第2回C++輪講

第2回の内容

演算子

第1回では比較演算子のみ説明したが，算術演算子，論理演算子を加えたものを記す

算術演算子	概要	比較演算子	概要	論理演算子	概要
+	加算	==	左右の値が等しい	&&	and
-	減算	!=	左右の値が等しくない	||	or
*	乗算	>=	左の値が右の値以上	!	not
/	除算	>	左の値が右の値より大きい		
%	剰余算	<=	右の値が左の値以上		
		<	左の値が右の値より大きい		

- ビット演算子というものもある

数値計算

C++では<cmath>で一般的な数学関数および各種マクロが提供されている
2つの整数の絶対値を求める

```
#include <iostream>
#include <cmath>
```

```
namespace name{
namespace t2cl{
    overflow_suru();
    overflow_suru2();
    overflow_suru3();
}
}
int main()
{
    name::t2cl::cmathExample();
    return 0;
}
void name::t2cl::cmathExample()
{
    // input
    std::cout << "一つ目の整数を入力してください" << std::endl;
    int num1;
    std::cin >> num1;

    std::cout << "二つ目の整数を入力してください" << std::endl;
    int num2;
    std::cin >> num2;

    // calc
    auto result = std::fabs(num1 - num2);

    // output
    std::cout << "絶対値:\t" << result << std::endl;
}
```

型

第一回で説明していない型の話

型	bit	概要
bool	1bit	boolern
char	8bit	文字
short	16bit	整数
int	32bit	整数
long	64bit	整数
long long	64bit以上	整数
std::size_t	環境依存	符号なし整数
float	32bit	浮動小数点数
double	64bit	浮動小数点数
std::string		文字列

overflow

整数型の最大値最小値は<climits>で確認できる

```
#include <iostream>
namespace name{
namespace t2cl{
    overflow_suru();
    overflow_suru2();
    overflow_suru3();
}
}
int main()
{
    name::t2cl::overflow_suru();
    std::cout << "-------------" << std::endl;
    name::t2cl::overflow_suru2();
    std::cout << "-------------" << std::endl;
    name::t2cl::overflow_sinai();
    return 0;
}
void name::t2cl::overflow_suru()
{
    char num = 0;
    for (int i = 0; i < 260; ++i)
    {
        ++num;
        std::cout << +num << std::endl;
    }
}
void name::t2cl::overflow_suru2()
{
    std::vector<unsigned char> num{ 100, 200, 0 };
    num[2] = num[0] + num[1];
    std::cout << +num[2] << std::endl;
}
void name::t2cl::overflow_sinai()
{
    std::vector<unsigned char> num{ 100, 200 };
    auto result = num[0] + num[1];
    std::cout << result << std::endl;
}
```

cast

型変換
CとC++ではcastが異なる
C++ではstatic_cast<type>を利用しよう

```
#include <iostream>
void name::t2cl::castExample()
```

```
{
    std::vector<double> num{ 1.7 ,0.3 }
    //int result = (int)(num[0] + num[1]); //C
    int result = static_cast<int>(num[0] + num[1]); //C++
    std::cout << result << std::endl;
}
```

stringstream

連番保存するときなどで便利

```
#include <iostream>
#include <string>
#include <sstream>
void name::t2cl::ssExample()
{
    for (int i = 0; i < 10; ++i)
    {
        std::stringstream ss;
        ss << "test";
        ss << i;
        std::string txt = ss.str();
        std::cout << txt << std::endl;
    }
}
```

動的配列（vector）

vector - cpprefjp C++日本語リファレンス
静的配列は<array>というヘッダがある
あまり使わないため第1回ではvectorを静的配列として紹介した
興味があれば各自で調べること

```
#include <iostream>
#include <vector>
void name::t2cl::vectorInfo(std::vector<int> &vec)
{
    std::cout << "size:\t" << vec.size() << std::endl;
    std::cout << "capacity:\t" << vec.capacity() << std::endl;
    for (auto && elem : vec)    // range_based for loops
    {
        // vectorはこのようなfor文の書き方もある
        std::cout << elem << std::endl;
    }
}
void name::t2cl::vectorExample()
{
    std::vector<int> num(5, 4); // (要素数, 初期値)
    auto copiedNum = num;
```

```
    vectorInfo(num);
    std::cout << "--------------------" << std::endl;
    num.push_back(3);       // 配列の後ろに3を追加
    vectorInfo(num);
    std::cout << "--------------------" << std::endl;
    num.clear();            // 配列の中身を削除
    vectorInfo(num);
    std::cout << "--------------------" << std::endl;
    num.shrink_to_fit();    // 確保している領域を削除
    vectorInfo(num);
    std::cout << "--------------------" << std::endl;
    vectorInfo(copiedNum);
}
```

