担当：富井尚志 (tommy@ynu.ac.jp)

基本的制御構造

今日は復習のような内容なので，演習中心でやってみようかな．．．

☆ 文（statement）

・計算は文(statement) を単位に行なわれる．
・文は

> 式自身またはその組み合わせ
> 制御構造

からなる．
例) 1 から 10 までの合計を求める．

```
int  i, sum;

sum = 0;
for ( i = 1; i < 11; i
++ ){
    sum += i;
}
```

i ++; は i = i + 1; と同じ．++ はインクリメント演算子と呼ばれる．

sum = sum + i; と同じ．

何度もやりましたが，sum = sum + i; は，「sum の元の値に i の値を加えた値を，sum の新しい値として代入しなさい」の意です．
数学の等式とは違います．

☆ 計算の順序付け

・通常，文の現われる順番に行なわれる
例)

```
n = 10;
s1 = 0;
i = n;
```

☆ 式と文

・式の後ろにセミコロン(;)をつけると文になる．
・式は値を持つ(計算結果)．→ 代入式も値を持つ．

☆ ブロック（複文）

・{ } で複数の文を囲むと単一の文と同等に扱われる．

```
{ 宣言
    文 1
    文 2
      :
}
```

☆ 選択計算（条件判断）

・ある条件に応じて別々の計算を行う．

```
if ( 式 )
    文 1
else
    文 2
```

・式が真(0 以外)であれば文 1 が実行され，偽であれば文 2 が実行される．
・else 以下は省略可
例)

```
if ( s1 = = s2 )
    printf("同じだよ¥n");
else
    printf("違うよ¥n");
```

条件式では演算子の優先順位に注意する必要がある．

優先度	演 算 子	結合規則
高い ↑	( ) [ ] -> .	左から右
	! ~ ++ -- + - * & (*type*) sizeof	右から左
	* / %	左から右
	<< >>	左から右
	< <= > >=	左から右
	== !=	左から右
	&	左から右
	^	左から右
	|	左から右
	&& (かつ)	左から右
	|| (または)	左から右
	?:	右から左
↓ 低い	= += -= *= /= %= &= ^= |= <<= >>=	右から左
	,	左から右

※ もし自信がなければ，括弧( )でくくっておく方が無難．

演習問題 1　（この中からランダムに出題しますのでワークシートに回答して下さい）

例）int 型変数 i に 10，j に 20 を代入するプログラムを作りなさい．

回答例 →

（main 関数の内側（赤字の部分）だけでも良いものとする）

```
#include<stdio.h>

int main(void)
{
    int i=10, j=20;
    return 0;
}
```

1－1　i が 10 なら“good”と表示する if 文を書きなさい．

1－2　i が 10 かつ j が 20 なら“good”，さもなければ“no good”と表示する if 文を書きなさい．（ヒント：“かつ”はどう表現する？）

1－3　i が 10 かつ j が 20，あるいは，i が 5 かつ j が 20 なら“good”と表示する if 文を書きなさい．（ヒント：“または”はどう表現する？）

1－4　int 型変数 i の値が偶数なら“偶数です”，奇数なら“奇数です”と表示する if 文を書きなさい．（ヒント：整数の剰余を求める演算子は%）

☆ 反復計算

・「同じような」計算をひとまとめにして，これくり返し行う．
　(a) ある条件が成立している間(成立していない間)繰り返す
　(b) 決められた回数だけ繰り返す

具体的には，
for, while, do～while がある

```
while ( 式 )
    文
```

・式が真の時に文が実行される．これを式が偽になるまで実行する．

```
int  i, sum;

sum = 0;
i = 1;
while ( i < 11 ){
    sum += i;
    i ++;
}
```

```
for( 式 1; 式 2; 式 3 )
    文
```

for の例は前出

・最初に一度だけ式 1 が実行される．
　式 2 が真の時に文が実行される．その後に，式 3 が実行される．
　これを式 2 が偽になるまで繰り返す．

1 から 10 までの和を求めている

```
do
    文
while ( 式 );
```

・まず，文が実行される．次に式が計算される．
　これを式が偽になるまで繰り返す．

```
int  i, sum;

sum = 0;
i = 1;
do {
    sum += i;
    i ++;
} while( i < 11 );
```

