C/C++プログラミング

第4回

反復処理の知識と演習

do文: C++

List 3-1 do文

```
----------------------------------------
// 入力された月の季節を表示（好きなだけ繰り返せる）

#include <string>
#include <iostream>

using namespace std;

int main()
{
        string retry;                   // もう一度？

        do {
                int month;
                cout << "季節を求めます。\n何月ですか：";
                cin >> month;

                if (month >= 3 && month <= 5)           // 3月・4月・5月
                        cout << "それは春です。\n";
                else if (month >= 6 && month <= 8)      // 6月・7月・8月
                        cout << "それは夏です。\n";
                else if (month >= 9 && month <= 11)     // 9月・10月・11月
                        cout << "それは秋です。\n";
                else if (month == 12 || month == 1 || month == 2) // 12月・1月・2月
                        cout << "それは冬です。\n";

                cout << "もう一度？ Y…Yes／N…No：";
                cin >> retry;
        } while (retry == "Y" || retry == "y");
}
----------------------------------------
```

do文: C++

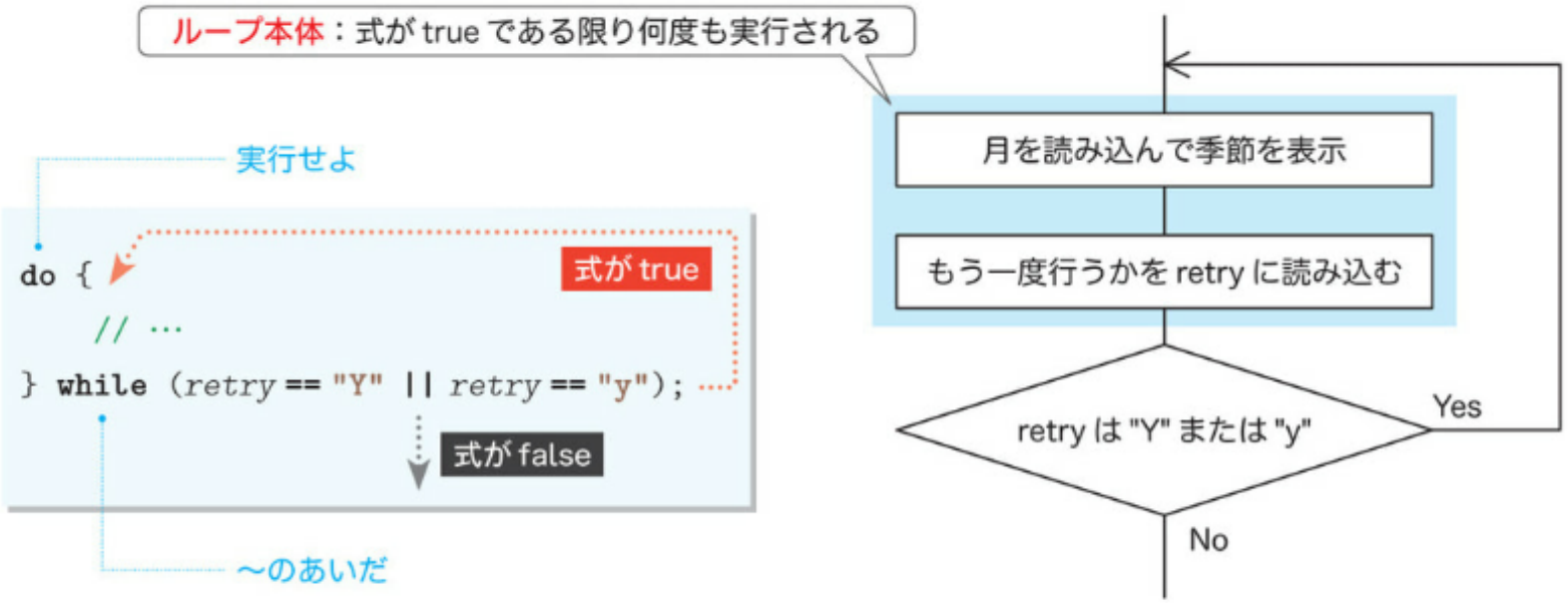


Fig.3-2　List 3–1 の do 文のフローチャート

do文: C++

「
```
#include <string>
string retry;
```
」
文字列のクラスを扱うためのヘッダのインクルードと，「string」というクラスの変数 retly の宣言.