- autoを用いてfor文を書く方法の他に，イテレータを用いて書く方法もある

while文

```
void name::t2cl::whileExample()
{
    int i = 0;
    while (i < 20)  // i<20を満たすならばループ継続
    {
        std::cout << ++i << std::endl;
    }
}
void name::t2cl::whileTest2()
{
    int i = 0;
    while (true)
    {
        std::cout << ++i << std::endl;
        if(i == 20) break;
    }
}
void name::t2cl::whileTest3()
{
    int i = 0;
    while (true)
    {
        std::cout << ++i << std::endl; // ++iをi++にしてインクリメントによる違いを確認しよう
        if(i < 20) continue;
        break;
    }
}
```

キーワード「continue」「break」

switch文

条件分岐が複数ある場合に使用 if文ときちんと使い分ける必要がある

```
void name::t2cl::switchTest(int casenum)    //casenumにいろんな値を入れて動作を確認
してみよう
{
    switch (casenum)
    {
    case 1:
        std::cout << "case 1" << std::endl;
    case 0:
        std::cout << "case 0" << std::endl;
        break;
    case 2:
        std::cout << "case 2" << std::endl;
        break;
    case 10:
        std::cout << "case 10" << std::endl;
        break;
    default:
        std::cout << "default" << std::endl;
        break;
    }
}
```

実行時間の測り方

<chrono>ヘッダを利用

```
#include <iostream>
#include <chrono>
#include <thread> // <thread>ヘッダは実行時間計測のときには不要
void name::t2cl::chronoExample()
{
    using clock = std::chrono::steady_clock;
    using time_resolution = std::chrono::microseconds;

    clock::time_point begin = clock::now();
    std::this_thread::sleep_for(time_resolution(3000)); //3000ms待つ
    clock::time_point end = clock::now();

    time_resolution elapsed_time = std::chrono::duration_cast<time_resolution>
(end - begin);
    std::cout << elapsed_time.count() << "microseconds" << std::endl;
}
```

乱数生成方法

<random>ヘッダを利用

```
#include <iostream>
#include <random>
void name::t2cl::randomExample()
{
    std::random_device seed_gen;
    std::default_random_engine engine(seed_gen());

    // -500以上500以下の値を等確率で発生させる
    std::uniform_int_distribution<int> dist(-500, 500);

    for (int n = 0; n < 10; ++n) {
        // 一様整数分布で乱数を生成する
        int result = dist(engine);
        std::cout << result << std::endl;
    }
}
```

Coding Rules

C++は複雑になりがち
バグの混入を防ぐための工夫をしよう

- グローバル変数を用いない
- グローバル変数かつ多用されているものはuppercaseにする（例：UPPERCASE）
- インデント（スペース4つ）をきちんとする（スペース4つ分のタブではない）
- 一文字の変数名など他人が見てもわからない命名をしない
  命名に悩む初学者におすすめ→ https://codic.jp/
- C++では関数や変数の命名はcamelcaseに従う
    - キーワード「camelcase」「snakecase」「chaincase」

```
void camelCaseFuncNameExample();
void snake_case_func_name_example();
```

- 警告は全て消す
    - 文字コードはSJISではなくUTF-8やUTF-16を使用
- 名前空間を適切に使用する
- バージョン管理をする
- 変数は使う直前で宣言しよう
  処理がコンパクトになるように工夫してわかりやすく！

