

1	内容	
2	前書き Visual Studio の使い方	4
3	プログラムの作成手順	4
4	・章末テスト	9
5	Lesson 1 はじめの一步	10
6	1.1 C++のプログラム	10
7	1.2 コードの入力	10
8	1.3 プログラムの作成	10
9	1.Ex C 言語の標準出力関数を知ろう(補足)	11
10	Lesson 1 章末テスト	11
11	Lesson 2 C++の基本	12
12	2.1 画面への出力	12
13	2.2 コードの内容	12
14	2.3 文字と数値	13
15	Lesson 2 章末テスト	15
16	Lesson 3 変数	16
17	3.3 型	16
18	3.5 変数の利用	16
19	3.5ex(補足) printf 関数を使用して、変数の値を出力する。	17
20	Lesson 3 中間テスト 1	17
21	3.6 キーボードからの入力	18
22	3.6.ex(補足) C 言語の標準入力関数 scanf_s	19
23	3.7 定数	19
24	Lesson 3 章末テスト	19
25	Lesson 4 式と演算子	20
26	4.1 式と演算子	20
27	Lesson4 中間テスト 1	21
28	4.2 演算子の種類	21
29	Lesson 4 中間テスト 2	22
30	Lesson 4 中間テスト 3	25
31	4.3 演算子の優先順位	25
32	4.4 型変換	25
33	Lesson 4 中間テスト 4	28
34	Lesson 4 章末テスト	29
35	Lesson 5 場合に応じた処理	30
36	5.1 関係演算子と条件	30

1	5.2 if 文 .....	30
2	5.2 実習 .....	30
3	5.3 if～else 文 .....	30
4	5.3 実習 .....	30
5	5.4 if～else if～else .....	31
6	5.4 実習 .....	31
7	5.ex_1 .....	31
8	Lesson 5 中間テスト 1 .....	33
9	5.5 switch 文 .....	33
10	5.6 論理演算子 .....	33
11	Lesson 5 章末テスト .....	35
12	Lesson 6 何度も繰り返す .....	36
13	Lesson6 を始める前に .....	36
14	6.1 for 文 .....	36
15	Lesson 6 中間テスト 1 .....	38
16	6.2 while 文 .....	39
17	6.3 do～while 文 .....	40
18	6.4 文のネスト .....	40
19	6.4 ex デバッガの機能のウォッチを使ってみよう。 .....	40
20	Lesson 6 中間テスト 2 .....	41
21	6.5 処理の流れの変更 .....	42
22	Lesson6 章末テスト .....	44
23	Lesson 9 配列 .....	45
24	9.1 配列の基本 .....	45
25	9.2 配列の宣言 .....	47
26	9.3 配列の利用 .....	47
27	Lesson_09 実習課題_1 .....	47
28	課題 1 .....	47
29	課題 2 .....	48
30	課題 3 .....	48
31	課題 4 .....	48
32	Lesson 9 中間テスト 1 .....	48
33	Lesson 7 配列 .....	49
34	7.1 関数の仕組みを知る .....	49
35	7.2 関数の定義と呼び出し .....	51
36	7.2 Ex_1 関数化を行う理由 .....	53

1	7.2 Ex_2 関数は分かりやすい名前をつける。.....	57
2	Lesson_07 ハンズオン 1.....	58
3	Lesson_07 実習課題_1.....	58
4	Lesson_07 中間テスト 1.....	58
5	7.3 引数.....	59
6	7.3 Ex_3 引数はローカル変数.....	59
7	7.3 Ex_2 引数の値渡し.....	60
8	7.3 Ex_2 引数の参照渡し.....	61
9	7.3 Ex_2.1 参照って何??? .....	61
10	Lesson_07 ハンズオン 2.....	62
11	Lesson_07 実習課題_2.....	62
12	Lesson_07 中間テスト 2.....	62
13	7.4 戻り値.....	63
14	7.4.1 サンプルプログラム.....	63
15	7.4.2 サンプルプログラム(浮動小数点を返す関数).....	64
16	7.4.3 サンプルプログラム(bool 型を返す関数).....	64
17	7.4.4 サンプルプログラム(if 文の中で関数呼び出し).....	65
18	7.4.5 サンプルプログラム(if 文の中で関数の戻り値をそのまま使う).....	65
19	7.4 Ex 関数名は分かりやすい名前を考えよう! .....	66
20	Lesson_07 ハンズオン 3.....	67
21	Lesson_07 実習課題_3.....	67
22	Lesson_07 中間テスト 3.....	67
23	7.6 関数宣言 .....	67
24	Lesson 11 いろいろな型.....	69
25	11.3 構造体.....	69
26	11.4 構造体の応用 .....	71
27	11.Ex 1 構造体の発展型としてのクラス.....	72
28	Lesson 11 ハンズオン_1.....	75
29	Lesson 11 実習課題_1.....	75
30	Lesson 11 実習課題_2.....	75
31	Lesson 11 実習課題_Ex_1.....	75
32	Lesson 11 中間テスト 1.....	75
33		
34		
35		
36		

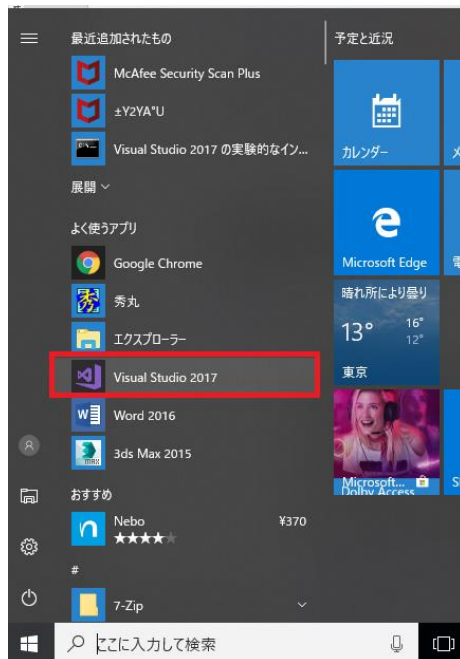
## 前書き Visual Studio の使い方

### プログラムの作成手順

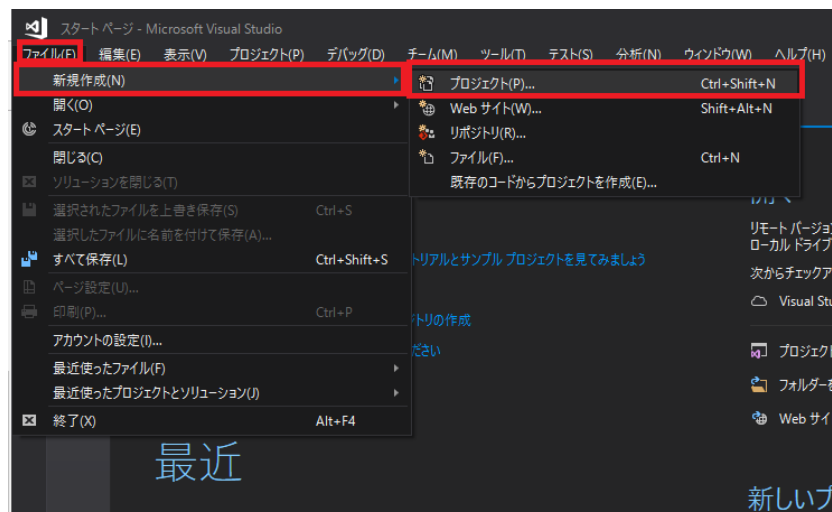
プロジェクト Lesson0 を作ってみましょう。大文字の入力は shift キーを押しながらで行えます。

### 1. プロジェクトの作成

Start メニューから VisualStudio を起動する。



ファイル→新規作成→プロジェクトを選択



ドキュメントの下に c\_purapura\_1 というフォルダを作って、そこにプロジェクトを作りましょう。\_(アンダースコア)は shift + / で入力できます。図1の赤枠を参照。

1 図 1



2

3 2. プロジェクトのプロパティを変更する。

4 ソリューションエクスプローラーでプロジェクトを選択→右クリック→プロパティ



5

6

7

8

9

10

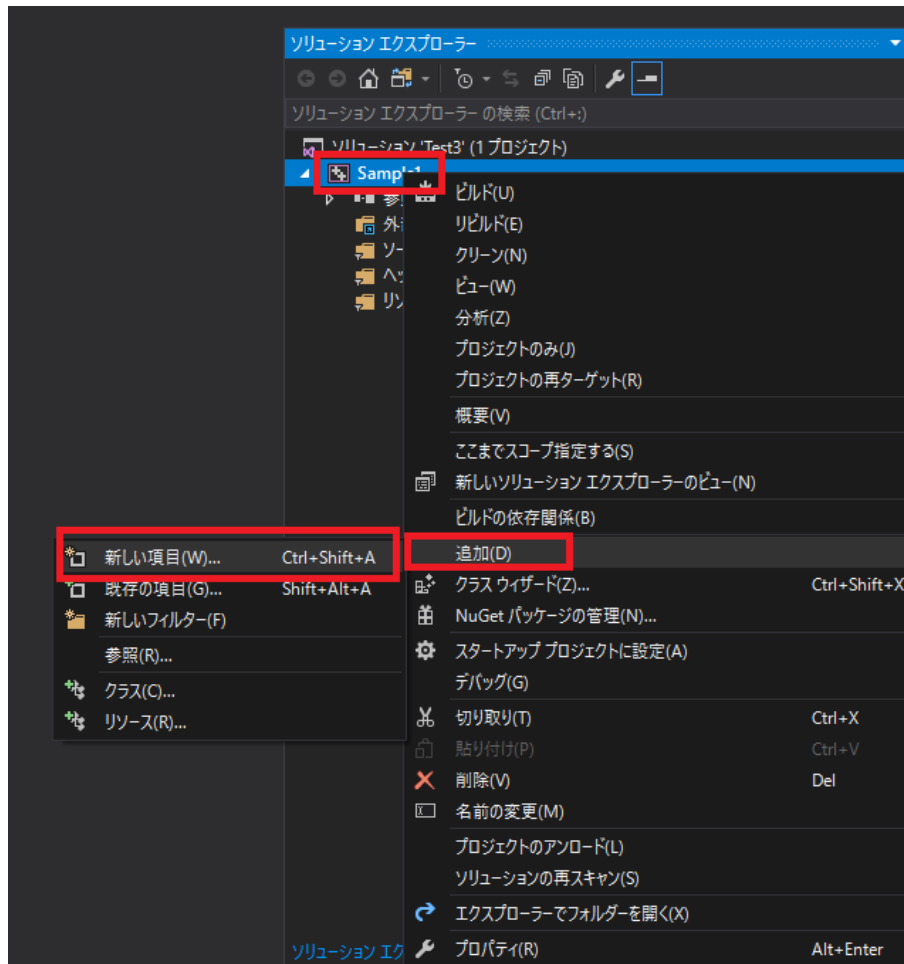
11

12

13

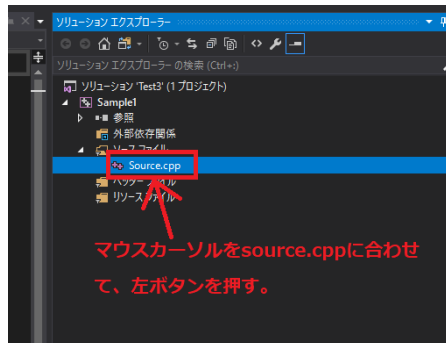
1 3.ソースファイルの追加

- 2 ソリューションエクスプローラーでプロジェクトを選択→右クリック→追加→新しい項  
3 目を選択。



#### 4. ソースコードの入力

追加したソースファイルを選択。



選択が出来たら、下記のようにコードを入力する。

```
#include <iostream>
using namespace std;

int main()
{
    cout << "ようこそC++へ! ¥n";

    return 0;
}
```

次の4点に注意する。

- ① スペースは半角。入力を忘れないように。図2，3の青枠。
- ② 大文字、小文字に注意。
- ③ #、<、(、)、{、}、”(ダブルクォーテーション)などは Shift キーを押しながら入力する。図2，3の赤枠。
- ④ ;(セミコロン)を忘れない。上記コード、図2，3の緑枠。
- ⑤ 日本語入力→英字入力は半角/全角ボタンで切り替えることができる。図2の黄色枠。

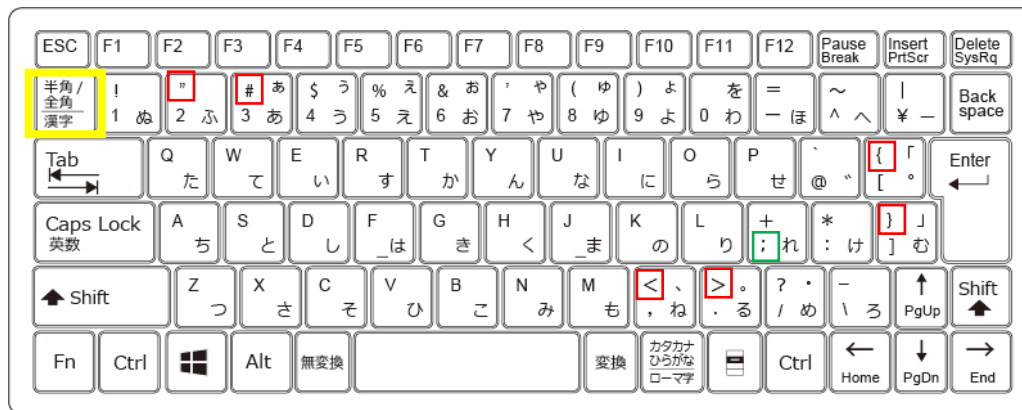
1 図 2

```
#include<iostream>
using namespace std;

int main()
{
    cout<<"ようこそC++へ!¥n";

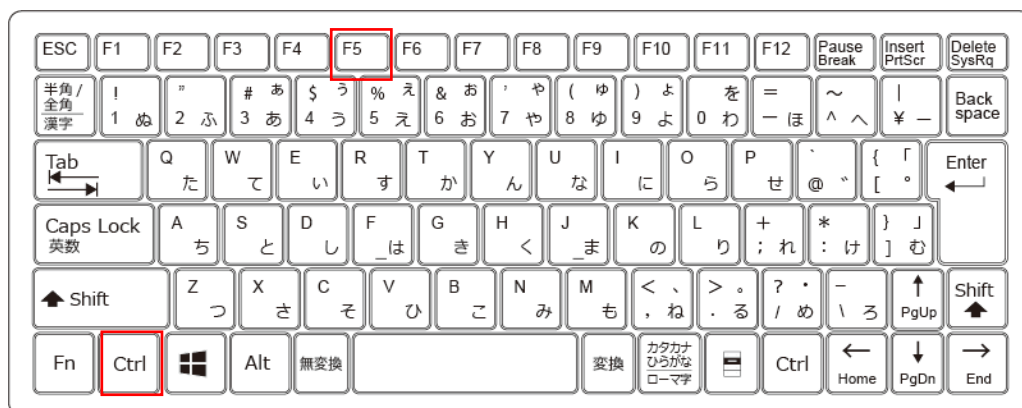
    return 0;
}
```

3 図 3



5. ソースコードのビルド→実行

6 Ctrl+F5 でデバッグなしで実行できます。下記図の赤枠。



6. コマンドプロンプト(黒いウィンドウ)がすぐに消える

9 この現象が発生した場合は、2 をやり直す。



・章末テスト

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSfWXuo1SqrWla4WEOIGEVUOJlqHg\\_vBP6p-5FZtuyqzRLNs5A/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSfWXuo1SqrWla4WEOIGEVUOJlqHg_vBP6p-5FZtuyqzRLNs5A/viewform?usp=sf_link)

