復習

C言語ソースプログラムの原型

```
#include      <stdio.h>

int main(void)
{
    printf("Hello World!¥n");
    printf("Hello Japanese Students!¥n");
}
```

オマジナイ

必ず必要

プログラム本体

必ず必要

大原則
プログラム本体内では上の行から下の行へ順に命令が実行される

例外
・ダブルクォーテーションの中
・コメント(後述)

```
Hello World!
Hello Japanese Students!
```

「メイン関数」と呼ばれる
この部分の書き方は幾つかある.

int main(void)	OK
int main()	OK
main()	OKだが警告あり

これらのいずれでも良い.
intやvoidの意味については後で学習.

ソースプログラムの注意

●すべて半角文字を用いる

●キーボードから直接入力できる文字のみを用いる

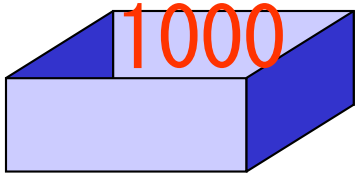
全角　ＡＢＣＤＥＦ　，．　；：”’　１２３４
半角　ABCDEF ,.;:"' 1234

漢字変換や全角文字を使わない

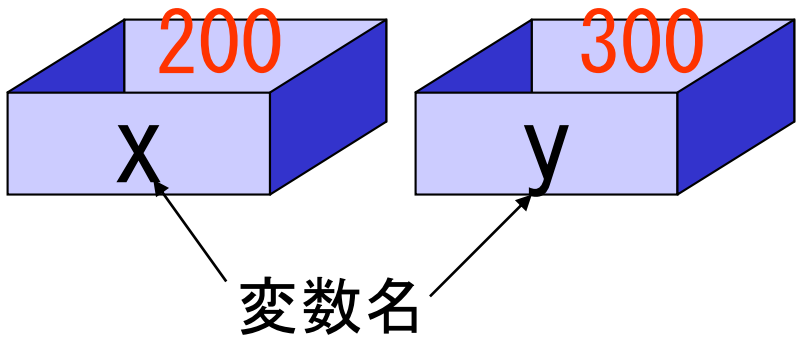
復習

変数とその種類

変数とは何か？
→データ(数値)を入れておく箱



変数名とは何か？
→箱に付ける名前



変数の種類(変数の型)
→入れるデータによって箱の種類が異なる

- int → integer (整数)
- char
- float → floating point (浮動小数点→実数)
- double

・・・その他たくさん

数学の場合

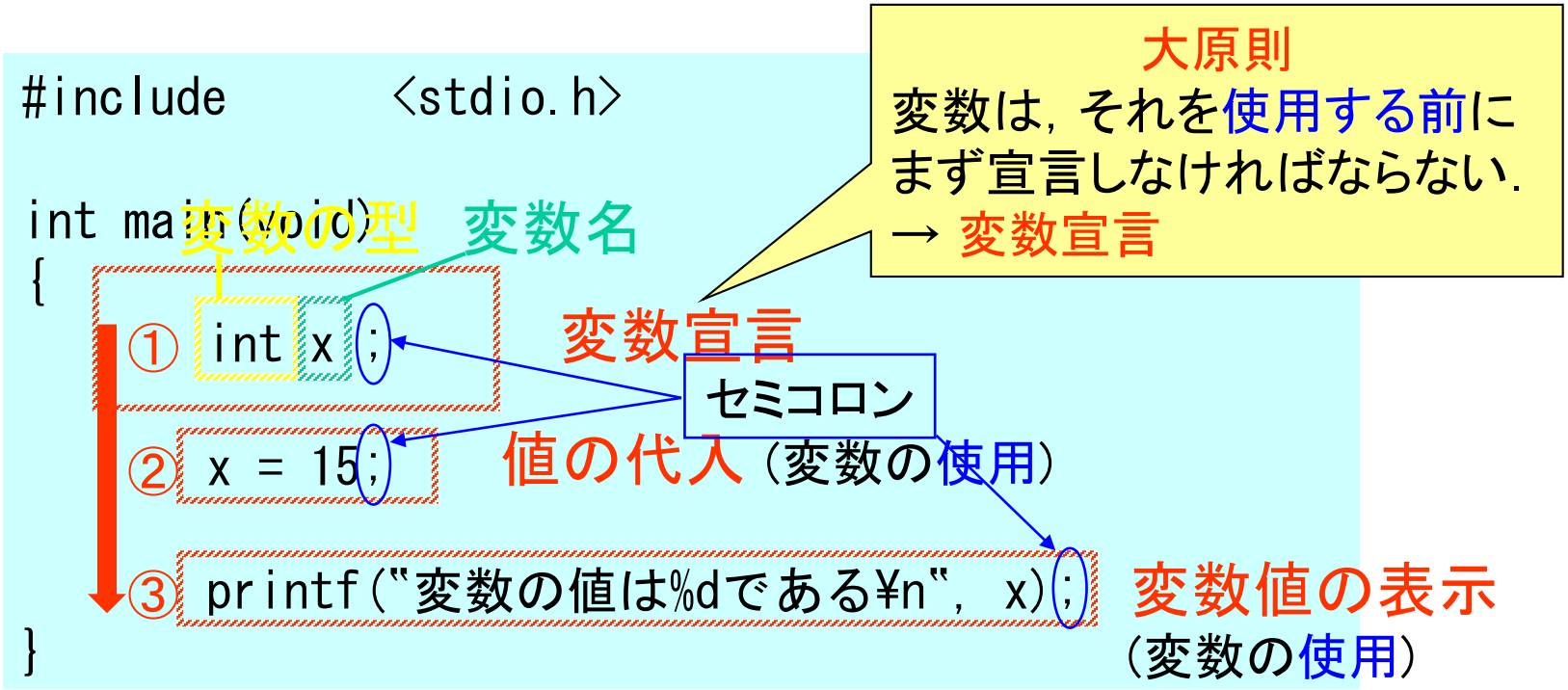
未知数

「変数xに・・を代入し・・」

x = 5
x = 1.3
x = 5/10

復習

変数の宣言とprintf()による値の表示




```
変数の値は15である
```

変数名(識別子)のルール

→ 変数名は1文字とは限らない．x, y, aa, xx, xy, aa1

→ 本スライド末尾

ソースプログラムの書式

復習

大原則
プログラム本体内では
(i)上の行から順に,
(ii)行内では左から右へ
命令文が実行される

```
#include<stdio.h>
