第2章「ループとカウンタ」

主なトピック

- ループと条件判断（C言語の復習）
- 名前空間(std::)

例えば，こんなプログラム (1/2)

```
//    1から100までの和を計算
sum = 0;
for(i = 1; i <= 100; i++)       {
      sum = sum + i;
}
```

- for()文による繰り返し（ループ）
- i という変数が繰り返し数を保存（カウンタ）

例えば，こんなプログラム (2/2)

```
//  3n + 1問題
n = 100;  //  n は任意の自然数
while(n != 1)  {
    if(n % 2 == 0)  {  //  2で割った時の余りが0, つまり偶数のとき
            n = n / 2;
    }
    else    {  //  奇数のとき
            n = 3 * n + 1;
    }
}
```

- while()文による繰り返し（ループ）
- n という変数とループの条件判定

ループとカウンタ

- ほとんどすべてのプログラムは
  - ループと
    - while(){}
    - for(){}
  - 変数（カウンタ）と
    - i ++;
    - n = 3*n + 1;
  - 条件分岐で
    - if(){} else if(){} else {}
- できている

while文 (1/5)

while(条件)
　　ステートメント;

- 条件が真の（成立している）間，ステートメントを繰り返し実行

while(条件) {
　　ステートメント1;
　　ステートメント2;
}

- 実行するステートメントが複数あるときはブロック{}の中に入れる

while文 (2/5)

- よくある失敗(1)
  - 実行されないかもしれないソース

```
// 確実な動作のためには i=0; がここに必要
while(i < 100){
    i ++;
}
```

  - iの値が初期化されていない（iの値は不定）
  - 条件が真かもしれない，偽かもしれない
  - while文の前に条件の初期値を確認すること

while文 (3/5)

- よくある失敗(2)
  - （ほぼ）無限ループ

```
i = 0;
while(i < 100){
    i --; // i++を間違えてしまう
}
```

  - 条件の更新を間違えると，終了しない
  - ちょっと複雑な条件や，複数の条件の組み合わせになると，よく間違えるので注意

while文(4/5)

- 何行あるか解らないデータを読み込むとき
- データファイル

  s0001 aaaa

  s0002 bbbb

  s0003 cccc
- データの最後はEOF(End of File; ファイルや入力の最後表す特別な記号)だとする
- キーボードからEOFを入力するには, UnixならCtrl+d, WindowsならCtrl+zをキー入力する

while文(5/5)

```
std::string id, name; //  IDと名前をstring型の変数に
int count; // 読んだデータの行数をint型の変数に
count = 0;
while(std::cin >> id >> name)     {
   count ++;
}
```

- 正常な入力データでは，whileの条件 std::cin >> id >> name は真になる
- 入力データがEOFだったら，条件は偽になる
- 1行読んだらcountが1増える，countにはデータの行数が入る

for文(1)

```
for(初期化ステートメント; 条件; 式){
    ステートメント;
}

int I, sum = 0;
for(I = 0; I < 100; i++){
    sum = sum + I;
}
```

- まず，初期化ステートメントを実行
  - 次に，条件が真の間，ステートメントを実行
  - 最後に式を実行
  - これを条件が偽になるまで繰り返す

for文(2)

- 同じ動作をする2つのループ式

```
for(初期化ステートメント; 条件; 式) {
    ステートメント;
}
```

```
初期化ステートメント;
while(条件)  {
    ステートメント;
    式;
}
```

- for文は初期化・条件・式を先頭で明示するため，一般的にプログラムミスが少ない
- while文は複雑なループを表現可能

条件分岐

```
if(条件1) {
        ステートメント1;
}
else if(条件2) {
        ステートメント2;
}
else {
        ステートメント3;
}
```

- 条件1が真ならばステートメント1を実行，条件2が真ならばステートメント2を実行，それ以外ならステートメント3を実行

条件の組み合わせ

- 論理演算子
  - AND, 論理積
    - 条件1 && 条件2
    - 条件1と条件2の両方が同時に真のときに，全体が真
    - if((a == 0) && (b == 0))
  - OR, 論理和
    - 条件1 || 条件2;
    - 条件1と条件2のどちらかが（両者でも構わない）真のときに，全体が真
    - if((a ==0) || (b == 0))

C++言語に時々でてくる変数の型

- size_type型
  - 例えば，std::string::size_type c;
  - 標準の名前空間(std::)に含まれる文字列型(string)のサイズや長さを表すための型(size_type)の，変数c
  - 長さを表すから符号なし，0以上の数
  - しかし，最大値はシステム依存
    - 127？ 255？ 32,767？ 65,535？ 2,147,483,648? 4,294,967,295?
  - 汎用的に使えるように size_typeという型
  - 実際はunsinged int型であることが多い
  - string型だけでなく，他の型にもsize_type型
    - vector<double>::size_type など

小技(1)

```
int sum = 0;
for(int i = 0; i < 100; ++i){
    int a;
    sum = sum + i;
}
```

- C++ ではどこででも変数宣言できる
- for()の中で変数宣言(int i)可能
- ブロック{}の中でも変数宣言可能(int a)
  - その変数はブロックの中だけで有効
  - ブロック外に出ると，その変数は破棄される

小技(2)

- C++言語だと，++iと書くことが多い
  - i++は式を評価してからインクリメント（1増やす）
  - ++iはインクリメントしてから式を評価

```
a = 3; std::cout << a++;
// 出力は3
a = 3; std::cout << ++a;
// 出力は4
```

  - for文だとi++も++iも同じ動作
  - まぎらわしい動作をしないようなソースを書く

```
a = 3; //  このように式を分割する
++a; // これならa++, ++aとも同じ動作になる
std::cout << a;
```

小技(3)

- std::を省略する方法
  - std::cin, std::stringなどstdが多くて大変
  - using std::cin;とすると，それ以降cinでOK
  - 例えばプログラムの先頭で

```
#include <iostream>
#include <string>
using std::cin;
using std::cout;
using std::string;
int main()          {
}
```

名前空間

- std::は名前空間(namespace)と呼ばれる
  - cin, cout, string, などたくさんの型や変数がstdの中で定義されている
- 用途に応じて名前空間を定義可能
- 同じ変数名が衝突しないように
  - prv::が自分で作った名前空間として
  - std::string と prv::string は別物として扱われる
  - 巨大なプログラムを複数人で作るときに便利
- using namespace std;
  - stdという名前空間で定義されている全体の識別子（型や変数）を利用可能
  - using std::cin;
    - stdという名前空間の中の識別子を個別に指定

¥nとendlの違い

- OSによって改行コードが異なる
  - Unix: CR (0x0d)
  - MS-DOS -> Windows : CR(0x0d) + LF(0x0a)
- Unixの¥nは0x0dだが，Windowsでは0x0d+0x0a
- コード変換しないでUnixからWindowにテキストファイルを持ってくると改行が乱れることがある
- なのでC++では，改行コードの機種依存を防ぐために，生の制御コード(¥n)ではなく，endlという抽象的なオブジェクトで改行を表現することにした

演習課題2のポイント

- while文で，EOFが入力されるまでstd::cinからデータの読み込み
  - 読み込んだ行数を変数に保存
  - １行読み込むたびに，画面に出力
  - 5行ごとに仕切りのフレームを入れる
  - 各行には，きれいに見えるように，適切に空白を入れる