## Lesson 1 はじめの一步

### 1.1 C++のプログラム

#### 機械語(p.3)

→こんなの。

```
3E 4D 66 02 0B 1A 68 8E 6E 4D 75 97 47 2A EC 9A
56 13 6F 94 EE E2 DE 1F 51 DE 79 DA 5E 56 45 69
81 91 A0 3A F3 7B E3 59 53 82 71 9E C1 E7 A7 03
A3 0F 91 E1 91 DF 6D BE F7 9D A5 25 1F 00 7F 6E
ED 7C 82 8B DE F1 8D C6 44 0B D0 01 58 11 1A 3C
63 4D 23 E1 22 92 73 6C C7 A6 FE 23 11 70 00 28
54 17 D3 F2 04 B7 96 7C 1F F2 21 7D C7 7F 87 77
2A 87 87 78 0B C0 6D F6 31 78 7D 82 7C 82 C4 26
E4 71 2B 93 95 01 0A 00 00 19 90 27 F0 36 43 52
81 00 9D A4 47 47 37 30 9E 36 01 F3 A1 37 AB 02
```

※図(機械語(マシン語))の一種。

もちろん、一文字でも間違えると動かない(汗))

### 1.2 コードの入力

#### 統合環境を使ってみる(p.5)

本校の授業では Visual Studio(統合環境)を使います。

#### テキストエディタにコードを入力する。(p.6)

新しいプロジェクト、Lesson1を作成して、ソースファイルを追加して、p.7のSample1.cppの内容を入力する。

#### コンパイルエラーを起こしてみる

コードを書くことができ、実行出来たら、わざとコンパイルエラーを起こしてどうなるか確認してみましょう。

### 1.3 プログラムの作成

#### コンパイラを実行する(P9)

オブジェクトファイルを確認してみる。

#### オブジェクトファイルをリンクする(P9)

実行ファイルができていることを確認する。

## 1.Ex C 言語の標準出力関数を知ろう(補足)

### printf 関数

cout は C++ の標準出力関数ですが、printf は C 言語の標準出力関数となります。C++ は C 言語のスーパーセット言語となっており、C 言語の機能はすべて使えます。では、下記のコードを入力してみてください。

```
#include <stdio.h>

int main()
{
    printf("ようこそ C++へ!");
    return 0;
}
```

C 言語検定では標準出力関数として、printf 関数を使用されていることと、ほかの言語でも似たような print 関数が用意されているので、こちらを使えるようになることも重要です。

## Lesson 1 章末テスト

下記の URL のテストを行いなさい。

<https://docs.google.com/forms/d/1NPbcFTvH37P5kJ3pHvOIB0GWXZoNYUG93aCOaEYAXYA/edit>

## Lesson 2 C++の基本

### 2.1 画面への出力

#### 新しいコードを入力する(p.16)

新しいプロジェクト Lesson2 を作成して、Sample.1.cpp の内容を入力して実行してください。

#### 色々な出力方法を知る(p.18)

Sample2.cpp の内容を入力して実行してください。

#### 改行には を使う

テキストにも書かれていますが、重要です。しっかりと覚えましょう。

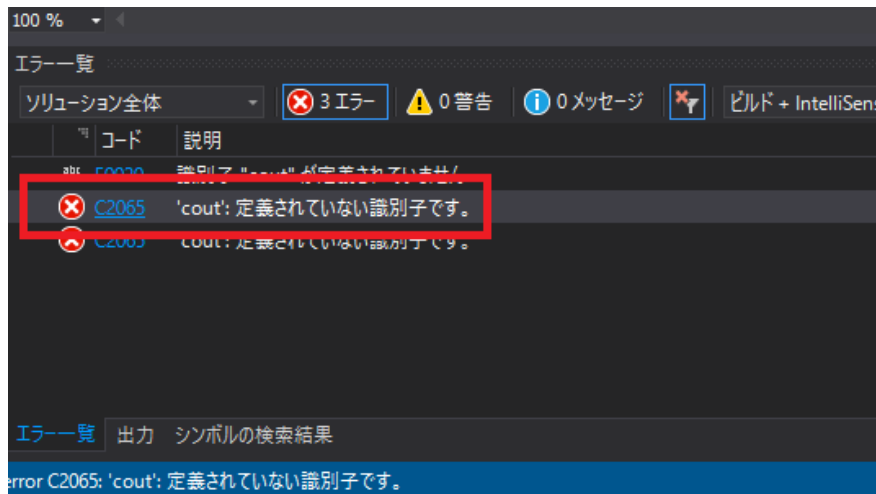
### 2.2 コードの内容

#### main()関数

main 関数はエントリーポイントと呼ばれる特殊な関数です。プログラムのスタート地点となる関数です。

#### #include <iostream>をコメントアウトしてみる。

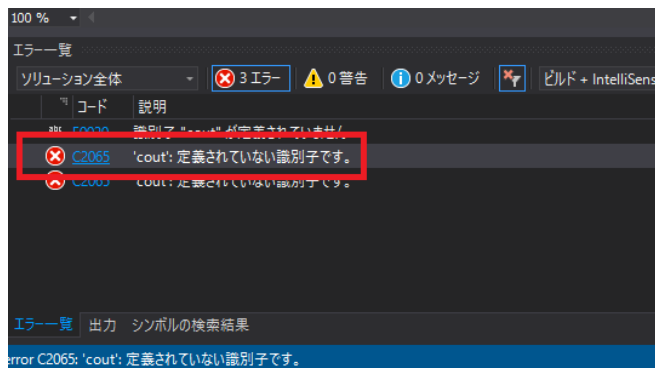
#include <iostream>をコメントアウトして、コンパイルしてみましょう。下記のようなエラーが表示されたと思います。



エラーをダブルクリックすると、エラーが起きている箇所にジャンプできます。試してみてください。

using namespace std;をコメントアウトしてみる。

こちらもコメントアウトしてコンパイルしてみてください。先ほどと同様のエラーが表示されたと思います。



## 重要

エラーをダブルクリックするとコンパイルエラーが起きている箇所にジャンプできることをしっかりと覚えましょう。

## 2.3 文字と数値

### 数値リテラル

123、579、30.0 など

### 文字リテラル

‘A’、‘B’、‘c’など。シングルクォーテーションで囲まれたもの。一文字を表す。文字列とは違う！！

### 文字列リテラル

“ABC”、“ようこそ C++”など。ダブルクォーテーションで囲まれたもの。

### エスケープシーケンス

全部を覚える必要はない。重要なのは下記の3点です。

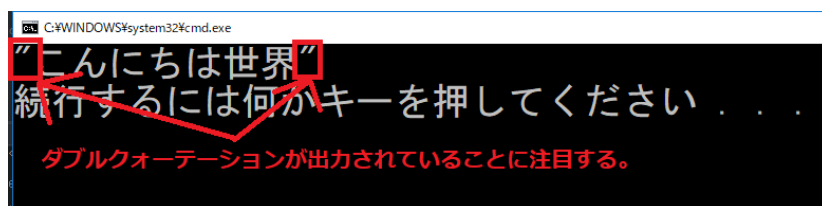
¥n 改行コード

¥0 文字列の終わり(詳細は Lesson9.7)

¥マークをつけることで、¥や’や”などの特殊文字を文字列に組み込むことができる。

→どういうこと？

例えば、下記のような文字列を表示する場合を考えてみましょう。



- 1 ダブルクォーテーションは文字列リテラルを囲むものという、特殊文字となっているため、  
2 下記のようなプログラムではコンパイルエラーになります。

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "こんにちは世界"¥n";
8     return 0;
9 }
```

- 3  
4  
5 文字としてダブルクォーテーションを出力したい場合は、下記のように記述する必要があります。  
6  
7

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     cout << "¥"こんにちは世界¥"¥n";
8     return 0;
9 }
```

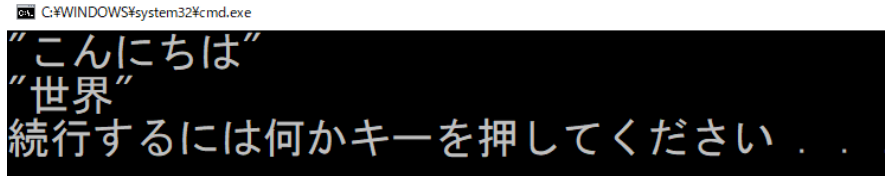
¥をつけることで、"はただの文字として扱われる。

- 8  
9 その他のエスケープシーケンスが必要になったら、ネットで検索をすればOKです。ググり  
10 ましょう。

- 11  
12  
13  
14

## 実習 1 (時間 5 分)

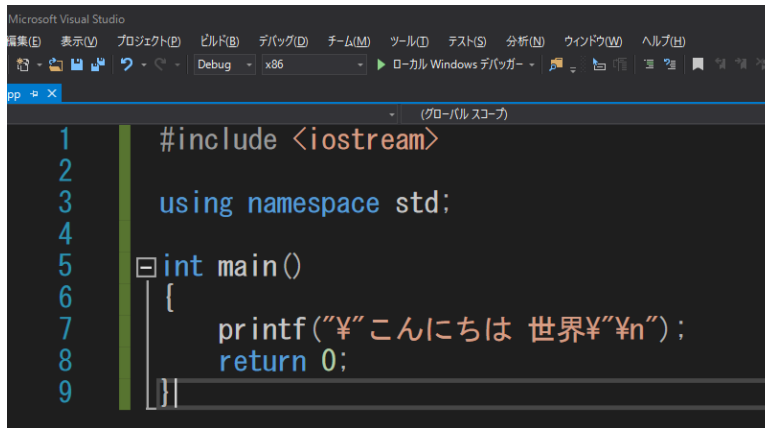
一度の cout の実行で下記のような表示ができるようにしてみよう。



```
C:\WINDOWS\system32\cmd.exe
"こんにちは"
"世界"
続行するには何かキーを押してください . . .
```

## printf 関数でも同じ

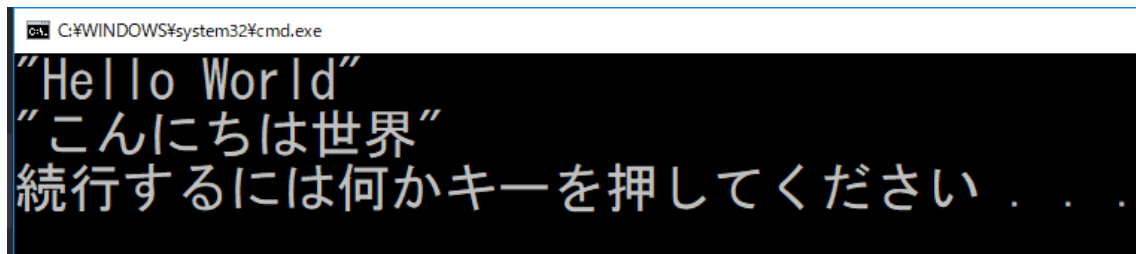
エスケープシーケンスのルールは printf 関数でも同じです。



```
Microsoft Visual Studio
編集(E) 表示(V) プロジェクト(P) ビルド(B) デバッグ(D) チーム(M) ツール(T) テスト(S) 分析(A) ウィンドウ(W) ヘルプ(H)
Debug x86 ローカル Windows デバッガ
pp (グローバル スコープ)
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     printf("¥こんにちは 世界¥¥n");
8     return 0;
9 }
```

## 実習 2 (時間 5 分)

一度の printf 関数の実行で下記のような表示ができるようにしてみよう。



```
C:\WINDOWS\system32\cmd.exe
"Hello World"
"こんにちは世界"
続行するには何かキーを押してください . . .
```

## 8 進数

10 を 8 とする表記法。

## 16 進数

10 を 16 とする表記法

## Lesson 2 章末テスト

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSdNjkkwZ4M0L5\\_ujSSc-GULWBWXiA\\_dRPSHZ8ct2XpUrocsjg/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSdNjkkwZ4M0L5_ujSSc-GULWBWXiA_dRPSHZ8ct2XpUrocsjg/viewform?usp=sf_link)

## Lesson 3 変数

### 3.3 型

数が多くて、いきなり覚えるのは難しいと思いますが、今は下記の4点を覚えてください。

整数型                      int

浮動小数点型              float

文字型                      char

unsigned をつけると      符号なしになる。(頭の隅の方にでも置いておいてください。)

```
int hoge = -10;              //hoge は符号付き整数型。負数も記憶できる。  
unsigned int hoge2 = 20; //hoge2 は符号なし整数型。負数は記憶できない。
```

### 3.5 変数の利用

#### 変数に値を代入する

```
int num;  
num = 3;
```

=が代入であることに注意してください。数学であればイコールですが、プログラムでは代入です。右辺の値を左辺に代入します。簡単に思えるかもしれませんが、実は、ここの理解で躓く人が多いです。

#### p.50 のコードを入力して結果を確認しよう

新しいプロジェクト Lesson3 を作成して、p.50 のコードを入力して確認しましょう。

#### 変数を初期化する

変数は宣言することで、数値を記録するための領域がメモリ上に確保されます。では下記のようなコードの場合、どのような値が表示されるのでしょうか？

```
int num;    //num という変数を用意する。  
cout << num << "¥n"; 何が表示される？
```

#### 変数の値を変更する

p.53 のコードを入力して、動作を確認しましょう。

#### 他の変数の値を代入する

p.55 のコードを入力して、動作を確認しましょう。

#### 値の代入についての注意

p.56 のコードを入力して、動作を確認しましょう。



1 double は float でも OK です。

2

3 3.5ex(補足) printf 関数を使用して、変数の値を出力する。

4 cout を使用して変数の中身を出力する方法は見てきましたが、printf 関数も変数の値を出力  
5 することができます。int 型の変数の値を出力する場合は下記のように記述します。

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int num = 10;
8     printf("numの値は%dです\n", num);
9     return 0;
10 }
```

%dの部分がnumの値に置き換わる。

6

7 また、下記のように記述することで、複数の変数の値を出力できます。