int main(void){
int x;x=15;printf("変数の値は%dである¥n",
x);}
```

OK！
基本的にどこで改行してもよいし，ブランク(スペース)はあっても無くてもよい

```
#include  <stdio.h>

int main(void)
{
    int x ;
    x = 15 ;
    printf("変数の値は%dである¥n", x);
}
```

単語の途中で改行してはダメ！

```
#include<stdio.h>
int mai
n(void)
{
    intx;
    x=15;
    pri ntf("変数の値は%dで
    ある¥n",  x);
}
```

intとxの間にスペースがないのでダメ！

単語の途中にスペースがあるのもダメ！

ダブルクォーテーション内で改行するのもダメ！

復習

変数の宣言いろいろ

整数型変数aaaの宣言．変数名は2文字以上でも良い

```
int aaa;
```

浮動小数点型(実数型)変数xxxの宣言

```
float xxx;
```

複数の変数を一つの命令文で宣言

```
int aa, bb;
```

最後はセミコロン

途中の区切りはコンマ

```
int cc = 10, d = 20;
```

複数の変数の宣言と初期値の代入(変数の初期化)を一つの命令文で実行

```
float xx = 1.5, yy = 20.3125;
```

複数の変数の宣言と初期化を一つの命令文で実行(float型の場合)

計算式の書き方

復習

```
float x1, x2, y;
x1 = 2e5;
x2 = 20.0;
y = x1 * x2 / 4.0;
printf("答えは%fだ¥n", y);
```

2e5は2x10^{5}を意味する

＊は乗算，/ は割り算の意味

指数表示

○e△ ⇔ ○x10$^{△}$

答えは1000000.000000だ

問題

実数3.5×10^{5}のプログラム中の表現で正しいのはどれか？

(A) 3.5*10e5 ×

(B) 3.5e5 ○

(C) 350000.0 ○

注意（例：$x = 10^5$の表現）

文字eの前の数字は省略できない

x = e5; × x = 1e5; ○

文字eの前に*を付けてはいけない

x = 1*e5; × x = 1e5; ○

解答

(B)と(C)