例）サンプルプログラム ctrlstruct.c（次頁）


```
/*****************************************************************

    アルゴリズムとデータ構造
    サンプルプログラム ctrlstruct.c
    <<制御構造>>

    copyright (c) 1995,96,97   T.Mori <mori@forest.dnj.ynu.ac.jp>
 *****************************************************************/
#include <stdio.h>

int main(void)
{
  int n,i,s1,s2,s3;

  n = 10;

  /* while を使って，1 から 10 までの合計を求める．*/
  /* 結果は s1 に入る */
  s1=0;
  i = n;
  while( i>0 ) {
    s1 += i;          /* s1 = s1 + i;    と等価 */
    i--;              /* i = i-1;   と等価 */
  }
  printf("s1 == %d¥n", s1);

  /* for を使って，1 から 10 までの合計を求める．*/
  /* 結果は s2 に入る */
  s2 = 0;
  for( i = n; i>0; i-- ) {
    s2 += i;
  }
  printf("s2 == %d¥n", s2);

  /* do-while を使って，1 から 10 までの合計を求める．*/
  /* 結果は s3 に入る */
  s3=0;
  i = n;
  do {
    s3 += i;
    i--;
  } while (i>0);
  printf("s3 == %d¥n", s3);

  /* 同じかどうか if を使って判定する．  */
  if ( s1 == s2 )
    printf("s1 と s2 は同じだよ¥n");
  else
    printf("s1 と s2 は違うよ¥n");

  if ( s2 != s3 )
    printf("s2 と s3 は違うよ¥n");
  else
    printf("s2 と s3 は同じだよ¥n");
    return 0;
}
```

s1 と i の値の変化の様子

	s1	i
while 以前	0	10
1 回目	10 (=0+10)	9
2 回目	19 (=10+9)	8
3 回目	27 (=19+8)	7
・・・・・・・・・・・・		
9 回目	54 (=52+2)	1
10 回目	55 (=54+1)	0
11 回目		

ループ 10 回目で i=0 となり，while(i>0)の(i>0)が偽になるため，11 回目は { } 内が実行されず，行番号 25 へ処理が移る．

ちょっと一言：
文が単文なら，括弧{ }でくくる必要はないが，単文でも括弧{ }でもくくる習慣をつけると良い．なぜか？　それは，新たに文を追加して複文にするときに楽だから．

```
if ( a > = b ){
    c = 1;
    d = 2;
}
```

さもないと括弧{ }も追加する必要がある．

例えばこの文を挿入する場合，書き込むだけで済む．

i--; は，i= i－1; と同じ．i の値を 1 減らすデクリメント．

= = : 等しい
! = : 異なる

```
57
58   【実行結果】
59   s1 == 55
60   s2 == 55
61   s3 == 55
62   s1 と s2 は同じだよ
63   s2 と s3 は同じだよ
64
65
```

演習問題2（この中からランダムに出題しますのでワークシートに回答して下さい）

２－１　1 から 100 までの偶数を全て画面に表示するプログラムを，for 文と if 文を使って作りなさい．（ヒント：制御変数として int 型変数 i を使う．）

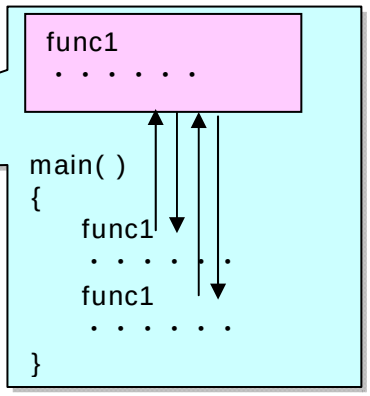
２－２　1 から 100 までの奇数を全て画面に表示するプログラムを，while 文を使って作りなさい．ただし，if 文を使ってはならない．
（ヒント：1 は奇数．次の奇数は 1 に何を足す？　以下同様．）

