Zoom URL

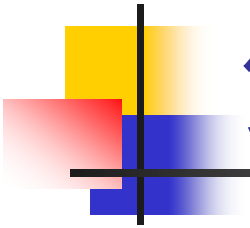
- 第4回演習は下記URLで実施します。
- 5/13（木）13:30 までにアクセスしてください。
（10分程前からアクセス可能になります）

https://us02web.zoom.us/j/84792016548?pwd=SmVXUGxTZ1FoQ3ZxTEFEMys5aGg5UT09

ミーティングID: 847 9201 6548
パスワード: csp

コンピュータサイエンスとプログラミング
第4回演習

2021年5月13日(木)

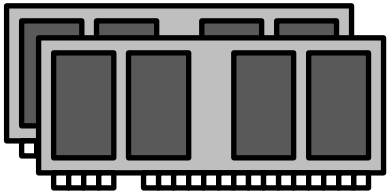


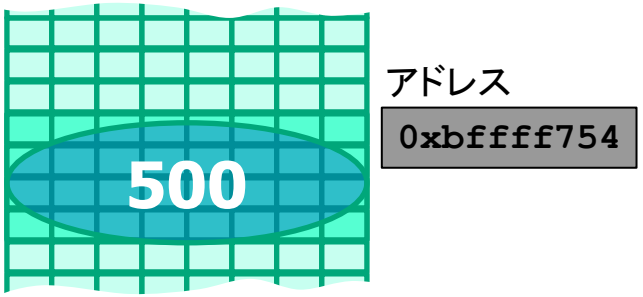
第4回演習課題の内容

C言語によるプログラミング入門④

- ポインタ
  - アドレスと変数
  - ポインタの宣言
  - 間接参照
  - 間接代入
  - ポインタ引数

アドレスと変数





- 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，値を取り出す操作

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のアドレスを出力　`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);
```

変数のアドレス，サイズ，値を調べる

【例題 4-1】

```
#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

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

ポインタの宣言：いくつかの例

- ポインタへのアドレスの代入
  （intへのポインタpに代入できるのは，int型の変数のアドレス）

```
int x = 77;
int *p;

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

```
int *p = &x;
```

- 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;`~~

変数のアドレスとポインタの値を出力

【例題 4-2】

```
#include <stdio.h>
int main(void){
    int x = 77;
    int *p = &x;

    printf("HENSU ¥t ADDRESS ¥t SIZE ¥t VALUE¥n");
    printf("x ¥t %p ¥t %d ¥t %d¥n", &x, sizeof(x), x);
    printf("p ¥t %p ¥t %d ¥t %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．

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

間接参照 :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;
```

ポインタを使った間接参照

【例題 4-3】

- ポインタpの値と間接参照によって参照される変数xの値を使って変数yを更新し，出力する．
- ポインタpの値や変数x,yのアドレスは変わらない．

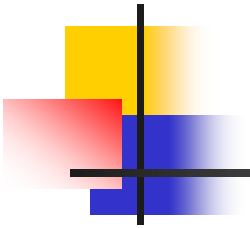
```
#include <stdio.h>
int main(void){
    int x = 77, y = 0;
    int *p = &x;

    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 %d¥n", &y, sizeof(y), y);
    printf("p ¥t %p ¥t %d ¥t %p¥n", &p, sizeof(p), p);

    y = *p +20;
    printf(" y = *p + 20 is executed.¥n");

    printf("x ¥t %p ¥t %d ¥t %d¥n", &x, sizeof(x), x);
    printf("y ¥t %p ¥t %d ¥t %d¥n", &y, sizeof(y), y);
    printf("p ¥t %p ¥t %d ¥t %p¥n", &p, sizeof(p), p);

    return 0;
}
```

ポインタの値を変更

【例題 4-4】

```
#include <stdio.h>
int main(void){
    int x = 77, y = 115;
    int *p;

    printf("HENSU ¥t ADDRESS ¥t VALUE ¥t INDIRECT¥n");
    printf("x ¥t %p ¥t %d¥n", &x, x);
    printf("y ¥t %p ¥t %d¥n", &y, y);

    p = &x;
    printf(" p = &x is executed.¥n");
    printf("p ¥t %p ¥t %p ¥t %d¥n", &p, p, *p);

    p = &y;
    printf(" p = &y is executed.¥n");
    printf("p ¥t %p ¥t %p ¥t %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， p=&yを実行した後，変数yを指すようになる．



【課題 4-3】

- 例題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．

【課題 4-4】

- 例題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．

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

（例）ポインタpが指す変数に10を代入する

```
*p = 10;
```

（注意）

ポインタの宣言でも，間接代入と同じ記号*を使用

＝は代入ではなく，初期値を指定するもの
＝の右側には初期値として設定したいアドレスを書く

（間違いの例） 宣言の時に数値を書くこと

× ~~`int *p = 10;`~~

ポインタを使った間接代入

【例題 4-5】

```
#include <stdio.h>
int main(void){
    int x = 77;
    int *p;

    p = &x;
    printf("HENSU ¥t ADDRESS ¥t VALUE ¥t INDIRECT¥n");
    printf("x ¥t %p ¥t %d¥n", &x, x);
    printf("p ¥t %p ¥t %p ¥t %d¥n", &p, p, *p);

    *p = 15;
    printf(" *p = 15 is executed.¥n");
    printf("p ¥t %p ¥t %p ¥t %d¥n", &p, p, *p);

    return 0;
}
```

- 間接代入によって，ポインタが指す変数の値が変わる．

【課題 4-5】

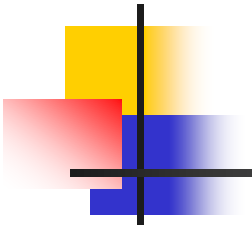
- 例題4-5のプログラムを実行し，動作を確認せよ

注意

- ポインタを使う前にアドレスを代入しておくこと
- ポインタを使う前にアドレスを代入せずに間接参照や間接代入をすると，誤ったメモリ領域を壊してしまう.

```
#include <stdio.h>

int main(void){
    int x = 77;
    int *p;

    printf("ポインタpが示す変数の値は，  %d  です. ¥n", *p);
    *p = 100;
    return 0;
}
```

- 8行目の誤ったメモリ領域（セグメント）への間接代入によって，プログラムが異常終了する.



【課題 4-6】
4つの変数を次のように定義する.

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

ポインタpを使って，変数xとyの値を交換して，次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.

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

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(actual argument):
    関数を呼び出すときに指定した引数
  - 仮引数(formal argument):
    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 ¥t %p ¥t %p¥n", &x, x);
    *x = (*x)*(*x);
}

int main(void){
    int b, x;

    printf("INPUT AN INTEGER  ");
    scanf("%d", &x);

    b = x;
    printf(" ¥t ADDRESS ¥t VALUE¥n");
    printf("Jitsu-Hikisu x ¥t %p ¥t %d¥n", &x, x);
    psquare(&x);
    printf("SQUARE of %d IS %d. ¥n", b, x);

    return 0;
}
```

【課題 4-7】

- 例題4-6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);
```

参考資料

C言語ポインタ完全制覇（技術評論社，ISBN4-7741-1142-2）

第4回演習課題のレポート提出

- 課題4-6，課題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: 2021年5月20日（木） 9:00
- CLEで提出する際のファイル名(半角英数)は下記の通りとする.
  - 課題4-6の場合，XXXX-kadai4-6.c
  - 課題4-8の場合，XXXX-kadai4-8.c
  - XXXXの部分は学籍番号下4桁である.