C++プログラミングI

第8回 テキスト入力処理

岡本秀輔

成蹊大学理工学部

cin の基本動作

- ホワイトスペース：スペース・タブ・改行の各文字
- `cin >> x` ではホワイトスペースを読み飛ばす

```
char ch {};
while (std::cin >> ch)
    std::cout << ch;
```

実行例：

```
this is a test.     <<--- 入力
thisisatest.        <<--- 出力
```

1文字ずつの処理

- noskipws: ホワイトスペース (ws) をスキップしない指示
- ファイルの内容をコピーするプログラムでもある

```
char ch {};
while (std::cin >> std::noskipws >> ch)
    std::cout << ch;
```

実行例：

```
this is a test.     <<--- 入力
this is a test.     <<--- 出力
```

入力文字のカウント

- cin を行った回数が入力文字数となる

```
int n {0};  // 入力文字数の合計
for (char ch{}; std::cin>> std::noskipws >>ch;)
    ++ n;
std::cout << n << "\n";
```

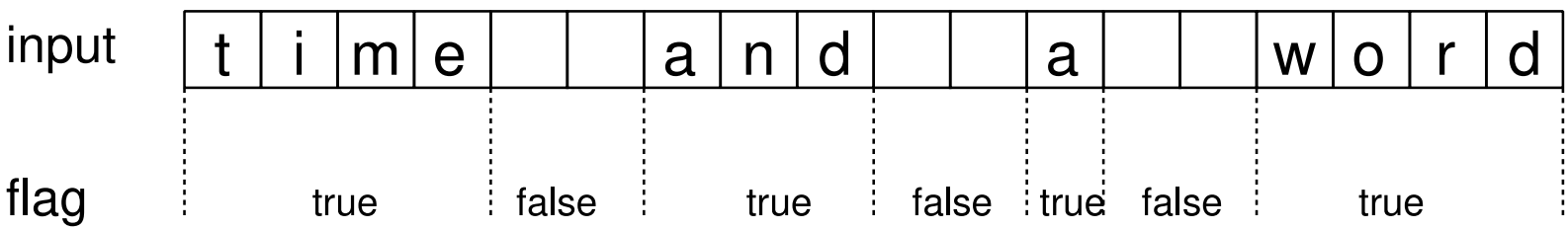
入力行のカウント

- 改行文字を数えれば行数が分かる
- '\n' 改行文字の役割は区切ること
- '\n' もホワイトスペースの一種

```
int n {0};  // 入力文字数の合計
for (char ch{}; std::cin>> std::noskipws >>ch;)
   if (ch == '\n')
      ++ n;
std::cout << n << "\n";
```

単語のカウント

- `flag` 変数を使って入力状態を把握
  - ホワイトスペースを読み込むと `false`
  - それ以外の文字を読み込むと `true`
- `false` から `true` に変化したときが単語の先頭文字



単語のカウント

- flag 変数は単語の外を表す flase から始まる
- 入力が ws 以外の文字で!flag ならば単語の先頭

```
int n {0};                  // 単語の数
bool flag {false};  // 単語中の文字かどうか
for (char ch{}; std::cin>> std::noskipws >>ch;)
    if (ch == ' ' || ch == '\n' || ch == '\t')
        flag = false;
    else if (!flag) {
        flag = true;    // 単語の先頭が見つかった
        ++ n;
    }
}
cout << n << "\n";
```

ASCIIコード

- American Standard Code for Information Interchange
- 文字を数値に対応させた表（値は覚えなくても良い）
- 文字'a' から'z' と、'A' から'Z' が連続する数値

