp/wakaba/server/ascii_code.html)

Q1.

- 小文字の'a'と、大文字の'A'のascii数値を表示してみよう。

ヒント: `printf("%d¥n",'文字');` で出力された値をasciiコード表の10進数の値と比べてみよう！

(制限時間:3分)

Q1.

- 小文字の'a'と、大文字の'A'のascii数値を表示してみよう。

ヒント: printf("%d¥n", '文字'); で出力された値をasciiコード表の10進数の値と比べてみよう！

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      char ch1 = 'a';
8      char ch2 = 'A';
9
10     printf("%c = %d¥n", ch1, ch1);
11     printf("%c = %d¥n", ch2, ch2);
12
13     /*main関数の戻り値*/
14     return 0;
15 }
```

Q1.

- 小文字の'a'と、大文字の'A'のascii数値を表示してみよう。

ヒント: printf("%d¥n", '文字'); で出力された値をasciiコード表の10進数の値と比べてみよう！

出力結果！


a = 97
A = 65

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      char ch1 = 'a';
8      char ch2 = 'A';
9
10     printf("%c = %d¥n", ch1, ch1);
11     printf("%c = %d¥n", ch2, ch2);
12
13     /*main関数の戻り値*/
14     return 0;
15 }
```



第四回 条件文

- if文
- if…else文
- if…else if文



第四回 条件文

- **if文**

if (条件){处理内容}

- **if…else文**

if (条件){处理内容} else{处理内容}

- **if…else if文**

if (条件){处理内容} else if (条件){处理内容}

if文

1. 条件文を書く
2. その下に条件を満たしたときに行う処理を書く。
3. 条件を満たすときのみ処理を行う。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*説明のための変数*/
8      int i = 10;
9
10     /*1. if(条件(なんでも良子))*/
11     if (i > 10)
12     {
13         /*2. 処理の内容*/
14         printf("i is more than 10");
15     }
16
17     /*main関数の戻り値*/
18     return 0;
19 }
```

if...else文

1. if文を書く。
2. if文の条件に当てはまらない場合だけ、elseで指定された処理を行う。

elseには条件をかけない
よ！！！！

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*説明のための変数*/
8      int i = 10;
9
10     /*1. if(条件(なんでも良子))*/
11     if (i > 10)
12     {
13         /*1.5.処理の内容*/
14         printf("i is more than 10");
15     }
16     /*2. else*/
17     else
18     {
19         /*2.5処理の内容*/
20         printf("i is not more than 10");
21     }
22
23     /*main関数の戻り値*/
24     return 0;
```

if...else if...文

1. if文を書く
2. ifで分岐させたくない条件等をelse ifの後ろに書く。
3. else ifの条件に当てはまる場合のみ else ifで指定された内容进行处理。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*説明のための変数*/
8      int i = 7;
9
10     /*1. if(条件(なんでも良い))*/
11     if (i > 10)
12     {
13         /*1.5. 処理の内容*/
14         printf("i is more than 10");
15     }
16     /*2. else if (条件文)*/
17     else if (i > 3 && i < 8)
18     {
19         /*2.5 処理の内容*/
20         printf("i is more than 4 and i is less than 8");
21     }
22     /*3. else*/
23     else
24     {
25         /*3.5 処理の内容*/
26         printf("i is not more than 10");
27     }
28
29     /*main関数の戻り値*/
30     return 0;
31 }
```

Q2.

- 任意の一文字(先ほどのch1のようなもの)に対して、以下のように条件を定義し、出力を確認せよ。

- 文字が数字の場合 -> number! と出力
- 文字が英小文字の場合 -> small char! と出力
- 文字が英大文字の場合 -> large char! と出力
- そのほかの場合 -> else! と出力

(例)

入力 char c = 'A'; の場合

出力は printf("large char !");

(制限時間15分)

- 任意の一文字(先ほどのch1のようなもの)に対して、以下のように条件を定義し、出力を確認せよ。

- 文字が数字の場合 -> number!と出力
- 文字が英小文字の場合 -> small char!と出力
- 文字が英大文字の場合 -> large char!と出力
- そのほかの場合 -> else! と出力
- (例)
- 入力 char c = 'A';の場合
- 出力はprintf("large char !");

- (制限時間15分)

Q2.

- ヒント
- さっきのAscii表を見てね！文字は、それそのもので数字として比べることができるよ！
- 例えば以下のように…
- `char c = 'r'`
- `'a' <= c <= 'z' → True`
- `'A' <= c <= 'Z' → False`

Switch文

1.switch(条件式)

2.{処理内容}

3.case文を書く

セットで**break;**が必要！

4.default文を書く

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*任意の一文字を定義！*/
8      int d = 2;
9
10     /*1.switch(条件式)*/
11     switch(d)
12     {
13         /*2.{処理内容}*/
14         /*3.case文*/
15         case 1:
16             printf("d = 1\n");
17             break;
18         case 2:
19             printf("d = 2\n");
20             break;
21         /*4.default文*/
22         default:
23             printf("d is not one and two!");
24     }
25
26     /*main関数の戻り値*/
27     return 0;
28 }
```