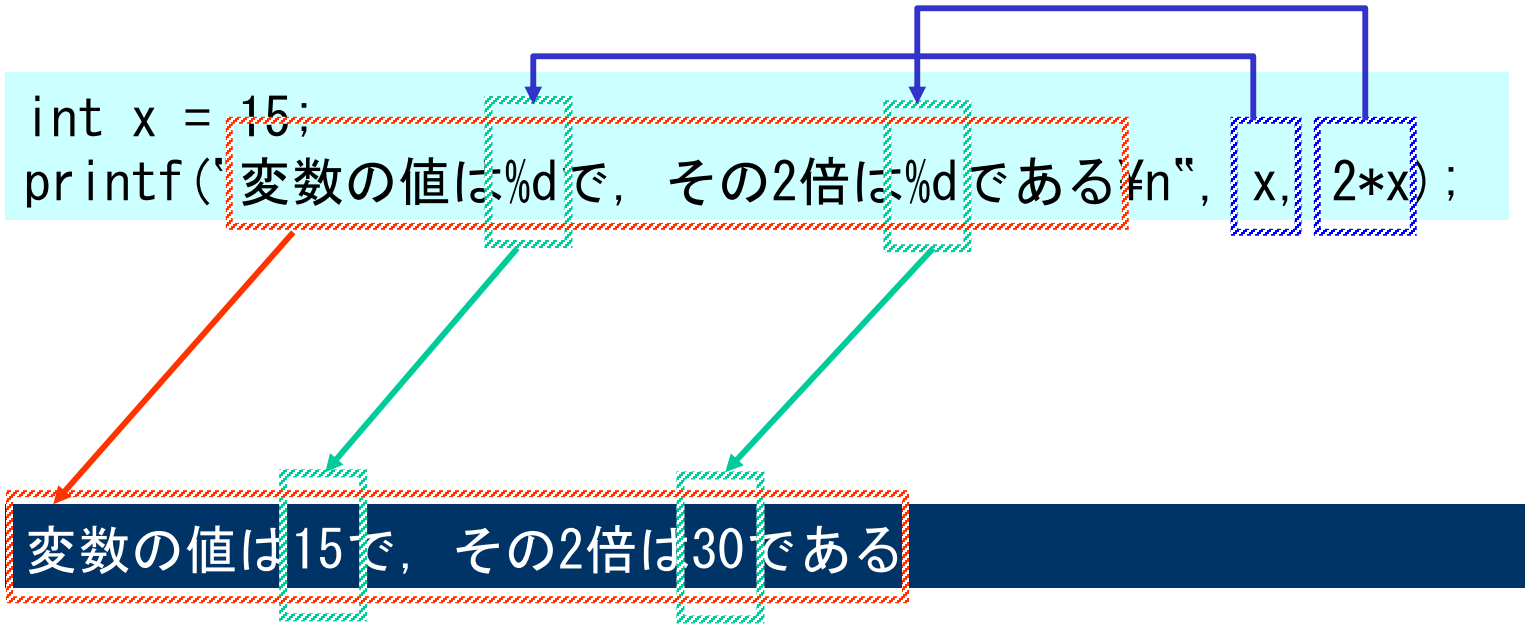
解説

「e5」が「×10^5」に相当するので，

3.5*10e5 と書くと，3.5×10×10^{5}の意味になる．

復習

printf()関数による変数値の表示



%d　は　int型変数の値を表示する

float型変数の値を表示するには？

→ %f %e %g などを用いる

→ 詳しくは，本スライド末尾

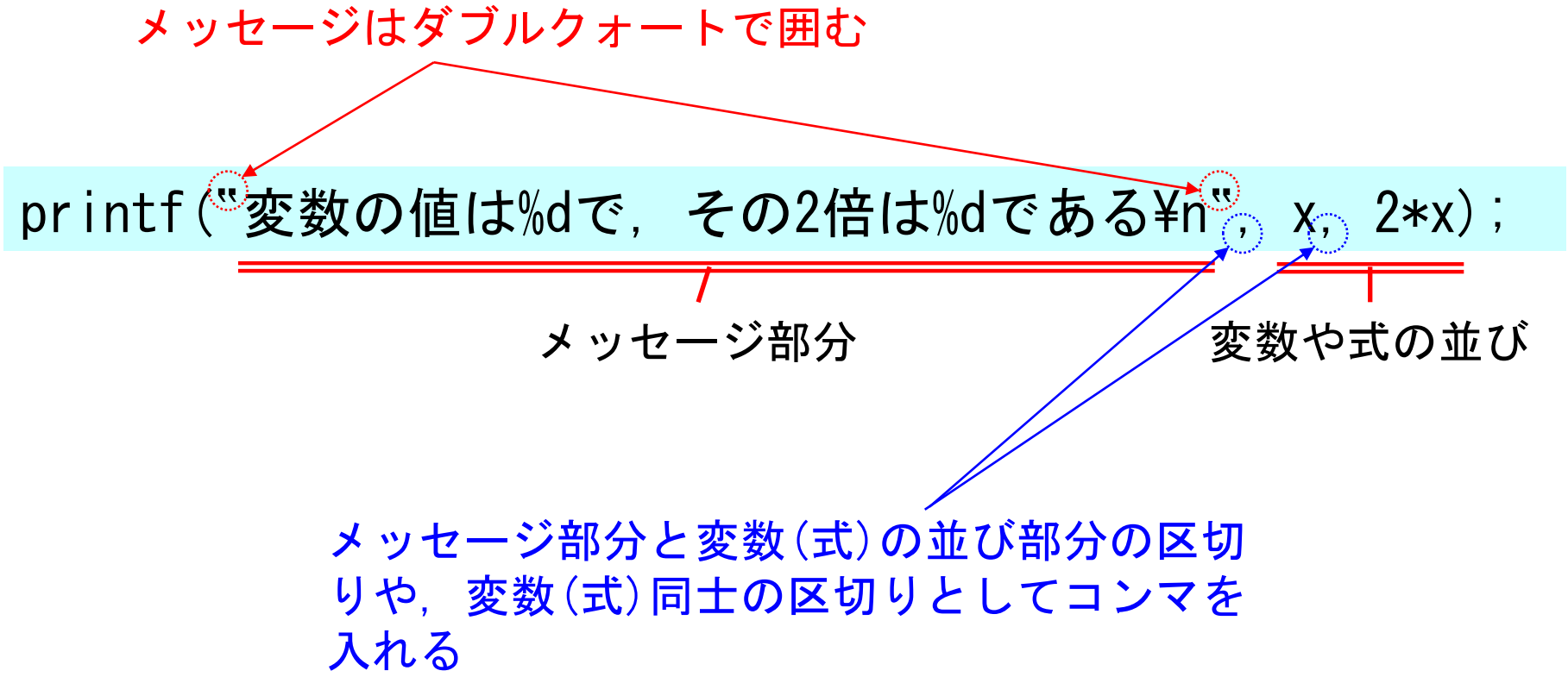
表示するべき変数(値)が不明

表示するべき変数(値)と%dの数が不一致

```
printf("変数の値は%dで，その2倍は%dである¥n");
printf("変数の値は%dで，その2倍は%dである¥n", x, 2*x, 3*x);
```

これらはコンパイルエラーにはならないが，表示がおかしくなる

printf()関数で変数値を表示する際の文法



浮動小数点数の代入と表示

```
int bb = 1300.5;
printf("bb = %d¥n", bb);
```

```
bb = 1300
```

int型に実数(浮動小数点数)を代入すると，少数以下は切り捨て

四捨五入ではない

int型で表すことができない数値はおかしな結果になる

```
int cc = 10.56e20;
printf("cc = %d¥n", cc);
```

```
1344274432
```

コンパイラからのメッセージ
warning = 警告

warning C4244: '=' : 'double' から 'int' に変換しました。データが失われているかもしれません。

```
float xx = 100.56e-20, yy = 10000;
printf("xx = %f,    yy = %f¥n", xx, yy);
```

```
xx = 0.000000,    yy = 10000.000000
```

変換文字%fは小数点形式で表示

```
printf("xx = %e,    yy = %e¥n", xx, yy);
```

```
xx = 1.005600e-018,    yy = 1.000000e+004
```

変換文字%eは指数形式で表示

```
printf("xx = %g,    yy = %g¥n", xx, yy);
```

```
xx = 1.0056e-018,    yy = 10000
```

変換文字%gは適切と思われる形式を自動的に選んで表示

復習

【付録】 変換文字(変換指定)の例

- printf中で%で始まる文字には変数の値が表示される
- 変換文字の例(詳細は教科書p.356(旧p.318)または参考書を見よ)

