工学研究部 24 年度 新入生講習

C 言語 - 23 みみ

第四回 ポインタ・配列・文字列

今回やること

- C 言語で一つ目の難関へようこそ
- ポインタを理解
- 配列と「文字の配列」である文字列で遊ぶ
- 本当に難しいので、「なんかわかったかも～」くらいで大丈夫
- 何度も何度も繰り返して自分のものにしてください！
- 難しいので練習問題が少なめです、時間が余ったら先週とかの発展で遊ぶ時間をとるかも

ポインタについて

第一回で変数をやりました。

ポインタを理解するために、まず、変数の実態について深掘りしたいと思います。

コンピュータ上では、変数はメモリに格納されています。

そして、変数の値が格納されているメモリの場所を指し示す「アドレス（住所）」というものが存在します。

ポインタは...

この「アドレス」のことです！！

実践

と言われてもピンとこないと思うので実例を見ながら理解していきましょう。

```c
#include <stdio.h>

int main(void){
  int a = 0;
  int *b = &a;

  printf("aの中身をそのまま: %d\n", a);
  printf("bの中身をそのまま: %x\n", b);
  printf("bが指しているアドレスの値: %d\n", *b);

  return 0;
}
```

出力結果

```
aの中身をそのまま: 0
bの中身をそのまま: cb4e843c
bが指しているアドレスの値: 0
```

何もわからないと思います...
が、何もわからなくて大丈夫です。（~~こんなの最初からわかる人いません~~）

ポインタ変数

アドレスを扱うために、「ポインタ変数」なるものを使用します。

ポインタ変数は、変数名の頭にアスタリスク（ `*` ）をつけます。例えば、整数型のポインタを宣言する場合は、

```
int *ptr;
```

のようにします。

アドレスを取得する

普通の（ポインタ変数ではない）変数のアドレスを取得するには、アンパサンド（`&`）を変数の前につけます。

先程の例を見ると、

```
int a = 0;
int *b = &a;
```

となっていますが、

これは、`b`というポインタ変数に、`&a`で得られる「`a`のアドレス」を代入しています。

…っと、そういえば！！！

scanf 関数使うとき`&`使ったな…

ん、てことはもしかして…

第二引数にはアドレスを渡している！！

```
scanf("%d", &a);
```

そうです、scanf 関数の第二引数には、値を格納したい変数のアドレスを渡す決まりになっていたんです！

少し詳しく説明すると、scanf 関数は、指定されたアドレス（番地）の場所に、標準入力から渡された値を格納してくれる、という関数だったんですね。

アドレスの先の値を参照する

与えられたアドレスの場所に入っている値を取り出したい or 編集したい場合もあると思います。

そんなときは、ポインタ変数の頭にアスタリスク（ `*` ）をつけましょう、アドレスの場所に入っている値を扱うことができます。

```
printf("b: %d\n", *b); // bのアドレスの場所に入っている値を表示
```

```
*b = 3; // bのアドレスの場所の値を「3」に変更
```

ポインタのまとめ

ここまで、ポインタについてかるぅ～く説明しましたが、こんな簡単にわかるはずはないので、

- 苦しんで覚える C 言語

あたりを、何度も手を動かして練習するとだんだん理解できるようになってくると思います。

配列

それでは、次に配列をやっていきます。

Ruby でも配列を使ったと思いますが、C 言語でも扱うことができます。

ここでは、一旦、整数型（ `int` ）の配列を扱ってみましょう。

配列を使って遊んでみます。長さ 10 の配列に奇数を順番に入れていき、それを表示するプログラムです。

```c
#include <stdio.h>

int main(void){
  int a[10];

  for(int i = 0; i < 10; i++){
    a[i] = 2*i - 1;
  }

  for(int i = 0; i < 10; i++){
    printf("%d ", a[i]);
  }
  printf("\n");

  return 0;
}
```

`配列[添字]` で配列の添字番目にアクセスすることができます。

Ruby とほとんど変わりませんね。

添字は 0 から始まることに注意！！

配列に関する主な文法（1/2）

要素数を指定して初期化

```
int a[10];
```

要素を指定して初期化（要素数は勝手に調節してくれる）

```
int a[] = {1, 2, 3, 4, 5};
```

※この記法は初期化のときのみ使用可能

配列に関する主な文法（2/2）

配列の i 番目（$0 \leqq i < n$）に代入

```
a[0] = 3; // 例）0番目に「3」を代入
```

配列の i 番目（$0 \leqq i < n$）を参照

```
printf("%d\n", a[0]) // 例）0番目を表示
```

配列とポインタの関係

ところで、配列とポインタにはふか～い関係があります。

配列の名前のみを渡すと、その配列の 0 番目のアドレスを帰します。

こんな感じです。

```
#include <stdio.h>

int main(void){
  int a[10];
  printf("a[0]のアドレス: %x\n", a);
  return 0;
}
```

このコードを実行すると、配列`a`の 0 番目、すなわち、`a[0]`のアドレスが表示されます。

```
a[0]のアドレス: d1b4dbd0
```

これが今から扱う文字列（文字`char`の配列）と深い関係があったりします。

> ちなみに、アドレスを表示するときに`%x`を使っていますが、これは 16 進数で表記してね、という意味です。
>
> また、実行するたびに値が変わると思いますが、これは毎回違う場所のメモリを使用するため、アドレスが変わってしまうことが理由です。

文字列

さて、皆さんお待ちかね、「文字列」を扱っていきます。

C 言語では文字（`char`）の配列として文字列を扱うようになっています。

こんな感じです。

```
char str[] = "abcdef";
```

表示するときは、以下のよう`%c`ではなく、`%s`を使い、`char`配列の先頭のポインタを渡します。

```
printf("%s\n", str);
```

scanf 関数で文字列を入力できるようにしてみる

ここまで得た知識をフル回転させます。

とりあえず以下のコードを書いてみてください。

```
#include <stdio.h>

int main(void){
  char str[256];
  printf("文字列を入力 > ");
  scanf("%s", str);

  printf("入力された文字列は: %s\n", str);
  return 0;
}
```

それでは一つづつ解説していきます。

`char` 配列の初期化

```
char str[256];
```

256 のサイズで `char` 型の配列を作ります。

先程は言いませんでしたが、終端文字 `\0` というものが文字列の最後には含まれるので、この配列で扱える最大の文字数は 255 文字となります。

また、日本語は「マルチバイト文字」といい、2 文字分使用するため、扱える文字数は半分になります（が、今回はあんまり気にする必要はない）。

標準入力から読み取る

```
scanf("%s", str);
```

おや、`&`がついていませんね。なぜでしょうか…？

先ほど、配列のところで取り上げたように、配列の名前を渡すと、その先頭のアドレスを帰してくれるんでした。

つまり、これは`str`という配列の先頭のアドレスを`scanf`関数に渡している、ということですね。

第二回で言った、`&`がいらない場合とは、こういうときのことだった、というわけです。

標準出力に表示

```
printf("入力された文字列は: %s\n", str);
```

表示するときは、先ほど紹介したように、`%c`ではなく`%s`を使い、`char`配列の先頭のポインタを渡します。

実行結果

```
文字列を入力 > ほげ
入力された文字列は: ほげ
```

うまく文字列を扱うことができましたね！

練習

それでは最後に練習問題です。

前回作った数あてゲームに、

- 最初に名前を入れてもらう
- 正解したら、「〇〇さんはすごいですね！」と褒める

機能を追加してみてください。

第四回 おわり

次回予告 : なにをやろうか...（未定）