ポインタ変数Ｉ

０．目次

１．ポインタ変数と変数

ポインタ変数には、記憶領域の番地が格納されている。通常の変数にはデータが格納されている。

宣言	意味
int *a;	ポインタ変数aは、整数型データが保存されている番地を格納している。
float *b;	ポインタ変数bは、実数型データが保存されている番地を格納している。
char *c;	ポインタ変数cは、文字型データが保存されている番地を格納している。

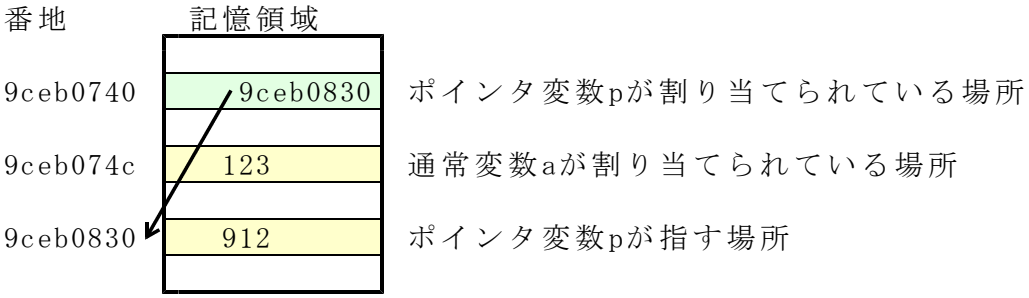
●プログラム（ptr111.c）

```
 1  /* << ptr111.c>> */
 2  #include <stdio.h>
 3  int main () {
 4    int a;  /* 変数aを整数型とする。*/
 5    int *p; /* ポインタ変数pが指すデータは整数である。*/
 6    a = 123;
 7    *p = 456;
 8    *p = *p*2;
 9    printf("通常変数aの記憶場所      = %x¥n",&a);
10    printf("通常変数aの値            = %d¥n",a);
11    printf("ポインタ変数pの記憶場所  = %x¥n",&p);
12    printf("ポインタ変数pの値        = %x¥n",p);
13    printf("ポインタ変数pの指す値    = %d¥n",*p);
14  }
```

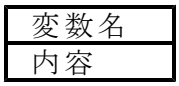
実行結果

```
% cc ptr111.c
% ./a.out
通常変数aの記憶場所      = 9ceb074c
通常変数aの値            = 123
ポインタ変数pの記憶場所  = 9ceb0740
ポインタ変数pの値        = 9ceb0830
ポインタ変数pの指す値    = 912
```

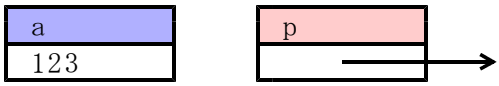
番地は、実行の都度変わる。



＆は変数に割り当てられた番地を求める演算子である。
＊はﾎﾟｲﾝﾀ変数が保持する番地の内容を求める演算子である。



左図は、変数名とその内容を意味する。
青色は通常の変数、赤色はポインタ変数を意味する。
ポインタ変数に保存されている番地は、矢印で表す。



２．ポインタ変数と配列

配列の実体はポインタである。
配列名がポインタ変数となる。
x[0]と*(x+0)、x[1]と*(x+1)、x[2]と*(x+2) は同等である。

●プログラム（ptr211.c）

```
 1  /* << ptr211.c>> */
 2  #include <stdio.h>
 3  int main () {
 4    int x[3]; /* 配列xを整数型とする。*/
 5    x[0] = 123;
 6    x[1] = 456;
 7    x[2] = 789;
 8    /* 配列的記述。*/
 9    printf("x[0] = %d¥n",x[0]);
10    printf("x[1] = %d¥n",x[1]);
11    printf("x[2] = %d¥n",x[2]);
12    /* ポインタ的記述。*/
13    printf("*(x+0) = %d¥n",*(x+0));
14    printf("*(x+1) = %d¥n",*(x+1));
15    printf("*(x+2) = %d¥n",*(x+2));
16  }
```

実行結果

```
% cc ptr211.c
% a.out
x[0] = 123
x[1] = 456
x[2] = 789
*(x+0) = 123
*(x+1) = 456
*(x+2) = 789
```

配列名を他のポインタ変数に代入することができる。

●プログラム（ptr221.c）

```
1  /* << ptr221.c>> */
2  #include <stdio.h>
3  int main () {
4    int x[3]; /* 配列xを整数型とする。*/
5    int *y;
6    x[0] = 123;
7    x[1] = 456;
8    x[2] = 789;
9    y = x;
10   /* 配列的記述。*/
11   printf("y[0] = %d¥n",y[0]);
12   printf("y[1] = %d¥n",y[1]);
13   printf("y[2] = %d¥n",y[2]);
14   /* ポインタ的記述。*/
15   printf("*(y+0) = %d¥n",*(y+0));
16   printf("*(y+1) = %d¥n",*(y+1));
17   printf("*(y+2) = %d¥n",*(y+2));
18 }
```