	0x00	0x10	0x20	0x30	0x40	0x50	0x60	0x70
+0x0	'\0'		' '	'0'	'@'	'P'	'`'	'p'
+0x1			'!'	'1'	'A'	'Q'	'a'	'q'
+0x2			'\"'	'2'	'B'	'R'	'b'	'r'
+0x3			'#'	'3'	'C'	'S'	'c'	's'
+0x4			'$'	'4'	'D'	'T'	'd'	't'
+0x5			'%'	'5'	'E'	'U'	'e'	'u'
+0x6			'&'	'6'	'F'	'V'	'f'	'v'
+0x7	'\a'		'\''	'7'	'G'	'W'	'g'	'w'
+0x8	'\b'		'('	'8'	'H'	'X'	'h'	'x'
+0x9	'\t'		')'	'9'	'I'	'Y'	'i'	'y'
+0xA	'\n'		'*'	':'	'J'	'Z'	'j'	'z'
+0xB	'\v'		'+'	';'	'K'	'['	'k'	'{'
+0xC	'\f'		','	'<'	'L'	'\\'	'l'	'|'
+0xD	'\r'		'-'	'='	'M'	']'	'm'	'}'
+0xE			'.'	'>'	'N'	'^'	'n'	'~'
+0xF			'/'	'\?'	'O'	'_'	'o'	

ASCII コードを仮定した文字の変換

- 単語の先頭文字を大文字にする
- ch - 'a' + 'A' に注意

```
bool flag {false};  // 単語中の文字かどうか
for (char ch{}; std::cin>> std::noskipws >>ch;)
   if (ch == ' ' || ch == '\n' || ch == '\t')
      flag = false;
   else if (!flag) {
      flag = true;   // 単語の先頭が見つかった
      if (ch>='a' && ch<='z') // 小文字ならば
         ch = ch-'a'+'A';     // 大文字に変換
   }
   cout << ch;
}
```

文字の変換とは

```
ch = 'd';
ch = ch - 'a' + 'A';
```

- ch = 'd'; とすると、ch は 100
- ch - 'a' は 100 - 97 = 3
- つまり、'd' は'a' から 1(b), 2(c), 3(d) 番目
- 'A' から数えて 3 番目の文字は D である
- ch - 'a' + 'A' は'd' を'D' に変換する。

ライブラリ関数の利用

- isspace() ホワイトスペースを見分ける
- islower() 英小文字であるか
- toupper() 小文字を大文字に変換

```
#include <cctype>

bool flag {false};
for (char ch{}; std::cin>>std::noskipws>>ch;){
   if (std::isspace(ch))
      flag = false;
   else if (!flag) {
      flag = true;
      if (std::islower(ch))
         ch = std::toupper(ch);
   }
   std::cout << ch;
}
```

ホワイトスペースの明示的な読み飛ばし

- 行頭のホワイトスペースを除去
- 改行のみの行も削除される

```
using std::cin, std::ws;

cin >> ws;     // 最初の ws
for (char ch{}; std::cin>>std::noskipws>>ch;){
   std::cout << ch;      // そのまま出力
   if (ch == '\n')
     cin >> ws;     // 次の行頭まで読み飛ばす
}
```

読み飛ばしの実行例

- 左にそろう
- 空白行もなくなる

入力

```
   abc
bcd
        cde


  defg
   efghi
```

出力

```
abc
bcd
cde
defg
efghi
```

string による単語のカウント

- string 型の入力では単語単位に読み取れる
- ホワイトスペースの読み飛ばしも起こる

```
int n {0}; // 入力単語数の合計
for (std::string s; std::cin>>s; )
   ++ n;
std::cout << n << "\n";
```

string による大文字変換

- 単語のすべての文字を大文字にする
- 範囲 for 文の auto は char
- ch はリファレンスのため更新すれば s も変化
- ホワイトスペースの個数が無視できる時には有効

```
for (std::string s; std::cin >> s; ) {
   for (auto& ch : s) {
      if (std::islower(ch))
         ch = std::toupper(ch);
   }
   std::cout << s <<"\n";
}
```

string ストリーム

- 目的
  - 文字列から整数や実数を取り出す
  - 整数や実数を文字列に変換する
  - cin・cout のようストリームとして使える
- ヘッダファイルと型名
  - <sstream>
  - istringstream 型：文字列から取り出す
  - ostringstream 型：文字列に変換する
- 類似機能の関数：
  - to_string()：整数や実数を文字列に変換
  - stoi()： 文字列を int 整数に変換
  - stod：文字列を double 実数に変換

