第14回 C++プログラミング実験II 実験課題

出題日:2022 年 7 月 11 日(月)
提出期限:2022 年 7 月 14 日(木)24:00

実験課題 14-1

　スケルトンファイル（ex14-1.cpp）の空欄箇所（デストラクタ：1−１＆1−２、exercise関数：２、カロリー摂取関数：３）を埋めて、自転車で運動した際に消費するカロリーと移動距離を出力するプログラムを作成せよ。1回の運動で1.8km移動、１回の運動でカロリーを1.2kJ消費し、残カロリーが１回の運動カロリー以下の場合、初期カロリーに摂取カロリーを加算するものとする。

```
ex14-1.cpp(main文除く）
#include <iostream>

using std::cin, std::cout, std::endl;

class roadcycle{
public:
    roadcycle(); //コンストラクタ
    //（1−1）←デストラクタ
    void exercise(); //運動
    void eat(double food); //カロリー摂取

private:
    double calorie; //カロリー
    double distance; //移動距離
};

//コンストラクタ
roadcycle::roadcycle(): calorie(1.0), distance(0)
{
    cout << "オブジェクト生成" << endl;
}
//デストラクタ
//（1ー2）←デストラクタ
{
    cout << "オブジェクト破棄" << endl;
}

//運動
void roadcycle::exercise()
{
    //（2）残カロリーが運動１回の消費カロリー以上なら移動：1回の運動で1.8km移動、１回の運動でカロリーを1.2kJ消費

    }
    cout << "移動距離: " << distance <<"km"<<endl;
    cout << "残カロリー: " << calorie <<"kJ"<<endl;
}
//カロリー摂取
void roadcycle::eat(double food)
{
        （3） //(１回の運動カロリー以下の場合、初期カロリーに摂取カロリーを加算
    }
    cout << "累積カロリー: " << calorie <<"kJ"<< endl; //累積カロリー表示
}
```

```
//実行例 1
% ./ex14-1
摂取カロリー: 0.1
オブジェクト生成
累積カロリー: 1.1kJ
移動距離: 0km
残カロリー: 1.1kJ
摂取カロリー: 0.2
累積カロリー: 1.3kJ
移動距離: 1.8km
残カロリー: 0.1kJ
オブジェクト破棄
```

```
//実行例 2
% ./ex14-1
摂取カロリー: 0.2
オブジェクト生成
累積カロリー: 1.2kJ
移動距離: 1.8km
残カロリー: 0kJ
摂取カロリー: 0.2
累積カロリー: 0.2kJ
移動距離: 1.8km
残カロリー: 0.2kJ
オブジェクト破棄
```

```
//実行例 3
% ./ex14-1
摂取カロリー: 11.1
オブジェクト生成
累積カロリー: 12.1kJ
移動距離: 1.8km
残カロリー: 10.9kJ
摂取カロリー: 11.1
累積カロリー: 10.9kJ
移動距離: 3.6km
残カロリー: 9.7kJ
オブジェクト破棄
```

実験課題 14-2

実験課題 5-1 で作成したハノイの塔のプログラムについて，生ポインタ版スタックのソースコードが ex14-2-main_raw.cpp である．main 関数や構造体とクラス内の流れは変更せずに，shared_ptr 版スタックに書き換えよ．（注意：生ポインタ版スタックの内容は講義資料の補足項目にあり，C++プログラミング III の受講準備となる）