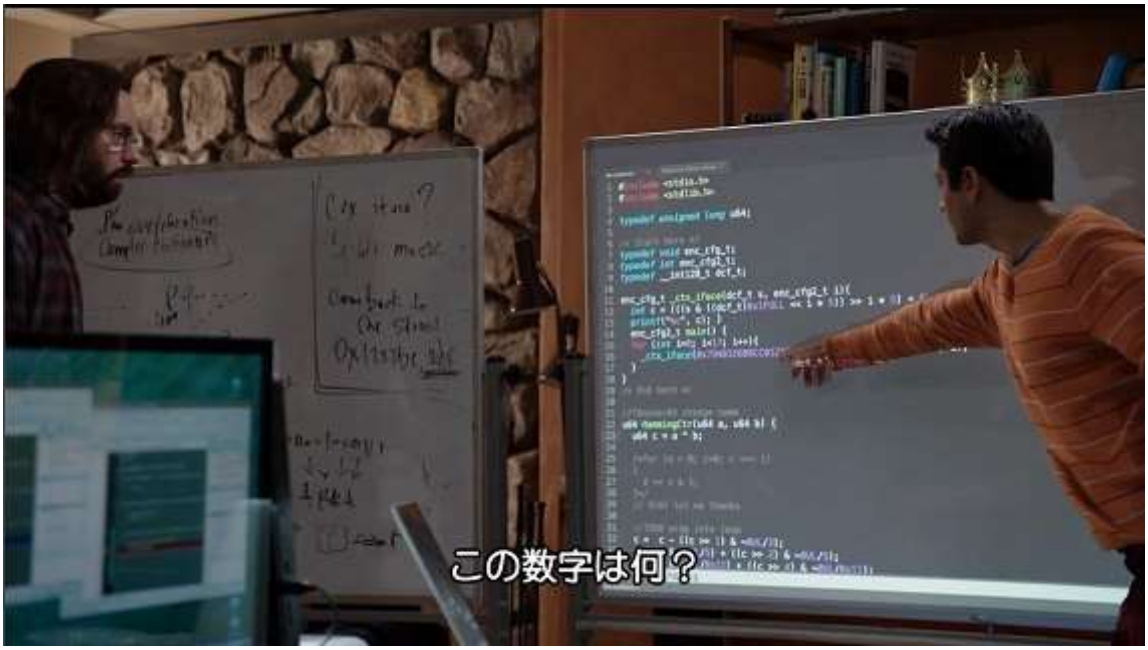
```
// 推奨されない例
int temp, i, j, k;
temp = 100;
// 処理
for(i = 0; i < 10; i++)
{
    for(j = 0; j < 10; j++)
```

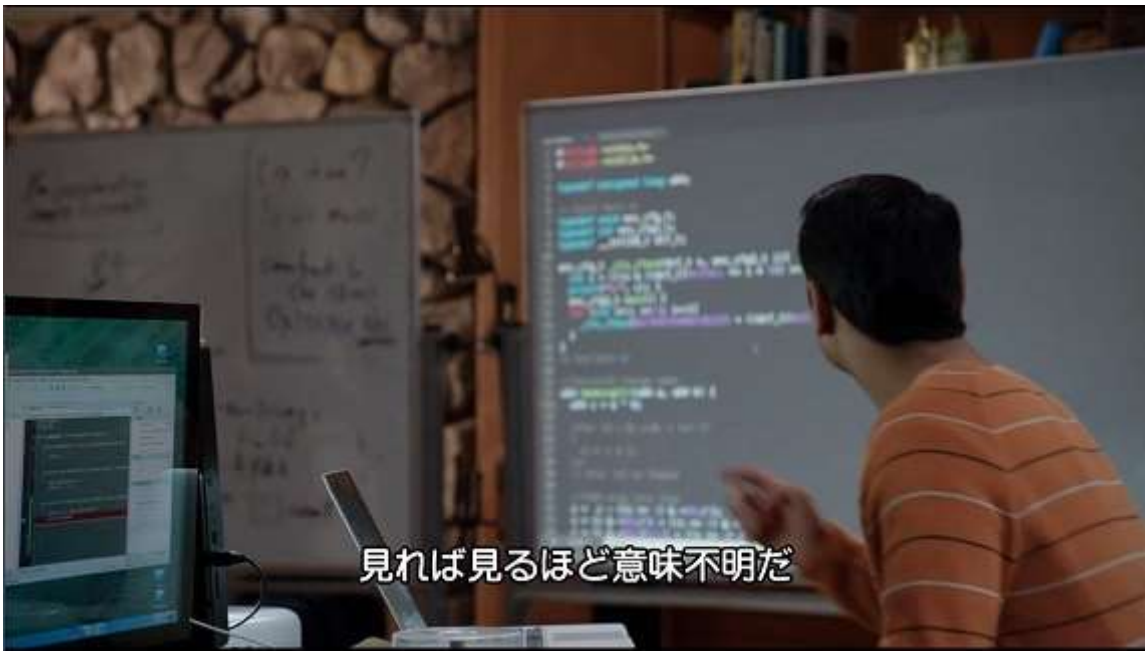
```
    {
        //いろんな処理
    }
}
```

```
// 改善例
int temp = 100;
// 処理
for(int i = 0; i < 10; ++i)
{
    for(int j = 0; j < 10; ++j)
    {
        int k;
        //いろんな処理
    }
}
```

