第4回
計算機基礎実習II

座席指定はありません。

Linux を起動して下さい。

計算機基礎実習II 2018 のウェブページから、以下の課題に自力で取り組んで下さい。

第3回の復習課題(rev03)

第4回の基本課題(base04)

第3回課題の回答例

ex03-3.c

```
#include <stdio.h>

int main() {
   int a, n, m;

   n = scanf("%d", &a);

   m = printf("a is %d\n", a);

   print("n=%d, m=%d\n", n, m);

   return 0;
}
```

1.1つの int 型変数 a を scanf() 関数でキーボードから読み込み、そのときの scanf() 関数の返り値を int型変数 n に代入する。

2.キーボードから読み込んだ a の値をprintf() 関数で出力し、そのときの printf() 関数の返り値を int 型変数 m に代入する。

3. n と m の値を printf() 関数で出力する。

ex03-4.c：ex03-3.c を while で繰り返し

```
#include <stdio.h>

int main() {
   int a = 1;
   int n, m;

   while (a > 0) {
      n = scanf("%d", &a);
      m = printf("a is %d\n", a);
      print("n=%d, m=%d\n", n, m);
   }

   return 0;
}
```

while の条件を満たすように適当な値で初期化

初期化されない場合、変数の値に何が入ってるかは不明

a は値が代入される前に読まれる＝初期化が必要

n, m は値が読まれる前に代入される＝初期化は不要

ex03-5.c

```
#include <stdio.h>

int main() {
   double x = 1;

   while (x != 0.0) {
      x /= 2.0;              /* x = x/2.0; */
      print("%e\n", x);      // x を指数表示
   }
   return 0;
}
```

一行コメントアウト：
この行の // 以降はコメントとして
コンパイラから無視される

実数変数(double型)もまた、どんな小さな値でも表現できる(変数の値として持てる)わけではない ＝ その限界を超えて小さくなった値は 0 として表現される

ex03-6.c

```
#include <stdio.h>
#include <limits.h>
#include <float.h>

int main() {
   printf("int型の最大値は %d\n", INT_MAX);
   printf("int型の最小値は %d\n", INT_MIN);
   printf("double型の最大値は %e\n", DBL_MAX);
   printf("double型の最小値は %e\n", DBL_MIN);

   return 0;
}
```

```
int型の最大値は 2147483647          // 2^31
int型の最小値は -2147483648         // -(2^31 + 1)
double型の最大値は 1.797693e+308
double型の最小値は 2.225074e-308
```

第4回の内容

試験実施の練習 ex04-1.c

いろいろな変数型(基本型、void型、符号なし変数)

異なる型での算術演算

代入による型変換

明示的な型の変換

変数の型(type)

基本型

整数型
（char, short, int, long）

浮動小数点型
（float, double, long double）

void 型

整数型(integer types)

整数を保持する変数や定数の型：
基本的に保持できるサイズが違うだけ

char 1バイト = 8ビット (-128〜127)

short 2バイト(以上) (-32,768〜32,767)

int 2または4バイト

long 4バイト(以上) (-2,147,483,648〜2,147,483,647)

（ **long long** 8バイト(以上) ANSI C99 より ）

大きさは処理系によって異なるが順序はある
char <= short <= int <= long <= long long

char 文字型(character type)

整数型 **char**： 1バイト = 8ビット (-128〜127)

整数型 ＝ 文字型？

ASCII(アスキー)文字コード

計算機は文字形をそのまま(画像として)扱うのは無駄 ⇒ 得意な数値にしたい

文字を数値として扱うために、**数値と文字を対応付けた表が文字コード**

ASCII文字コード：
7ビットの整数(0〜127)と英数字などを対応付けたよく使われている文字コード（以下に一部を抜粋）

9 → \t	47 → /	65 → A	97 → a
10 → \n	48 → 0	66 → B	98 → b
	49 → 1	67 → C	99 → c
33 → !	50 → 2	68 → D	100 → d
34 → "	51 → 3	69 → E	101 → e
35 → #	52 → 4	70 → F	102 → f
36 → $	53 → 5	71 → G	103 → g
37 → %	54 → 6	72 → H	104 → h
38 → &	55 → 7	73 → I	105 → i

日本語は？
その他の文字コード：EUC, JIS, Shift-JIS, UTF-8, UTF-16 など

char 文字型(character type)

整数型 **char**： 1バイト = 8ビット (-128〜127)