%d	整数型(int型)の変数を10進数で表示
%o	整数型(int型)の変数を8進数で表示
%x	整数型(int型)の変数を16進数で表示
%f	実数型(浮動小数点型，float型)の変数を小数点形式(mmm.ddddddd)で表示
%lf	実数型(浮動小数点型，double型)の変数を小数点形式(mmm.ddddddd)で表示
%e	実数型(浮動小数点型，float型，double型)の変数を指数形式(m.ddddddd e ±xx)で表示
%g	%fと%eの形式の内，適切なほうを自動的に選択して表示
%c	char型の変数を文字で表示
%s	char型の配列(ポインタ)を文字列で表示

いろいろな変数型(1)

1バイト = 8ビット

0000 0000	0
0000 0001	1
～	～
1111 1111	255

4バイト = 32ビット

0000 0000 0000 0000 0000 0000 0000 0000	0
0000 0000 0000 0000 0000 0000 0000 0001	1
～	～
1111 1111 1111 1111 1111 1111 1111 1111	4,294,967,295

いろいろな変数型(2)

char

1バイト　→　英数字1文字を入れるのにぴったり
アスキーコード　→　文字と文字列で学習

int

4バイト　もっとも標準的な整数型

float

2進法で10進実数を表わすので誤差がある
（有効数字8桁程度）

4バイト　単精度実数型(単精度浮動小数点型)

double

floatより高精度(有効数字15桁程度)

8バイト　倍精度実数型(倍精度浮動小数点型)

計算機における有効桁数と計算精度

有効数字

数学　$3 \times \frac{1}{3} = 1$

計算機　$3 \times 0.33333333 = 0.99999999$

有効数字（有効桁数8桁）

２進法による小数点以下の数値の表現

２進数　１１１１．１１１１　＝ 15.9375

$1\times2^3 + 1\times2^2 + 1\times2^1 + 1\times2^0$
$= 8 + 4 + 2 + 1$
$= 15$

$1\times2^{-1} + 1\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$
$= 0.5 + 0.25 + 0.125 + 0.0625$
$= 0.9375$

少数以下の2進数による10進数値の表現

$(100)_2 = 1 \times 2^2 = 4$
$(10)_2 = 1 \times 2^1 = 2$
$(1)_2 = 1 \times 2^0 = 1$
$(0.1)_2 = 1 \times 2^{-1} = 0.5$
$(0.01)_2 = 1 \times 2^{-2} = 0.25$
$(0.001)_2 = 1 \times 2^{-3} = 0.125$
$(0.0001)_2 = 1 \times 2^{-3} = 0.0625$

少数以下の10進数値には，有限桁数の２進法では表せない数値がある．

丸め誤差

問題：10進数の 0.1 は2進数で正確に表せるか？

$(0.001)_2 = 1 \times 2^{-3} = 0.125$ 大きすぎ！
$(0.0001)_2 = 1 \times 2^{-4} = 0.0625$ 小さすぎ！
$(0.00011)_2 = 1 \times 2^{-4} + 1 \times 2^{-5} = 0.0625 + 0.03125 = 0.09375$ まだ小さい！
$(0.000111)_2 = 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-6}$
$= 0.0625 + 0.03125 + 0.015625 = 0.109075$ 大きすぎ！
$(0.0001101)_2 = 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-7}$
$= 0.0625 + 0.03125 + 0.0078125 = 0.1015625$ まだ大きい！
$(0.00011001)_2 = 1 \times 2^{-4} + 1 \times 2^{-5} + 1 \times 2^{-8}$
$= 0.0625 + 0.03125 + 0.00390625 = 0.09765625$ 惜しい！

課題の考察

課題1-3
次に述べるプログラムを作成し，ソースプログラムと実行結果を提出せよ．三つのfloat型変数xxとyyとzを宣言し，xxには1.0×10^{10}を，yyには1.0×10^{11}を，またzには1500の数値を代入する．これらの変数の値を次のように表示し，さらにxxとzの積を表示するプログラム．

```
xx=10000000000.000000であり，
yy=99999997952.000000であり，
z=1500.000000である．
いまxxとzをかけるとその答えは15000000266240.000000だ
続行するには何かキーを押してください．．．
```

考察： yyの値，またxxとzの積の計算結果に端数(注)が出ていておかしいが，これはプログラム上の間違いではない．では，なぜこのようになるのだろうか？　どうすれば，もっと正確な答えがでるだろうか？　理由を考えてみよ．わかった人はレポートに考察を書くこと．

考察

10進実数は2進法で完全に表わせないため誤差がある．float型変数の有効数字は8桁程度のため，より精度の高いdouble型を使用すれば改善される．

定数値でよくある間違い

```
int aa, bb, cc = 1;
aa = 1/3*30;
bb = cc/3*30;
printf("aa = %d    bb = %d¥n", aa, bb);
```