さらに可読性，メンテナンス性，速度などを改善するためにも暇なときに目を通しておこう

- Google C++ Style Guide
- Effective C++ など





第2回課題

the2ndCppLecture_name.hpp，the2ndCppLecture_name.cppを作成し以下の関数を作成せよ
全ての関数は名前空間name，t2cl内に置け
関数run()で全ての関数が実行できるようにすること
引数の命名は変えてもよい
エラーの対処法などには目を通すこと

sortについてはOpen Data Structuresの227ページ以降を参考にするとよい
OpenDataStructures.jp

the2ndCppLecture_name.hpp

```
namespace name{
namespace t2cl{
    // 2次元座標，(x1，y1)，(x2，y2)を入力し，2点間の距離を求める関数
    double calcTheDistanceBetween(const int coordinateX1, const int coordinateY1
```

```
, const int coordinateX2, const int coordinateY2);
    // 半径x[m]の球の表面積と体積を求める関数
    // 円周率はPIを使うこと
    constexpr double PI = 3.14;
    double calcSphereVolume(const double radius);
    double calcSphereSurfaceArea(const double radius);
    // bubble sort（値の入れ替えには第1回課題で作成したswap関数を使用せよ）
    void bubbleSort(std::vector<int> &list);
    // quick sort
    void quickSort(std::vector<int> &list, const int left, const int right);
    //maerge sort
    void mergeSort(std::vector<int> &list, const int left, const int right);
    // sort3種の実行時間の計測と比較をせよ
    void comparisonOfSortingMethods(const std::string & filepath = "");
    void run(int num = 0); //ヘッダではこのように宣言すること
}
}
```

the2ndCppLecture_name.cpp

```
double name::t2cl::calcTheDistanceBetween(const int coordinateX1, const int
coordinateY1 , const int coordinateX2, const int coordinateY2)
{
    return 0.0;
}
double name::t2cl::calcSphereVolume(const double radius)
{
    return 0.0;
}
double name::t2cl::calcSphereSurfaceArea(const double radius)
{
    return 0.0;
}
void name::t2cl::bubbleSort(std::vector<int> &list)
{

}
void name::t2cl::quickSort(std::vector<int> &list, const int left, const int
right)
{

}
void name::t2cl::mergeSort(std::vector<int> &list, const int left, const int
right)
{

}
void name::t2cl::comparisonOfSortingMethods(const std::string & filepath)
{

}
void name::t2cl::run(int num)
```

```
{
    switch(num){
    default:
    case 1:
        std::cout <<"run calcTheDistanceBetween"<< std::endl;
        std::cout << t2cl::calcTheDistanceBetween(1, 1, 5, 4) << std::endl;
        if(num != 0) break;
    case 2:
        std::cout <<"run calcSphereVolume"<< std::endl;
        std::cout << t2cl::calcSphereVolume(10) << std::endl;
        if(num != 0) break;
    case 3:
        std::cout <<"run calcSphereSurfaceArea"<< std::endl;
        std::cout << t2cl::calcSphereSurfaceArea(10) << std::endl;
        if(num != 0) break;
    case 4:
        std::cout <<"run comparisonOfSortingMethods"<< std::endl;
        t2cl::comparisonOfSortingMethods("output.txt");
        if(num != 0) break;
    }
}
```