プログラミング演習第二回　令和4年10月12日

計算するプログラム

例えば、二つの数の和、例えば1+2を求めるプログラム

```
int main(int ac, char *av[])
{
    1+2;
}
```

これは、1+2の計算は行なわれるが計算結果は何にも使われないし、
表示もされないので何もしないのと同じこと。
結果を表示するには、printfを使って、

```
int main(int ac, char *av[])
{
    printf("%d¥n",1+2);
}
```

計算結果を保存する

あるいは、計算結果を保存するには、

```
int main(int ac, char *av[])
{
    int x;
    x=1+2;
}
```

保存する場所を確保

‘=‘で代入（保存）される

このプログラムのままでは、1+2の計算は行なわれるがなにも出力（表示）されないので、printfを使って出力を行なう。

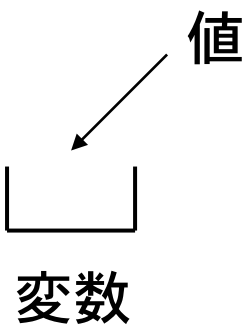
```
int main(int ac, char *av[])
{
    int x;
    x = 1+2;
    printf(“%d¥n”,x);
}
```

変数

プログラム中で、値を格納するには変数 variable を用いる

変数は、格納する値の型によって、整数型、文字型、などの型 type をもつ

変数を使うには、利用に先立って変数の宣言 declaration をしなければならない



変数の値はコンピュータのメモリ上に格納される。具体的にメモリのどの場所に格納されるかは言語処理系が自動的に扱うので、プログラマ（特に初級者）が意識する必要はない。

変数とは値（データ）を格納する容器

値には、整数値、実数値、文字などの様々な型がある

変数の名前

変数や、関数には名前を付ける。

名前は、英字で始まり英字あるいは数字を並べたものであれば良い。
（名前に下線記号文字‘_’を使うことができるが、先頭が_で始まる名前はシステムで使用されていることがあるので、使わない方が良い。）

A	aとは区別される
a123	
This_is_a_long_name	
0xx	数字で始まる名前はダメ
_012	下線記号で始まる名前は許されるが使わないように（システムで使っているかも！）

整数型

整数の値を保持する変数を整数型という

変数宣言には `int` を用いる(英語で整数はInteger)

```
int main(int ac,
                char *av[])
{
   int x;

   x = 1;
   printf("x = %d\n",x);
}
```

整数型の変数 x を宣言

変数 x に整数値 1 を代入

変数 x の値を出力

変数の宣言には、変数の型(整数型の場合 int)に引き続き、変数名を書く。

変数名は原則として自由に付けられる。最初の 1 文字は英字でなければならない。また、C 言語のキーワードは使用できない。

= は、右辺の値を左辺の変数へ代入する代入演算子。

printf による変数の出力

```
printf("x = %d\n", x);
```

書式　　変数名

書式 `"x = %d\n"` の意味

%d は、変数を整数値として出力することを示す変換指定

変換指定 %d が無ければ `printf("x = \n");` は、単に文字列リテラル `"x = \n"` の出力になる。

変換指定は必ず対応する変数と対になっている必要がある。

```
printf("x = %d\n",x);
```

変数の入力と出力

キーボードから整数値を入力し、その値を出力するプログラム

```
int main(int ac, char *av[])
{
   int x;

   printf("整数値を入力せよ:");
   scanf_s("%d", &x);
   printf("入力した整数値は %d です\n",x);
}
```

キーボードから変数の値を入力するにはライブラリ関数 scanf() を用いる

関数 scanf_s()

キーボードから値を整数値として読み込み、変数 x に格納する

```
scanf_s("%d", &x);
```

書式　&変数名

scanf_s()による入力では、
変数名の前に & をつける!

書式 "%d" の意味

%d はキーボードから入力する値を整数値として取り扱うことを指定する変換指定。

&x の意味

& をアドレス演算子という。変数名の前に付けると、その変数が格納されているメモリ上の場所を表す。

scanf_s の書式の変換指定と&変数は必ず対になっていなければならない

```
scanf_s("%d", &x);
```

printf の書式と異なり、scanf_s の書式には変換指定以外の文字は書かない

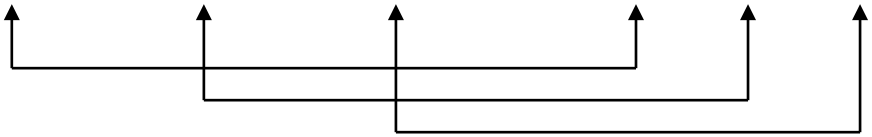
複数の変数を使う

複数の変数を宣言するには、型宣言の後に変数名をコンマで区切って書く

```
int x, y, z;
```

printf を用いて複数の変数を出力するには、書式に複数の変換指定を指定する。書式の後、コンマで変数を区切って記述。

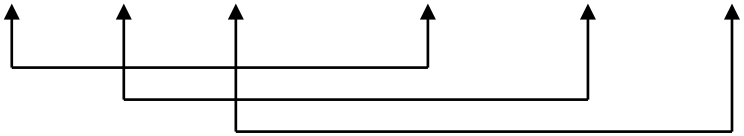
```
printf("x=%d,y=%d,z=%d\n", x, y, z);
```



