コンピュータサイエンスとプログラミング 演習課題

第４回演習課題　ポインタ

2021 年 5 月 13 日

1. アドレスと変数

概要

- アドレス(address)
  - コンピュータのメモリ(memory)の特定の場所を示す番地
  - 1 バイト(8bit)単位のメモリを識別するための数値
- 変数はメモリ上の連続した領域に対応づけられている．（その領域の大きさは変数の型によって異なる）
- 変数の代入・参照
  - その変数に対応するメモリ領域（アドレスによって指定される）に値を書き込んだり，値を取り出したりする操作

アドレス演算子　&

- 変数名の前に**&**（アンパサンド）を付けると，変数の値ではなく，その変数のアドレスが得られる．
  （例）変数 **x** のアドレス　`&x`
  （例）変数 **x** の値を出力　`printf("%p", &x);`
  %p で，引数の値(アドレス）は先頭に**"0x"**をつけた 16 進数で表示される．
- アドレスは変数ではなく値なので，代入演算子の左側には書けない．
  ×　~~`&x =10;`~~
- 変数に対応するメモリ領域のサイズ(単位はバイト）は **sizeof** 演算子を利用してわかる．
  `int s = sizeof(x);`

【例題 4-1】変数のアドレス，サイズ，値を調べる

```
#include <stdio.h>

int main(void){
    int x = 77;
    double y = 3.14;
    printf("HENSU ¥t ADDRESS ¥t SIZE ¥t VALUE¥n");
    printf("x ¥t %p ¥t %d ¥t %d¥n", &x, sizeof(x), x);
    printf("y ¥t %p ¥t %d ¥t %f¥n", &y, sizeof(y), y);
    return 0;
}
```

【課題 4-1】例題 4-1 のプログラムの **x** と **y** で，何種類かの型や値を試してみよ

2. ポインタの宣言

- ポインタ(pointer)はアドレスを格納する変数
- ポインタの宣言
  - 変数名の前に*****（アスタリスク）を付ける
  - 型名は，ポインタによって参照される変数の型（被参照型）を指定する
  - ポインタの型はポインタ型

  （例）**int** 型の変数のアドレスを格納するポインタ **p** の宣言（**p** は **int** へのポインタと呼ばれる）

  `int *p;`　　型名　*****ポインタ変数名
  `int *p = &x;`　　変数 **x** のアドレスをポインタ **p** の初期値として与える

- ポインタにアドレスではなく値を代入するのは誤り
  ×~~`p =100;`~~

【例題 4-2】変数のアドレスとポインタの値を出力

```
#include <stdio.h>

int main(void){
    int x = 77;
    int *p = &x;
    printf("HENSU  ADDRESS  SIZE  VALUE¥n");
    printf("x   %p   %d   %d¥n", &x, sizeof(x), x);
    printf("p   %p   %d   %p¥n", &p, sizeof(p), p);
    return 0;
}
```

- **int** 型の変数 **x** のアドレスをポインタ **p** に代入し，**x** と **p** のアドレスと値を出力する．
- ポインタ **p** の値と変数 **x** のアドレスが同じなので，**x** のアドレスが **p** に代入されたことが確認できる．
- ポインタ **p** も変数なのでアドレスを持っている．

【課題 4-2】

例題 4-2 のプログラムに，**double** 型の変数 **y** （初期値は 3.14）とポインタ **py** の宣言を追加し，変数 **y** のアドレスをポインタ **py** に格納する処理と，**y** と **py** のアドレス，サイズ，値を出力するプログラムを作り，動作を確認せよ．

3. 間接参照： ポインタを使って変数の値を参照する

- ポインタ **p** に変数 **x** のアドレスが格納されている時，ポインタ **p** は変数 **x** を指すという
- 間接参照
  - ポインタが指す変数の値を参照すること
  - ポインタの変数の前に*を付けることで，そのポインタが指す変数を参照できる

    `*`ポインタ変数名　　（*を間接参照演算子と呼ぶ）

（例）

- ポインタ **p** が変数 **x** を指していて，変数 **x** に 77 が格納されている時，次の変数 **y** には 97 が代入される．

  ```
  y = *p + 20;
  ```

- 間接参照演算子は乗算演算子よりも優先順位が高いが，迷ったら，**(*p)** のように括弧を使うと良い．

  ```
  y = *p * *q;
  ```

【例題 4-3】ポインタを使った間接参照

```
#include  <stdio.h>

int main(void){
    int x = 77, y = 0;
    int *p = &x;
    printf("HENSU  ADDRESS  SIZE  VALUE ¥n");
    printf("x   %p   %d   %d¥n", &x, sizeof(x), x);
    printf("y   %p   %d   %d¥n", &y, sizeof(y), y);
    printf("p   %p   %d   %p¥n", &p, sizeof(p), p);
    y = *p + 20;
    printf(" y = *p + 20 is executed.¥n");
    printf("x   %p   %d   %d¥n", &x, sizeof(x), x);
    printf("y   %p   %d   %d¥n", &y, sizeof(y), y);
    printf("p   %p   %d   %p¥n", &p, sizeof(p), p);
    return 0;
}
```

