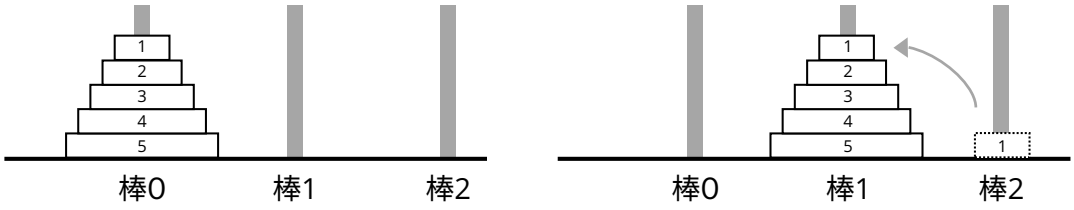
第 5 回 C++ プログラミング II 実験

出題日：2022 年 5 月 9 日（月）

課題 1　(提出期限 5 月 12 日 (木) 24:00)

スタックを使ってハノイの塔というパズルを作りたい．ハノイの塔のパズルとは次のようなものである．「下の図のように 3 本の棒と大きさの異なる穴の開いた円盤 n 枚がある．最初は下の方が大きい順に円盤が棒 0 に積まれている．この円盤を一枚ずつ別の棒に移動させて全ての円盤を棒 1 に移動させなさい．ただし，円盤を移動する際には移動させる円盤よりも小さい円盤の上に積んではいけない.」（ハノイの塔に関する詳しい説明は Wikipedia ハノイの塔などを参照せよ．）`ex05-1-main.cpp` を完成させてハノイの塔のパズルを完成させなさい.



- `ex05-1-main.cpp` のプログラムでは，棒をスタック，円盤を 1 から n の整数で管理する．$n = 2$ に設定されているが，他の整数に変えても動作することを確認すること．CoursePower に提出する際の実行例は $n = 2$ の場合のみでよい（実行例 1 を真似すること）.
- 初期状態を設定するため，棒 0 を表すスタック `Hanoi[0]` に 1 から n の整数を先頭 (上) から小さい順になるように格納する (空欄 1)．まず，while 文 (33 行目から 47 行目) をコメントアウトして，初期状態が正しく表示されるか確かめてみること.
- 毎ステップごとに 2 つの整数 `From` と `To` を標準入力から取得し，棒 `Hanoi[From]` の先頭の円盤を棒 `Hanoi[To]` に移動可能ならば移動させる (空欄 $2-3$).
- 以下のいずれかの場合が移動可能でない場合である.
  - 0 から 2 までの整数以外を受け取ったとき.
  - 棒 `Hanoi[From]` に円盤が無かったとき (空欄 $2-1$).
  - 棒 `Hanoi[From]` の先頭の数が棒 `Hanoi[To]` の先頭の数よりも大きかったとき (空欄 $2-2$).
- スタックが空の場合に `top()` を使うとエラーになるので，空欄 $2-2$ を埋める際には注意すること.
- 空欄 1，空欄 2-3 には複数の文を書いて構わない.

ソースコード 1　ex05-1-main.cpp

```cpp
#include <iostream>
#include <vector>
#include <stack>
using std::cout, std::cin, std::stack, std::vector;

const int n{2}; //ハノイの塔の高さ(円盤の枚数)

void print(vector<stack<int>> vs){ //ハノイの塔を出力する関数
    for(int i=0; i<n; i++){
        cout << "␣";
        for(int j=0; j<3; j++){
            if( vs[j].size()<(n-i) )cout << "␣␣␣␣␣␣";
            else{
                cout << "--" << vs[j].top() << "--␣";
                vs[j].pop();
            }
        }
        cout << "\n";
    }
    cout << "␣------------------␣\n";
    cout << "␣␣[0]␣␣␣[1]␣␣␣[2]␣␣\n";
}

int main(){
    vector<stack<int>> Hanoi(3); //3つの棒(スタック)を用意

    /* (1) 棒0(Hanoi[0])の初期状態を設定せよ */

    stack<int> goal{Hanoi[0]}; //棒0の状態を終了判定のためにコピーしておく
    int k{0}; //円盤を動かす回数のカウンター
    print(Hanoi); //初期状態を出力
    int From{}, To{};
    while(cin >> From >> To){ //円盤の移動元と移動先を入力

        //(2) 空欄を埋めて円盤を適切に動かせるようにしなさい
        if( From<0 || From>2 || To<0 || To>2 ) cout << "cannot␣move\n";
        else if( /* 空欄2-1 */ ) cout << "cannot␣move\n";
        else if( /* 空欄2-2 */ ) cout << "cannot␣move\n";
        else{ /* 空欄2-3 */ }

        print(Hanoi); //移動後の状態を出力
        k++; //カウンターに+1
        if(Hanoi[1]==goal){ //終了判定
            cout << "Finished!␣" << k << "steps\n";
            break;
        }
    }
}
```