Switch文

1.switch(条件式)

2.{処理内容}

3.case文を書く

セットで**break;**が必要！

4.default文を書く

出力！

d = 2

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*任意の一文字を定義！*/
8      int d = 2;
9
10     /*1.switch(条件式)*/
11     switch(d)
12     {
13         /*2.{処理内容}*/
14         /*3.case文*/
15         case 1:
16             printf("d = 1\n");
17             break;
18         case 2:
19             printf("d = 2\n");
20             break;
21         /*4.default文*/
22         default:
23             printf("d is not one and two!");
24     }
25
26     /*main関数の戻り値*/
27     return 0;
28 }
```


Switch文

セットで**break;**が必要！

break;がないと当てはまった
case文の後ろを全部実行する…

出力！

```
d = 2
```

```
d is not one and two!
```

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int main(void)
6  {
7      /*任意の一文字を定義！*/
8      int d = -2;
9
10     /*1.switch(条件式)*/
11     switch(d)
12     {
13         /*2.{処理内容}*/
14         /*3.case文*/
15         case 1:
16             printf("d = 1\n");
17             // break;
18         case 2:
19             printf("d = 2\n");
20             // break;
21         /*4.default文*/
22         default:
23             printf("d is not one and two!");
24     }
25
26     /*main関数の戻り値*/
27     return 0;
28 }
```

関数プロトタイプ宣言ってなーに？

- 前回やったstrlenも同じ。
- 自分で作った関数を宣言すること！
- 一歩間違えるとプログラムが迷子…

やり方

1. **呼び出し元の関数の前**に定義を書く。
2. それを呼び出す。
3. おしまい。

関数プロトタイプ宣言

1. 呼び出し元の関数の**前**に定義を書く。

2. それを呼び出す。

3. おしまい。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*1.今回はmain関数で呼び出すので、その前で定義!*/
5  /*戻り値の型 関数名 (引数の型)*/
6  void print_hello(void)
7  {
8      /*処理内容を書く*/
9      printf("Hello!\n");
10 }
11
12 /*戻り値の型 関数名 (引数の型)*/
13 int main(void)
14 {
15     /*2.呼び出す!*/
16     print_hello();
17
18     /*main関数の戻り値*/
19     return 0;
20 }
```

関数プロトタイプ宣言

1. 呼び出し元の関数の**前**
に定義を書く。

2. それを呼び出す。

3. おしまい。

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*1. 今回はmain関数で呼び出すので、その前で定義！*/
5  /*戻り値の型 関数名 (引数の型)*/
6  void print_hello(void)
7  {
8      /*処理内容を書く*/
9      printf("Hello!\n");
10 }
11
12 /*戻り値の型 関数名 (引数の型)*/
13 int main(void)
14 {
15     /*2. 呼び出す！*/
16     print_hello();
17
18     /*main関数の戻り値*/
19     return 0;
20 }
```

演習問題！

Q3.関数プロトタイプ宣言を使って、
前回やってみたmy_strlenを定義してみよう！

定義:

名前 → my_strlen

戻り値 → int型

引数 → char s[] (文字型配列)

処理内容:

s[]の長さを数えて返す。

(制限時間10分)

演習問題！

ヒント

(‘¥0’に行くまでインクリメントすればよかった…よね？)

Q3.関数プロトタイプ宣言を使って、前回やってみたmy_strlenを定義してみよう！

定義:

名前 → my_strlen

戻り値 → int型

引数 → char s[] (文字型配列)

処理内容:

s[]の長さを数えて返す。

(制限時間10分)