変換指定と変数は必ず対を形成する。

scanf を用いて複数の変数を入力するには、書式に複数の変換指定を指定。書式のあと、アドレス演算子と変数名をコンマで区切って記述。

```
scanf_s("%d %d %d", &x, &y, &z);
```



キーボードから入力する値は空白（スペース）で区切って入力する。

たし算の計算

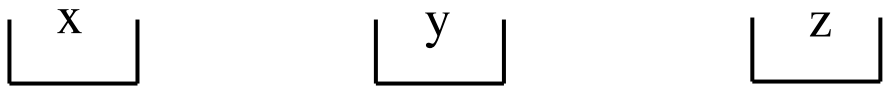
問題：

キーボードから2つの整数値を読み込んで、その和を計算して出力するプログラムを作れ。

考え方：

読み込んだ2つの整数値を格納するために、整数型の変数2つが必要

たし算の結果を格納するためにもう一つの変数が必要



入力値1　　入力値2　　たし算の結果

変数 x と y は scanf_s で入力する。たし算の結果 z は printf で出力する。

たし算のプログラム

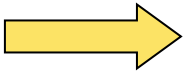
```
#include “pch.h”
#include <iostream>

int main(int ac, char *av[])
{
  int x, y, z;

  printf("整数値を2つ入力せよ:");
  scanf_s("%d %d", &x, &y);
  z = x + y;
  printf("%dと%dの和は%dである\n", x, y, z);
}
```

+ は算術演算子の一つで和を計算する。算術演算子は他には
-, *, /, % がある。それぞれ、差、積、商、余りを計算する。

整数値同士の算術演算

C では、整数型 int 同士の算術演算 +, -, *, /, % の演算結果はすべて整数型になる

```
int + int (加算)
int - int (減算)
int * int (乗算)
int / int (除算)
int % int (剰余)
```



演算結果はすべて整数(int)となる

```
1 + 2  -> 3
5 - 10 -> -5
4 * 6  -> 24
35 / 4 -> 8
35 % 4 -> 3
```

整数同士の割り算には注意が必要!

実数型の場合、加減乗除算は通常の計算、
剰余は商を整数とした時の余り

整数型の範囲

整数型は通常 4 バイト(32ビット)で表される(言語処理系により異なる)。つまり 2 の 32乗 ＝ 4294967296 通りの表現しか出来ない。正と負の整数を考えると、約±20億の範囲の整数しか扱えない。

取り扱い可能な整数値の最大最小値は、limits.h というヘッダファイルで、INT_MAX, INT_MIN として記載されている。

```
#include “pch.h”
#include <iostream>
#include <limits.h>

int main(int ac, char *av[])
{
  printf("整数値の最大値は %d\n", INT_MAX);
  printf("整数値の最小値は %d\n", INT_MIN);
}
```

実数の取り扱い

整数型の変数は整数値しか格納できない。実数を扱う型に浮動小数点型 (floating type) がある。浮動小数点型は 32ビットで表現され 10進数での有効桁数は 7 桁。変数宣言には `float` を用いる。

同じく実数を取り扱う型に、倍精度浮動型（単に倍精度型ともいう）がある。倍精度浮動型は 64ビットで表現され有効桁数は 17 桁。通常はこちらの倍精度浮動型を用いて実数を取り扱う。変数宣言には `double` を用いる。

倍数度型の変数の宣言

```
double x, y;
```
変数 x と y を倍精度として宣言

```
x = 3.1415;
```
変数 x に実数 3.1415 を代入

```
y = 5.0;
```
変数 y に実数 5 を代入

↑

値が実数であることを明示する場合は 5.0 という具合に書く

実数の出力と入力

printf を用いて倍精度型の変数の値を出力するには、変換指定 `%lf` を用いる。

scanf を用いて値を倍精度型としてキーボードから入力するには変換指定 `%lf` を用いる。

```
int main(int ac, char *av[])
{
  double x;

  scanf_s("%lf", &x);
  printf("%lf\n", x);
}
```

```
int型   %d
double型  %lf
float型   %f
```

実数と整数との演算結果

倍精度型同士の算術演算結果は、倍精度型になる。

整数型と倍精度型の算術演算結果は、倍精度型になる。

整数型で宣言された変数に倍精度型の値を代入すると、少数部分が
切り捨てられて代入される。

```
double x;
int y;

x = 3.1415;
y = 10;
printf("%lf\n", x+y);
```

x+y の型は double になるので
変換指定は %lf でなければならない

```
double x;
int seisu;

x = 3.1415;
seisu = x;
printf("%d\n", seisu);
```