２－３　1000 より小さい 7 の倍数を全て画面に表示するプログラムを do～while 文を使って作りなさい．
（ヒント：7 から順に 7 ずつ足していけば良いのでは？）

２－４　while 文 と do～while 文 の違いを簡潔に述べよ．

２－５　100 以下の素数を全て画面に表示するプログラムを作りなさい．何を使っても良い．
（ヒント：素数は 1 と自分自身だけを約数としてもつ整数ですよね．
他の数で割り切れる数は表示しなければ良いですね．）

関　数



☆ 計算に対する名前付け

・「同じような」計算が幾つか現れる場合
　→ 名前を付けて再利用する.
　　一般的には**サブルーチン(subroutine)**などと呼ばれる.
・異なる部分に関する情報を**パラメタ(引数)**で与える.
・C 言語ではすべての計算部分に対して「**関数**」という形で名前づけがされる.
（元になる部分は main という名前をつける）

関 数	→ ある値を与えると「対応する」値が得られる
与える値	→「引き数」
得られる値	→「関数の値」(下の関数呼出を見よ)

◎ 関数定義の一般形:

```
得られる値の型
関数名(引数型 1 仮引数 1, 引数型 2 仮引数 2, ... )
{
    変数型 1 変数 11, 変数 12, ...;
    変数型 2 変数 21, 変数 22, ...;

    文 1
    文 2
      :

    return 戻り値;
}
```

例)

```
int
summation( int n1, int n2 )
/* 整数 n1 から n2 までの和を */
/* 求めて返す関数             */
{
    int  i, sum=0;

    for ( i = n1; i <= n2; i ++ ){
        sum = sum + i;
    }
    return sum;
}
```

・void 型 → 値を返さない関数の型
・関数内で定義された変数はその中でのみ使われる(cf. 外部変数)
・仮引数(パラメータ)は局所的

void 型の関数の例：

```
void
display_answer( int number )
{
    printf("**********¥n");
    printf("   答えは %d です. ¥n", number);
    printf("**********¥n");
}
```

引数で渡された整数型変数 number を画面に表示するだけの関数. 戻り値はない. 呼ぶ側では,
　display_answer(n);
などとするだけ.

◎ 関数呼出

・関数内部に記述された計算を行ない値を求める.
・関数呼出

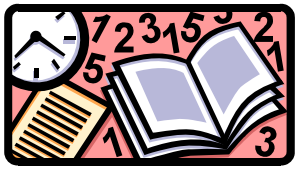
```
関数名( 値 1, 値 2, 値 3, ... )
```

は，式である. → 関数呼出自身が**値(戻り値)**を持つ.

```
main( )
{
    int n;
    ・・・・・・
    n = summation( 1, 100 );
    ・・・・・・
}
```

※関数の本体が，ブロックそのものであることに注意.
・C 言語は call by value による受け渡し(cf. call by reference)
・呼び出し側で関数の返す値を知る必要がある→プロトタイプ宣言

例) サンプルプログラム func.c(次ページ)



```
/*****************************************************************

    アルゴリズムとデータ構造
    サンプルプログラム　func.c
    <<関数の定義>>

    copyright (c) 1995,96,97　T.Mori <mori@forest.dnj.ynu.ac.jp>
 *****************************************************************/
#include <stdio.h>

/*
 * 関数 power のプロトタイプ宣言
 * すべての関数は使用される前に宣言される必要がある.
 * 宣言をしないと暗黙の宣言となる.
 * (暗黙の宣言は「返り値が int 型. 引数には制限なし.」)
 *
 * 関数 power は
 *    第一引数: 浮動小数点数
 *    第二引数: 整数
 *    返り値:   浮動小数点数
 * と宣言されている.
 */
float power( float f, int p );

/*
 * プログラム本体 main もまた関数
 * 返り値ならびに引数に関する宣言がないので,
 *    引数:   特に指定なし
 *    返り値: 整数
 * という暗黙の宣言(定義)がなされたのと同じ.
 */
int main(void)
{
  printf("2.0 ^ 3 = %f¥n", power(2.0, 3));
  return 0;
}

/*
 *  関数 power の定義
 */
float
power( float base, int p )
{
  float x;

  x = 1.0;
  while(p-- > 0)
    x *= base;

  return x;
}
/*
p-- > 0 において, p-- は p = p-1 と同じであるが, 引き算が行なわれるのは,
p > 0 を計算した「後」である.
逆に,  --p > 0 の場合は, 引き算は p > 0 を計算する「前」に行なう.
```

