2006 年 12 月 14 日（木）実施

ポインタ

アドレス

C 言語のプログラムを実行する際，主記憶装置（メインメモリ）上にプログラムを配置して，中央処理装置（CPU）で解釈・実行することは，第 1 回の教材で述べたが，プログラムで利用される変数，定数，配列，関数もメインメモリ上でプログラムそのものとは別の場所（データ領域）に配置される。このとき，メインメモリ上に格納されたプログラムやデータの場所を特定するための**アドレス**（番地）はメモリ空間をバイト単位で区切って番号付けされて表される。このとき，何番地にプログラムの先頭が配置されるかはそのときの状況により，一定していない。

C 言語では，データがメインメモリ上で占める領域が何バイト分となるかは，データ型ごとに定められている。例えば，char 型のデータは 1 バイト分の領域を占有する。int 型の場合には，何バイト分を占めるかは CPU に依存し，32 ビット CPU の場合には通常 4 バイト分を占める。

ポインタ変数，アドレス演算子，間接演算子

プログラミングの用語では，一般に他のデータを指し示すデータを**ポインタ**と呼び，C 言語では他のデータを指し示すためにアドレスを利用する。C 言語で利用できるポインタには，次のようなものがある。

1）変数を指すポインタ　　2）配列を指すポインタ　　3）ポインタを指すポインタ
4）ポインタを指すポインタを指すポインタ　　5）ポインタ配列を指すポインタ
6）構造体を指すポインタ　　7）関数を指すポインタ　　8）ファイルを指すポインタ

アドレスは数値として表されるため，C 言語ではその数値を格納する変数として**ポインタ変数**が利用できる。ポインタ変数を用いるには，先ず次のように変数名に * を付けて宣言を行う。

データ型 *変数名;

ここでのデータ型は，ポインタ変数が指しているデータのデータ型である。(ポインタ変数そのものがメインメモリ上で占有するバイト数はアドレスを表現するのに必要なバイト数である。)

例）int *p;　　/* int 型のデータを指すポインタ変数 p */

宣言されたポインタ変数に，指し示すデータをセットするには，そのデータの格納場所を表すアドレスを代入する。指し示すデータが通常の変数の場合，**アドレス演算子** & を付けてアドレスを表す。次の例の代入が行われると，「p は x を指している」または「p は x へのポインタを保持している」と言う。

例）p = &x;

p が x を指しているとき，**間接演算子** * を p に付けて *p とすると，p が指している領域（x）を表す。このとき，*p への代入は x への代入，*p の参照は x の参照となる。

代入の例）*p = 100;　　/* p 及び x が int 型の場合（x = 100; と同じ働き） */

参照の例）printf("p の指す先に格納されているデータは%d¥n", *p);

例題 1

次のプログラムを入力し，翻訳・編集して実行形式のファイルを作成し，実行せよ。ここで，ソースプログラム名は prog9-1.c とする。

```
/* prog9-1.c */
#include <stdio.h>

int main(void)
{
    int x;
    int *p1, *p2;

    p1 = &x;
    *p1 = 80;

    p2 = p1;

    printf(" x の値は%d¥n", x);
    printf("&x の値は%#x¥n", &x);
    printf("p1 の値は%p¥n", p1);
    printf("p1 の指す先の値は%d¥n", *p1);
    printf("p2 の値は%p¥n", p2);
    printf("p2 の指す先の値は%d¥n", *p2);

    return 0;
}
```

【解説】

1. printf 中の変換指定 %#x はアドレスを 16 進数表記（0x から始まる）に変換する。なお，%p は void へのポインタ（あらゆる型へのポインタ）実引数の値を印字可能文字列に変換する。

2. ポインタ p1 の参照はアドレスを得るので，別のポインタ p2 への代入が可能である。&x は定数なので，これに他のアドレスを代入することはできない。

例題 2

次のプログラムを入力し，翻訳・編集して実行形式のファイルを作成し，実行せよ。ここで，ソースプログラム名は prog9-2.c とする。

```
/* prog9-2.c */
#include <stdio.h>

int main(void)
{
    int i;
    unsigned int ui;
    long l;
    float f;
    double d;
    long double ld;

    int *pi;
    unsigned int *pui;
    long *pl;
```

```
    float *pf;
    double *pd;
    long double *pld;

    printf("         int 型変数の大きさは%d¥n", sizeof i);
    printf("unsigned int 型変数の大きさは%d¥n", sizeof ui);
    printf("        long 型変数の大きさは%d¥n", sizeof l);
    printf("       float 型変数の大きさは%d¥n", sizeof f);
    printf("      double 型変数の大きさは%d¥n", sizeof d);
    printf(" long double 型変数の大きさは%d¥n", sizeof ld);
    printf("         int 型ポインタ変数の大きさは%d¥n", sizeof pi);
    printf("unsigned int 型ポインタ変数の大きさは%d¥n", sizeof pui);
    printf("        long 型ポインタ変数の大きさは%d¥n", sizeof pl);
    printf("       float 型ポインタ変数の大きさは%d¥n", sizeof pf);
    printf("      double 型ポインタ変数の大きさは%d¥n", sizeof pd);
    printf(" long double 型ポインタ変数の大きさは%d¥n", sizeof pld);

    return 0;
}
```

【解説】

sizeof は変数やポインタがメインメモリ上で占める領域の大きさをバイト単位で計算する。

例題 3

次のプログラムを入力し，翻訳・編集して実行形式のファイルを作成し，実行せよ。ここで，ソースプログラム名は prog9-3.c とする。

```
/* prog9-3.c */
#include <stdio.h>

void swap1(int, int);
void swap2(int *, int *);

int main(void)
{
    int x, y;

    printf("x の値を整数で入力してください：");
    scanf("%d", &x);

    printf("y の値を整数で入力してください：");
    scanf("%d", &y);

    swap1(x, y);
    printf("swap1 を実行した結果, x の値は%d, y の値は%d¥n", x, y);

    swap2(&x, &y);
    printf("swap2 を実行した結果, x の値は%d, y の値は%d¥n", x, y);

    return 0;
}

void swap1(int a, int b)
{
```

```
    int temp;

    temp = a;
    a = b;
    b = temp;
}

void swap2(int *a, int *b)
{
    int temp;

    temp = *a;
    *a = *b;
    *b = temp;
}
```

【解説】

関数の引数には変数の値が渡される。引数にポインタを用いた場合には，アドレスが渡されるため，ポインタが指す先の変数の領域を操作することが可能となる。

演習 1

char 型変数及び char 型ポインタ変数の大きさを求めて表示するプログラムを作成せよ。ここで，ソースプログラム名は ex9-1.c とする。