答え

- できたかな？
- 確認するところまでいかなくても大丈夫！

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型と名前)*/
5  int my_strlen(char s[])
6  {
7      /*文字数を数えるための変数*/
8      int i = 0;
9
10     /*回数がわからない時の繰り返し構文*/
11     while (s[i] != '\0')
12     {
13         i++;
14     }
15     return (i);
16 }
17
18 /*戻り値の型 関数名 (引数の型)*/
19 int main(void)
20 {
21     /*確かめるための変数*/
22     char s[10] = "You!";
23
24     /*確認の仕方！*/
25     printf("s's length = %d", my_strlen(s));
26
27     /*main関数の戻り値*/
28     return 0;
```


演習問題！

Q4.関数プロトタイプ宣言を使って、
my_is_lowerを定義してみよう！

定義:

名前	→	my_is_lower
戻り値	→	int型 (0か1を返す)
引数	→	char c (1文字)

処理内容:

cが小文字なら1, そのほかなら0を返す
(制限時間10分)

演習問題！

ヒント

一文字は数字として
比べられた…。って
ことは `'a' <= c <=`
`'z'` が条件になるはず

Q4.関数プロトタイプ宣言を使って、
`my_is_lower`を定義してみよう！

定義:

名前	→	<code>my_is_lower</code>
戻り値	→	int型 (0か1を返す)
引数	→	char c (1文字)

処理内容:

cが小文字なら1, そのほかなら0を返す
(制限時間10分)

答え

- できたかな？
- 確認するところまでいかななくても大丈夫！

c is lower!

出力！

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int my_is_lower(char c)
6  {
7      /*小文字の時は1を返す*/
8      if ('a' <= c && c <= 'z')
9          return 1;
10     /*それ以外は0を返す！*/
11     else
12         return 0;
13 }
14
15 /*戻り値の型 関数名 (引数の型)*/
16 int main(void)
17 {
18     /*確かめるための変数*/
19     char c = 'i';
20
21     /*確認の仕方！*/
22     if (my_is_lower(c) == 1)
23         printf("c is lower!\n");
24     else
25         printf("c is no lower!\n");
26
27     /*main関数の戻り値*/
28     return 0;
29 }
```

答え2

変態チックな答え方も全然あり

c is lower!

出力！

```
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int my_is_lower(char c)
6  {
7      /*小文字の時は1を返す*/
8      /*それ以外は0を返す!*/
9      return ('a' <= c && c <= 'z');
10 }
11
12 /*戻り値の型 関数名 (引数の型)*/
13 int main(void)
14 {
15     /*確かめるための変数*/
16     char c = 'i';
17
18     /*確認の仕方!*/
19     if (my_is_lower(c) == 1)
20     {
21         printf("c is lower!\n");
22     }
23     else
24     {
25         printf("c is no lower!\n");
26     }
27
28     /*main関数の戻り値*/
29     return 0;
30 }
```

演習問題！

Q5. さっきのQ4を使って文字列の小文字を大文字にしよう！

```
char s[] = "AbcDw";
```

となっているとき、printfで出したい出力は以下の通り！

ABCDW

ただし、小文字を大文字にするには小文字から('a' - 'A')を引けばよい。

(制限時間15分)

答え

- できたかな？
- これで今季のc言語勉強会はおわり！
- お疲れさまでした！

```
1  /*参照するライブラリを指定*/
2  #include <stdio.h>
3
4  /*戻り値の型 関数名 (引数の型)*/
5  int my_is_lower(char c)
6  {
7      /*小文字の時は1を返す*/
8      if ('a' <= c && c <= 'z')
9          return 1;
10     /*それ以外は0を返す！*/
11     else
12         return 0;
13 }
14
15 /*戻り値の型 関数名 (引数の型)*/
16 int main(void)
17 {
18     /*確かめるための変数*/
19     char s[] = "AbcDw";
20     int i = 0;
21
22     printf("%s\n", s);
23     /*確認の仕方！*/
24     while (s[i] != '\0')
25     {
26         if (my_is_lower(s[i]) == 1)
27             s[i] = s[i] - ('a' - 'A');
28         i++;
29     }
30     printf("%s\n", s);
31     /*main関数の戻り値*/
32     return 0;
33 }
```

最後に

プログラミングはこれがよいという趣向はあっても、絶対そうでないという答えはない。

無限にやり続けられるパズルゲームと同じ。

パズドラより、こっちの方が面白いと思う人は少ないかもしれないけど(私はパズドラ好き。)、ちよつとでも興味を持ってくれたらうれしい。

FIN

