キーボードから入力

値の入力を扱ったプログラミングができるようになることを目標にします.

標準入力(キーボード)から文字を入力し，文字の表示や数値計算をするプログラムを学んでいきます.
原理まで理解しようとするとかなり難しい章になります.

キーボードから入力するプログラムを作るときに,
「コードを同じように真似して記述するれば使える」
ことを目標にしてください.

（教科書p.71）

キーボードから入力する

教科書p.71 List3-5は名前と年齢をたずねて，その名前と10年後の年齢を答えるプログラムです．まずは，実行例の確認です．

実行例（教科書p.73）

あなたの名前を入力してください。	←表示
不老長寿	←入力
不老長寿さん、こんにちちは。	←表示
年齢を入力してください。	←表示
120	←入力
今 120 歳とすると，10年後は 130 歳ですね。	←表示

では，この入力はどのように行っているのか見てみましょう．

（教科書p.71）

List3-5

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define BUFFER_SIZE 256
 5:
 6:  int main(void);
 7:  void get_line(char *buffer, int size);
 8:
 9:  int main(void)
10:  {
11:      char buffer[BUFFER_SIZE];
12:      int age;
13:
14:      printf("あなたの名前を入力してください。¥n");
15:      get_line(buffer, BUFFER_SIZE);
16:      printf("%sさん、こんにちは。¥n", buffer);
17:
18:      printf("年齢を入力してください。¥n");
19:      get_line(buffer, BUFFER_SIZE);
20:      age = atoi(buffer);
21:      printf("いま %d 歳とすると、10年後は %d 歳ですね。¥n", age, age + 10);
22:
23:      return 0;
24:  }
```

（表示が小さいので教科書を見てください）

関数のプロトタイプ宣言
第8章で説明するので，今はこのまま書いてください

入力待ちの状態の個所
自作関数get_line内で
キーボードから文字列を得ています

List3-5　続き