```
1 #include <iostream>
2
3 using namespace std;
4
5 int main()
6 {
7     int num = 10;
8     int num2 = 20;
9     printf("numの値は%dです。num2の値は%dです。n", num, num2);
10     return 0;
11 }
```

8

9

10 Lesson 3 中間テスト 1

11 下記の URL のテストを行いなさい。

12 [https://docs.google.com/forms/d/e/1FAIpQLScDqY-](https://docs.google.com/forms/d/e/1FAIpQLScDqY-iHav90m71Sr1uTCXsHrTW1t5TJLkW1oNA0xNZUAtfmg/viewform?usp=sf_link)

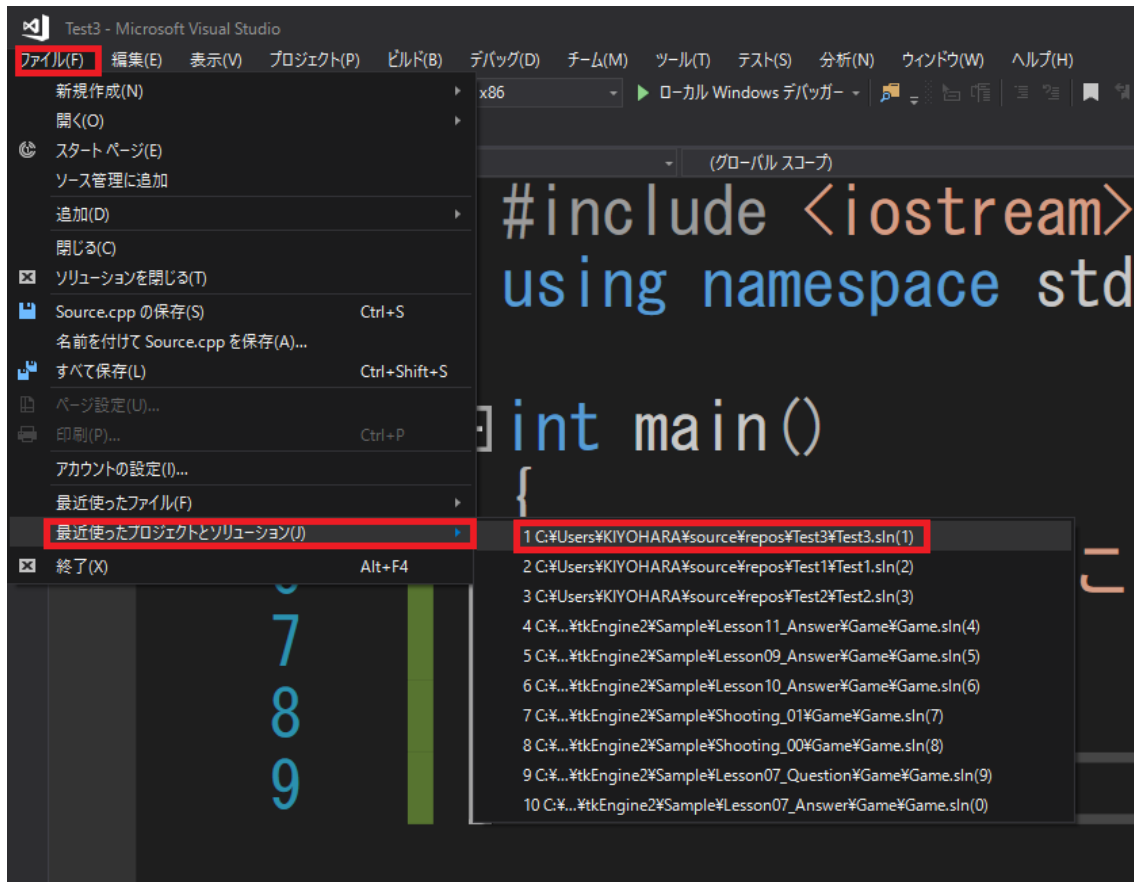
13 [iHav90m71Sr1uTCXsHrTW1t5TJLkW1oNA0xNZUAtfmg/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLScDqY-iHav90m71Sr1uTCXsHrTW1t5TJLkW1oNA0xNZUAtfmg/viewform?usp=sf_link)

14

### 1 3.6 キーボードからの入力

#### 2 キーボードから入力する

3 最近使ったプロジェクトから Lesson3 を起動して、P.60 の Sample5.cpp の内容を打ち込んで下さい。



#### 5 C++標準入力関数 cin

7 cin を実行するとキーボードからの入力待ちになります。

#### 9 二つ以上の数値を入力する

10 p.61 の Sample6 の内容を打ち込んでください。

### 3.6.ex(補足) C 言語の標準入力関数 scanf\_s

C++の標準入力関数は cin ですが、C 言語にも同様の関数の scanf\_s があります。下記のよ  
うなコードを入力することで、cin と同じ動作になります。入力して確認してください。

```
1  #include <iostream>
2
3  using namespace std;
4
5  int main()
6  {
7      int num;
8
9      printf("整数を入力して下さい。\\n");
10
11     scanf_s("%d", &num); 入力された値が、numに入る。
12                             &がついていることに注意す
13                             る!!!
14
15     printf("%dが入力されました。", num);
16     return 0;
17 }
```

(注意)正確には、C 言語の標準入力関数は scanf 関数なのですが、この関数はセキュリティホールが存在しているため、マイクロソフトがより安全な scanf\_s 関数を用意しています。使い方は scanf と同じなので、混乱しないようにしてください。

### 3.7 定数

P.63 の Sample7.cpp のコードを入力してみてください。

変更することのできない変数に const をつける。

プログラマのミスで値を変更してしまうヒューマンエラーをなくす。  
今、定数の利点を理解するのは難しいと思うので、これは後期に詳しくやります。

### Lesson 3 章末テスト

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSfgUfWrA1P6vhlj-UqzQr3CBhHYmOxpzsFYOo0mdsmuQpSU9A/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSfgUfWrA1P6vhlj-UqzQr3CBhHYmOxpzsFYOo0mdsmuQpSU9A/viewform?usp=sf_link)

## Lesson 4 式と演算子

### 4.1 式と演算子

#### 式の値を出力する(p.71)

新しいプロジェクト Lesson4 を作成して、Sample1.cpp を入力して実行してください。  
プログラムの世界では\*が掛け算になります。\*は shift+；、と+は shift+；で入力できます。  
下記の図の赤枠を参照。



#### 色々な演算をする(p.71)

下記の2点をしっかりと意識しながら、Sample2.cpp のコードを入力してください。

- ・ 変数と変数の演算を行える。
- ・ 変数と数値リテラルとの演算を行える。

#### 変数 num1 の値に 1 を足し、その値を再度 num1 に代入する(p.73)

下記のコードに注目してください。

```
num1 = num1 + 1;
```

数学的にはおかしい式です。 = の記号が等しいではなく、代入であったことを思い出してください。右辺の結果(num1+1)を左辺に代入しているため、このような記述が可能になります。

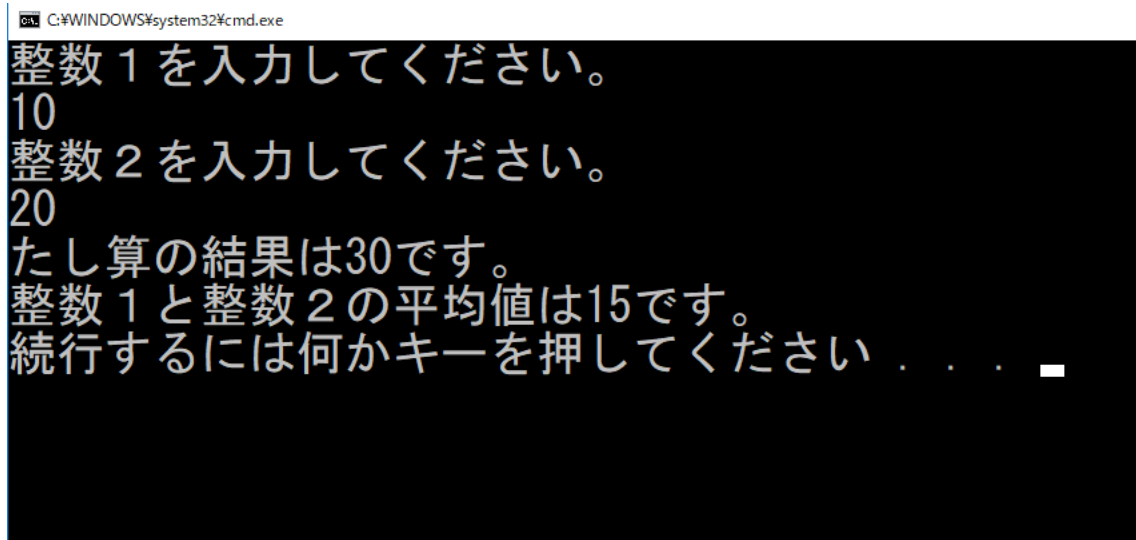
#### キーボードから入力した値を足し算する(p.74)

Sample3.cpp の内容を入力して、実行してください。

## 実習 1(10 分)

Lesson4 の内容を改造して、下記のような表示をできるようにしなさい。

また、除算の演算子は / です。



```
C:\WINDOWS\system32\cmd.exe
整数 1 を入力してください。
10
整数 2 を入力してください。
20
たし算の結果は30です。
整数 1 と整数 2 の平均値は15です。
続行するには何かキーを押してください . . .
```

## Lesson4 中間テスト 1

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSfyMQIL4RSN9mjOLdaeRnAJv2mDBi4U-WjkY4mywBdxOnw-uQ/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSfyMQIL4RSN9mjOLdaeRnAJv2mDBi4U-WjkY4mywBdxOnw-uQ/viewform?usp=sf_link)

## 4.2 演算子の種類

### いろいろな演算子(p.76)

非常に多くの演算子があります。残念なことに、ここに記載されている演算子はすべて重要です。世のプログラマーはすべてを覚えていると思います。でも、安心してください。覚えようとしなくてもプログラムを書いていると、自然と覚えてしまいます。

まずは四則演算(+ - × ÷) +  $\alpha$  をマスターしましょう。

では、Lesson4 を最近使ったプロジェクトから開いて、p.77 の Sample4.cpp のコードを入力して、実行してください。

## 1 実習 2(10 分)

2 Lesson4 を改造して、次の動画のような挙動になるプログラムを作成しなさい。

3 出席番号で 0 ～ 4 のグループに振り分けるプログラム。

4 <https://www.youtube.com/watch?v=lRwpa-nbEuI&feature=youtu.be>

## 6 インクリメント・デクリメント演算子(p.79)

7 これも四則演算子なのですが、少し変わっています。よく使う演算子なのでしっかりと覚えましょう。

## 10 インクリメント・デクリメントの前置と後置(p.80)

11 前置と後置で実行結果が変わることがあります。普段プログラムを書くときは、この規則を意識しなくてもいいように書く方が優れている場合がほとんどですが、資格・検定の試験でこれを問う問題がでることがあるので、覚えましょう。

14 Lesson4 に p.81 の Sample.cpp のコードを入力して、実行してください。

15 下記が前置と後置の挙動の違いの覚え方です。

17 ・前置なので、代入する前にインクリメント・デクリメントする。

18 ・後置なので、代入した後でインクリメント・デクリメントする。

20 ただし、これに依存するようなコードを書くことは可読性を下げることになるので、下記のようなコードを書くことを推奨します。

### 23 代入する前にインクリメントしたい場合

```
a++; //インクリメントしてから  
b = a; //代入する。
```

### 25 代入した後でインクリメントしたい場合

```
b = a; //代入してから、  
a++; //インクリメントする。
```

## 27 Lesson 4 中間テスト 2

28 下記の URL のテストを行いなさい。

29 [https://docs.google.com/forms/d/e/1FAIpQLSdAUAv0FP5S5zUoNbznYQUhUVoGBI82IVA6wdh3LZrc0I-oRA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSdAUAv0FP5S5zUoNbznYQUhUVoGBI82IVA6wdh3LZrc0I-oRA/viewform?usp=sf_link)

## 代入演算子(p.83)

Lesson4 に p.85 の Sample6.cpp を入力して実行してください。

## sizeof 演算子

この演算子は、配列(Lesson9)の要素数を調べるときなどに使用されることがあります。国家試験でもよく出てくる演算子です。例えば下記のように使います。

```
int num = sizeof(int);
```

この演算子は読んで字のごとく、「size of int」int 型の(of) サイズ(size)を求めてくれます。では、int 型のサイズはいくつだったのでしょうか？教科書の p.43 に戻って確認してみましょう。

## 確認テスト

下記のプログラムの実行結果を答えなさい。

```
int main()
{
    int hoge = 0;
    int size = sizeof(hoge);
    cout << size << "¥n";
    return 0;
}
```

答え

\_\_\_\_\_

```
int main()
{
    char hoge = 'w';
    int size = sizeof(hoge);
    cout << size << "¥n";
    return 0;
}
```

答え

\_\_\_\_\_

## 1 4Hands-On 2

2 Lesson 4 に下記のコードを入力して動作を確認しなさい。

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      double d = 10.0;           //doubleは8バイト。
7      int a = 10;                //intは4バイト。環境によるけど。
8
9      int hoge = sizeof(d) / sizeof(a);
10     cout << hoge << "¥n";
11     return 0;
12 }
```

## 3 シフト演算子(p.88)

4 下記の2点を覚える。

- 5 ・ 1ビット左にシフトすると値は倍になる。
- 6 ・ 1ビット右にシフトすると値は半分になる。

## 7 Hands-On 3

8 Lesson4 に下記のコードを入力して動作を確認しなさい。

```
1  #include <iostream>
2  using namespace std;
3
4  int main()
5  {
6      int num = 10;
7      num = num << 1;           //1ビット左にシフト。
8      cout << num << "¥n";     //20と表示される。
9
10     num <<= 2;                //2ビット左にシフト。
11     cout << num << "¥n";     //80と表示される。
12
13     num = num >> 1;           //1ビット右にシフト。
14     cout << num << "¥n";     //40と表示される。
15
16     num >>= 2;                //2ビット右にシフト。
17     cout << num << "¥n";     //10と表示される。
18     return 0;
19 }
```



## Lesson 4 中間テスト 3

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLScoCxdw4wk-kCYEglROeYF9pFD-tRO5MOZj9S-LSrc4pMY2Jw/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLScoCxdw4wk-kCYEglROeYF9pFD-tRO5MOZj9S-LSrc4pMY2Jw/viewform?usp=sf_link)

### 4.3 演算子の優先順位

#### 演算子の優先順位とは(P.90)

下記の 2 点を抑えましょう。

- ・四則演算の優先順位は数学と同じ。
- ・優先順位が分からなかったら()を使えば良い。

この 2 点を押さえておけば、p.91～p92 の表は覚える必要はありません。この表を答えられるプログラマーなどいないのですから。

#### 同じ優先順位の演算子を使う(P93)

これも難しいことを考える必要はありません。抑える点は同じです。

- ・四則演算の優先順位は数学と同じ。
- ・優先順位が分からなかったら()を使えば良い。

### 4.4 型変換

#### 大きなサイズの型に代入する(P.94)

Lesson4 に Sample8.cpp の内容を入力して実行してください。

#### 小さなサイズの型に代入する(P.95)

小さなサイズの型に代入すると、値が失われるのは正しいですが、このテキストに書かれている内容は、ちょっと不正確です。この Sample9 は正しくは、浮動小数点型の変数の値を整数型の変数に代入すると、小数点の値が失われるです。

Lesson4 に下記のコードを入力して実行してください。

```
#include <iostream>
using namespace std;

int main()
{
    float fnum = 50.5f;    //floatは32bitの浮動小数点。
    int inum = fnum;        //intも32bitなので、値は失われないはず？
    cout << inum << '¥n'; //教科書の説明は正しくない。小数点は失われる！
    return 0;
}
```

- 1 では、「小さなサイズの型に代入すると値が失われる」の正しい説明となるコードを見てみ  
2 ましょう。  
3 Lesson4 に下記のコードを入力して実行してください。

```
#include <iostream>
using namespace std;

int main()
{
    int inum = 100000;    //intは32bitは符号付きの整数型。
                        //表現できる範囲は、-2,147,483,648 ~ 2,147,483,647
    short snum;          //shortは16bitの符号付きの整数型。
                        //表現できる値の範囲は、-32,768 ~ 32,767
    snum = inum;          //snumに100000という値は表現できないので、代入すると・・・
    cout << snum << '¥n'; //値が失われる！
    return 0;
}
```

