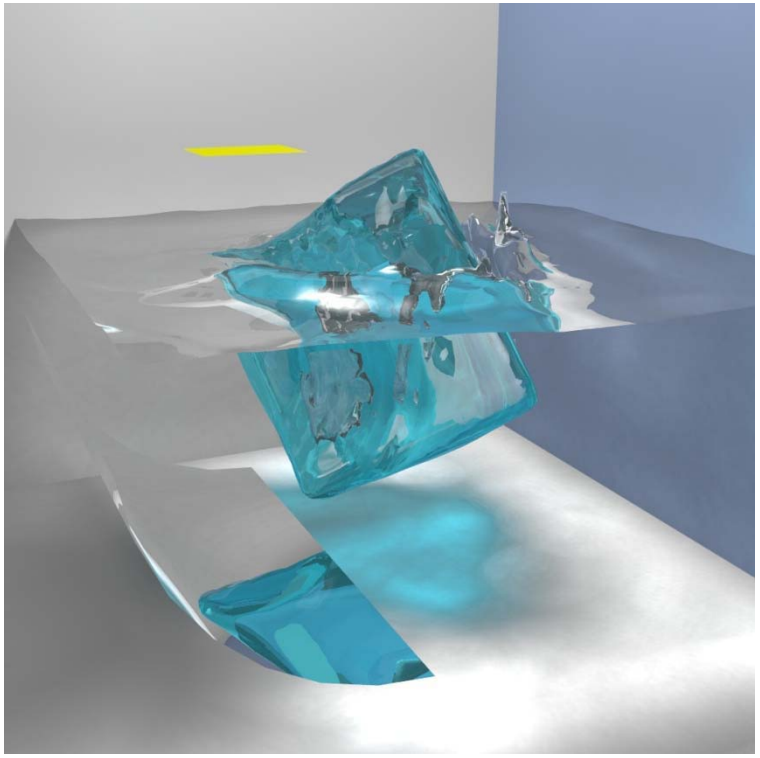
プログラミング

第5週



静岡大学　工学部機械工学科
知能・材料コース　ロボット・計測情報分野　臼杵 深
光電・精密コース　光ナノバイオ分野　居波 渉

講義の前に（重要な情報）

・講義の担当

第4週～第7週は臼杵が担当

・期末試験

クラスⅡ：2月9日（金）10:20-11:50

・再試験

期末試験で60点未満の場合，再試験となる.

2月26日または2月27日の予定

講義アウトライン

- 復習
  - 課題の復習
  - 配列 (p. 42)
- C言語の基礎
  - 配列の誤用
  - 論理演算子
  - インクリメント/デクリメント演算子
  - 演算子の優先順位と結合規則

復習:C言語の特徴

1. 小文字でプログラムを書く
2. 簡潔な表現ができる
3. 演算子が豊富
4. ポインタを用いる
5. データ型が豊富
6. 関数で構成される
7. 構造化制御文が備わっている
8. プリプロセッサ付きである
9. 入出力処理や文字列処理は関数で行う
10. 特殊文字の表現が可能
11. 関数プロトタイプを宣言する

復習:C言語のルール p-34

1. プログラムは関数で構成される
2. プログラムはmain関数が１つ必要である
3. 関数は　{　で始まり　}　で終わる
4. 文には;(セミコロン)が必要である
5. コメントは /* と */ で囲む
6. 変数は宣言してから使用する
7. 変数の値を出力するには書式指定が必要である
8. プログラムは字下げをすると見やすくなる
9. 名前は英数字と_(下線)を使って書く
12. 不可視コードはエスケープシーケンスで記述できる

¥n, ¥t, ¥0, ¥', ¥" など

復習:C言語のルール p-34

1. プログラムは関数で構成される
2. プログラムはmain関数が１つ必要である
3. 関数は　{　で始まり　}　で終わる
4. 文には;(セミコロン)が必要である
5. コメントは /* と */ で囲む
6. 変数は宣言してから使用する
7. 変数の値を出力するには書式指定が必要である
8. プログラムは字下げをすると見やすくなる
9. 名前は英数字と_(下線)を使って書く
12. 不可視コードはエスケープシーケンスで記述できる

    ¥n（改行）, ¥t（タブ）¥0（ヌル）,

    ¥', ¥"（シングル，ダブルクオーテーションの表示））

```
include<stdio.h>

int main(void){
   int a; float b; double c;

      /* a, b, cに数値を読み込む */
   scanf(“%d%f%lf”, &a, &b, &c);

      /* 1行でa, b, cを順番にコンマで区切り出力する。最後に改行する。*/
   printf(“%d,%f,%f¥n”, a, b, c);

    return 0;
}
```

・3から99までの3の倍数で7の倍数でない数を出力するプログラムを作成せよ。
(for文, if文を使用すること)

```
for( i=3; i <= 99; i = i + 3){
    if( i%7 != 0){
        printf(“%d¥n”,i);
    }
}
```

```
for( i=3; i <= 99; i++){
    if( i%3 == 0){
        if( i%7 != 0){
            printf(“%d¥n”,i);
        }
    }
}
```