「do { }」
「while ( )」
do と while をペアで使う．これを do 文といい，繰り返し動作（ループ）が行われる．
まず do の中身（中括弧で囲まれたブロック）を一通り実行し，その後，while の小括弧の条件文の真偽を確認する．真ならまた do に戻って繰り返し動作を行う．偽なら while 文の下に進みループから抜け出す.

「while (retry == "Y" || retry == "y")」
条件文の中身の「||」は「または」という意味．つまり
「retry == "Y"」が成り立つか「retry == "y"」が成り立てば全体として「真」という意味.

以降，C++とCでは繰り返し動作のプログラムの文法には大きな違いはないので，Cは省略.

while文: C++

List 3-4 while文

```
----------------------------------------
// 正の整数値を０までカウントダウン（その１）

#include <iostream>

using namespace std;

int main()
{
        int x;

        cout << "カウントダウンします。\n";
        do {
                cout << "正の整数値：";
                cin >> x;
        } while (x <= 0);

        while (x >= 0) {
                cout << x << "\n";        // xの値を表示
                x--;                      // xの値をデクリメント（値を一つ減らす）
        }
}
----------------------------------------
```

while文: C++

「
```
while (x >= 0) {
                        cout << x << "\n";
                        x--;
}
```
」
while 文だけでも繰り返し動作は実現できる．ここでは while の右の小括弧の中の条件式が「真」であれば中括弧の中身を一通り実行し，また while の行に戻る，という動作．
中括弧の中の最後の行「x--;」は「xの値を1減らしてxに格納せよ」という意味．つまり，
x = x – 1;
と同じ．「--」を「減分演算子」と呼ぶ．また x-- の動作を「デクリメントする」という．
これにより中括弧の最後に x が1減り，while 文にもどり，x >= 0 かどうかを確認，真の間は while 文を繰り返し，x が負になったらループから抜け出すという動作になる．

「
```
while (x >= 0)
                        cout << x-- << "\n";
}
```
」
上と同等のプログラム．ただし「cout << x--」は，「x をデクリメントする前の値」を cout に表示し，「その後」x をデクリメントする，という動作になる．これを「後置減分演算子」という．逆に x を 1 増やすには，「x++」（後置増分演算子）とする．これを「インクリメントする」という．

while文: C++

「++x」，「--x」
これらも先と同様に，xの値を１増やす，あるいは１減らすという動作．ただし
 y = ++x;
の動作は「xを1増やした後の値をyに代入する」となる．これらを，「前置増分演算子」，「前置減分演算子」と呼ぶ.

for文: C++

List 3-11 for文

```
---------------------------------------
// 読み込んだ個数だけ*を表示（for文）

#include <iostream>

using namespace std;

int main()
{
        int n;
        cout << "何個*を表示しますか：";
        cin >> n;

        for (int i = 0; i < n; i++)
                cout << '*';
        cout << '\n';
}
---------------------------------------
```

for文: C++

「
```
for (int i = 0; i < n; i++)
        cout << '*';
```
」
for の次の小括弧の中の条件式が真ならば，その次の実行文を実行し，偽なら実行文の下に飛ぶ．これを「for文」という．「for ()」の行は，

for (A; B; C)

という形式で書かれる．

Aが「初期化」，　　int i = 0

Bが「条件式」，　　i < n

Cが「後処理」　　i++

上の例の場合，for文の1巡目では，まずAで変数 i が宣言 &i =0と初期化される．次にそのiの値とnとを比べて i < n なら真となり， cout << '*'; が実行される．その後，Cの i++ が実行され，i が1増え，また for の行に戻り2巡目以降を繰り返す．ただし，2巡目以降は A の「初期化」は実行されない．「'*'」のシングルクォーテーションは1文字を扱う場合．