実行結果

```
% cc ptr221.c
% a.out
y[0] = 123
y[1] = 456
y[2] = 789
*(y+0) = 123
*(y+1) = 456
*(y+2) = 789
```

配列名xはポインタ変数であるが他のポインタ変数から代入はできない。

●プログラム（ptr231.c）

```
1  /* << ptr231.c >> */
2  #include <stdio.h>
3  int main() {
4    int x[3],*y;
5    y = x;  /* OK */
6    x = y;  /* エラー */
7  }
```

実行結果

```
% cc ptr231.c
pointer23.c: In function ‘main’:
pointer23.c:6: error: incompatible types when assigning to type
  ‘int[3]’ from type ‘int *’
```

３．ポインタ変数と構造体

構造体は、複数のデータをまとめて扱う。構造体の定義は次のように行う。

```
struct 構造体名 {
  メンバー１;
  メンバー２;
  …
  メンバーｎ;
};
struct 構造体名 変数名;
```

新たなデータ型を定義したと考えられる。

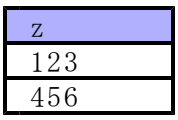
構造体を変数で宣言した場合、宣言時に記憶領域が確保される。
メンバーを指定するのに、演算子.を使う。

●プログラム（ptr311.c）

```
/* << ptr311.c >> */
#include <stdio.h>
int main () {
  struct complex {
    int rpart;
    int ipart;
  };
①  struct complex z;
②  z.rpart = 123;
③  z.ipart = 456;
  printf("rpart:%d ipart:%d¥n",z.rpart,z.ipart);
}
```

実行結果

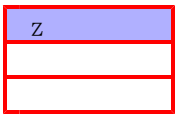
```
rpart:123 ipart:456
```



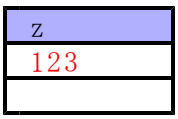
●動作

① struct complex z;

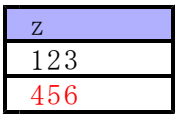
記憶領域が確保される。



② z.rpart = 123;



③ z.ipart = 456;



構造体をポインタ変数で宣言した場合、記憶領域は確保されないので、自分で確保する必要がある。メンバ指定には、演算子->を使う。

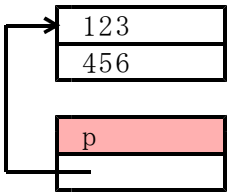
●プログラム（ptr321.c）

```
/* << ptr321.c >> */
#include <stdio.h>
#include <stdlib.h>
int main () {
  struct complex {
    int rpart;
    int ipart;
  };
  struct complex *p;

①  p = (struct complex *)malloc(sizeof(struct complex));
②  p->rpart = 123;
③  p->ipart = 456;
④  printf("rpart:%d ipart:%d¥n",p->rpart,p->ipart);
   free(p);
⑤  printf("rpart:%d ipart:%d¥n",p->rpart,p->ipart);
}
```

実行結果

```
rpart:123 ipart:456
rpart:0 ipart:0
```



	意味
sizeof(型名)	指定した型名に必要なバイト数を返す。
malloc(サイズ)	サイズで指定された大きさの記憶領域が確保され先頭番地が返される。

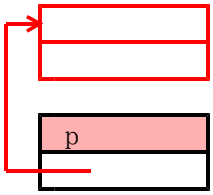
（注意１）sizeofは演算子で、mallocは関数である。
（注意２）(struct complex *)をキャストという。
（注意３）確保した記憶領域を開放するのに、free関数を使う。
（注意４）malloc関数、free関数を使うときは、標準ヘッダファイルstdlib.hをインクルードしておく必要がある。

●動作

① `struct complex *p;`

構造体のデータを格納する記憶領域は確保されていない。

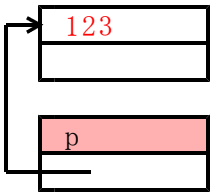
② `p = (struct complex *)malloc(sizeof(struct complex));`



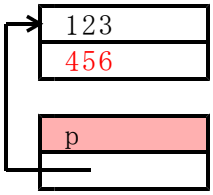
構造体のデータを格納する記憶領域が確保された。

`sizeof(struct complex)`で構造体complexの大きさが求められ、mallocでその大きさの記憶領域が確保され、先頭番地が返されてくる。`(struct complex *)`で、返されてきた先頭番地が構造体complexの境界に合うように変換される。