```
25:
26:  void get_line(char *buffer, int size)
27:  {
28:      if (fgets(buffer, size, stdin) == NULL) {
29:          buffer[0] = '¥0';
30:          return;
31:      }
32:
33:      for (int i = 0; i < size; i++) {
34:          if (buffer[i] == '¥n') {
35:              buffer[i] = '¥0';
36:              return;
37:          }
38:      }
39:  }
```

キーボードから入力した文字列を配列に格納するための自作関数

関数fgets
ファイルから文字列を得る関数．stdin(標準入力:キーボード)から文字列を得ています
プログラムで実際に入力待ちになっている個所がここです

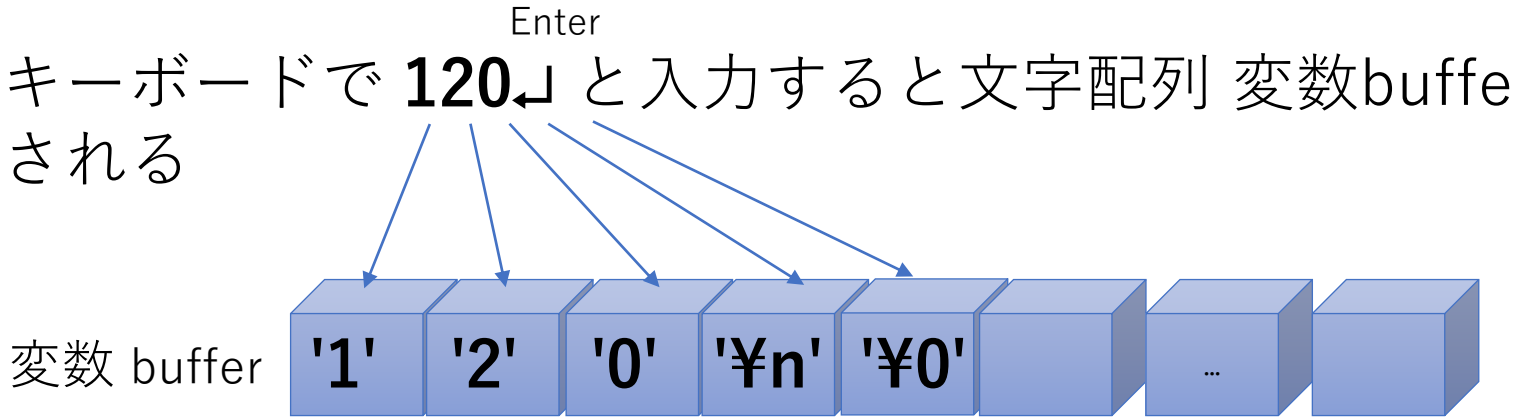
まだ習っていないものがいくつもあるので，いまは中身は理解できなくてもよいです．
7行目と26～39行目をセットにして，そのまま利用できるようになってください．
今後の入力を伴うプログラムでは，コピー・貼り付けなどして利用してください．

関数fgets

fgets(buffer, size, stdin);

stdin(標準入力，通常はキーボード)から，改行文字('¥n')が現れるか，ファイルの終端に達するか，または，読み込んだ文字数が(size – 1)になるまで文字列を読み込み，文字配列(buffer)に格納する．
（配列の詳細については第9章で説明します．ここでは文字を1つ入れられる箱が並んでいると思ってください）

キーボードで **120↵** と入力すると文字配列 変数bufferに文字が1個ずつ格納される



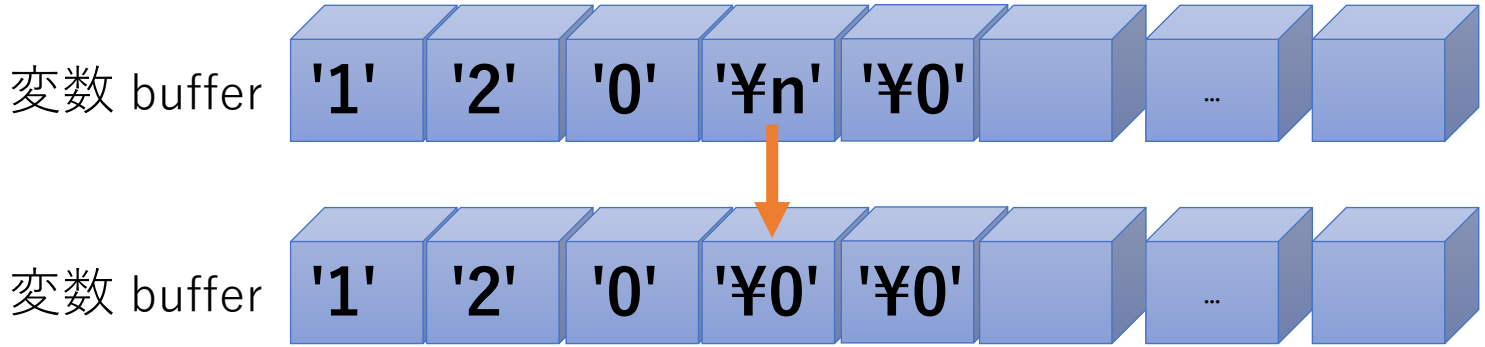
↑ 文字列の終わりの印(詳細は第9章にて)

（教科書p.77）

自作関数get_line

get_lineは標準で用意されている関数ではなく，この教科書の作者が自作した関数です．

先ほどの関数fgetsで文字配列に格納すると，改行文字('¥n')も入るので，それをナル文字('¥0')に書き換えています．
欲しい情報は入力された文字列だけなので，余計な改行文字をなくしています（33～38行目の処理で行っています．詳しい説明は今後していきます）



↑ 文字列の終わりの印(詳細は第9章にて)

（教科書p.74）

文字の配列

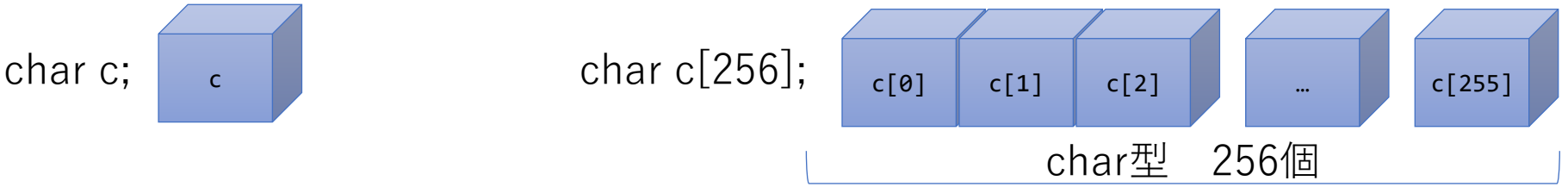
では，文字の配列はどのように定義するか?

整数型の変数定義は，int a; のようにしました.

浮動小数点数型の変数定義は，double x; のようにしました.

文字型の変数定義は，char c; のようにします．（復習p.60）

1文字入れられる変数(箱)が定義できます．先ほどのように幾つも入れられるようにするには，
char c[256]; のようにすると256個の箱が並んだ変数を定義できます.



（教科書p.74）

文字の配列

配列の大きさは入力する可能性のある文字数以上確保しておく必要があります．List3-5では，11行目でbufferという文字の配列を定義してします．

char buffer[BUFFER_SIZE];

char buffer[256]; と書かずに，後から大きさを簡単に変更できるように，4行目でオブジェクト形式マクロを利用しています．

#define BUFFER_SIZE 256

行末の；は書きません．

オブジェクト形式マクロとは（１）

オブジェクト形式マクロ(通称：定数(ていすう)マクロ)は，

#define BUFFER_SIZE 256

#define 置換文字列(マクロ名) 置換する数値

このように定義すると，コンパイル前に行う準備処理(プリプロセッサ)で文字列BUFFER_SIZEが数値256に置き換えられた後に，