for文: C++

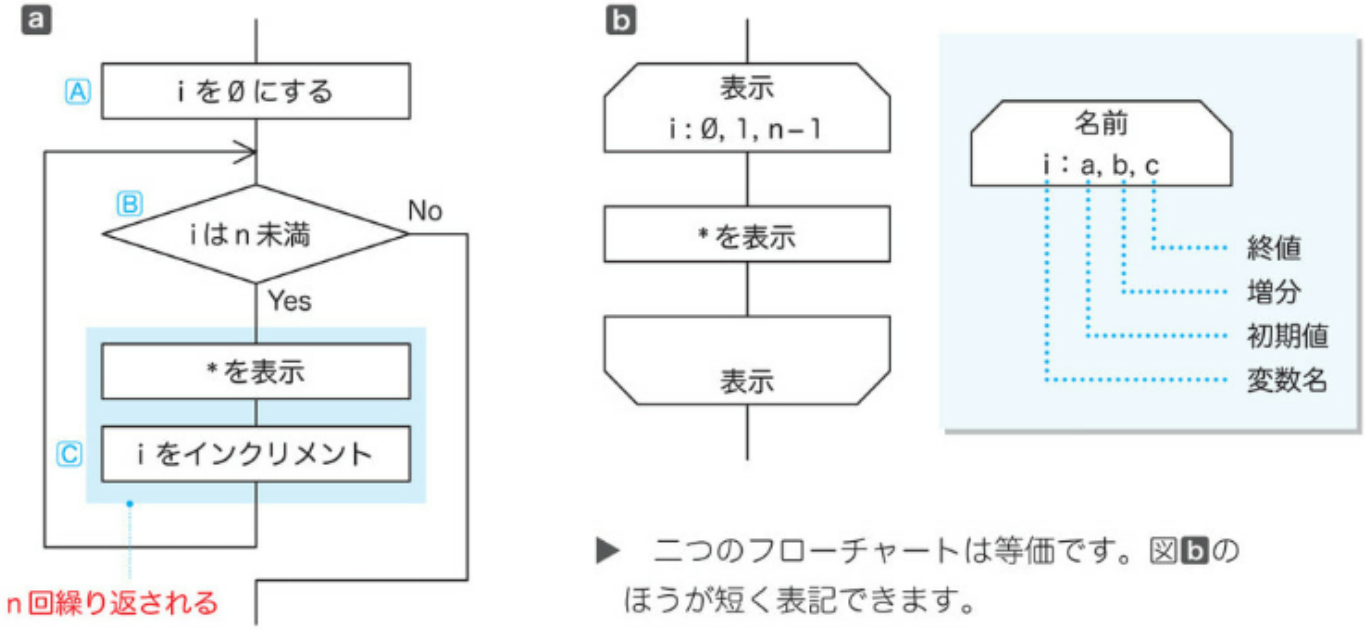


Fig.3-13 List 3–11 の for 文のフローチャート

多重ループ: C++

List 3-14 多重ループ

```
---------------------------------------
// 九九の表を表示

#include <iomanip>
#include <iostream>

using namespace std;

int main()
{
        for (int i = 1; i <= 9; i++) {
                for (int j = 1; j <= 9; j++)
                        cout << setw(3) << i * j;
                cout << '\n';
        }
}
---------------------------------------
```

多重ループ: C++

「
```
for (int i = 1; i <= 9; i++) {
        for (int j = 1; j <= 9; j++)
                cout << setw(3) << i * j;
        cout << '\n';
}
```
」
for 文の中にもう一つ for 文が含まれている例．これを「二重ループ」という．同様に「三重ループ」，「四重ループ」… などがあり，まとめて「多重ループ」という．

この例の場合の動作は，まず外側の赤いfor文の1巡目（i=1）が実行される．実行内容は，すぐに内側の青いfor文が実行される．

青いfor文ではまず1巡目（j=1）が実行され，j<=9が真の間，繰り返し実行される．j<=9が偽となったら，つまり j = 10 となったら，青いfor文から抜け出す．

その後「cout << '\n';」が実行され，外側の赤いfor文の実行文の1巡目が終了し，i++ で i をインクリメントし，赤いfor文の2巡目に進む．2巡目に青いfor文が実行される際，青いfor文の1巡目はまた j は j=1に初期化される．以降，これを繰り返す．

「 cout << setw(3) << i * j; 」
setw(3)は，この後の出力を「少なくとも」3桁の幅で実行せよという意味．これを用いるには，#include <iomanip> にあるようにヘッダ iomanip のインクルードが必要．