- 4  
5 **キャスト演算子を使う(P.96)**  
6 明示的な型変換を行う。  
7 →教科書に書かれていることをしたところで、結果は何も変わらない。  
8 →え？じゃあキャストってなんのためにするの？  
9 型変換を行うと、小さなサイズの型に代入すると値が失われてしまいます。つまり下記のよ  
10 うなコードを書いてしまった場合、致命的な不具合を生み出すことがあります。  
11  
12 (このコードは書かなくていい)

```
#include <iostream>
using namespace std;

int g_playerHp = 100000;    //int型のプレイヤーのHPを表すグローバル変数。

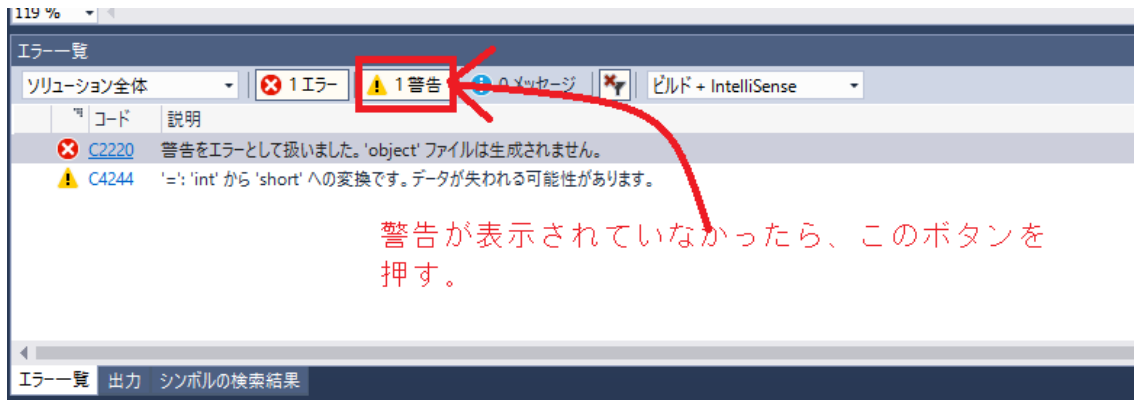
int main()
{
    short playerHp = g_playerHp;    //プレイヤーのHPをshort型のローカル変数に代入！
    playerHp -= 100;                //プレイヤーのHPを100減らすことが目的なのだが・・・
    g_playerHp = playerHp;          //変更したHPを書き戻す。
    cout << g_playerHp << '¥n';    //なんてこった。
    return 0;
}
```

- 13 このようなヒューマンエラーを防ぐために、明示的ではない、小さな型への変換を行うコー  
14 ドを書いた場合にコンパイルエラーにすることができます。  
15 次の設定を行ってから、コンパイルを行ってください。

- 16  
17  
18  
19  
20  
21



- 1 この設定で Lesson4 のコンパイルを行うと、下記のようなエラーと警告が出てくると思っ  
2 ます。



- 3  
4  
5 こうすることで、意図していない型変換を行うコードを書いてしまった場合はコンパイル  
6 エラーとなって、ヒューマンエラーをなくすることができます。そして、このエラーは下記  
7 のように明示的にキャストすることによって、消すことができます。

```
#include <iostream>
using namespace std;

int main()
{
    int inum = 100000;           //intは32bitは符号付きの整数型。
                                //表現できる範囲は、-2,147,483,648 ~ 2,147,483,647
    short snum;                 //shortは16bitの符号付きの整数型。
                                //表現できる値の範囲は、-32,768 ~ 32,767
    snum = (short)inum;         //snumに100000という値は表現できないので、代入すると・・・
    cout << snum << '¥n';      //値が失われる！
    return 0;
}
```

- 8  
9 つまり、キャストというのは「小さな型変換で、値が失われるのは知っているけど、これは  
10 正しいコードだから、黙ってコンパイルしろ！」ということをコンパイラに教えてやる行為  
11 となります。

## 12 13 Lesson 4 中間テスト 4

- 14 下記の URL のテストを行いなさい。

15 [https://docs.google.com/forms/d/e/1FAIpQLScNMpAM-h4\\_Bek28g8\\_-tVNiXn8Rj-](https://docs.google.com/forms/d/e/1FAIpQLScNMpAM-h4_Bek28g8_-tVNiXn8Rj-bos_NAHbE0QlagOpyTA/viewform?usp=sf_link)  
16 [bos\\_NAHbE0QlagOpyTA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLScNMpAM-h4_Bek28g8_-tVNiXn8Rj-bos_NAHbE0QlagOpyTA/viewform?usp=sf_link)

- 17  
18

## 異なる型どうして演算する(P.98)

この説明もちょっと正しくありません。例えば下記のコードの場合、結果は小さい型の浮動小数点となります。

```
#include <iostream>
using namespace std;

int main()
{
    long long llValue = 10; //long longは64bitの整数型。
    float fValue = 0.5f;    //floatは32bitの浮動小数点型。
    long long llResult = llValue * fValue; //教科書の説明なら64bitの整数型になるはずが、
                                         //32bitの小数点型になっている!!!

    return 0;
}
```

型変換のルールは下記です。

- ・ どちらか一方のオペランドが浮動小数点なら、結果は浮動小数点になる。
- ・ オペランドが両方とも整数型 or 浮動小数点型なら、型サイズの大きい方になる。

## Lesson 4 章末テスト

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSeDBBADoB0WS-1l8sT8foEhJoobCE0-\\_bRu\\_UVdXioWAZqWgA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSeDBBADoB0WS-1l8sT8foEhJoobCE0-_bRu_UVdXioWAZqWgA/viewform?usp=sf_link)

## Lesson 5 場合に応じた処理

### 5.1 関係演算子と条件

#### 条件の仕組みをみる(p.106)

例えばゲームであれば、「もしも、コントローラーの A ボタンが押されたら」  
→ジャンプする

#### 条件を記述する(p.107)

C++では==が数学の=と同じになる。左辺と右辺が等しければ true を返す演算子。  
関係演算子は、条件が成立する時に true、成立しなければ false を返してくる演算子です。

### 5.2 if 文

#### if 文の仕組みを知る(p.111)

新しいプロジェクト Lesson5 を作成して、Sample1.cpp の内容を入力して、実行してください。

#### if 文で複数の分を処理する(p.113)

Lesson5 を改造して、Sample2.cpp のコードを入力して、実行してください。

#### ブロックにしないと(p.116)

if 文を使うときは、条件を満たすときの処理が、たとえ 1 文であったとしても、必ず{}で囲むのをお勧めします。ただし、{}で囲んでいないコードを読むこともあるので、知識として知っておくことは重要です。

### 5.2 実習

下記の動画のようなプログラムを実装しなさい。

<https://www.youtube.com/watch?v=EM0hp-jT15I&feature=youtu.be>

### 5.3 if~else 文

Lesson5 を改造して、Sample3.cpp の内容を入力して、実行してください。

### 5.3 実習

下記の動画のようなプログラムを実装しなさい。

<https://www.youtube.com/watch?v=qEYbe7RDEkg&feature=youtu.be>

## 5.4 if~else if~else

### if~else if~else の仕組みを知る (p.122)

Lesson5 を改造して、Sample4.cpp を入力して実行してください。

## 5.4 実習

下記の仕様を満たすプログラムを実装しなさい。

- ・年齢の入力を促す。
- ・20 歳未満なら、「未成年ですね」と表示する。
- ・20 歳以上なら「成人ですね」と表示する。
- ・ただし、下記の年齢の場合は長寿の祝いを表示する。

60 歳の場合は「還暦おめでとうございます。」

77 歳の場合は「喜寿おめでとうございます。」

88 歳の場合は「米寿おめでとうございます。」

99 歳の場合は「白寿おめでとうございます。」

下記の動画を参考にして実装しなさい。

<https://www.youtube.com/watch?v=JmguNYN1aaU&feature=youtu.be>

## 5.ex\_1

if 文は bool 型の true、false の値によって分岐します。そのため、下記のようなコードも合法です。

```
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "年齢を入力してください。¥n";
    cin >> age;

    bool result = age >= 20; //>=のような関係演算子はbool型の値を返す。

    if (result) { //if文は単にbool型の値によって分岐するだけなので、これもOK。
        cout << "あなたは成人ですね。¥n";
    }

    return 0;
}
```

- 1 また、整数の 0 は false に 0 以外は true に暗黙的に変換されるため、下記のようなコードも  
2 合法です。

```
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "年齢を入力してください。¥n";
    cin >> age;

    //int型の変数ageをif文にそのまま書く。
    //整数型は暗黙的に、bool型に変換される。
    if (age) {
        cout << "あなたは0歳ではないですね。¥n";
    }
    else {
        cout << "あなたは0歳ですね。¥n";
    }
    return 0;
}
```

- 3  
4 ! (否定演算子)を使うと、bool 型の結果が反転します。

```
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "年齢を入力してください。¥n";
    cin >> age;

    bool result = age >= 20; //>=のような関係演算子はbool型の値を返す。

    //resultの結果を反転しているので、このif文の条件が成立するということは、
    //未成年だということになる。
    if (!result) {
        cout << "あなたは未成年ですね。¥n";
    }
    return 0;
}
```

- 5 下記のようにも書ける。

```
#include <iostream>
using namespace std;

int main()
{
    int age;
    cout << "年齢を入力してください。¥n";
    cin >> age;

    //age>=20の結果を反転しているので、このif文の条件が成立するということは、
    //未成年だということになる。
    if (!(age >= 20)) {
        cout << "あなたは未成年ですね。¥n";
    }
    return 0;
}
```

- 6



## Lesson 5 中間テスト 1

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSfu0l2a0RE\\_FFheu8jhl6mmcJVzDEyxFF6kIh-vOvKfVe5RE6w/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSfu0l2a0RE_FFheu8jhl6mmcJVzDEyxFF6kIh-vOvKfVe5RE6w/viewform?usp=sf_link)

### 5.5 switch 文

#### switch 文の仕組みを知る (p.126)

switch 文で出来ることは、if～else if ～else 文でもできるので、どちらを使っても OK。

Lesson5 を改造して、Sample5.cpp を入力して実行してみてください。

#### break 文が抜けていると (p.129)

break 文がないと下のケースが実行されます。このようなことを意図して行うコードもありますが、慣れないうちは必ず break を書くようにしましょう。

### 5.6 論理演算子

#### 論理演算子の仕組みを知る (p.131)

(条件A) && (条件B)

→条件Aが true かつ 条件Bが true

(条件A) || (条件B)

→条件Aが true または 条件Bが true

では、&&(論理積)を使えるケースを見てみましょう。

例) 20 歳以上かつ男性かどうかを判断する場合の条件式(論理演算を使わない場合)

```
#include <iostream>

using namespace std;

int main()
{
    int age, gender;
    cout << "年齢を入力してください。¥n";
    cin >> age;
    cout << "性別を入力してください。0 : 男性、1 : 女性¥n";
    cin >> gender;

    //&&はif文のネストでも表現できるが・・・
    if (age >= 20) {
        if (gender == 0) {
            cout << "あなたは20歳以上で、男性ですね。¥n";
        }
    }

    return 0;
}
```

1 例) 20 歳以上かつ男性かどうかを判断する場合の条件式(論理演算を使う場合)

```
#include <iostream>

using namespace std;

int main()
{
    int age, gender;
    cout << "年齢を入力してください。¥n";
    cin >> age;
    cout << "性別を入力してください。0 : 男性、1 : 女性¥n";
    cin >> gender;

    //&&を使って、条件をまとめることができる。
    if (age >= 20 && gender == 0) {
        cout << "あなたは20歳以上で、男性ですね。¥n";
    }

    return 0;
}
```

2

3 続いて、||(論理和)を使えるケースを見てみましょう。

4 例)飲食店で女性と子供は 2 割引のサービスを行っている場合の条件式(論理演算を使わない  
5 場合。)

```
#include <iostream>

using namespace std;

int main()
{
    //例えば、飲食店で女性と子供は2割引のサービスを行っている場合。
    int age, gender;
    cout << "年齢を入力してください。¥n";
    cin >> age;
    cout << "性別を入力してください。0 : 男性、1 : 女性¥n";
    cin >> gender;

    //||は複数のif文でも表現できるが・・・
    if (age < 13) {
        cout << "あなたは2割引きのサービスを受けられます。¥n";
    }
    if (gender == 1) {
        cout << "あなたは2割引きのサービスを受けられます。¥n";
    }

    return 0;
}
```

6

7

8

9

10

11

12

13

例)飲食店で女性と子供は 2 割引のサービスを行っている場合の条件式(論理演算を使う場合。)

```
#include <iostream>

using namespace std;

int main()
{
    //例えば、飲食店で女性と子供は2割引のサービスを行っている場合。
    int age, gender;
    cout << "年齢を入力してください。¥n";
    cin >> age;
    cout << "性別を入力してください。0 : 男性、1 : 女性¥n";
    cin >> gender;

    ///||を使って、条件をまとめる。
    if (age < 13 || gender == 1) {
        cout << "あなたは2割引のサービスを受けられます。¥n";
    }

    return 0;
}
```

更に論理演算を組みあわせると、じゃんけんの勝敗判定も行えます。

```
//playerという変数にプレイヤーの手、comという変数にコンピュータの手の情報が入っている。
//0がグー、1がチョキ、2がパーです。
if (( player == 0 && com == 1 )      //プレイヤーがグー、コンピュータがチョキ。
    || ( player == 1 && com == 2 )    //プレイヤーがチョキ、コンピュータがパー。
    || ( player == 2 && com == 0 )    //プレイヤーがパー、コンピュータがグー。
) {
    cout << "あなたの勝ちです。¥n";
}
```

## Lesson 5 章末テスト

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLScWLHDqPBaJc3ISuLyL13M93NhYJJwO010PPaCld7PK5EUltw/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLScWLHDqPBaJc3ISuLyL13M93NhYJJwO010PPaCld7PK5EUltw/viewform?usp=sf_link)

## Lesson 6 何度も繰り返す

### Lesson6 を始める前に

Lesson 6 に記載されているサンプルコードですが、今後の勉強に悪い影響を与えかねない要素があるので、教科書のサンプルコードは打ち込まずに、PDF ファイルのサンプルコードを打ち込んでください。この理由は Lesson9 で配列を勉強する時に説明します。また、ループの繰り返しの処理は、for 文、while 文、do~while 文の 3 つがありますが、for 文だけ覚えてもらえば OK です。while 文、do~while 文で出来ることは、すべて for 文で実現可能なので、プログラムに慣れてきたときに覚えてもらえれば十分です。ただし、国家試験、検定にはすべて出てくる可能性があるなので、国家試験を取りたい人は、すべて覚える必要があります。

### 6.1 for 文

#### for 分の仕組みを知る (p.142)

VisualStudio で Lesson\_6 のプロジェクトを作成して、Sample1.cpp を入力してください。

for 文は一行に 3 つの処理が記述されていることに注意！

for( int i = 0 ; i < 5; i++ )

ループ変数 i の宣言と初期化

ループの継続条件

ループブロックの処理を  
一度、実行した後で行わ  
れる処理。

#### Sample1.cpp

```
#include <iostream>
using namespace std;

int main()
{
    for (int i = 0; i < 5; i++) {
        cout << "繰り返しています。¥n";
    }
    cout << "繰り返しが終わりました。¥n";
    return 0;
}
```

- 1 もう少し、見ていきましょう。下記のコードを見てください。  
2 Sample1.cpp は下記のようなコードと同義です。

```
#include <iostream>
using namespace std;

int main()
{
    int i = 0; //変数iの宣言と初期化。
LOOP: //これはラベルと呼ばれるもの。
    if (i < 5) { //ループの継続判定。
        cout << "繰り返しています。¥n";
        i++; //ループ変数のインクリメント。
        goto LOOP; //goto文でラベルLOOPにジャンプする。
    }
    cout << "繰り返しが終わりました。¥n";
    return 0;
}
```