```
char buffer[BUFFER_SIZE];
```

⇩ コンパイル前に置き換えられる．

```
char buffer[256];
```

としてコンパイルされる．

プログラムで同じ値が何か所にもあり，1か所変更することで，簡単に対応したい場合などに使用すると便利

オブジェクト形式マクロとは（２）

補足

- マクロは他のデータ型の値でも使うことができる
- どこに書いてもよいが，マクロ定義の次の行からその定義が有効になるため，コンパイルエラーにならないよう通常は#include文のあとに書く
- マクロ文の最後にセミコロン(;)はつかない
- マクロ名は通常の変数名などと区別しやすいように，大文字とする習慣がある
- プログラム中のキーになる数値は，直接埋め込まずにマクロを利用

※ C言語におけるマクロとは，プログラム中のある特定の文字列を
別の特定の文字列へ 置換することを指す.

文字列を表示

キーボードから入力した文字列は変数bufferに格納されるので，printfで表示させる（List3-5 14～16行目の処理）

```
printf("あなたの名前を入力してください。¥n");
get_line(buffer, BUFFER_SIZE);                    ←入力
printf("%sさん、こんにちは。¥n", buffer);          ←表示
```

文字列を表示する場合は，書式文字列**%s**を使用する

ここでは，関数get_line,fgets,printf内で配列変数を添字を省略して配列変数名だけにすると覚えておいてください

- 配列変数は，添字を省略して配列変数名だけにすると，配列の先頭アドレスへのポインタを表します（&buffer[0] 詳細はp.264で説明します）

（教科書p.77）

文字列を整数に変換する

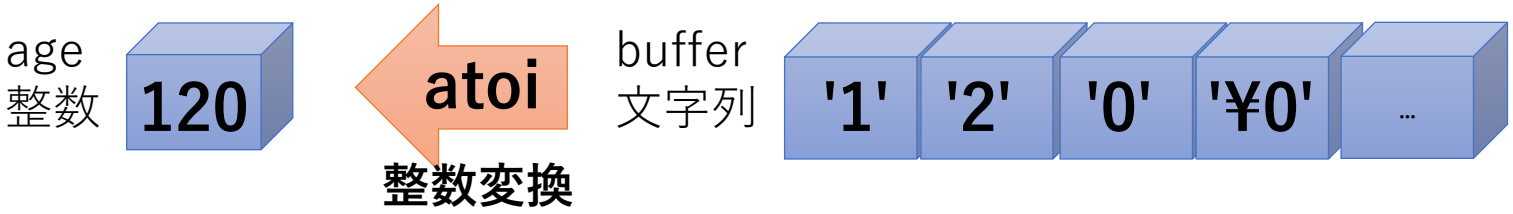
キーボードから入力した文字列を整数に変換する（List3-5 18～20行目の処理）