多重ループ: C++

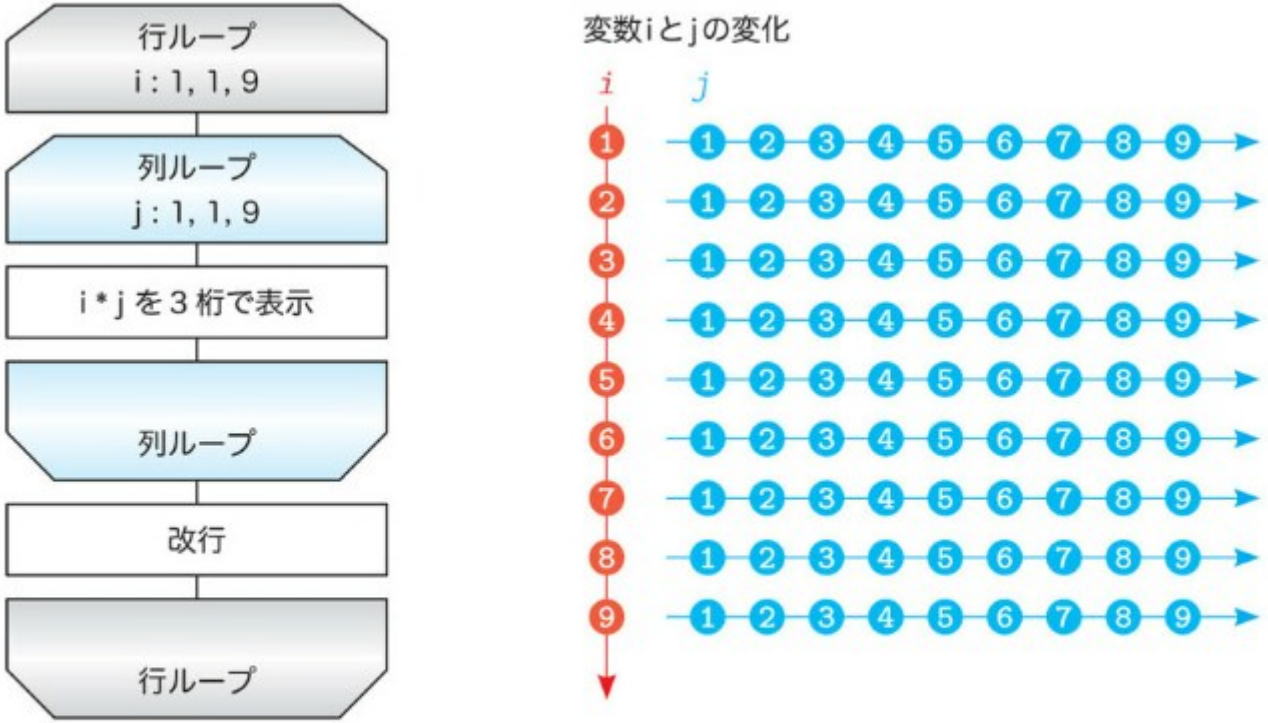


Fig.3-14　九九の表を表示する二重ループのプログラムの流れ

break文: C++

List 3-17 break文

```
// 読み込んだ整数を加算（0が入力されたら終了）

#include <iostream>

using namespace std;

int main()
{
        int n;                                  // 加算する個数
        cout << "整数を加算します。\n";
        cout << "何個加算しますか：";
        cin >> n;

        int sum = 0;            // 合計値
        for (int i = 0; i < n; i++) {
                int t;
                cout << "整数（0で終了）：";
                cin >> t;
                if (t == 0) break;                      // for文から抜け出る
                sum += t;
        }
        cout << "合計は" << sum << "です。\n";
}
```

break文: C++

「
```
if (t == 0) break;
```
」
if文の条件式が真のとき，右側の実行文が実行される．ここでは break という動作が実行される．break の意味は，「このif文の行を直接含んでいる for ループ」から抜け出せ，という意味．多重ループの場合に多重ループ全体から抜け出すという「意味ではない」．
例えば次の2重ループの場合，

```
for (   ) {
   for (   ) {
      …
      if (   ) break;
   }
   x=y;
}
```

青の break文が実行されると，この break 文を直接含んでいる青のforループから抜け出し，一つ上の赤のforループのx=y;を実行する，という動作になる．

continue文: C++

List 3-19 continue文

```
// 読み込んだ整数を加算（負の値は加算しない）

#include <iostream>

using namespace std;

int main()
{
        int n;                                          // 加算する個数
        cout << "整数を加算します。\n";
        cout << "何個加算しますか：";
        cin >> n;

        int sum = 0;   // 合計値
        for (int i = 0; i < n; i++) {
                int t;
                cout << "整数：";
                cin >> t;
                if (t < 0) {
                        cout << "\a負の数は加算しません。\n";
                        continue;
                }
                sum += t;
        }
        cout << "合計は" << sum << "です。\n";
}
```

continue文: C++

「

```
for (int i = 0; i < n; i++) {
                int t;
                cout << "整数：";
                cin >> t;
                if (t < 0) {
                        cout << "\a負の数は加算しません。\n";
                        continue;
                }
                sum += t;
}
```