【課題 4-3】

例題 4-3 のプログラムを実行して動作を確認し，ポインタによる間接参照について理解せよ．

【例題 4-4】ポインタの値を変更

```
#include <stdio.h>

int main(void){
    int x = 77, y = 115;
    int *p;
    printf("HENSU  ADDRESS  VALUE  INDIRECT¥n");
    printf("x   %p   %d¥n", &x, x);
    printf("y   %p   %d¥n", &y, y);
    p = &x;
    printf(" p = &x is executed.¥n");
    printf("p   %p   %p   %d¥n", &p, p, *p);
    p = &y;
    printf(" p = &y is executed.¥n");
    printf("p   %p   %p   %d¥n", &p, p, *p);
    return 0;
}
```

ポインタ **p** は最初変数 **x** を指しているが，12 行目の **p=&y** を実行した後，変数 **y** を指すようになる．

【課題 4-4】

例題 4-4 のプログラムを実行して動作を確認し，ポインタ **p** が指す変数が変化していることを理解せよ．

4. 間接代入： ポインタを使って変数の値を変更する

- ポインタが指す変数の値を変更する

  `*ポインタ変数名 = 式;`　　（*****は間接参照演算子）

  （例）ポインタ **p** が指す変数に 10 を代入する

  `*p = 10;`

【例題 4-5】ポインタを使った間接代入

```
#include<stdio.h>

int main(void){
        int x = 77;
        int *p;
        p = &x;
        printf("HENSU  ADDRESS  VALUE  INDIRECT ¥n");
        printf("x   %p   %d¥n", &x, x);
        printf("p   %p   %p   %d¥n", &p, p, *p);
        *p = 15;
        printf(" *p = 15 is executed.¥n");
        printf("p   %p   %p   %d¥n", &p, p, *p);
        return 0;
}
```

【課題 4-5】

例題 4-5 のプログラムを実行し，間接代入によって，ポインタが指す変数の値が変わることを確認せよ．

【課題 4-6】

4 つの変数を下の左のように定義する．ポインタ **p** を使って，変数 **x** と **y** の値を交換して，右の実行例のように出力するプログラムを作成・実行し，動作を確認せよ．

```
int x = 77;
int y = 115;
int tmp;
int *p = &tmp;
```

```
% ./a.exe
HENSU  ADDRESS        VALUE
x      0xbffff754     77
y      0xbffff750     115
 exchange x and y
x      0xbffff754     115
y      0xbffff750     77
```

5. 関数の呼び出し

- 関数は呼び出し時に指定された値（引数）を使って計算し，結果（返却値）を **return** 文で返す.
- 引数
  - 実引数： 関数を呼び出すときに指定した引数
  - 仮引数： 関数定義の **( )** の中に書いた引数
- 値による呼び出し（call by value）
  - 関数を呼び出すとき，実引数の値を仮引数へコピーしてから関数を実行する.
  - 関数を呼び出した後でも実引数の値は変化しない
- 参照による呼び出し（call by reference）
  - 変数のアドレスを実引数に指定する.
  - 実引数に指定したアドレスが仮引数にコピーされ，仮引数のポインタを使って間接代入することで，実引数の値を変更する.

ポインタ引数 （ポインタ型の仮引数）

参照による呼び出しを行う関数を定義する時は，仮引数をポインタ型として宣言する.

【例題 4-6】関数の参照による呼び出しで，ポインタ引数 x が指す変数の値を 2 乗する

```
#include <stdio.h>

void psquare (int *x){
      printf("Kari-Hikisu  x  %p  %p¥n", &x, x);
       *x = (*x)*(*x);
}

int main(void){
       int b, x;
       printf("INPUT AN INTEGER  ");
       scanf("%d", &x);
       b = x;
       printf("    ADDRESS    VALUE¥n");
       printf("Jitsu-Hikisu  x  %p   %d¥n", &x, x);
       psquare(&x);
       printf("SQUARE of %d IS %d. ¥n", b, x);
       return 0;
}
```

【課題 4-7】 例題 6 のプログラムを実行し，**psquare** 関数の実引数 **x** で指定したアドレスが仮引数 **x** にコピーされていることを確認せよ

【課題 4-8】 ポインタ引数 **x** が指す変数の値を **d** だけ増やす **pinc** 関数を定義し，例題 6 に習って **main** からこの関数を呼び出すプログラムを作成して，動作を確認せよ. `void pinc(int *x, int d);`

レポート課題の提出について

- 課題 4-6，4-8 を実施し，レポート課題として提出すること（提出期限：2020 年 5 月 20 日 09:00）
- CLE で提出する際のファイル名（半角英数）は以下の通りとする（XXXX は学籍番号下 4 桁）.
  - 課題 4-6 の場合，XXXX-kadai4-6.c
  - 課題 4-8 の場合，XXXX-kadai4-8.c