- 3 注意！今回 for 文の説明のために goto 文を使用しましたが、goto 文はスパゲッティコード(読  
4 みづらいコード)を生み出しやすくするものとして、多くの開発で使用することが非推奨とな  
5 っている構文です。絶対に使用しないように！！

- 6  
7 for 文の 3 つの式の意味が分かれば、下記のようなコードが書けることが分かります。  
8 下記のコードを入力して、F10 キーでステップ実行を行い、処理の流れを確認してくださ  
9 い。

```
#include <iostream>
using namespace std;

int main()
{
    //ループ変数は0以外で初期化することもできるし、
    //ループの継続判定も i>=0 などにもできる。
    //ループブロックの処理が終わった後の処理もデクリメントでも良い。
    for (int i = 5; i >= 0; i--) {
        //ループ変数はループ内で使用できる。
        cout << i << "¥n";
    }
    return 0;
}
```

- 10  
11 このコードを goto 文を使って書くと下記のようになります。

```
#include <iostream>
using namespace std;

int main()
{
    int i = 5; //変数iの宣言と初期化。
LOOP: //これはラベルと呼ばれるもの。
    if (i >= 0) { //ループの継続判定。
        cout << i << "¥n";
        i--; //ループ変数のインクリメント。
        goto LOOP; //goto文でラベルLOOPにジャンプする。
    }
    return 0;
}
```

## 変数をループ内で使う (p.145)

ここまでの例で見てきたように、ループ変数をループ内で使うことができます。Lesson\_6 に Sample2.cpp を入力して実行してください。

## for 文を応用する (p.146)

Lesson\_6 に Sample3.cpp と Sample4.cpp を入力して、動作を確認してください。

Sample3.cpp (入力した数だけ\*を表示する。)

```
#include <iostream>
using namespace std;

int main()
{
    int num;
    cout << "いくつ*を表示しますか?¥n";
    cin >> num; //数を入力させる。

    for (int i = 0; i < num; i++) {
        cout << "*"; //入力した数だけ*を繰り返し表示する。
    }

    cout << "¥n";
    return 0;
}
```

ループの終了判定に変数を使  
えることに注目！

Sample4.cpp (入力した数までの合計を求める。)

```
#include <iostream>
using namespace std;

int main()
{
    int kazu;
    int goukei = 0;

    cout << "いくつまでの合計を求めますか?¥n";
    cin >> kazu; //数を入力させる。

    for (int i = 1; i <= kazu; i++) {
        goukei += i; //例えば、kazuに5が入力されたら、iの値は1～5となる。
        //なので、iの値を加算していくと1～5までの数値の合計になる。
    }

    cout << "1 から" << kazu << "までの合計値は" << goukei << "です。¥n";
    return 0;
}
```

## Lesson 6 中間テスト1

下記のURLのテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLScAPHvnMY3l2-](https://docs.google.com/forms/d/e/1FAIpQLScAPHvnMY3l2-PuMblKSBgfP0XIBRcexBqBHIzURobXUUCUFA/viewform?usp=sf_link)

[PuMblKSBgfP0XIBRcexBqBHIzURobXUUCUFA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLScAPHvnMY3l2-PuMblKSBgfP0XIBRcexBqBHIzURobXUUCUFA/viewform?usp=sf_link)

## 6.2 while 文

### while 文の仕組みを知る (p.149)

while 文は、for 文を goto 文に置き換えて記述したものとよく似ています。

Sample5.cpp を入力して実行出来たら、F10 を押して、ステップ実行をおこなって動作を確認しなさい。

#### Sample5.cpp

```
#include <iostream>
using namespace std;

int main()
{
    int i = 0;          //これがループ変数。
    while (i < 5) {      //これがループの継続条件。
        cout << i << "番目の繰り返しです。¥n";
        i++;            //これがループの処理が終わったときの処理。
    }
    cout << "繰り返しが終わりました。¥n";
    return 0;
}
```

### 条件の記述を省略する (p.151)

p.152 のようなコードを書くときは for ではなく、while が使われることが多いです(私もたぶん while を使います)。while が使われる理由は、このようなコードを書くときは、for 文に比べて、キータイプの量が減ることと(キータイプ量が減ると腱鞘炎に苦しむプログラマーが減りますよね?)、while を覚えている人からすると、読みやすいコードになるからです。

では、Sample6.cpp を記入して、動作を確認してください。

#### Sample6.cpp

```
#include <iostream>
using namespace std;

int main()
{
    int num = 1;
    while (num == 1) { //numが1の間は処理を繰り返す。
        cout << "整数を入力してください。(0で終了)¥n";
        cin >> num;
        cout << num << "が入力されました。¥n";
    }
    cout << "繰り返しが終わりました。¥n";
    return 0;
}
```

### 6.3 do～while 文

たぶん学生のうちに使うことはないと思います。do～while 文は下記の一点だけを覚えておきましょう。

**ループの中の処理が必ず最低一回は行われる。**

### 6.4 文のネスト

#### 文をネストする(p.157)

Sample8.cpp を入力して動作を確認したら、F10 を押してステップ実行で処理の流れを確認しなさい。

Sample8.cpp

```
#include <iostream>
using namespace std;

int main()
{
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 3; j++) {
            cout << "iは" << i << ", jは" << j << "¥n";
        }
    }
    return 0;
}
```

ループ変数が i と j で違うことに注意！

#### 6.4 ex デバッガの機能のウォッチを使ってみよう。

プログラムを書いていくうえで、強力な武器となるデバッガの機能のウォッチ(変数の値を見るため)を使って、6.4 のループ変数の i と j の中身を見てみましょう。ウォッチの使い方は下記の動画を参考にしてみてください。

<https://www.youtube.com/watch?v=dAwYii65J40&feature=youtu.be>



1 if 文などと組み合わせる('p.159)

2 Sample9.cpp を入力して動作を確認してください。動作が確認出来たら、下記の動画のよ  
3 うに、ステップ実行を行い、変数 i、j、ch をウォッチに追加して、処理の流れを追いかける  
4 さい。

5 **Sample9.cpp**

```
#include <iostream>
using namespace std;

int main()
{
    int ch = 0;
    for (int i = 0; i < 5; i++) {
        for (int j = 0; j < 5; j++) {
            if (ch == 0) {
                cout << '*' ;    /**を出力したら、次は-を表示
                                //するように、chに1を代入。
                ch = 1;
            }
            else {
                cout << '-' ;    /**を出力したら、次は*を表示
                                //するように、chに0を代入。
                ch = 0;
            }
        }
        cout << "\n";          //内側のループが終わったら改行します。
    }
    return 0;
}
```

6  
7 動画(音声が入っているので再生する時は注意してください。)

8 <https://www.youtube.com/watch?v=MPCf2NUSCUA&feature=youtu.be>

9  
10 Lesson 6 中間テスト2

11 下記の URL のテストを行いなさい。

12 <https://docs.google.com/forms/d/e/1FAIpQLSfPy->

13 [snsI\\_VVDKWYV18xdYtVq171je48\\_McWFPbH9udXML3sQ/viewform?usp=sf\\_link](snsI_VVDKWYV18xdYtVq171je48_McWFPbH9udXML3sQ/viewform?usp=sf_link)

## 6.5 処理の流れの変更

### break 文の仕組みを知る (p.161)

Sample10.cpp を入力して、動作を確認したら、F10 キーでステップ実行して、処理の流れを確認しなさい。

Sample10.cpp

```
#include <iostream>
using namespace std;

int main()
{
    int res;
    cout << "何番目でループを注視しますか？ (1~10) \n";
    for (int i = 0; i < 10; i++) {
        cout << i << "番目の処理です。 \n";
        if (i == res) {
            break; //指定した回数で繰り返しを終了します。
        }
    }
}
```

break 文は、例えば下記のような仕様を実装する時などに使えます。RPG の製作を行っていて、一緒に戦う仲間の AI を作っている場合を考えて下さい。「ヒーラーの AI であれば、HP が 500 以下になった味方にヒーリングを行う。」といった仕様を実装することがあるかもしれません。そのような場合に、下記のコードのように味方から HP 500 以下のキャラクターを検索する必要があります。

```
for (int i = 0; i < mikataCharacterNum; i++) {
    if (mikataCharacterHP[i] < 500) {
        //500以下のキャラクターを発見したので、ヒールをかけて
        //ループを抜ける。
        .
        .
        省略
        .
        .
        break;
    }
}
```

- 1 繰り返しをネストしている場合、その内側の文で **break** 文を使うと、外側のブロックに処理  
2 が移る(p.162)

- 3 これは下記のようなコードの話です。入力して動作を確認したのち、F10 でステップ実行  
4 を行い、処理の流れを確認してください。

```
#include <iostream>
using namespace std;

int main()
{
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 5; j++) {
            if (j == 3) {
                break;
            }
            //jは2までしか出力されない!
            cout << "i : " << i << ", j : " << j << "¥n";
        }
    }
    return 0;
}
```

内側のループを  
抜ける。

- 5  
6 **continue** 分の仕組みを知る(p.165)  
7 Sample12.cpp の内容を入力して動作を確認してください。

```
#include <iostream>
using namespace std;

int main()
{
    int res;
    cout << "何番目の処理を飛ばしますか？(0~9)¥n";
    cin >> res;

    for (int i = 0; i < 10; i++) {
        if (i == res) {
            continue;
        }
        cout << i << "番目の処理です。¥n";
    }
    return 0;
}
```

8  
9  
10  
11  
12  
13  
14  
15  
16

1 continue 文は下記のような仕様を実装したい場合に使えます。

2 「1000 人のグループの名簿から 20 歳の人にだけ、成人おめでとうと表示する。」

3 この仕様を continue 文を使って、実装すると下記ようになります。

```
for (int i = 0; i < 1000; i++) {  
    if (ageList[i] != 20) {  
        //20歳以外はスキップ。  
        continue;  
    }  
    cout << "成人おめでとう。¥n";  
}
```

4  
5 ただし、このコードは下記のように書くこともできます。

```
for (int i = 0; i < 1000; i++) {  
    if (ageList[i] == 20) {  
        cout << "成人おめでとう。¥n";  
    }  
}
```

6 このように、continue 文を使用しなくても、ほかの方法があるため、今無理して覚える必要  
7 はないと思います。

8  
9 Lesson6 章末テスト

10 下記の URL のテストを行いなさい。

11 [https://docs.google.com/forms/d/e/1FAIpQLSe9Qi4I82H6LzhIYxmw-wPiGth-](https://docs.google.com/forms/d/e/1FAIpQLSe9Qi4I82H6LzhIYxmw-wPiGth-ihf3PCZ0zmdcEPjd4MANgg/viewform?usp=sf_link)  
12 [ihf3PCZ0zmdcEPjd4MANgg/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSe9Qi4I82H6LzhIYxmw-wPiGth-ihf3PCZ0zmdcEPjd4MANgg/viewform?usp=sf_link)

## Lesson 9 配列

### 9.1 配列の基本

#### 配列の仕組みを知る (p.256)

「ただし、こうしてたくさんの変数が登場すると、コードが複雑で読みにくくなってしまう場合があります。(p.256)」

→読みやすいコードを書くと言ことは、ソフトウェアを開発していくうえで、とても重要要素となります。今のうちから意識してコードを書いていきましょう。

例えば、あなたがMMORPGの100人参加可能な、レイドバトルのボスキャラクターのAIを実装している場合を考えてみて下さい。そのボスキャラクターのAIは「HPが一番低いプレイヤーに対して攻撃を行う。」という仕様だったとします。配列を知らない不幸なプログラマー(プロのプログラマーに、そんな人はいないと思いますが)は、悪夢のようなコードを書いてしまうでしょう。

```
//プレイヤー構造体。
struct Player {
    int hp;
    int attack;
    .
    .
    省略
    .
};

//プレイヤー100人を表すグローバル変数を100個用意する。
Player player00;
Player player01;
Player player02;
Player player03;
Player player04;
Player player05;
Player player06;
Player player07;
Player player08;
Player player09;
Player player10;
.
.
省略
.
Player player98;
Player player99;

//攻撃対象となるプレイヤーを検索する関数。
Player* FindAttackTargetPlayer()
{
    Player* attackTargetPlayer = nullptr;

    //一番HPが小さい奴を探す。
    int hp = 9999999; //とりあえずあり得ないくらい大きくしておく。40億超えたらダメだよ。
    //プレイヤー00
    if (player00.hp < hp) {
        //プレイヤー00の方がHPが小さいので更新。
    }
}
```

```

        hp = player00.hp;
        attackTargetPlayer = &player00;
    }
    //プレイヤー01
    if (player01.hp < hp) {
        //プレイヤー01の方がHPが小さいので更新。
        hp = player01.hp;
        attackTargetPlayer = &player01;
    }
    //プレイヤー02
    if (player02.hp < hp) {
        //プレイヤー02の方がHPが小さいので更新。
        hp = player02.hp;
        attackTargetPlayer = &player02;
    }
    //プレイヤー03
    if (player03.hp < hp) {
        //プレイヤー03の方がHPが小さいので更新。
        hp = player03.hp;
        attackTargetPlayer = &player03;
    }
    //プレイヤー04
    if (player04.hp < hp) {
        //プレイヤー04の方がHPが小さいので更新。
        hp = player04.hp;
        attackTargetPlayer = &player04;
    }
    //プレイヤー05
    if (player05.hp < hp) {
        //プレイヤー05の方がHPが小さいので更新。
        hp = player05.hp;
        attackTargetPlayer = &player05;
    }
    //プレイヤー06
    if (player06.hp < hp) {
        //プレイヤー06の方がHPが小さいので更新。
        hp = player06.hp;
        attackTargetPlayer = &player06;
    }
    //プレイヤー07
    if (player07.hp < hp) {
        //プレイヤー07の方がHPが小さいので更新。
        hp = player07.hp;
        attackTargetPlayer = &player07;
    }
    .
    .
    省略
    .
    .
    //プレイヤー98
    if (player98.hp < hp) {
        //プレイヤー98の方がHPが小さいので更新。
        hp = player98.hp;
        attackTargetPlayer = &player98;
    }
    //プレイヤー99
    if (player99.hp < hp) {
        //プレイヤー99の方がHPが小さいので更新。
        hp = player99.hp;
        attackTargetPlayer = &player98;
    }
    //見つかった攻撃対象のプレイヤーを返す。
    return attackTargetPlayer;
}

```

1 もし、このプログラムを配列を知っていた幸福なプログラマーが書けばこうなるでしょう。

```
//プレイヤー構造体。
struct Player {
    int hp;
    int attack;
    .
    .
    省略
    .
    .
};

//プレイヤー100人を表すグローバル変数を要素数100の配列として個用意する。
Player player[100];

//攻撃対象となるプレイヤーを検索する関数。
Player* FindAttackTargetPlayer()
{
    Player* attackTargetPlayer = nullptr;

    //一番HPが小さい奴を探す。
    int hp = 9999999; //とりあえずあり得ないくらい大きくしておく。40億超えたらダメだよ。
    for (int i = 0; i < 100; i++) {
        if (player[i].hp < hp) {
            //このプレイヤーの方がHPが小さいので更新する。
            hp = player[i].hp;
            attackTargetPlayer = &player[i];
        }
    }
    //見つかった攻撃対象のプレイヤーを返す。
    return attackTargetPlayer;
}
```

