第 13 回 C++プログラミング実験 II

出題日:2022 年 7 月 4 日(月)
提出期限:2022 年 7 月 7 日(木)24:00

実験課題 13-1

実行結果例のようにコマンド引数として 2 つの文字列を入力すると,

- 2 つ目のコマンド引数が -m の場合には,「Hi, (1 つ目のコマンド引数)! Good Morning!」,
- 2 つ目のコマンド引数が -e の場合には,「Hi, (1 つ目のコマンド引数)! Good Evening!」,
- 上記以外の場合には,「Hi, (1 つ目のコマンド引数)!」

と出力するように, 以下のプログラムを完成させよ.

ex13-1-main.cpp

```
#include <iostream>

int main(int argc, char *argv[]){
    // 空欄 1: Hi, (1 つ目のコマンド引数)!と出力する.

    if(/* 空欄 2-1: 2 つ目のコマンド引数の 1 文字目が - の場合 */){
        // 空欄 2-2: 2 つ目のコマンド引数の 2 文字目が m なら Good Morning!,
        // e なら Good Evening! と出力する.
    }
    std::cout << "¥n";
}
```

実行結果 1

```
$ g++ -std=c++17 ex13-1-main.cpp -o ex13-1
$ ./ex13-1 Taro -m
Hi, Taro! Good Morning!
```

実行結果 2

```
$ g++ -std=c++17 ex13-1-main.cpp -o ex13-1
$ ./ex13-1 Hanako -e
Hi, Hanako! Good Evening!
```

第 13 回 C++プログラミング実験Ⅱ

出題日:2022 年 7 月 4 日(月)
提出期限:2022 年 7 月 7 日(木)24:00

実験課題 13-2

CoursePower 上の ex13-2-main.cpp に示すのは，ある特定の整数値の要素が配列(要素数 5 の配列)に含まれているかどうかを調べるプログラムの一部である．実行例を参考にして，5 つの空欄を埋めることでプログラムを完成させよ．

実行結果1 下線部は入力箇所

```
$ g++ -std=c++17 ex13-2-main.cpp -o ex13-2
$ ./ex13-2
Input five integers to an array.
x[0] : 11
x[1] : 22
x[2] : 33
x[3] : 44
x[4] : 55
The value which you want to find is : 44
The value is in x[3].
```

実行結果2 下線部は入力箇所

```
$ g++ -std=c++17 ex13-2-main.cpp -o ex13-2
$ ./ex13-2
Input five integers to an array.
x[0] : 11
x[1] : 22
x[2] : 33
x[3] : 44
x[4] : 55
The value which you want to find is : 66
Not found.
```

参考:

- array_search 関数について:
  - 要素数 n の配列 a から，値が key の要素を先頭から順に探索して，見つけた要素の添字に相当する値を返す関数である．ただし，探索に失敗した場合に返すのは −1 とする．このように，先頭要素から順に値を比較していく探索法は，**線形探索**(linear search)あるいは**逐次探索**(sequential search)と呼ばれる．
  - 関数内で宣言されるポインタ p は，a の値で初期化されているため，配列 x の先頭要素を指すことになる．
  - ポインタ p が指す要素の値 *p が，探索すべき値 key と等しければ，if 文の条件が成立するため，探索成功である．また，key と等しくなければ，p++ の実行によって，ポインタ p を更新して次の要素に着目する．
- 空欄(5)について:
  - 左オペランドの代入式「idx = /*空欄(5)*/」と「-1」とが等しくないかどうかの判定が if 文の条件判定である．
  - このように，式の中に式を詰め込んだ表現は C++のプログラムで多用されるので，見てすぐに理解できるようにしておくとよい．

第 13 回 C++プログラミング実験Ⅱ

出題日:2022 年 7 月 4 日(月)
提出期限:2022 年 7 月 7 日(木) 24:00

実験課題 13-3

2 次元のデータの組が(x_0, y_0) (x_1,y_1), ・・・, (x_{n-1},y_{n-1}) と n 個与えられたとき, これらの点を通る関数 $f(x)$はラグランジュ補間を用いて次のように表すことができる.