入力用の string ストリーム

- std::istringstream 型
- 初期値は string 変数または文字列リテラル

```
// cin のような入力変換を行う文字列ストリーム
string str = "1 2 3.4 2 4 6.8";
std::istringstream iss {str};
int x, y; double z;
while (iss >> x >> y >> z)
   cout <<2*x<<", "<<2*y<<", "<<2*z<<"\n";
```

出力用の string ストリーム

- std::ostringstream 型
- string へ出力
- .str() で取り出す

```
// cout のような出力変換を行う文字列ストリーム
std::ostringstream oss;
oss <<"abc: "<< 6 <<" "<< 1.5 <<" ";
string s1 = oss.str(); // この時点の文字列
oss <<"xyz: "<< 7 <<" "<< 2.5; // さらに追加
string s2 = oss.str(); // この時点の文字列
cout << s1 << " ::: " << s2 <<"\n";
```

三種のストリーム

- ストリーム型
  - 入出力：istream・ostream 型 (cin・cout)
  - ファイル：ifstream・ofstream 型
  - string：istringstream・ostringstream 型
- 共通の機能
  - >> によるストリームからの取り出し
  - << によるストリームへの出力
  - getline() 行単位の処理

1行ずつの処理

- getline() 関数の利用
- 第1引数はストリーム型の変数（リファレンス）
- 第2引数は string 変数（リファレンス）
- 行末の'\n' は読み捨てられる

```
// 1行ずつ読み込んで処理する
for (std::string line;
     std::getline(std::cin,line); ){
   ++ num;                    // 行数が分かる
   if (maxline.size() < line.size()) {
      maxline = line;
      maxline_num = num;
   }
}
```

行の途中から行末までの入力

- getline() は入力位置から改行文字まで読み込む
- 改行を取り除いて string 変数に設定
  - in >> x >> y で行の途中まで読み込む
  - getline(in, s) で行の残りを読み込む

```
std::ifstream in("input.txt");
if (!in) { return 1; }
int x, y; std::string s;
while (in >> x >> y && std::getline(in, s))
   std::cout << x * y << s << "\n";
```

```
$ cat input.txt
32 300 White Chocolate
42 430 Orange Cookie
53 380 Lemon Macaroons
$ ./a.out
9600 White Chocolate
18060 Orange Cookie
20140 Lemon Macaroons
```

区切り文字ごとの処理

- getline() は'\n' を区切り文字にしている
- 第 3 引数の指定でこれを別の文字に変更できる

```
std::string line {"a,b,c,d"};
std::istringstream iss(line);
for (std::string s; std::getline(iss,s,','); )
   std::cout << s << '\n';
```

出力：

```
a
b
c
d
```

CSV ファイルとは

- 表計算ソフトや DB データのテキスト保存形式
- デファクトスタンダード
- たくさんの変種
- 共通部分
  - レコードと呼ぶ関係するデータを 1 行ごとにまとめる
  - レコードが複数の値を持つならばカンマ文字で区切って値を並べる

```
White Chocolate, 32,300
Orange Cookie, 42,430
Lemon Macaroons, 53, 380
```

CSV ファイルの処理

- 1 行ずつ取り出し、カンマ区切りで取り出す
- ifs:ファイルストリーム, iss:string ストリーム

```
for (string line; getline(ifs, line); ) {
    std::vector<string> v;
    std::istringstream iss(line);
    for (string s; getline(iss, s, ','); )
        v.push_back(s);
    if (v.size() < 3) {
        cout << "line error\n";
        continue;
    }
    int num {std::stoi(v[1])};
    int price {std::stoi(v[2])};
    cout << v[0] <<": "<< num*price << "\n";
}
```