2 とても短くシンプルで分かりやすいプログラムになりました。

3

## 4 9.2 配列の宣言

5 配列の添え字は0からはじまるので、最後の添え字は「要素数-1」となる。

6

## 7 9.3 配列の利用

8 新しいプロジェクト、Lesson09 を作成して Sample1.cpp を入力してください。

## 9 配列を初期化する(p.262)

10

## 11 Lesson\_09 実習課題\_1

12 下記の URL から zip ファイルをダウンロードして、次の1~4の実習を行いなさい。

13 <https://1drv.ms/u/s!AnpsoGtakIkVi41brqVH5jmOnCTrVQ>

## 14 課題 1

15 サンプルプログラムの kadai\_09\_01 を改造して、試験の合計点を求めて表示するプログ  
16 ラムを作成しなさい。

17

18

## 課題 2

サンプルプログラムの kadai\_09\_02 を改造して、試験の平均点を求めて表示するプログラムを作成しなさい。

## 課題 3

サンプルプログラムの kadai\_09\_03 を改造して、試験の平均点以上の点数を取っている学生の人数を表示するプログラムを作成しなさい。

## 課題 4

サンプルプログラムの kadai\_09\_04 を改造して、試験の点数を昇順（小さい順）に表示できるようにしなさい。

### 課題 4 のヒント動画

ヒント 1 値の交換のアルゴリズムの解説と、ソースコードの解説

<https://www.youtube.com/watch?v=Y-nZ8dYaY-A&feature=youtu.be>

ヒント 2 バブルソートアルゴリズムの解説

<https://www.youtube.com/watch?v=xvbUuWzBOHY&feature=youtu.be>

ヒント 3 バブルソートのソースコード解説

<https://www.youtube.com/watch?v=gOmWRxBH8WM&feature=youtu.be>

## Lesson 9 中間テスト 1

下記の URL のテストを行いなさい。

<https://docs.google.com/forms/d/e/1FAIpQLSe->

[NfkV14Qi6lv8wgo6qYVaa8ODC3LK89Qy35QdTtaVa\\_7juCA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSe-NfkV14Qi6lv8wgo6qYVaa8ODC3LK89Qy35QdTtaVa_7juCA/viewform?usp=sf_link)



## Lesson 7 配列

### 7.1 関数の仕組みを知る

「C++でも、複雑なコードを書くようになってくると、たびたび行わなければならない一定の処理が出てくる場合があります。(P.170)」

例えば、これまでいくつかの関数をみなさんは使ってきています。

- ・コンソールに文字列を表示する処理

printf 関数、cout 関数(実はこれも関数。ただしちょっと特殊な書き方をしている所以需要。)

- ・ユーザーからの入力を受け付ける関数。

scanf 関数、cin 関数(cout と同様に、これも関数。)

- ・キーボードの入力判定

GetAsyncKeyState 関数。これは microsoft が提供している関数です。

- ・パックマン、落ち物などの文字表示関数。

kbcDrawMoji 関数。この関数は私が作成した関数です。

もしこの関数という機能がなかったら、あなたはコンソールに文字を出力する時に下記のようなコードを毎回コピペで複製する羽目になります！！

```
int main()
{
    //Hello worldと表示する！
    char* fmt = "Hello world\n";
    for (;;) {

        unsigned long long ui;
        long long i;
        char *s = NULL;
        double d = 0.0;
        int sign = 0;

        int flags = 0;
        int length = 0;
        int precision = 0;
        int tmp = 0;
        INTEGER int_type = 0;

        while (*fmt && *fmt != '%') __putc(*fmt++);
        if (*fmt == '%0') { va_end(ap); break; }
        fmt++;

        while (mystrchr("+-#0", *fmt)) {
            switch (*fmt++) {
                case '¥': flags |= THOUSAND_GROUP; break;
                case '-': flags |= LEFT_JUSTIFIED; break;
                case '+': flags |= WITH_SIGN_CHAR; sign = '+'; break;
                case '#': flags |= ALTERNATIVE; break;
                case '0': flags |= ZERO_PADDING; break;
                case ' ': flags |= WITH_SIGN_CHAR; sign = ' '; break;
            }
        }
    }
}
```

```

if (*fmt == '*') { length = va_arg(ap, int); fmt++; }
else { while (_isnumc(*fmt)) length = (length * 10) + _ctoi(*fmt++); }

if (*fmt == '.')
{
    fmt++;
    if (*fmt == '*') { fmt++; precision = va_arg(ap, int); }
    else { while (_isnumc(*fmt)) precision = precision * 10 + _ctoi(*fmt++); }
}

while (mystrchr("hljzt", *fmt)) {
    switch (*fmt++) {
        case 'h': int_type--; break;
        case 'l': int_type++; break;
        case 'j': /*intmax : long */
        case 'z': /*size : long */
        case 't': /*ptrdiff : long */
            int_type = L; break;
    }
}

switch (*fmt) {

case 'd':
case 'i':
    i = get_signed(ap, int_type);
    if (i < 0) { i = -i; sign = '-'; }
    put_integer(__putc, i, 10, length, sign, flags);
    break;

case 'u':
    ui = get_unsigned(ap, int_type);
    put_integer(__putc, ui, 10, length, sign, flags);
    break;

case 'o':
    ui = get_unsigned(ap, int_type);
    put_integer(__putc, ui, 8, length, sign, flags);
    break;

case 'p':
    length = sizeof(long) * 2;
    int_type = L;
    sign = 0;
    flags = ZERO_PADDING | ALTERNATIVE;
case 'X':
    flags |= CAPITAL_LETTER;
case 'x':
    ui = get_unsigned(ap, int_type);
    put_integer(__putc, ui, 16, length, sign, flags);
    break;

case 'c':
    i = get_signed(ap, C);
    __putc(i);
    break;

case 's':
    s = va_arg(ap, char *);
    if (s == NULL) s = "(null)";
    tmp = strlen(s);
    if (precision && precision < tmp) tmp = precision;

```

```

        length = length - tmp;
        if (!(flags & LEFT_JUSTIFIED))
        {
            while (length > 0) { length--; __putc(' '); }
        }
        while (tmp-->) { __putc(*s++); }
        while (length-- > 0) { __putc(' '); }

        break;

    case '%':
        __putc('%');
        break;

    default:
        while (*fmt != '%') fmt--;
        break;
    }

    fmt++;
}
return 0;
}

```

1

2 このプログラムは Hello world という文字列を表示するためだけに 127 行ものプログラム  
3 を書いています。これが一度だけというならまだしも、なにか文字列を表示するたびに、あ  
4なたはこれをコピーアンドペーストする必要があるのです！

5 しかし、関数を利用すると、この 127 行ものプログラムをまとめることができ、下記  
6のように簡単に再利用することができます。

```

int main()
{
    //HelloWorldと表示。
    printf("Hello world\n");
    //こんにちは世界と表示。
    printf("こんにちは世界\n");
    return 0;
}

```

7

## 8 7.2 関数の定義と呼び出し

### 9 関数を定義する

10 関数を定義する＝関数の処理を実装する＝関数の本体といった意味合いになります。

11 後々関数宣言というものが出てきますが、それと混合しないように注意してください。

12 関数定義は関数の本体、プログラムの本体といった意味です。

13 関数は下記のように定義します。

14

15

16

# 戻り値 関数名( 引数リスト )

{

}

波カッコの間が関数の処理!!!

戻り値と引数リストは 7.3 と 7.4 で説明します。今は関数名だけ覚えてください。

## 関数呼び出し

新しいプロジェクト、Lesson7 を作成して Sample1.cpp を入力してください。

関数を呼び出すには下記ように書く必要があることに注意してください。

# 関数名();

カッコが必要なことを忘れてしまう学生が多いです。()を忘れないようにしてください。

Sample1.cpp が入力出来たら、VisualStudio のステップイン(F11)を使って、処理の流れを追いかけてみてください。

## 参考動画

<https://www.youtube.com/watch?v=vljzUcT3XiU&feature=youtu.be>

## 関数を何度も呼び出す

Sample2.cpp を入力して下さい。

入力出来たら、ステップインを使って、処理の流れを追いかけてみてください。

## 7.2 Ex\_1 関数化を行う理由

### プログラムを綺麗に書くため

Lesson7 では関数化を行う理由として、「プログラムの再利用性」を挙げています。これはもちろん正しいのですが、再利用を行うことができるプログラムというのは、それほど多くありません。そのため、それが最大の理由だと考えてしまうと、関数化を行うことが難しくなってきます。しかし、多くのプロジェクトでは、一か所でしか呼ばれない関数というのが沢山あります。つまり、**再利用されていない関数**が沢山あるのです。しかも、それらの関数は、再利用されている関数よりも多いのです！

では、なぜ関数化するのでしょうか？答えは、プログラムを見やすく、綺麗に描くためです。下記のプログラムは制作演習で行っている落ち物ゲームのゲームループです。

### 関数化を行っていない落ち物ゲームのゲームループ

```
//ここからゲームループ。
while (true) {
    //1フレームの開始。
    //フォントのサイズの設定。
    const HANDLE hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CONSOLE_FONT_INFOEX info = { sizeof info };
    info.dwFontSize.X = 32;
    info.dwFontSize.Y = 32;
    SetCurrentConsoleFontEx(hConsole, 0, &info);

    //コンソールの縦と横の行数を設定。
    HANDLE hCons = GetStdHandle(STD_OUTPUT_HANDLE);
    COORD dwSize = { 9999, 9999 };
    SetConsoleScreenBufferSize(
        hCons,          // HANDLE hConsoleOutput
        dwSize);        // COORD dwSiz

    //コンソールの高さを設定。
    SMALL_RECT consoleWindow = { 0, 0, 20, 20 };
    BOOL ret = SetConsoleWindowInfo(
        hCons, // HANDLE hConsoleOutput
        TRUE,  // BOOL bAbsolute
        &consoleWindow); // CONST SMALL_RECT *lpConsoleWindow

    //フレームバッファをクリア。
    memset(sFrameBuffer, 0, sizeof(sFrameBuffer));
    //背景を描画。
    //二次元配列のmapを参照して、背景を描画していく。
    for (int y = 0; y < 16; y++) {
        for (int x = 0; x < 16; x++) {
            if (map[y][x] == 0) {
                //二次元配列のmap[y][x]に0が入っていたら、
                //座標x, yの場所に空白文字を描画する。
                kbcDrawMoji(x, y, ' ');
            }
            else if (map[y][x] == 1) {
                //二次元配列のmap[y][x]に1が入っていたら、
                //座標x, yの場所に@を描画する。
                kbcDrawMoji(x, y, '@');
            }
        }
    }

    //プレイヤー1の移動処理。
    for (int playerNo = 0; playerNo < 2; playerNo++) {
```

```

if (playerNo == 0) {
    //1プレイヤー。
    //プレイヤー 1 の移動処理。
    if (GetAsyncKeyState(VK_LEFT) != 0) {
        //左のキーが押されたときの処理をここに記述する。
        player[playerNo].posX--;
    }
    if (GetAsyncKeyState(VK_RIGHT) != 0) {
        player[playerNo].posX++;
    }
}
else if (playerNo == 1) {
    //2プレイヤー。
    //続いてプレイヤー 2 の移動処理。
    if (GetAsyncKeyState('A') != 0) {
        //左のキーが押されたときの処理をここに記述する。
        player[playerNo].posX--;
    }
    if (GetAsyncKeyState('D') != 0) {
        player[playerNo].posX++;
    }
}
//壁判定。
if (player[playerNo].posX < 1) {
    player[playerNo].posX = 1;
}
if (player[playerNo].posX > 14) {
    player[playerNo].posX = 14;
}
//落ち物との衝突判定。
for (int i = 0; i < otimonoArray.size(); i++) {
    if (otimonoArray[i].posX == player[playerNo].posX
        && otimonoArray[i].posY == player[playerNo].posY
    ) {
        otimonoArray[i].dead = 1;
        //ゲット出来ていたらカウントアップ。
        player[playerNo].score++;
    }
}
//プレイヤー 1 を描画する。
if (player[playerNo].posX < FRAME_BUFFER_WIDTH && player[playerNo].posY <
FRAME_BUFFER_HEIGHT) {
    sFrameBuffer[player[playerNo].posY][player[playerNo].posX] = 'P';
}

//落ち物のプログラムを実行。
gameCount++;
if (gameCount == 10) {
    //30フレーム経過したら新しい落ち物を発生させる。
    Otimono otimono;
    //落ち物のX座標をランダムで決定。
    otimono.posX = (rand() % 14) + 1;
    otimono.posY = 0;
    //ゲームカウントを0に戻す。
    gameCount = 0;
    otimonoArray.push_back(otimono);
}
//全ての落ち物を落下させる。
for (auto& otimono : otimonoArray) {
    if ((gameCount % 5) == 0) {
        //5フレームに一度落下する。
        otimono.posY += 1;
    }
}

```

```

//リストから削除。
otimonoArray.erase(std::remove_if(otimonoArray.begin(), otimonoArray.end(), [&](auto& otimono)-
>bool {
    return otimono.posY == 18 || otimono.dead == 1;
}), otimonoArray.end());

//落ち物を描画。
for (auto& otimono : otimonoArray) {
    if (otimono.posX < FRAME_BUFFER_WIDTH && otimono.posX < FRAME_BUFFER_HEIGHT) {
        sFrameBuffer[otimono.posY][otimono.posX] = '*';
    }
}

//スコアを表示する。
char scoreText[256];
sprintf_s(scoreText, "Player1 スコア %d", player[0].score);
for (int i = 0; scoreText[i] != '\0'; i++) {
    kbcDrawMoji(0 + i, 18, scoreText[i]);
}
sprintf_s(scoreText, "Player2 スコア %d", player[1].score);
for (int i = 0; scoreText[i] != '\0'; i++) {
    kbcDrawMoji(0 + i, 19, scoreText[i]);
}

//1フレームの終了。
//フレームバッファの内容を画面に表示する。
for (int i = 0; i < FRAME_BUFFER_HEIGHT; i++) {
    for (int j = 0; j < FRAME_BUFFER_WIDTH; j++) {
        std::cout << sFrameBuffer[i][j];
    }
    std::cout << "\n";
}

//32ミリ秒眠る。
Sleep(16);
//標準出力コンソールのハンドルを取得。
hCons = GetStdHandle(STD_OUTPUT_HANDLE);
COORD pos;
pos.X = 0;
pos.Y = 0;
//標準コンソールのカーソルを0行目、0列に戻す。
SetConsoleCursorPosition(hCons, pos);
CONSOLE_CURSOR_INFO cci;
// CONSOLE_CURSOR_INFO構造体の現在の状態を取得する
GetConsoleCursorInfo(hCons, &cci);
// メンバ変数であるbVisibleがカーソルの表示・非表示を制御する変数なので、これをFALSEにする事でカーソルを非表示にできる
cci.bVisible = FALSE;
// 変更した構造体情報をコンソールWindowにセットする
SetConsoleCursorInfo(hCons, &cci);
}

```

1  
2  
3  
4  
5  
6  
7  
8  
9

いかがでしょうか。これが関数化を行っていない落ち物のゲームループのプログラムです。150 行ほどのプログラムとなっております。では、続いて、関数化を行っている落ち物のゲームループを見てみましょう。

## 1 関数化を行っている落ち物ゲームのゲームループ

```
//ここからゲームループ。
while (true) {
    //1フレームの開始。
    kbcBeginFrame();
    //背景を描画。
    DrawBackground();
    //プレイヤーの処理。
    for (int i = 0; i < 2; i++) {
        MovePlayer(player[i], i);
        //落ち物との衝突判定。
        CheckHitPlayerAndOtimono(player[i]);
        //プレイヤー 1 を描画する。
        DrawPlayer(player[i]);
    }
    //落ち物の描画処理。
    DrawOtimono();
    //落ち物の移動処理。
    MoveOtimono();
    //スコアを表示する。
    DrawScore(player);
    //1フレームの終了。
    kbcEndFrame();
}
```