復習：配列 – その1（1次元配列）

【説明】 p. 42

配列は、同じ名前で操作される同じ型のデータを集めたもの。一次元配列の宣言の基本的な書式は次のとおり。

データ型名　変数名[サイズ];

配列確保の例

```
int a[10];          /* int型変数10個からなる配a          */
int b[10], c[5];    /* int型変数10個からなる配列bと      */
                    /* 5個からなる配列　c                 */
double d[20];       /* double型変数20個からなる配列　d */
```

復習：配列 – その1（1次元配列）

このとき確保される要素は0からはじまる。したがって「`int　a[10];`」と宣言したとき利用できる配列は、`a[0]`～`a[9]`までの10個である。配列の使い方は普通の変数と同様で、`a[5]`とすればaという配列の6番目の要素に代入したり値を参照したりできる。

またコンパイラは、実行時に配列の境界をチェックしない。それは、「`int a[10];`」の宣言で使える配列要素は`a[0]`～`a[9]`までだが、このときに、

```
a[10] = 100;
a[25] = 100;
```

といった代入をしても、強引に実行する。それは`a[0]`から11番目や26番目の要素に該当するアドレスの内容を無理矢理書き変えることになります。したがって、プログラムを書く際は境界を越えた代入をしないように注意しなければならない。

復習:配列:用例 その1

【用例】リスト p-43

p. 43

```
#include <stdio.h>
int main(void)
{
    int n; int dt[10];
    dt[0]=10; dt[9]=99; n=5; dt[n]=55; dt[2]=dt[9];

    printf("dt[0]=%d¥n",dt[0]);
    printf("dt[2]=%d¥n",dt[2]);
    printf("dt[5]=%d¥n",dt[5]);
    printf("dt[9]=%d¥n",dt[9]);
    return 0;
}
```

復習:配列:用例 その1

【用例】リスト p-43

p. 43

```
#include <stdio.h>
int main(void)
{
    int n; int dt[10];
    dt[0]=10; dt[9]=99; n=5; dt[n]=55; dt[2]=dt[9];

    printf("dt[0]=%d¥n",dt[0]);
    printf("dt[2]=%d¥n",dt[2]);
    printf("dt[5]=%d¥n",dt[5]);
    printf("dt[9]=%d¥n",dt[9]);
    return 0;
}
```

実行結果

```
dt[0]=10
dt[2]=99
dt[5]=55
dt[9]=99
```

復習:配列:用例 その2

配列の値の加算

```
#include <stdio.h>
int main(void)
{
    int i,s;
    int a[10]={1,2,3,5,7,11,13,17,19,23};
    s=0;
    for(i=0; i<=9; i++){
        s+=a[i];
    }
    printf("sum=%d¥n",s);
    return 0;
}
```

？？演算子p. 61

計算結果
？？

復習:配列:用例 その2

配列の値の加算

```
#include <stdio.h>
int main(void)
{
    int i,s;
    int a[10]={1,2,3,5,7,11,13,17,19,23};
    s=0;
    for(i=0; i<=9; i++){
        s+=a[i]; /* s=s+a[i]; と同じ */
    }
    printf("sum=%d¥n",s);
    return 0;
}
```

複合代入演算子p. 61

計算結果
sum=1+2+3+…+19+23

復習：配列：課題４

課題4の2.4
教科書P-68の配列の初期化を利用して以下の数値の2乗を計算して出力するプログラムを作成せよ。

配列の初期化の例

```
int a[10]={1,2,3,5,7,11,13,17,19,23};
```

配列：課題4の解答

```
#include <stdio.h>
int main(void)
{
    int i;
    int a[10]={1,2,3,5,7,11,13,17,19,23};
    int b[10];
    for(i=0; i<=9; i++){
        b[i]=a[i]*a[i];
    }
    for(i=0; i<=9; i++){
        printf("%d¥n",b[i]);
    }
    return 0;
}
```

配列の誤用 その1

```
#include <stdio.h> int main(void)
{
    int i; int a[];
    float s;
    float b[]={1.2f,2.3f,3.5f};

    double c[3]={2.5,3.4,1.2, 5.6};

    b[]=6.7f;
    b[-1]=2.6f;
    b[3]=5.8f;
```

配列の誤用 その1

```
#include <stdio.h>
int main(void)
{
    int i;
    int a[];/* 誤り:初期化されない配列は必ずサイズを宣言する.     */
    float s;
    float b[]={1.2f,2.3f,3.5f};  /* 正しい宣言 */
                                 /* 配列のサイズは3となる */
    double c[3]={2.5,3.4,1.2,5.6};/* 誤り:サイズが初期化と異なる    */
    b[]=6.7f;       /* 誤り:添字がないので配列の何番目か決められない*/

    b[-1]=2.6f;      /* 誤り:負の添字は許されない */
    b[3]=5.8f;       /* 誤り:サイズは3なので, 添字は0,1,2のみ */
```

配列の誤用 その2

```
    b[0]=b[1]+b[];

    /* 配列に格納された値の平均を求める */
    s=0;
    for(i=1;i<=3;++i)
    {
        s+=b[i];
    }
    s/=3;
    printf("平均 = %f￥n",s);
    return 0;
}
```

