# プログラミング演習
# ～ポインタ～

## 1　目的

C 言語におけるポインタの使い方を理解する．代表的な使用方法としては

- 動的配列の実現
- 関数の多出力化

などがあるが，今回は特に動的配列の実現を通してポインタを学習する．またバンディット問題 [1] 等を例題にしながら実践し理解を深める．

## 2　製作対象 ポインタの理解

int 型は整数を記憶できる変数，float 型や double 型は実数を記憶できる変数，そして char 型は文字 (1 文字) を記憶できる変数である．これらに対してポインタは，アドレス値を記憶できる変数である．

ここでアドレス値の説明を簡単に行う．変数のみならずプログラム等コンピュータの様々な情報はコンピュータのメモリ上に記憶されている．このメモリ上に記憶されている情報を自由に記憶したり取り出したりして使うためには，”どこに”必要な情報が記憶されているか分かる必要がある (図 1)．そのために，メモリには”住所 (番地，アドレス)”が割り振られており，○×のデータは△□という場所にある，という住民登録表のようなものがある．プログラム中で int 型の変数を宣言すると，OS によって自動的に空いているアドレスにその変数用の領域が確保される仕組みとなっている．

ポインタ変数は，この”アドレス”という数値を記憶できる変数である．例えば

- int* a;
- double* b;

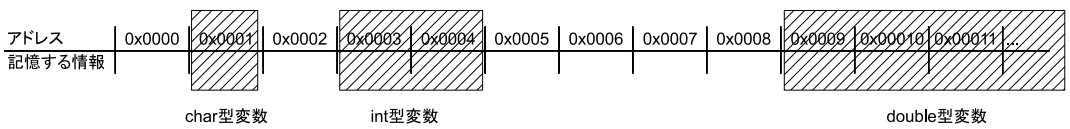


図 1: メモリ空間表記-アドレスと値-

の変数である a，b はアドレスを記憶できる変数である．ここで，どちらもアドレスを記憶できるのだが，int 型のポインタ，double 型のポインタという違いがある．これは，int 型と double 型の変数が必要とするサイズが異なることから同じアドレス値であっても区別する必要が出てくる場合が考えられるからである．

ここで，ポインタが記憶するアドレス値は，プログラム中で

- a = 0x001;　(アドレス表記では数値の前に 0x をつける)

と書いても代入する事が可能だが，一般的ではない．最も簡単かつ一般的な使い方としては，他の変数のアドレスを記憶することである．2.2 プログラムでは，代入が 15 行目および 17 行目であり，表示が 20 行目および 21 行目である．このように，変数の前に&をつけるとその変数のアドレスとなる．

## 2.1 準備

ディレクトリ ”programming07”を作成する．今回の演習では，プログラムの作成や必要ファイルのダウンロードは，programming07 ディレクトリで行う．

## 2.2 プログラム

```
1  /*********************************
2          ポインタの基礎 1
3  *********************************/
4
5  #include <stdio.h>
6
7  int main(){
8    int i;
9    int* j;
10   double d;
11   double* e;
12
13   /* 値の代入 */
14   i=13;
15   j=&i;
16   d=3.141592;
17   e=&d;
18
19   /* 値の出力 */
20   printf("i=%d, j=%p\n",i,j);
21   printf("d=%lf, e=%p\n", d,e);
22
23   return 0;
```

```
24  }
```

### 2.3 コンパイル

上記のプログラムを作成し，ファイル名 pointer-test1.c として保存してコンパイルを行う．コンパイル後の名前を ptest1 とする．

> gcc -o ptest1 pointer-test1.c

### 2.4 動作実験

コンパイルしたプログラムを実行し，出力を見て”アドレス”が記憶されていることを確認する．ポインタの出力では”0x”がついていないので数値データと間違えないようにする．

## 3 製作対象 ポインタの使い方-基礎編-

2 ポインタの理解ではポインタが他の変数と同じ変数であり，代入・出力ができることを見た．次に，このようなポインタがどのように使えるかについて述べる．その便利な使い方・実用的な使い方については動的配列の実現・関数の多出力化 [3] などで述べる．