2  
3 いかがでしょうか？関数化を行っている方だと、1フレームに何をしているのか？という  
4 処理の流れが一目霊山ではないでしょうか？この二つのプログラムは処理的には同じこと  
5 をしています。ですが、1フレームの処理の流れをどちらが把握しやすいか？と問われると  
6 後者だと思います。このプログラムで網掛けになっている関数は再利用性を考慮して関数  
7 化を行っているわけではありません。一か所にしか記述されていませんね。そして、驚くべ  
8 きことにすべての関数がそうです。

9 関数化の最も大きなメリットは、「意味のある処理に名前を付けて、分割することでコー  
10 ドが読みやすくなる」ということだということが分かります。関数の再利用性は棚上げにし  
11 て、まずは処理を分割するということを行っていきましょう。

12  
13 この二つのプログラムは下記の URL にアップされています。両方とも動作するので、興  
14 味がある方はどうぞ。

15 <https://1drv.ms/u/s!AnpsoGtakIkVi41lbigVIJ8xjmOKHw>

16

17

18

19

20

21

22

23



## 7.2 Ex\_2 関数は分かりやすい名前をつける。

関数化を行う最も大きな理由として、**読みやすい、理解しやすい**プログラムを書くというものがあるということを勉強しました。さて、関数の名前というのは自由に決めていいということを学びましたが、ではどんな名前でもつけてもいいのでしょうか？（まあ不幸なことに、それでもコンパイルは通るのですが）下記のプログラムを見てみてください。

```
//ここからゲームループ。
while (true) {
    //1フレームの開始。
    A();
    //背景を描画。
    B();
    //プレイヤーの処理。
    for (int i = 0; i < 2; i++) {
        C(player[i], i);
        //落ち物との衝突判定。
        D(player[i]);
        //プレイヤー 1 を描画する。
        E(player[i]);
    }
    //落ち物の描画処理。
    F();
    //落ち物の移動処理。
    G();
    //スコアを表示する。
    H(player);
    //1フレームの終了。
    I();
}
```

さて、いかがでしょうか？このプログラムは理解しやすいでしょうか？コメントがあるから、まだ何とかなる気がしますか？では、コメントをあまり書かない、ずぼらなプログラマーが書いたプログラムを見てみましょう。

```
//ここからゲームループ。
while (true) {
    A();
    B();
    for (int i = 0; i < 2; i++) {
        C(player[i], i);
        D(player[i]);
        E(player[i]);
    }
    F();
    G();
    H(player);
    I();
}
```

どうでしょうか？もはや、このプログラムは暗号文のようなものです。A();という関数呼び出しが、どんな処理をしているかさっぱり分かりません。では、しっかりと分かりやすい関数名を付けたプログラムを見てみましょう。

1

```
//ここからゲームループ。  
while (true) {  
    kbcBeginFrame();  
    DrawBackground();  
    for (int i = 0; i < 2; i++) {  
        MovePlayer(player[i], i);  
        CheckHitPlayerAndOtimono(player[i]);  
        DrawPlayer(player[i]);  
    }  
    DrawOtimono();  
    MoveOtimono();  
    DrawScore(player);  
    kbcEndFrame();  
}
```

2    こちらもコメントはありませんが、関数名を読めば、何となく何をしているのかが分かりま  
3    す。これが分かりやすい、綺麗なプログラムです。プログラマーは理系っぽいというイメー  
4    ジがあるかもしれませんが、実は多彩な能力が求められます。このように分かりやすい名前  
5    を考える力は、文系の人の方が得意だと思われます。

6

## 7    Lesson\_07 ハンズオン 1

8    サンプルプログラムの Lesson\_07\_01/Question を使用して、ハンズオン 1～3 を行いなさ  
9    い。

## 10   Lesson\_07 実習課題\_1

11    サンプルプログラムの Lesson\_07\_01/Question を使用して、実習 1～3 を行いなさい。

12

13    サンプルプログラムの URL

14    <https://1drv.ms/u/s!AnpsoGtakIkVi41mnfcGk3d6lgyIsw>

15

16    実習課題の解説動画

17    [https://www.youtube.com/watch?v=Tr0u\\_uXMYsk&feature=youtu.be](https://www.youtube.com/watch?v=Tr0u_uXMYsk&feature=youtu.be)

18

## 19   Lesson\_07 中間テスト 1

20    下記の URL のテストを行いなさい。

21    [https://docs.google.com/forms/d/e/1FAIpQLSfLoJBcmo9nAZqAPd6QBH6pfn6muV94rzzxCbIjAgGnY9cZloA/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSfLoJBcmo9nAZqAPd6QBH6pfn6muV94rzzxCbIjAgGnY9cZloA/viewform?usp=sf_link)

22

23

24

25

26

27

28

## 7.3 引数

### 引数を渡して関数を呼び出す(p.181)

Sample3.cpp を入力してください。

### キーボードから入力する(p.183)

Sample4.cpp を入力してください。

### 複数の引数をもつ関数を使う(p.185)

Sample5.cpp を入力して下さい。

## 7.3 Ex\_3 引数はローカル変数

関数の引数はローカル変数と呼ばれるものとなります。ローカル変数ってなに？って思うと思いますが、みなさんがこれまで学んできた変数がローカル変数です。ローカル変数は関数の中で定義されて、その関数の中でだけ有効な変数です(ちょっと嘘です。詳しくは10.1の範囲で勉強します)。

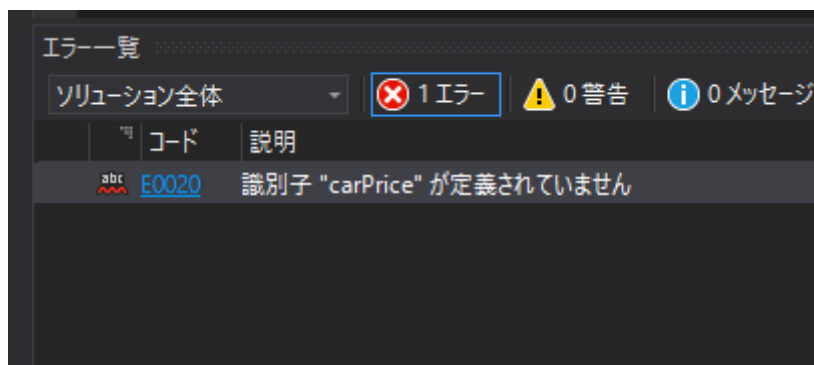
下記のプログラムを入力してください。コンパイルエラーが出るはずです。

```
#include <iostream>
using namespace std;

void buy()
{
    //carPriceはローカル変数なのでアクセスできないよ!!!
    cout << carPrice << "万円の車を買いました。¥n";
}

int main()
{
    int carPrice = 100;    //車の値段。
    //buyという関数を呼び出す。
    buy();
    return 0;
}
```

下記のようなコンパイルエラーが発生したのではないですか？



carPrice という変数は main 関数の中で定義されていて、main 関数の中でだけ有効なローカル変数と呼ばれるものです。そのため、buy 関数では使用することができないため、

- 1 carPrice なんて変数はありませんよ？といった意味のエラーが発生します。そこで、buy 関  
2 数に車の値段を渡すことができる引数を追加しましょう。  
3 下記のプログラムを入力して、コンパイルエラーを修正して下さい。

```
#include <iostream>
using namespace std;

void buy( int price )
{
    cout << price << "万円の車を買いました。¥n";
}

int main()
{
    int carPrice = 100;    //車の値段。
    //buyという関数を呼び出す。
    buy( carPrice );
    return 0;
}
```

- 4 網掛けになっている箇所が修正箇所です。  
5 関数の引数は、その関数のローカル変数になります。つまり、price という変数は buy 関数  
6 の中でだけ有効な変数ということです。

7

### 8 7.3 Ex\_2 引数の値渡し

- 9 7.3 Ex\_1 の引数渡しは値渡しと呼ばれる渡し方で、変数のコピーを渡しています。下記の  
10 コードを入力して動作を確認してください。

```
#include <iostream>
using namespace std;

void Add( int v )
{
    v += 10;
}

int main()
{
    int x = 10;
    Add(x);
    cout << "xの値は" << x << "です。¥n";
    return 0;
}
```

- 11 Add 関数の中で v に 10 加算していますが、x の値は 20 ではなく、10 と表示されたと思  
12 います。これは Add 関数の変数 v が、main 関数の変数 x のコピーだからです。このプログラ  
13 ムは下記のようなコードが実行されていると考えてください。

```
#include <iostream>
using namespace std;

int main()
{
    int x = 10;
    //Add関数はこんな処理が実行されていることと同じ。
    int v = x; //vにxの値をコピー！
    v += 10;    //vを10加算。
    cout << "xの値は" << x << "です。¥n"; //vが加算されているだけなので、xは当然10!
    return 0;
}
```

14

では、次の節では変数の値渡しではなく、参照渡しについて見ていきましょう。

## 7.3 Ex\_2 引数の参照渡し

さて、引数を値渡しで関数に渡すと、呼び出し元の変数の値は変更されないことを学びました。しかし、関数の中で値を変えたい場合は結構あります。そこで、引数の参照渡しについて学びましょう。

### 7.3 Ex\_2.1 参照って何???

では、引数の参照渡しを学ぶ前に、そもそも参照ってなに？というところから学びましょう。参照とは、変数の別名定義となります。では、下記のコードを入力して動作を確認してください。

```
#include <iostream>
using namespace std;

int main()
{
    int x = 10; //xは本名。
    int& v = x; //vはxの別名!!! (芸名やペンネームみたいなもの)
    v += 10;    //vを10加算。
    //vはxの別名だけで正体は同じなので、当然xの値も変わっている。
    cout << "xの値は" << x << "です。¥n";
    return 0;
}
```

下記のように記述すると、vは参照となります。

# int& v = x;

この場合、vはxの別名となります。芸能人の芸名みたいなものですね。名前は違うけど、正体は同じとなります。これを使うと、関数の中で変更した値を、呼び出し元に反映させることができます。下記のコードを入力して動作を確認してください。

```
#include <iostream>
using namespace std;

//vは参照!!!
void Add(int& v)
{
    v += 10;
}

int main()
{
    int x = 10;
    //Add関数の引数vは参照なので、xの別名となる。
    Add(x);
    //Add関数の中で10加算されているので20と表示される!!!
    cout << "xの値は" << x << "です。¥n";
    return 0;
}
```

Lesson\_07 ハンズオン 2

サンプルプログラムの Lesson\_07\_02/Question を使用して、ハンズオン 1～3 を行いなさい。

Lesson\_07 実習課題\_2

サンプルプログラムの Lesson\_07\_02/Question を使用して、実習 1～3 を行いなさい。

サンプルプログラムの URL

<https://1drv.ms/u/s!AnpsoGtakIkVi41n5ETsy70uYECQtg>

ハンズオン 2 の動画

<https://www.youtube.com/watch?v=-NVyUeAXYtg&feature=youtu.be>

実習課題 2 の解説動画

[https://www.youtube.com/watch?v=LgpJbw54\\_eU&feature=youtu.be](https://www.youtube.com/watch?v=LgpJbw54_eU&feature=youtu.be)

Lesson\_07 中間テスト 2

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSeAMKOEI\\_7SVdpkahmPo5DGLh8luvXFVAXn-cy1j7yejyZhag/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSeAMKOEI_7SVdpkahmPo5DGLh8luvXFVAXn-cy1j7yejyZhag/viewform?usp=sf_link)

## 1 7.4 戻り値

### 2 戻り値の仕組みを知る

3 「ブロックの最後に行き着かなくても、return 文が処理されたところで関数の処理が終了  
4 します。」 VisualStudio のプロジェクト、Lesson7 を立ち上げて、下記のサンプルプログラ  
5 ムを入力してください。

#### 6 7.4.1 サンプルプログラム

```
#include <iostream>
using namespace std;

//
// 概要
// クラスの最高点数を求める関数です。
// 引数
// score[] 点数の配列。
// numStudent 生徒数
// 戻り値
// 最高点数が返ります。
//
int GetMaxScore(int score[], int numStudent)
{
    //最高点を線形探索で探す。
    int maxScore = 0;
    for (int i = 0; i < numStudent; i++) {
        if (score[i] == 100) {
            //100点が見つかったので、
            //これ以上の点数はないので、探索を打ち切る。
            return 100;
        }
        if (score[i] > maxScore) {
            //この点数の方が高い。
            maxScore = score[i];
        }
    }

    return maxScore;
}

int main()
{
    //Aクラスのスコア。
    int scoreA[4] = { 60, 20, 80, 90 };
    int saikouTokuten = 0;
    //Aクラスの最高点を求める。
    saikouTokuten = GetMaxScore(scoreA, 4);
    cout << "Aクラスの最高点は" << saikouTokuten << "です。¥n";

    //続いて、Bクラスの最高得点を求める。
    int scoreB[4] = { 20, 100, 80, 30 };
    //F11でステップインして、21行目のreturnで関数が終了しているのを確認してみよう。
    saikouTokuten = GetMaxScore(scoreB, 4);
    cout << "Bクラスの最高点は" << saikouTokuten << "です。¥n";

    return 0;
}
```

7

8

9

- 1
- 2 戻り値の型は int とは限らない。浮動小数点を返す場合は float 型。
- 3 下記のサンプルプログラムを入力してください。
- 4 7.4.2 サンプルプログラム(浮動小数点を返す関数)

```
#include <iostream>
using namespace std;

//大きい方の値を返す関数。
//浮動小数点の値を返す関数。
float Max(float x, float y)
{
    if (x > y) {
        //xの方が大きい。
        return x;
    }
    //yの方が大きい。
    return y;
}

int main()
{
    float v1 = Max(10.5f, 2.5f);
    float v2 = Max(25.53f, 20.12f);
    cout << "v1の値は" << v1 << "です。¥n";
    cout << "v2の値は" << v2 << "です。¥n";
    //引数に変数を渡すこともできる。
    float v3 = Max(v1, v2);
    cout << "v3の値は" << v3 << "です。¥n";
    return 0;
}
```

- 5
- 6 true、false を返す場合は bool 型となる。
- 7 下記のサンプルプログラムを入力してください。
- 8 7.4.3 サンプルプログラム(bool 型を返す関数)

```
#include <iostream>
#include <Windows.h> //WindowsAPIを使うので、windows.hをインクルードする。
using namespace std;

//キーボードのAが押されているかどうかをbool型の
//値で返す関数。
bool IsPressA()
{
    if (GetAsyncKeyState('A') != 0) {
        //キーボードのAが押されている。
        return true;
    }
    return false;
}

int main()
{
    //ゲームループ。
    while (true) {
        bool result = IsPressA();
        if (result == true) {
            //キーボードのAが押された。
            MessageBox(NULL, "Aが押された", "通知", MB_OK);
        }
    }
    return 0;
}
```



- 1 7.4.3 のプログラムは下記のようにも書ける。
- 2 7.4.4 サンプルプログラム(if 文の中で関数呼び出し)

```
#include <iostream>
#include <Windows.h> //WindowsAPIを使うので、windows.hをインクルードする。
using namespace std;

//キーボードのAが押されているかどうかをbool型の
//値で返す関数。
bool IsPressA()
{
    if (GetAsyncKeyState('A') != 0) {
        //キーボードのAが押されている。
        return true;
    }
    return false;
}

int main()
{
    //ゲームループ。
    while (true) {
        if (IsPressA() == true) { //if文の中で関数を呼び出す。IsPressAの戻り値がtrueなら
            //キーボードのAが押された。
            MessageBox(NULL, "Aが押された", "通知", MB_OK);
        }
    }
    return 0;
}
```