```
printf("年齢を入力してください。 ¥n");
get_line(buffer, BUFFER_SIZE);      ←入力
age = atoi(buffer);                 ←整数変換
```

atoi(文字列);　　atoiを使うときは #include <stdlib.h> が必要

文字列で表現された数値をint型の数値に変換する関数
ASCII to integer の言葉の略

変換不能な英字などの文字列の場合は0を返すが，数値が先頭にあればその値を返す



（教科書p.71）

List3-5を入力し実行せよ

List3-5

```
 1:  #include <stdio.h>
 2:  #include <stdlib.h>
 3:
 4:  #define BUFFER_SIZE 256
 5:
 6:  int main(void);
 7:  void get_line(char *buffer, int size);
 8:
 9:  int main(void)
10:  {
11:      char buffer[BUFFER_SIZE];
12:      int age;
13:
14:      printf("あなたの名前を入力してください。¥n");
15:      get_line(buffer, BUFFER_SIZE);
16:      printf("%sさん、こんにちは。¥n", buffer);
17:
18:      printf("年齢を入力してください。¥n");
19:      get_line(buffer, BUFFER_SIZE);
20:      age = atoi(buffer);
21:      printf("いま %d 歳とすると、10年後は %d 歳ですね。¥n", age, age + 10);
22:
23:      return 0;
24:  }
```

関数atoiを使うのに必要

（25行目以下省略）

実行結果

```
あなたの名前を入力してください。
長岡太郎
長岡太郎さん、こんにちは。
年齢を入力してください。
65
いま 65 歳とすると、10年後は 75 歳ですね。
```

入力待ちになるので，
入力して[Enter]する

すべて入力すると大変なので，別途用意したテキストファイル get_line.txt を開いて，
コードをコピーして7行目と26行目以降に貼り付けて利用してください．

確認問題

次のプログラムはキーボードから文字列を入力して，入力した文字列を整数に変換し，二乗した値を画面に表示するプログラムである．（ア）～（オ）の空欄を埋めてプログラムを完成させよ．なお，標準入力から文字列を取得する自作関数 get_line は適切に組み込まれているものとする．

```
#include <stdio.h>
#include <  （ア）  >

#define BUFFER_SIZE 256

int main(void);
void get_line(char *buffer, int size);

int main(void)
{
    char buf[BUFFER_SIZE];
    int n;

    （イ）("整数を入力してください：");
    get_line(buf, BUFFER_SIZE);                    // 標準入力から文字列を取得
    （ウ） = （エ）(buf);                          // 文字列を整数に変換して変数に代入

    printf("入力した整数の二乗は（オ）です。¥n", n * n);          // 値を画面表示

    return 0;
}
```

実行例

整数を入力してください：5
入力した整数の二乗は25です。

練習問題

- 教科書p.91の問題3-2を作成しなさい
  ファイル名は，ex3-2などとするとよいです．
  平均を計算して表示するので，浮動小数点数型(double型)で年齢の変数を定義する必要があります．

- 時間内に終わらない場合は宿題とします
  解答例は，教科書p.93～にあります．
  （解答例の10～12行目に#include文が記載されているようであれば，間違えて印刷されているので不要です．新しい版では訂正されているかもしれません．2019年初版では10～12行目は印刷間違い）

第3章で学んだこと

- ☑ 変数の定義・代入・参照
- ☑ 変数の型と値
- ☑ 文字の配列
- ☑ オブジェクト形式マクロ
- ☑ 文字列を表示する書式文字列%s
- ☑ 文字列を整数に変換する関数atoi,　#include <stdlib.h>
- ☑ 自作関数get_lineの使い方