プロトタイプ宣言がないと，main()中での関数 power が未定義になる．なお，プロトタイプ宣言部に関数の本体を書いておけば問題ないが，使用する関数が多数あるときなどを考慮し，全ての関数をプロトタイプ宣言として最初の部分にまとめておくと分かり易い．

関数の型で改行せず,
float power(float base, int p)
{
・・・・・・・・・・・・・
としても良い（好みの問題）.

x *= base; は,
x = x * base;
と同じこと.

ちょっと難しいので無理に使わなくても良い

```
56    */
57
58    【実行結果】
59
60    2.0 ^ 3 = 8.000000
61
```

演習問題3（この中からランダムに出題しますのでワークシートに回答して下さい）

3－1　引数 n1 から n2（n1,n2 は int 型）までの整数の和を求める関数 summation を作りなさい（ヒント：引数は２つです.）

3－2　3-1 で作った関数 summation を使って，5 から 20 までの整数の和を求めて画面に表示するプログラムを作りなさい.（ヒント：特になし）

3－3　画面に I love you. と 1 回書く関数 disp を作りなさい.
（ヒント：この関数の型は？）

3－4　3-3 の関数 disp を使って，画面に 100 回 I love you. と書くプログラムを作りなさい.（ヒント：特になし）

3－5　引数 n が素数なら 1，素数ではないなら 0 を返す関数 prime を作り，それを使って，1 から 100 までの素数だけを画面に表示するプログラムを作りなさい.（ヒント：特になし）

3－6　if 文と関数を使って，ごく簡単な“アドベンチャーゲーム”を作りなさい.
例えば実行時は次のようなもののことです.
今目の前に箱が落ちている．どうする？(1)開ける(2)蹴る
（1 を選んだとき）煙が出てあなたは老人になりました．Game Over!
（2 を選んだとき）中からお金が出てきました．(1)拾う(2)無視する
・・・・・・・・・・・・・・・・・・・・

（ヒント：例えばこんな感じで作ります：

```
printf(”今目の前に箱が落ちている．どうする？(1)開ける(2)蹴る”);
scanf(”%d”,&n);
if (n == 1){
    printf(”煙が出てあなたは老人になりました．Game Over!¥n”);
    else {
        printf(”中からお金が出てきました．(1)拾う(2)無視する”);
        scanf(”%d”,&n);
        if (n == 1){
                ・・・・・・・・・・・・・・・・
```

良いゲームができたら送って下さい．結構面白いですよ！友達同士でやってみるのも良いですよ．こんな風に楽しみながらプログラミングをするのが１番です．“習うより慣れよ”，“好きこそものの上手なれ”です．

応用問題コーナー

問題１

要素数が 10 個の整数型の配列 number[10]に，キーボードから任意の自然数を代入し，その中の最小値と最大値を求めて表示するプログラムを作りなさい．

（ヒント1）キーボードから整数 n を入力するときは，scanf(”%d”,&n)；とします．ここで &n は n の格納されている記憶領域のアドレスを指しますが，今はあまり考えずに，やり方だけ覚えて下さい．

問題２

キーボードから任意の自然数を入力し，2進数に直して表示するプログラムを作成せよ．すなわち，整数が8なら 1000，整数が 10 なら 1010 など．なお，この問題では負の数や0などの誤りの入力は考慮しなくてよい．

（ヒント1）元の整数を 2 で割って商と余りを求める処理を順次繰り返せばいいですね．

（ヒント2）計算しながら余りを出力するとビット列が逆になってしまいますから，配列を用意して代入し，後で逆順で表示するとよいでしょう．難しいですか？