整数型 ＝ 文字型 !!

```
#include <stdio.h>
```

文字として扱うときは '' で囲む　　"" で囲むと文字列になる

```
int main() {
   int a1 = 65;        // int型に数値 65 を代入
   int a2 = 'A';       // int型に文字 'A' を代入
   char c1 = 66;       // char型に数値 66 を代入
   char c2 = 'B';      // char型に文字 'B' を代入

   c2 = c2 + 1;
```

char は整数型なので演算できる！

```
   printf("a1 は整数値なら%dであり、文字なら%cである\n", a1, a1);
   printf("a2 は整数値なら%dであり、文字なら%cである\n", a2, a2);
   printf("c1 は整数値なら%dであり、文字なら%cである\n", c1, c1);
   printf("c2 は整数値なら%dであり、文字なら%cである\n", c2, c2);

   return 0;
}
```

変数の型(type)

基本型

整数型
（char, short, int, long）

浮動小数点型
（float, double, long double）

void 型

浮動小数点型(floating point types)

実数を保持する変数や定数の型：
保持できるサイズと精度が異なる

float 4バイト = 32ビット

double 8バイト = 64ビット

long double 10バイト= 80ビット

大きさはC言語規格で決められていないが、ほとんどの処理系で
IEEE754–1985 という規格に沿っており上の通りだと思ってよい

変数の型(type)

基本型

整数型
（char, short, int, long）

浮動小数点型
（float, double, long double）

void 型

void型

値がないことを明示的に示すための型

```
#include <stdio.h>

void main(void) {
   int a;          // int型の変数 a は定義できるが、
   void v;         // void型の変数定義はエラーになる

   return;
   // return 0;
}
```

main() 関数の返り値（出力）は「無い」

main() 関数の引数（入力）は「無い」

void な main() 関数は何も値を返さない

void　　main()　　void

unsigned (符号なし変数)

int など整数型の変数は、変数定義の際に unsigned を型名の前に付けることで、符号なし変数として定義することができます。

```
#include <stdio.h>

int main() {
   int a;              // int型 (-2,147,483,648〜2,147,483,647)
   unsigned int ua;    // unsigned int型 (0〜4,294,967,296)

   unsigned double ux; // unsigned double は無いのでエラー！

   ua = -1;            // ただし負数を代入できてしまうので注意！
   ua = 4294967296

   printf("ua = %u\n", ua);

   return 0;
}                                  %u で unsigned int として表示
```

第4回の内容

試験実施の練習 ex04-1.c

いろいろな変数型(基本型、void型、符号なし変数)

異なる型での算術演算

代入による型変換

明示的な型の変換

異なる型での算術演算

異なる型同士の演算はより**上位**の型に自動的に（暗黙的に）変換されて計算される

int < long < long long < float < double < long double

（char, short の演算は全て自動的に int に変換される）

```
int i = 2;       long l = 3;
float f = 2.1;   double d = 3.0;

i+l;            // i は自動的に long に変換され i+l は long (5)

i*f;            // i は自動的に float (2.1) に変換され i*f は float (4.2)

i*0.1;          // i は自動的に double (2.0) に変換され i*0.1 は double (0.2)

(i/l)*f;        // i は自動的に long に変換され i/l は long (0)
                // long (0) は自動的に double に変換され (0.0)*f は double (0.0)

i*(l/0.1);      // l は自動的に double に変換され l/0.1 は double (30.0)
                // i は自動的に double に変換され (30.0)*f は double (60.0)
```

代入による型変換

異なる変数型への代入は自動的に（暗黙的に）その変数型に変換代入される

a = b;　// b の値は常に a の変数型に変換され代入される

```
int i = 2;       long l = 3;
float f = 2.1;   double d = 3.0;

i = 0.1;        // 0.1 は自動的に int に変換され i は int (0)

i = f;          // f は自動的に int に変換され i は int (2)

l = f*i;        // i は自動的に float (2.0) に変換され f*i は float (4.2)
                // float (4.2) は自動的に long に変換され l は long (4)

d = i%l/f;      // i は自動的に long に変換され i%l は long (2)
                // long (2) は自動的に float に変換され
                //                          (2.0)/f は float (0.952381)
                // float (0.952381) は自動的に double に変換され
                //                                d は double (0.952381)
```

明示的な型の変換

キャスト(cast) によって明示的に型変換

(type)x;　// x の型は明示的（強制的）に type型に変換される

```
int i = 2;        long l = 3;
float f = 2.1;    double d = 3.0;

(int)0.1;         // 0.1 は int に変換され int (0)

l/(double)i;      // i は double に変換され double (2.0)
                  // l も double に変換され double (1.5)

d*(int)f;         // f は int に変換され int (2)
                  // int (2) は double に変換され d*(2.0) は double (6.0)

(int)f*d;         // f は int に変換され int (2)
                  // int (2) は自動的に double に変換され double (6.0)

(int)(f*d)        // f は自動的に double に変換され f*d は double (6.3)
                  // double (6.3) は int に変換され int (6)
```