- 3
- 4 7.4.4 のプログラムは下記のようにも書けます。
- 5 7.4.5 サンプルプログラム(if 文の中で関数の戻り値をそのまま使う)

```
#include <iostream>
#include <Windows.h> //WindowsAPIを使うので、windows.hをインクルードする。
using namespace std;

//キーボードのAが押されているかどうかをbool型の
//値で返す関数。
bool IsPressA()
{
    if (GetAsyncKeyState('A') != 0) {
        //キーボードのAが押されている。
        return true;
    }
    return false;
}

int main()
{
    //ゲームループ。
    while (true) {
        if (IsPressA()) { //IsPressAの戻り値がtrueなら条件成立。
            //キーボードのAが押された。
            MessageBox(NULL, "Aが押された", "通知", MB_OK);
        }
    }
    return 0;
}
```

- 6 if 文の判定はどの書き方でも構いません。自分が分かりやすい書き方をしてください。た
- 7 だし、他人が書いたコードを読むことはたくさんあるので、色々な書き方に慣れておきまし

よう。

## 7.4 Ex 関数名は分かりやすい名前を考えよう！

分かりやすいプログラムを書くことは非常に重要です。そして、分かりやすいプログラムを書くためには、分かりやすい名前を考えることが重要です。分かりやすい名前を考えるときに、大きな指針となるのが、プロジェクトで統一された規則を守ることでしょう。ここでは、多くのプロジェクトで採用されているいくつかの命名規則を紹介します。

### 動詞+目的語とする

例えばプレイヤーを検索する関数を作る場合、FindPlayer といった関数名にします。Find(動詞)+Player(目的語) 敵を攻撃する関数を作る場合は、AttackEnemy といった関数名にします。

### Upper camel case(アッパーキャメルケース)

単語の先頭を大文字にするという命名規則です。FindPlayer や AttackEnemy などです。

また、Upper camel case にはパスカルケースと呼ばれるものと、ローワーキャメルケースと呼ばれるものがあります。

パスカルケース

→単語の先頭はすべて大文字。FindPlayer、AttackEnemy など。

→関数名、構造体名、クラス名などでよく使われる。

ローワーキャメルケース

→最初の単語だけ小文字。findPlayer、attackEnemy など。

→変数名でよく使われる。

### 判定を行う関数には Is~を使う

これは日本語圏のプログラマーが良く使うもので、英語的には正しくないのですが、関数名の先頭に Is がついていると大抵 bool 値を返す関数です。

Is this a pen?といった感じで、疑問文で使われる Is からきています。例えば、下記のよう

```
if (IsPressA()) {  
    //キーボードのAが押された。  
    MessageBox(NULL, "Aが押された", "通知", MB_OK);  
}
```

## Lesson\_07 ハンズオン 3

サンプルプログラムの Lesson\_07\_03/Question を使用して、ハンズオン 1～2 を行いなさい。

## Lesson\_07 実習課題\_3

サンプルプログラムの Lesson\_07\_03/Question を使用して、実習 1～5 を行いなさい。

サンプルプログラムの URL

<https://1drv.ms/u/s!AnpsoGtakIkVi-0viStE3apdAzUxXw>

## Lesson\_07 中間テスト 3

下記の URL のテストを行いなさい。

[https://docs.google.com/forms/d/e/1FAIpQLSeAMKOEI\\_7SVdpkahmPo5DGLh8luvXFVAXn-cy1j7yejyZhag/viewform?usp=sf\\_link](https://docs.google.com/forms/d/e/1FAIpQLSeAMKOEI_7SVdpkahmPo5DGLh8luvXFVAXn-cy1j7yejyZhag/viewform?usp=sf_link)

## 7.6 関数宣言

**関数宣言**は「こんな関数がどこかにありますよ」、とコンパイラに教えるもの。

これまで作ってきたものは**関数定義**。関数定義は関数の本体。下記のコードを入力してください。max 関数が定義されていませんというエラーが発生するはずです。

```
#include <iostream>

using namespace std;

int main()
{
    int a = 10;
    int b = 20;
    //max識別子が見つかりませんでした、というエラーが出る。
    int c = max(a, b);
    cout << "大きい値は" << c << "です。¥n";
    return 0;
}

//大きいほうの値を返す関数。
int max(int x, int y)
{
    if (x > y) {
        //xのほうが大きい。
        return x;
    }
    //yのほうが大きい。
    return y;
}
```

コンパイラはコードを上から下に向かってコンパイルしていくため、下に何かが書かれているか分かりません。そのため、このようなエラーが発生してしまいます。このようなときに関数宣言を使いましょう。

では、プログラムを下記のように書き換えてください。

```

#include <iostream>

using namespace std;

//これが関数宣言
int max( int x, int y );

int main()
{
    int a = 10;
    int b = 20;
    //max識別子が見つかりませんでした、というエラーが出る。
    int c = max(a, b);
    cout << "大きい値は" << c << "です。¥n";
    return 0;
}

//大きいほうの値を返す関数。
int max(int x, int y)
{
    if (x > y) {
        //xのほうが大きい。
        return x;
    }
    //yのほうが大きい。
    return y;
}

```

1  
 2  
 3  
 4  
 5  
 6  
 7  
 8  
 9  
 10  
 11  
 12  
 13  
 14  
 15  
 16  
 17  
 18  
 19  
 20  
 21

## Lesson 11 いろいろな型

### 11.3 構造体

#### ユーザー定義型

ユーザー(プログラマー)が自由に作成することができる型。C++ II で勉強する `enum` や `class` もユーザー定義型となる。

#### 組み込み型

`int`、`float`、`char`、`bool` などは組み込み型。最初から言語に組み込まれているので組み込み型と呼ばれます。

### 構造体の仕組みを知る

構造体は異なる型の値をまとめるための型となっています(p339)

→なぜまとめるのか？

→そのほうが綺麗でわかりやすいプログラムが書けるから。

→この発展形がオブジェクト指向。

Sample2.cpp を入力してください。

### 構造体を初期化する

波かっこの初期化は初期化リストと呼ばれるもので、C++のほぼすべての型の初期化で使える方法です。

`int` 型の初期化

```
int hoge = {0}; //もちろん OK
```

`int` の配列型の初期化。

```
int hoge[4] = { 4, 5, 6, 7 }; //
```

構造体の配列の初期化。

```
//Car型の配列。  
Car car[4] = {  
    1111, 10.0, 2222, 16.5, 3333, 8.4, 4444, 10.5  
};
```

下記のようにも書ける。こちらのほうが理解しやすいコードになっていると思います。

```
//Car型の配列。  
Car car[4] = {  
    { 1111, 10.0 }, //0番目  
    { 2222, 16.5 }, //1番目  
    { 3333, 8.4 },  //2番目  
    { 4444, 10.5 }  //3番目  
};
```

- 1 下記のサンプルプログラムを入力してください。

```
#include <iostream>

using namespace std;

//車の構造体。
struct Car {
    int number;
    double gas;
};

int main()
{
    //Car型の配列。
    Car car[4] = {
        { 1111, 10.0 }, //0番目
        { 2222, 16.5 }, //1番目
        { 3333, 8.4 },  //2番目
        { 4444, 10.5 }  //3番目
    };
    //車の情報を表示する。
    for (int i = 0; i < 4; i++) {
        cout << i << "番目の車のナンバーは : " << car[i].number << "です。¥n";
        cout << i << "番目の車のガソリン量は : " << car[i].gas << "です。¥n";
    }
    return 0;
}
```

- 2 構造体に代入する

- 3 Sample3.cpp を入力してください。

- 4 構造体の値のコピーのルールは組み込み型(int、float など)の変数と同じです。

- 5 つまり、下記のようなコードを描いた場合でも car1.gas の値は変わりません。

```
Car car1 = { 1234, 25.5 };
Car car2 = { 4567, 52.2 };

car2 = car1; //car2にcar1をコピー。
car2.gas = 10.0; //car2 と car1 は別物なので、car1.gas の値は変わらない。
```

- 6

- 7 下記のプログラムを入力して、動作を確認してください。

```
#include <iostream>

using namespace std;

//車の構造体。
struct Car {
    int number;
    double gas;
};

int main()
{
    Car car1 = { 1234, 25.5 };
    Car car2 = { 4567, 52.2 };

    car2 = car1; //car2にcar1をコピー。
    car2.gas = 10.0; //car2とcar1は別物なので、car1.gasの値は変わらない。

    cout << "car1.gasの値は" << car1.gas << "です。¥n";
    return 0;
}
```

## 11.4 構造体の応用

### 引数として構造体を使う

Sample4.cpp を入力してください。

Sample4.cpp の引数渡しは値渡しなので、下記のようなコードを書いた場合、おそらくプログラマーが意図した動作はしません。

```
#include <iostream>

using namespace std;

//車の構造体。
struct Car {
    int mileage;    //走行距離。マイルージ。)
    int number;     //ナンバー
    double gas;     //ガソリン量。
};

//車を動かす関数。
void MoveCar (Car car)
{
    car.mileage += 20;    //走行距離を20km増やす。
    car.gas -= 1.5;      //ガソリン量を減らす。
}

int main()
{
    //走行距離0、ナンバー1234、ガソリン量25.5で初期化する。
    Car car = { 0, 1234, 25.5 };

    //MoveCar関数を使って車を動かす。
    MoveCar(car);

    //さて結果は？
    cout << "走行距離は" << car.mileage << "です。¥n";
    cout << "ガソリン量は" << car.gas << "です。¥n";

    return 0;
}
```

関数の中で変更した値を、呼び出し元の変数に反映させたい場合は参照(変数の別名)を使えばよかったのを思い出してください。先ほどのコードを下記のように書き換えてください。書き換える箇所は網掛けになっている箇所だけです。

```

#include <iostream>

using namespace std;

//車の構造体。
struct Car {
    int mileage;    //走行距離。マイルージ。)
    int number;    //ナンバー
    double gas;    //ガソリン量。
};

//車を動かす関数。
void MoveCar (Car& car)
{
    car.mileage += 20;    //走行距離を20km増やす。
    car.gas -= 1.5;    //ガソリン量を減らす。
}

int main()
{
    //走行距離0、ナンバー1234、ガソリン量25.5で初期化する。
    Car car = { 0, 1234, 25.5 };

    //MoveCar関数を使って車を動かす。
    MoveCar(car);

    //さて結果は？
    cout << "走行距離は" << car.mileage << "です。¥n";
    cout << "ガソリン量は" << car.gas << "です。¥n";

    return 0;
}

```

ここだけ変更する！

型名の後ろに&をつけると参照になります。

## 11.Ex 1 構造体の発展型としてのクラス

さて、ついに C++ のオブジェクト指向の核となるクラスの勉強を行うための準備が整いました。C++ は C 言語のスーパーセットの言語となっています。ここまで勉強してきた内容は、どちらかというと C 言語の機能です。C++ は雑な説明をすると、C 言語にオブジェクト指向の機能を追加した言語といえます。

これも雑な説明になるのですが、クラスは構造体と関数をひとまとめにしたものと言えます。

では、11.4 のサンプルプログラムをクラスを利用するコードに書き換えてみましょう。



1

```
#include <iostream>

using namespace std;

//Carクラスの宣言。
class Car {
public:
    //これらをメンバ変数という。
    int mileage;           //走行距離。マイルージ。)
    int number;            //ナンバー
    double gas;            //ガソリン量。
    //そしてこれをメンバ関数という。
    void Move();
};

//CarクラスのMove関数の定義。
void Car::Move()
{
    //クラスのメンバ関数では、この関数を呼び出した
    //インスタンスのメンバ変数にアクセスできる!!!!
    mileage += 20;        //走行距離を20km増やす。
    gas -= 1.5;           //ガソリン量を減らす。
}

int main()
{
    //走行距離0、ナンバー1234、ガソリン量25.5で初期化する。
    Car car = { 0, 1234, 25.5 };

    //Move関数を使って車を動かす。
    car.Move();

    //さて結果は?
    cout << "走行距離は" << car.mileage << "です。¥n";
    cout << "ガソリン量は" << car.gas << "です。¥n";

    return 0;
}
```

2

3 クラスの実体(変数などのこと)のことをインスタンスといいます。クラスのインスタンスは  
4 複数作成することができます。先ほどのプログラムを下記のように改造してみてください。  
5 網掛けになっている箇所が追加されたコードです。

6

7

8

9

10

11

12

13

14

15

1

```
#include <iostream>

using namespace std;

//Carクラスの宣言。
class Car {
public:
    //これらをメンバ変数という。
    int mileage;           //走行距離。マイルージ。)
    int number;           //ナンバー
    double gas;           //ガソリン量。
    //そしてこれをメンバ関数という。
    void Move();
};

//CarクラスのMove関数の定義。
void Car::Move()
{
    //クラスのメンバ関数では、この関数を呼び出した
    //インスタンスのメンバ変数にアクセスできる！！！！
    mileage += 20;        //走行距離を20km増やす。
    gas -= 1.5;          //ガソリン量を減らす。
}

int main()
{
    //走行距離0、ナンバー1234、ガソリン量25.5で初期化する。
    Car car = { 0, 1234, 25.5 };
    //Move関数を使って車を動かす。
    car.Move();
    //さて結果は？
    cout << "走行距離は" << car.mileage << "です。¥n";
    cout << "ガソリン量は" << car.gas << "です。¥n";

    //2台目
    Car car2 = { 0, 2222, 30.0 };
    //Move関数を使って車を動かす。
    car2.Move();
    //さて結果は？
    cout << "走行距離は" << car2.mileage << "です。¥n";
    cout << "ガソリン量は" << car2.gas << "です。¥n";
    return 0;
}
```

2 クラスは後期の C++ II でさらに詳しく見ていきますので、ここでは紹介だけにとどめてお  
3 きます。

4

5

6

7

8

9

10

11

12

13

サンプルプログラム Lesson\_11\_01 は下記の URL からダウンロードして、下記のハンズオンと実習を行いなさい。

[https://1drv.ms/u/s!AnpsoGtakIkVi\\_MYnn\\_\\_\\_HfLkZI4cQ](https://1drv.ms/u/s!AnpsoGtakIkVi_MYnn___HfLkZI4cQ)

#### Lesson 11 ハンズオン\_1

サンプルプログラム Lesson\_11\_01/Question を使用して、ハンズオン 1～4 を行いなさい。

#### Lesson 11 実習課題\_1

Lesson\_11\_01/Question を使って、下記の動画のようなプレイヤーを追いかける敵キャラクターを実装しなさい main.cpp の実習ヒント①～⑤を参考に実装しなさい。ただし、ヒントに従わずに自由な方法で実装するのも OK です。

<https://youtu.be/e53lS03xo-A>

#### Lesson 11 実習課題\_2

Lesson\_11\_01/Question を使って、下記の動画のように 4 体敵キャラクターを表示して、プレイヤーを追いかけるようにしなさい。

<https://youtu.be/6EVFBGOi0ck>

#### Lesson 11 実習課題\_Ex\_1

実習課題\_2 のプログラムを Player クラスと Enemy クラスを利用するようにリファクタリングを行いなさい。

(ちょっと難しいという人は、実習課題\_ex\_1\_Answer を見ても OK、構造体を勉強した今だからこそ、クラスを使ったコードも見よう！)

#### Lesson 11 中間テスト 1

下記の URL のテストを行いなさい。

<https://goo.gl/forms/tDxlbEJstAF7ziG52>