どんな結果になる？

```
aa = 0    bb = 0
```

1 / 3 * 30
= 0 * 30
= 0

整数同士の演算なので中間結果も整数値
1÷3 = 0.333... = 0

```
int aa;
aa = 1.0/3*30;
printf("aa = %d", aa);
```

プログラムの世界では...
1.0　浮動小数点数(double型)
1　整数

```
aa = 10
```

1.0 / 3 * 30
= 0.333... * 30
= 10

浮動小数点数と整数の演算結果はdouble型

いろいろな演算子(1)

代入演算子

'='は等号ではなく代入の意味

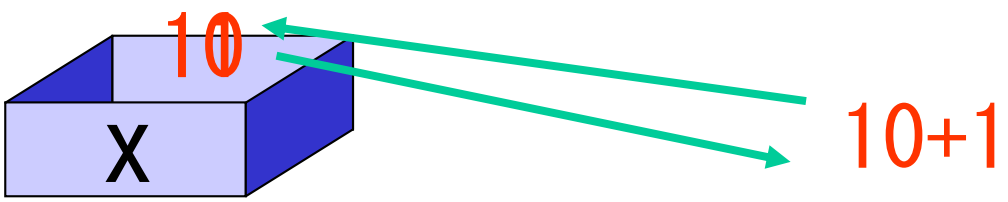
```
x = 10;
x = x + 1;
```

x の解は不定？？？

= は ← の意味

```
x = x + 1;
```
⟶ x ← x + 1

変数(箱) xから値を取り出して，1を足して変数xに代入する



いろいろな演算子(2)

算術演算子

+	足し算
−	引き算
*	掛け算
/	割り算
%	剰余算（例）x = 5 % 3 ⇒ x = 2

割り算の余り
5÷3 = 1 … 2

++	インクリメント演算子 aa++ は aa = aa+1 の省略形 → aaの値を1増やす
−−	デクリメント演算子 aa−− は aa = aa−1 の省略形 → aaの値を1減らす

```
aa = 10;
aa++;          /* この文を実行した後のaaの値は11になる */
```

aa = aa++; と書くのは誤り！

理由
式中の aa++ を aa = aa+1 で書き換えると aa = aa = aa+1

実は，インクリメント演算子の性質のため，aa=aa++;と書いても正しい結果が得られる．その理由が説明できない人は，この書き方は間違いだと覚えておこう！

いろいろな演算子(3)

その他の演算子

+=　-=　*=　/=　%=
<<　>>　^　,

関係演算子と論理演算子

>　>=　<　<=　==　!=
&&　||　!

選択制御文(条件分岐)で学習

scanf()関数による数値の入力

```
int x;
printf("値を入力してください. ");
scanf("%d", &x);
printf("入力された値は, %dです. ¥n", x);
```

```
値を入力してください. 15
入力された値は, 15です.
```

ユーザーがキーボードから入力

「ユーザー」＝プログラムを実行して使う人

```
int aa;
scanf(" %d ", & aa );
```

整数値の入力では%dとする.

変数名の前には&をつける

```
float bb;
scanf(" %f ", & bb );
```

実数値の入力では%fとする.

scanf()関数の注意

```
値はいくらですか？15
```

ダブルクォーテーション内に%d
や%f以外の文字を入れてはダメ

```
int x;
scanf("値はいくらですか?%d¥n", &x);
```

×

```
int x;
printf("値はいくらですか?¥n");
scanf("%d¥n", &x);
```

×

¥nで改行して、次の行で
入力されるのでダメ

×

```
値はいくらですか？
15
```

ダブルクォーテーション内に%d
や%f以外の文字を入れてはダメ

```
int x;
printf("値はいくらですか?");
scanf("%d", &x);
```

○

```
値はいくらですか？15
```

一つの値の入力には，
一つのscanf()を使う．
（複数の変数に一気に入
力することはできない）

Error!!

error C4996: 'scanf': This function or variable may be unsafe. Consider using scanf_s instead. To disable deprecation, use _CRT_SECURE_NO_WARNINGS. See online help for details.

エラー C4996: 'scanf' : この関数または変数は安全ではないかもしれません. 代わりに scanf_s を用いることを検討してください. このエラーを無効にするためには_CRT_SECURE_NO_WARNINGS を用いてください. 詳細はオンラインヘルプを参照してください.