3.1 プログラムで，ポインタの値の代入は 13 行目で行っており，i のアドレス値を記憶している．この状態でポインタの使い方の説明をする．

ポインタは，値 (アドレス値) をそのまま使用することは少ない．基本的な使い方は，*を変数の前につけて使う．ポインタが記憶しているアドレスを address とすると，*を変数の前につけると address の場所に記憶している情報そのものを表す．この場合，j=&i でポインタ j が i のアドレスを記憶していると，*j は i のアドレスに記憶している情報，つまり i そのものを表す．よって，21 行目以降のように，i に対して演算を行っても，*j に対して演算を行っても i と*j は同じ値となる．

以上より，使い方としては，変数の前に&が付く場合となにも付かない場合，そしてポインタ変数特有として変数の前に*が付く場合について述べた．ポインタの説明としては，これですべてであり，これをもとにして様々な使い方がある．配列との関係も含めると図 2 のようになる．ここで，配列との関係について少々述べておく．配列の演習 [2] での配列は

- double array[10]; ← 定義
- array[3] = 25.3; ← 使用

であり，”[”と”]”でくくった番号が要素番号であった．この配列の名前 (ここでは array) は，実は特殊なポインタ変数であり，先頭要素のアドレス (array[0] のアドレス，&array[0]) を保持しているものである．

### 3.1 プログラム

```
1   /*********************************
2             ポインタの基礎２
3   *********************************/
4
```

```
5  #include <stdio.h>
6
7  int main(){
8    int i;
9    int* j;
10
11   /* 値の代入 */
12   i=13;
13   j=&i;
14
15   /* 値の出力 */
16   printf("    アドレス: &i=%p, &j=%p\n", &i, &j);
17   printf("          値: i=%d, j=%p\n", i, j);
18   printf("ポインタ使用: *i=なし, *j=%d\n", *j);
19
20   /* ポインタの特性 j=&i で *j と i と同じ物になる */
21   printf("i=%d, *j=%d\n",i,*j);
22   i=i+2;
23   printf("i=%d, *j=%d\n",i,*j);
24   *j=*j*5;
25   printf("i=%d, *j=%d\n",i,*j);
26
27   return 0;
28 }
```

### 3.2 コンパイル

上記のプログラムを作成し，ファイル名 pointer-test2.c として保存してコンパイルを行う．コンパイル後の名前を ptest2 とする．

> gcc -o ptest2 pointer-test2.c

| 具体例 | 接頭語 / 意味 | * | なし | & |
|---|---|---|---|---|
| int a; など通常変数 (a=3の場合) | 変数 | | 変数の記憶する値 (この場合整数値 3) | 変数のアドレス |
| int* b;などポインタ変数 (b=&aの場合) | ポインタ変数 | 変数の記憶するアドレスが記憶する情報 | 変数の記憶する値 (この場合アドレス値 &a) | 変数のアドレス |
| 配列の名前 (int c[10]の場合のc) | ポインタ変数と同等 | 変数の記憶するアドレスが記憶する情報 (配列の先頭c[0]となる) | 変数の記憶する値 (この場合配列の先頭アドレス値) | 変数のアドレス |
| 配列の要素 (int c[10]の場合のc[0]) | 変数と同等 | | 変数の記憶する値 (この場合整数値c[0]の値) | 変数のアドレス |

図 2: 変数の接頭語と変数・ポインタ・配列の関係

### 3.3 動作実験

コンパイルしたプログラムを実行し，ポインタの理解を深める．

### 3.4 調べてみよう

配列とポインタの関係について調べてみよう．

### 3.5 やってみよう

double 型についても 3.1 プログラムと同様のプログラムを作成し，四則演算を行ってみよう．

## 4 製作対象 ポインタの使い方-動的配列-

配列の演習 [2] では，配列の使い方を学んだ．その配列の制約として，事前に何個の記憶領域を持つか (要素の数，double reward[x]; の x の数) を決めておかなければならなかった．配列の演習 [2] では，バンディットからの報酬を配列 reward に記憶していたが，double reward[10]; と宣言すると 10 個の報酬しか記憶できなかった．