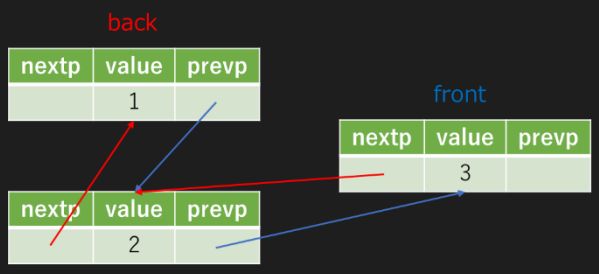
```
// ex14-2-main_raw.cpp
#include <iostream>

using std::cout, std::cin, std::endl;

struct Node
{
    int idata;
    Node *next;
    Node(int a, Node *np) : idata{a}, next{np} {}
    ~Node() { cout<<"dtor: "<<idata <<"¥n"; }
    void print() { cout << idata << " "; }
};

class Stack
{
    Node *top;   // スタックトップ
    int length; // スタック内のデータ数（スタックの長さ）
public:
    Stack() : top{nullptr}, length{0} {} // コンストラクタ
    void push(int d);                    // スタックにデータを push する
    int back();                          // スタックトップのデータを返す
    void pop();                          // スタックトップのデータを取り出す（取り除く）
    bool empty() { return length == 0; } // スタックが空かの判定
    int size() { return length; }        // スタックの長さを返す
    void print();                        // スタックの内容をすべて表示する
    ~Stack()          // デストラクタ（先頭から順に pop する）
    {
        while (top != nullptr)
            pop();
    }
};

void Stack::push(int d)
{
    top = new Node{d, top};
    length++;
}

int Stack::back() { return top->idata; }

void Stack::pop()
{
    auto tmp{top};    // 解放する対象を覚えておく
    top = top->next; // 先頭の変更
    delete tmp;       // 解放
    length--;
}

void Stack::print()
{
    for (Node *p{top}; p; p = p->next)
        cout << p->idata;
}
int main()
{
    Stack s, g, b;          // ３本の杭，s:スタート，g:ゴール，b:バッファ，
    int n;                  // 円盤の数
    std::string from, to;   // 始点と終点の入力
    Stack *fp, *tp;         // 始点と終点の杭を指すポインタ
```

```cpp
// ex14-2-main_raw.cpp つづき
    cout << "円盤の数--> ";
    cin >> n;
    for (int i{n}; i > 0; i--)   s.push(i);

    for (int i{0}; i < n - s.size(); i++)    cout << "-";
    s.print();
    cout << "-S¥n";
    for (int i{0}; i < n - g.size(); i++)    cout << "-";
    g.print();
    cout << "-G¥n";
    for (int i{0}; i < n - b.size(); i++)    cout << "-";
    b.print();
    cout << "-B¥n";

    cout << "移動指示 : S (or s), G (or g), B (or b)¥n";
    while (cout << "From -> To : " && cin >> from >> to)
    {
        if (from == "S" || from == "s")          fp = &s;
        else if (from == "B" || from == "b")     fp = &b;
        else if (from == "G" || from == "g")     fp = &g;
        else
        {
            cout << "杭は S, B, G の中から選んでください¥n";
            continue;
        }
        if (to == "S" || to == "s")          tp = &s;
        else if (to == "B" || to == "b")     tp = &b;
        else if (to == "G" || to == "g")     tp = &g;
        else
        {
            cout << "杭は S, B, G の中から選んでください¥n";
            continue;
        }

        if (fp->empty())
        {
            cout << "cannot move¥n";
            continue;
        }
        if (!tp->empty() && tp->back() < fp->back())
        {
            cout << "cannot move¥n";
            continue;
        }
        tp->push(fp->back());
        fp->pop();

        for (int i{0}; i < n - s.size(); i++)    cout << "-";
        s.print();
        cout << "-S¥n";
        for (int i{0}; i < n - g.size(); i++)    cout << "-";
        g.print();
        cout << "-G¥n";
        for (int i{0}; i < n - b.size(); i++)    cout << "-";
        b.print();
        cout << "-B¥n";

        if (g.size() == n)
        {
            cout << "Finished!¥n";
            break;
        }
    }
    return 0;
}
```

```
//実行例
./a.out
円盤の数--> 2
12-S
---G
---B
移動指示 : S (or s), G (or g), B (or b)
From -> To : s b
dtor: 1
-2-S
---G
-1-B
From -> To : s g
dtor: 2
---S
-2-G
-1-B
From -> To : b g
dtor: 1
---S
12-G
---B
Finished!
dtor: 1
dtor: 2
```

実験課題 14-3

第 6 回授業で扱ったリストの簡易版を，shared_ptr を用いて実装する．リストの各ノードは，下の図に示す通りその前後のノードのポインタを持つものとする．このとき，空欄(1)~(3)を埋め，push・pop 関数を完成させなさい．ただし，ノードの解放順序は問わないものとする（実行例では 123 の順だが，321 でもよい）．




```cpp
//ex14-3-main.cpp
#include <iostream>
#include <memory>
using std::cout;

using Ptr = std::shared_ptr<struct Node>;

struct Node
{
    int value;
    Ptr nextp, prevp;
    Node(int a, Ptr n, Ptr p) :value{ a }, nextp{ n }, prevp{ p } {}
    ~Node() { cout << "dtor: " << value << "¥n"; }
};

class List {
    Ptr front, back;
    int length;
public:
    List() :front{ nullptr }, back{ nullptr }, length{ 0 }{}
    void push_front(int);
    void pop_front();
    void print_back() const;
    ~List(){ while (front != nullptr) pop_front(); }
};

void List::push_front(int d)
{
    if(length==0){
        /* (1) */
        back = front;
    }
    else{
        Ptr tmp = front;
        /* (2) */
        tmp->prevp = front;
    }
    length++;
}
void List::pop_front()
{
    /* (3) */
    length--;
}

void List::print_back() const
{
    for (Ptr p{ back }; p; p = p->prevp)
        cout << p->value << " ";
    cout << "¥n";
}

int main() {
    List s;
    for (int i; std::cin >> i; s.push_front(i));
    s.print_back();
}
```

実行例 1

```
> echo 1 2 3 | ./a.out
1 2 3
dtor: 1
dtor: 2
dtor: 3
```

実行例 2

```
> echo 1 3 5 6 7 9 13 | ./a.out
1 3 5 6 7 9 13
dtor: 1
dtor: 3
dtor: 5
dtor: 6
dtor: 7
dtor: 9
```