⇒ この関数は安全でない！

これは主にシステムプログラムを書くプログラマへの警告
要するに言いたいことは・・・
「この関数を用いたプログラムは，プログラマがしっかりしてないとウィルスの餌食になるかもしれません」

⇒ 学習には無関係なので無視する.

- Visual Studio 2013以降ではエラーになるため，無視できない！
- Visual Studio 2013以降でも，プロジェクトの設定でSDLチェックを外すと無視できる（エラーでなく警告がでる）.
- Visual Studio 2019での対処方法は次ページで説明.

scanf()関数がエラーになる場合の対処方法

対処方法１
scanf()関数の代わりにscanf_s()関数を用いる

推奨できません！

資格試験(情報処理技術者試験等)でもバツになるかも！

理由：
scanf_s()関数はマイクロソフト独自仕様なので、他のシステムやコンパイラでは使えないことが多い！

Visual Studio 2019でもこれがおススメ

対処方法２
ソースプログラムの1行目に次の行を書いておく

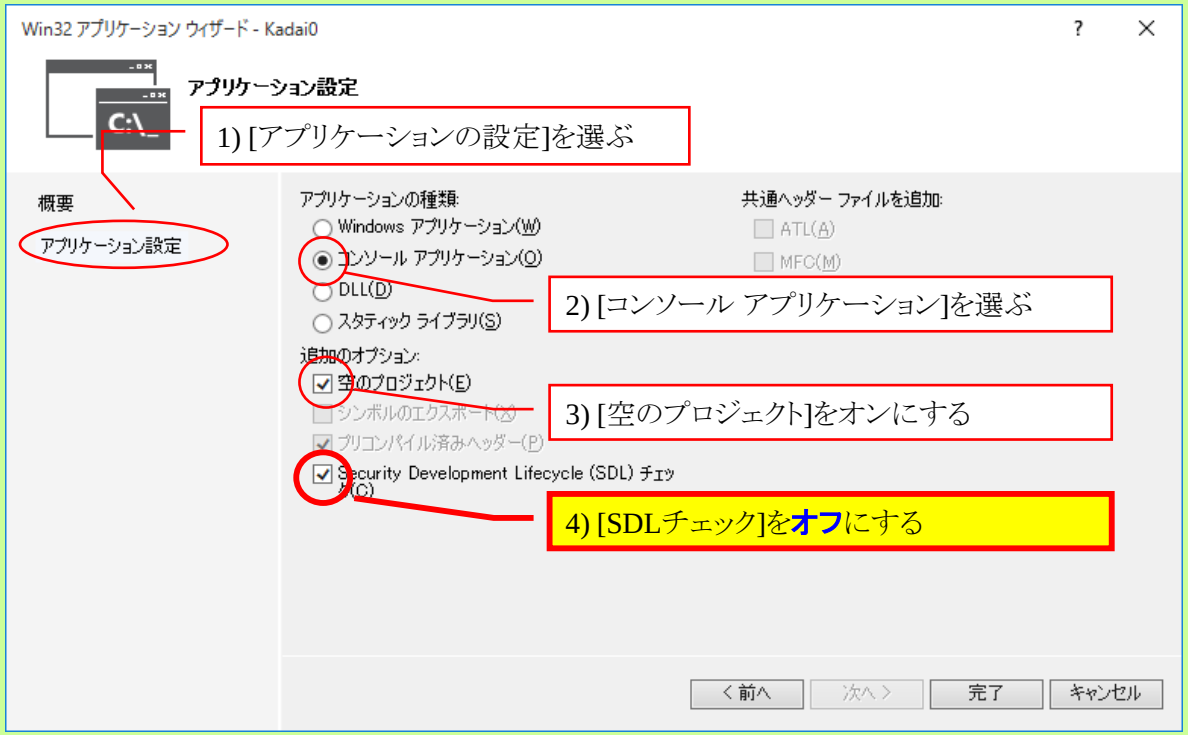
```
#define _CRT_SECURE_NO_WARNINGS
#include <stdio.h>
・・・・
scanf(・・・);
```

Visual Studio 以外のコンパイラではこの行は無視される

scanf()関数がエラーにならず警告も出ない！

scanf()関数がエラーになる場合の対処方法

対処方法3
Win32アプリケーションウィザードで「SDLチェック」をオフにする



scanf()関数がエラーにならずに警告になるので，この警告を無視する．