配列の誤用 その2

```
    b[0]=b[1]+b[];  /* b[]は誤り:添字がない */

    /* 配列に格納された値の平均を求める */
    s=0;
    for(i=1;i<=3;++i)  /* 誤り:正しくは for(i=0;i<3;++i) */
    {
        s+=b[i]; /* s=s+b[i]; と同じ */
    }
    s/=3;           /* 正しい:s=s/3; と同じ */
    printf("平均 = %f￥n",s);
    return 0;
}
```

その他の間違い

●printfのデータの型が合っていない

```
float a;
a=1.5f;
printf(“a=%d¥n”, a);
```

●全角文字の使用“　”で囲まれたところとコメント文はOK

●変数の初期化をしていない

復習：データ型のバイト数

p. 67

課題4の2.3

バイトとは何か述べよ。また、教科書**p-67**の**sizeof**関数を利用して，**int**型，**float**型，**double**型のバイト数を表示するプログラムを作成せよ。

復習：データ型のバイト数

p. 67

課題4の2.3

バイトとは何か述べよ。また、教科書p-67の**sizeof**関数を利用して，**int**型， **float**型， **double**型のバイト数を表示するプログラムを作成せよ。

コンピュータのメモリ素子ひとつを表す単位のことをビットという.
1ビットで1か0という1組の値を表現できる.
8ビット分をまとめてバイトという単位で表す.
1バイトで2^8=256種類の値の区別が可能.

復習：データ型のバイト数

p. 67

課題4の2.3

バイトとは何か述べよ。また、教科書p-67のsizeof関数を利用して，int型，float型，double型のバイト数を表示するプログラムを作成せよ。

```
#include <stdio.h>
int main(void)
{
    printf("%d¥n",sizeof(int));
    printf("%d¥n",sizeof(float));
    printf("%d¥n",sizeof(double));
    return 0;
}
```

論理演算子　p-58

論理演算子

p. 58

真偽値を否定したり、複数の条件を組み合わせる。

```
if(a == 10 && b == 20){ /* aが10でかつbが20ならば  */
  ...
}
```

演算子	説明	使用例
!	否定	`if(!flg) /* flgが0であれば真 */` （`if(flg==0)`と同じ）
&&	論理積	`if(10 < a && a < 20)`
||	論理和	`if(a == 2 || a == 4)`

論理演算子の使用例

```
#include <stdio.h>

int main(void)
{
  int a;

  for(a = 1; a <= 5; a++) {
    printf("----%d¥n", a);
    if(2 <= a && a <= 4)
        printf("2以上かつ４以下¥n");
    if(a < 2 || 4 < a)
        printf("2未満または４より大¥n");
    if(!(a == 1 || a == 3))
        printf("1または３、ではない¥n");
  }
  return 0;
}
```

実行結果どうなるか？

インクリメント/デクリメント演算子 p.59

演算子	説明	使用例
++	１加算	**++a;** あるいは **a++;**
--	１減算	**--a;** あるいは **a--;**

a++; あるいは **++a;** は **a=a+1;** と等価
a--; あるいは **--a;** は **a=a-1;** と等価

間違え易いポイント

a=++b; **-> b=b+1; a=b** （前置型）

a=b++; **-> a=b; b=b+1;** （後置型）

算術演算子
– その２（優先順位(p.68)と結合規則）

【説明】これまでに説明した算術演算子（他にも演算子はあるが）には、優先順位がある。これはたとえば、「1+2*3」では2*3が先に計算されて、結果は7になるといったことである。もし「1+2」を優先して演算したいときは、優先順位が最も高い（ ）を用いて、順位を切り替える。すなわち「(1+2)*3」とする。

結合規則

もうひとつ演算子には結合規則というものがある。一つの式の中に同順位 の演算子が存在した場合、結合規則に基づいて優先評価される。たとえば、

```
a = 10;
b = 20;
a = b = 30;
```

では変数 a と b の値は共に30 になる。これは演算子' = 'の結合規則が右結合的(右から左に結合する)となっているからである。このため式はまず右側の「b = 30」が実行され、そのあと「a = b」が実行される。

算術演算子
– その2（優先順位と結合規則）

また

```
8 / 4 * 2
```

という式では '/' と '*' の優先順位は等しいけれど左結合であるため、

```
( 8 / 4 ) * 2
```

として働き、結果は 4 となる。

優先順位が同じ場合は、結合規則に従う。

優先順位と結合規則（P68）

種類	演算子結合法則	→は左から右に結合	
	カッコ	( )	→
	否定	!	←
	乗除	* / %	→
	加減	+ -	→
	比較	< <= > >=	→
	等値	== !=	→
	論理積	&&	→
	論理和	||	→
	代入	=, +=, -=, *=, /=, %=	←
	コンマ	,	→

まとめ

- C言語の基礎
  - 配列の誤用
  - 論理演算子 (p. 58)
  - インクリメント/デクリメント演算子 (p. 59)
  - 演算子の優先順位と結合規則 (p. 68)