実行例 1

```
$ ./ex05-1
 --1--
 --2--
 -----------------
   [0]    [1]    [2]
0 2   (← 自分で入力)

 --2--        --1--
 -----------------
   [0]    [1]    [2]
0 2   (← 自分で入力)
cannot move

 --2--        --1--
 -----------------
   [0]    [1]    [2]
0 1   (← 自分で入力)

        --2-- --1--
 -----------------
   [0]    [1]    [2]
2 1   (← 自分で入力)
        --1--
        --2--
 -----------------
   [0]    [1]    [2]
finished! 4steps
```

実行例 2

(円盤の枚数を $n = 3$ に変更)

```
$ ./ex05-1b
 --1--
 --2--
 --3--
 -----------------
   [0]    [1]    [2]
0 1   (← 自分で入力)

 --2--
 --3-- --1--
 -----------------
   [0]    [1]    [2]



途中省略



0 1   (← 自分で入力)
        --1--
        --2--
        --3--
 -----------------
   [0]    [1]    [2]
finished! 7steps
```

問題２、queue を用いて，入力された文字列の履歴を最大 3 件まで保存するプログラムを作成せよ。

実行例をもとに，以下のプログラムを完成せよ。過去の入力ほど queue の先頭に保存されるものとし，履歴が 3 件を超えるときは最も過去の履歴が queue から削除されるものとする（つまり，queue の先頭の要素が削除される）。

```
#include <iostream>
#include <queue>
#include <string>

int main() {
    std::string s{};

   // queueの初期化


    // 5回の入力を実行
    for (int i = 0; i < 5;i++) {
        // cin で値を入力する

        // queue へ格納

        // queue が 4 以上のときに先頭の要素を取り出す

        // queue の中身を表示，注意：begin()関数は使えない

        std::cout << "¥n";
    }
    return 0;
}
```

実行例

```
Input: I
history: I
Input: can
history: I can
Input: sing
history: I can sing
Input: a
history: can sing a
Input: song
history: sing a song
```

課題３ （提出期限：5 月 12 日（木）24:00）

deque を用いて，入力された文字の履歴を最大 3 件まで保存したり，入力履歴を表示したりするプログラムを考える．

下に示すソースコード ex05-3-main.cpp は，キーボードから入力された文字列を string の str で受け取り，入力された文字列に応じて履歴の保存・履歴の表示・直前の操作の取り消し（Undo）・処理の終了を行うプログラム（未完成）である．入力された文字列は，過去の入力ほど deque の history の先頭に保存されるものとし，履歴が 3 件を超えるときは最も過去の履歴が deque の history から削除されるものとする．ただし，“show” と入力されたときは history のデータを古い文字列から順に表示し，“undo” と入力されたときは history のデータから最新の文字列を履歴から削除し，“exit” と入力されたときはプログラムが終了するようになっている．

実行例をもとに，ex05-3-main.cpp を完成させなさい．また，完成したプログラムをコンパイルし，実行例１および実行例２と同じ結果になるかを確認しなさい．

ソースコード 1　ex05-3-main.cpp

```cpp
#include <iostream>
#include <deque>
#include <string>

int main() {
    std::deque<std::string> history;
    std::string str{};

    while(true) {
        // cin で文字列を入力（str へ格納）
        std::cout << "Input: ";
        std::cin >> str;

        // 入力された文字列により処理を変更
        if (str == "show") {
            //
            // show を入力したときのプログラムを作成せよ
            //
        }
        else if (str == "undo") {
            std::cout << "Undo\n";
            //
            // undo を入力したときのプログラムを作成せよ
            //
        }
        else if (str == "exit") {
            // exit を入力すると終了
            std::cout << "Finished." << std::endl;
            return 0;
        }
        else {
            //
            // そのほかの文字列を入力したときのプログラムを作成せよ
            //
        }
    }
}
```

実行例 1

```
./ex05-3
Input: abc
Input: 123
Input: show
History: abc 123
Input: hello
Input: world
Input: show
History: 123 hello world
Input: exit
Finished.
```

実行例 2

```
./ex05-3
Input: abc
Input: 123
Input: undo
Undo
Input: xyz
Input: show
History: abc xyz
Input: undo
Undo
Input: undo
Undo
Input: show
History:
Input: exit
Finished.
```