ここで，while 文や for 文に工夫を凝らし，好きなだけバンディットを試行 (プレイ) し，好きなだけバンディットからの報酬を記憶することを考える．具体的に考えてみると，例えば最初にユーザから試行回数を入力してもらい，その回数だけバンディットを試行することを考える。すると，プログラムの実行毎に配列の大きさを，プログラムの実行毎に変えなければならないが，配列だけでは実現は不可能である．

そこでポインタを用いて，プログラムの実行中に自由な大きさの配列 (と同等なもの) を定義・使用する方法を説明する．

int 型の配列と同等なポインタを取得する方法を次に示す．他の型の場合は，int を他の型に呼び変えてみること．

- int n; /* 要素数 */
- int* a;
- n=3; /* 要素数 3 とする */
- a = malloc(sizeof(int)*n);
- a[0]=10, a[1]=14,a[2]=20;
- free(a);

この為には，malloc 関数を用いる．malloc 関数は，指定されたサイズだけ変数用にメモリ領域を確保するものである．また，sizeof 関数は大正となる型のサイズを知るためのものである．sizeof(int) で int 型のサイズを取得し，それに要素の数だけかけてやると，結果として int 型の変数を 3 個 (n 個) 記憶するだけのメモリを確保できる．

使用方法は，配列と同様に，”[と”] を用いることができる．また，確保した領域は，自動的に解放されないので手動で解放する必要がある．これが free 関数である．

## 4.1 プログラム

```
1  /*******************************
2             動的配列
3  *******************************/
4
5  #include <stdio.h>
6  #include <stdlib.h>
7
8  int main(){
9    int i, tmp, loop;
10   int* info;
11
12   printf("記憶する整数の個数を入力:");
13   scanf("%d",&loop);
14
15   /* loop 個の配列を取得 */
16   info = malloc(sizeof(int)*loop);
17
18   /* 配列に入れてみる */
19   for(i=0 ; i<loop ; i++){
20     printf("%d 個目の記憶:数を入力してください:",i+1);
21     scanf("%d", &tmp);
22     info[i]=tmp;
23   }
24
25   /* 配列を出力 */
26   printf("記憶した数の羅列\n");
27   for(i=0 ; i<loop ; i++){
28     printf("info[%d]:%d\n",i, info[i]);
29   }
30
31   free(info);
32   return 0;
33 }
```

## 4.2 コンパイル

上記のプログラムを作成し，ファイル名 pointer-test3.c として保存してコンパイルを行う．コンパイル後の名前を ptest3 とする．

> gcc -o ptest3 pointer-test3.c

## 4.3 動作実験

コンパイルしたプログラムを実行し，ポインタの理解を深める．

## 4.4 調べてみよう

動的配列をとる関数 malloc について詳細に調べてみよう．

## 4.5 やってみよう

### 4.5.1 n 回繰り返しバンディットプログラム

配列の演習 [2] での 2.2 プログラムでは，配列を用いたバンディットプログラムを作成した．これを拡張し，バンディットを 10 回繰り返すものから，ユーザに繰り返し回数を問うプログラムを考える．4.1 プログラムを参考にし，最初に繰り返し回数を入力させるようにし，その回数だけバンディットを試行する．そして総スコア・平均スコアを出力するプログラムを作成してみよう．

プログラムの概要は以下のようになる．

- ユーザに繰り返し回数 loop を入力してもらう．
- 繰り返し命令文で loop 回繰り返し
  - 腕選択
  - バンディット試行
- 総スコア・平均スコア出力

### 4.5.2 未定回数繰り返しバンディットプログラム

4.5.1 を更に拡張し，事前に繰り返し回数を決めないバンディットプログラムを考える．この場合，バンディット繰り返し部分となる for 文または while 文の中で，”終了しますか?(Y/N):”とユーザに問い，終了するまで繰り返し，最後に総スコア・平均スコアを出力するプログラムを作成してみよう．

プログラムの概要は以下のようになる．

- 繰り返し命令文　ユーザから終了の合図があるまで
  - 腕選択
  - バンディット試行
  - 終了しますか?Y/N
- 総スコア・平均スコア出力

# 5　参考文献

## 参考文献

[1] プログラミング演習～ガイダンス～

[2] プログラミング演習～配列～

[3] プログラミング演習～関数～