実数から整数への変換

実数型から整数型に変換（代入）すると少数以下が
切り捨てられる。

四捨五入するには、0.5を加えて代入すれば良い。
小数点以下2桁目を四捨五入して小数点以下一桁で
表示するには、10倍して、小数以下を四捨五入して
0.1倍する。

```
double x,z;
int y;

x = 3.6415;
y = x*10+0.5;
z = y*0.1;
printf(”%lf -> %lf\n", x, z);
```

計算機で扱う数（整数と浮動小数点数）

計算機で扱っている整数や浮動小数（浮動小数点数）は数学で扱う数とよく似ているがかなり違うものである。たいていはその違いを意識すること無く扱っても問題は起こらない。しかしながら、十分にその違いを認識してプログラムを作る必要がある。

大きさに制限がある。範囲を超えたとき、

浮動小数は無限(inf)あるは非数(NaN)

整数は負の値になったりする

整数の除算では小数以下が切り捨てられる

算術演算子の優先度

算術演算子を複数組み合わせる場合、各演算子は数学と同じ優先順位で実行される

かけ算と割り算 (*， /) は、たし算と引き算 (+， –) よりも優先度が高い

a + b*c とは、a に、b と c の積を足すことを意味する

優先度を変えるには数学と同様、カッコ () を用いる。

(a+b)*c とは、a と b の和に c を掛けることを意味する。

以上の優先度は、整数・実数を問わず成り立つ。

よく使う変換指定

`%d`　int 型を10進数表記に変換

`printf("%5d\n",x);`　int 型の変数 x の値を右詰め 5 桁で表示（10進法）

`scanf_s("%d", &x);` キーボードから入力された値を整数値として変数 x に格納

`%f`　float 型を 10進数表記に変換

`printf(”%f\n",x);`　変数 x の値を表示

`printf("%10.5f\n",x);`　変数 x の値を右詰め 10桁、小数点以下 5桁で表示

`%lf`　double 型を 10進数表記に変換（printfの書式変換としても使用可能）

`printf("%15.5lf\n",x);`　変数 x の値を右詰め 15桁、小数点以下 5桁で表示

`scanf_s("%lf", &x);`　キーボードから入力された値を double型として変数 xに格納

`%e`　実数を指数部付き10進数表記に変換

`printf("%10.4e\n", x);`　変数 x の値を右詰め 10桁、精度 4桁で表示

変換指定は他にもたくさんある。詳細は C 言語の教科書を参照

課題 1

キーボードから 2つの整数値を読み込み、加減乗除と余りを計算するプログラムを作れ。

動作例:

```
整数値を２つ入力せよ：24 7
入力した数は 24 と 7 です
24 + 7 = 31
24 – 7 = 17
24 * 7 = 168
24 / 7 = 3
24 % 7 = 3
```

← 空白文字(スペース)で区切って入力

“24%7”のように出力文に%の文字が含まれている時はprintfの書式で
printf(“%d %% %d¥n”, a,b);
と、%を二つ重ねる。

茶色はプログラムの出力結果

課題 2

(Metric)BMI指標は、身長（m）と体重（kg）から

$$BMI = \frac{\text{体重}}{\text{身長} \times \text{身長}}$$

で求められる。
　身長と体重、二つの数値を入力して BMIを出力するプログラムを作れ。身長と体重は浮動小数で入力する。

```
身長（m）と体重（kg）を入力：1.7 55
BMIは19.0です。
```

茶色はプログラムの出力結果

注意：　身長が変数h、体重が変数wにそれぞれ保存されているとすると、計算式は

$w / (h*h)$

となる。

課題 3

キーボードから 3つの整数値を読み込み、合計値と平均値を計算するプログラムを作れ。（合計値は整数で、平均値は浮動小数点数で計算すること。

動作例：

```
整数値を３つ入力せよ：24 5 50
入力した数は 24 と 5 と 50 です
合計値は 79
平均値は 26.333
```

空白文字（スペース）で区切って入力

茶色はプログラムの出力結果

課題 3（続き）

平均を計算する時に、合計値を3で割ると正しい答えが求められない。
それでh、合計値は整数でそれを整数の3で割ると、少数以下が切り捨てられる。小数点以下まで正しい値を求めるには、整数の割り算でなく浮動小数での割り算を行う必要がある。整数変数に代入された値を浮動小数点数として扱うにはキャストを用いることができる。あるいは、3ではなく、3.0で割り算することで、浮動小数での割り算を計算することができる。

```
int x、y、z;
int m;          //合計値を保持する変数
double a;       //平均値を保持する変数
scanf("%d%d%d", &x, &y, &z);
m = x + y + z;        // 合計の計算
a = (double)m/3;      //平均の計算
```

intをdoubleに変換

変数aには、合計値を浮動小数に変換した値を3で割った結果が代入される。