$$f(x) = \frac{(x-x_1)(x-x_2)\cdots(x-x_{n-1})}{(x_0-x_1)(x_0-x_2)\cdots(x_0-x_{n-1})}y_0 + \frac{(x-x_0)(x-x_2)\cdots(x-x_{n-1})}{(x_1-x_0)(x_1-x_2)\cdots(x_1-x_{n-1})}y_1 \cdots \frac{(x-x_0)(x-x_1)\cdots(x-x_{n-2})}{(x_{n-1}-x_0)(x_{n-1}-x_1)\cdots(x_{n-1}-x_{n-2})}y_{n-1}$$

$$= \sum_{i=0}^{n-1} y_i \left(\prod_{j=0}^{n-1} \frac{x-x_j}{x_i-x_j} \right) \quad (j \neq i) \qquad = y$$

このラグランジュ補間式を用いることにより, 与えられたデータ点以外の点について計算することができる.

例えば, (x, y) : (0, 0), (1, 3) ,(2, 6), (3, 9), (4, 12), (5, 15) の 5 点のデータ点が与えられていたとする. これらをもとに, ラグランジュ補間を使って $x = 0.5, 1.5, 2.5\ 3.5, 4.5$ に対応する y の値を求めるとき, 上式に (x_0, y_0) から (x_5, y_5) を代入し, 各 x に対応する y を計算すると, $y = 1.5, 4.5, 7.5, 10.5, 13.5$ が求められる. 本課題では, この y の算出をプログラミングにて配列を利用して行う(今回は, $f(x)$の係数を求める必要はない).

① ラグランジュ補間を行うため, 関数 double lagrange(DATA*, int, double) を定義し, main 関数内の配列 XY1 にて与えられたデータ点 5 組以外の点($x = 0, 0.1, 0.2$ ・・・, 1.0) の y を求め, 表示せよ. lagrange 関数は, 仮引数として, 1 次元配列(DATA 型), 配列の要素数(int 型), データ点以外の x の値(double 型)を持ち, y の値を return する.
また, DATA は, double 型の変数 2 つをもつ構造体である.

② main 関数内で与えられたデータ XY1[ ] について, 計算・出力できたことを確認した後, xydata.csv ファイル中のデータ点を読み込み, 配列 XY2[10] へ代入し, XY2 配列にて与えられた点以外の点($x = 0, 0.2, 0.4$ ・・・, 2.0) について, 同様に計算・出力を行うこと (必要に応じて std::stod( ) を利用してよい) .

ex13-3.cpp

```
#include<iostream>
#include<iomanip>
#include<fstream>

struct DATA{
    double x{},y{};
};

double lagrange(DATA*,int,double);

int main(){
    DATA XY1[]={{0.15,0.63},{0.25,0.52},{0.35,0.47},{0.55,0.55},{0.95,1.43}};
    DATA XY2[10]{};
    int n{11};

    std::cout<<" x    y¥n";
    for(int i=0;i<n;i++){
        std::cout<<std::fixed<<std::setw(3)<<std::setprecision(1)<<0.1*i<<std::setw(6)
                 <<std::setprecision(2)<<lagrange(XY1,std::size(XY1),0.1*i)<<"¥n";
    }
    std::cout<<"¥n";

    //②ここにファイル読み込みと XY2 配列へのデータの代入を書く
    std::cout<<" x2   y2¥n";
    for(int i=0;i<n;i++){

        std::cout<<std::fixed<<std::setw(3)<<std::setprecision(1)<<0.2*i<<std::setw(6)
                 <<std::setprecision(2)<<lagrange(XY2,std::size(XY2),0.2*i)<<"¥n";
    }
}

//①ここに lagrange 関数を定義する.
```

実行結果 下線部は入力箇所

```
$./ex13-3
 x    y
0.0  0.91
0.1  0.71
0.2  0.57
0.3  0.49
0.4  0.47
0.5  0.51
0.6  0.61
0.7  0.77
0.8  0.99
0.9  1.27
1.0  1.61

 x2   y2
0.0  0.50
0.2  0.14
0.4  0.01
0.6  0.01
0.8  0.04
1.0 -0.00
1.2 -0.20
1.4 -0.65
1.6 -1.45
1.8 -2.70
2.0 -4.50
```