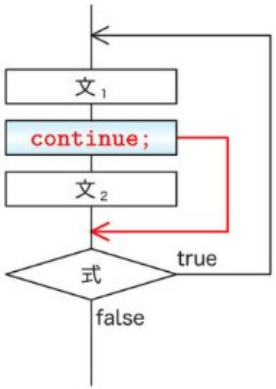
」

continue文は，それが実行されると，そのcontinue文を直接含むループの一番最後にjumpせよ，という動作になる．上の例の場合，continue文を直接古むループは青のforループであるが，その最後，つまり「sum += t; の次」に jump する．

continue文: C++

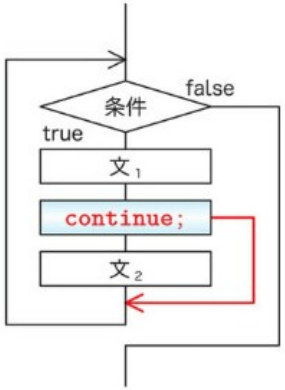
a do文

```
do {
    文1
    continue;
    文2
} while (式);
```



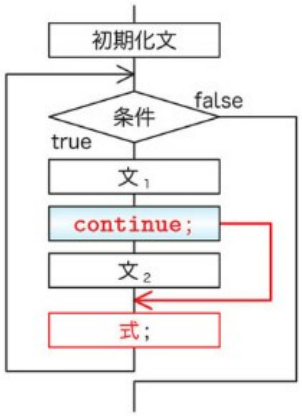
b while文

```
while (条件) {
    文1
    continue;
    文2
}
```



c for文

```
for (初期化文 条件; 式) {
    文1
    continue;
    文2
}
```



注意：$文_2$ はスキップされるが
式の評価はスキップされない

繰返し文中で continue 文が実行されると、ループ本体の残りの部分の実行がスキップされる

Fig.3-18 continue 文の働き

goto文: C++

List 3-20 goto文

```
// 読み込んだ整数を加算（9999が入力されると強制終了）

#include <iostream>

using namespace std;

int main()
{
        int n;                                          // 加算する個数
        cout << "整数を加算します。\n";
        cout << "何個加算しますか：";
        cin >> n;
        cout << "9999で強制終了します。\n";

        int sum = 0;    // 合計値
        for (int i = 0; i < n; i++) {
                int t;
                cout << "整数：";
                cin >> t;
                if (t == 9999)
                        goto Exit;
                sum += t;
        }
        cout << "合計は" << sum << "です。\n";

Exit:
        ;
}
```

goto文: C++

「

```
if (t == 9999)
                                        goto Exit;
                        sum += t;
            }
            cout << "合計は" << sum << "です。\n";

Exit:
            ;
```

」

goto文は，ループのどこにあっても無関係に，指定の行に jump するという動作をする．
上の例の場合，「goto Exit;」の 「Exit」という「ラベル」の付いた行に jump する．
ラベルの名称は任意に設定できる．行のラベルのつけ方は行頭に
「Exit:」（ラベル＋コロン）とする．上の例の場合，この行は，「;」のみなので，なにも実行しない，となる．

課題

本日作成したC++のプログラム（ソースコード），それぞれの実行ファイル一式をzipし，UNIPA上にアップロードする.

課題

Q1：平均値の計算
（**プログラム名：ex04_1.cpp**）

while ループを使って整数を読み込み、0未満になったら停止し、平均値（float 型）を計算して出力します。ただし、有効な非負の整数が一つも入力されなかった場合は、「平均値は計算できません。」と出力してください。

【実行例】

例1
入力：
1⏎
3⏎
-1⏎
出力：
2

例2
入力：
2⏎
3⏎
-1⏎
出力：
2.5

例3
入力：
-2⏎
出力：
平均値は計算できません。

␣: space
⏎ : enter

課題

Q2：右揃えの下三角
（**プログラム名：ex04_2.cpp**）

整数 N を読み込み、N 行からなる下三角（ピラミッドの右半分）の * を出力してください。各行の * の前には、適切な数の空白（スペース）を挿入して、全体が右揃えになるようにします。入力 N が0以下なら「無効な入力です。」と出力して終了。

【実行例】

例1
入力：
2⏎
出力：
␣*
**

例2
入力：
5⏎
出力：
␣␣␣␣*
␣␣␣**
␣␣***
␣****
*****

例3
入力：
-1⏎
出力：
無効な入力です。

␣: space
⏎ : enter