③ `p->rpart = 123;`



④ `p->ipart = 456;`



⑤ `free(p);`

４．ポインタ変数と線形リスト

一連のデータをポインタでつなぎ合わせたものを線形リストという。

４．１　線形リストの作成と表示

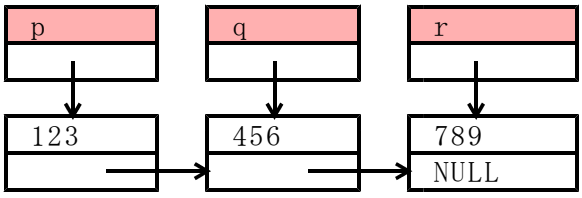
●プログラム（ptr411.c）

```
/* << ptr411.c >> */
#include <stdio.h>
#include <stdlib.h>
int main () {
  /* 構造体の定義。*/
  struct NODE {
    int info;
    struct NODE *next; /* nextは、ポインタ変数。*/
  };
  /* 構造体の宣言。*/
①  struct NODE *p,*q,*r,*s;

②  p = (struct NODE *) malloc(sizeof(struct NODE));
③  p->info = 123;
④  q = (struct NODE *) malloc(sizeof(struct NODE));
⑤  q->info = 456; p->next = q;
⑥  r = (struct NODE *) malloc(sizeof(struct NODE));
⑦  r->info = 789; q->next = r;
⑧  r->next = NULL;
  /* 線形リストをたどる。*/
  s = p;
  while( s != NULL ) {
    printf("%d¥n",s->info);
    s = s->next;
  }
⑨  free(p); free(q); free(r);
}
```

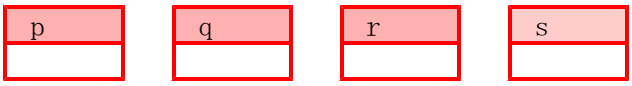
実行結果

```
123
456
789
```

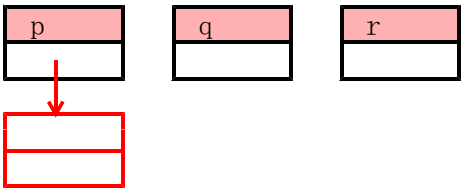


●動作

① struct NODE *p,*q,*r,*s;

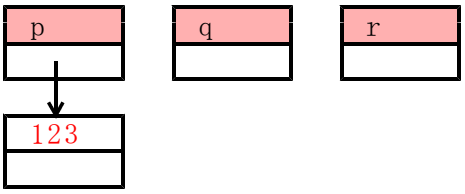


② p = (struct NODE *) malloc(sizeof(struct NODE));

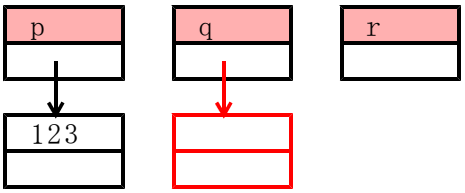


sizeof(struct NODE)で構造体NODEの大きさが求められ、mallocでその大きさの記憶領域が確保され、先頭番地が返されてくる。(struct NODE *)で、返されてきた先頭番地が構造体NODEの境界に合うように変換される。

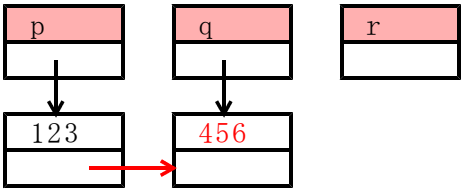
③ p->info = 123;



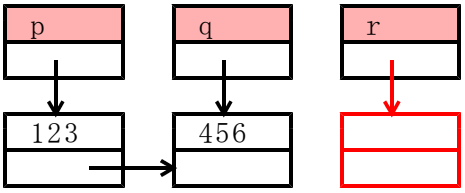
④ q = (struct NODE *) malloc(sizeof(struct NODE));



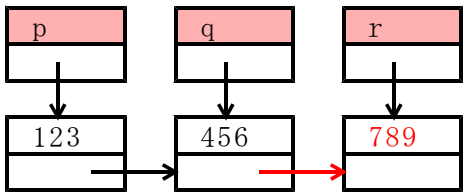
⑤ q->info = 456; p->next = q;



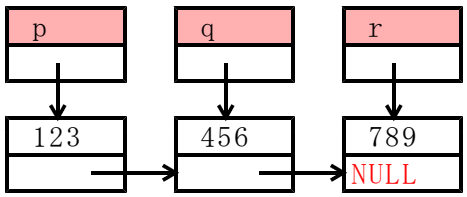
⑥ r = (struct NODE *) malloc(sizeof(struct NODE));



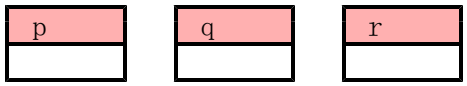
⑦ r->info = 789; q->next = r;



⑧ r->next = NULL;



⑨ free(p); free(q); free(r);



５．問題

問題１

１人の学生について、３科目の成績（国語、数学、英語）をもつとき、
つぎの構造体で表現する。

```
1  struct student {
2    int kokugo;
3    int suugaku;
4    int eigo;
5  }
```

１人分の成績データを読み込んだ後、平均値を出力するプログラムを作成せよ。
ただし、構造体とポインタ変数を使うこと。

```
1  /* << ptr511.c >> */
2  #include <stdio.h>
3  #include <stdlib.h>
4  int main () {
5    int sum;
6    struct student {
7      int kokugo;
8      int suugaku;
9      int eigo;
10   };
11   struct student *p;
12   p = (struct student *)malloc(sizeof(struct student));
13   scanf("%d%d%d",&p->kokugo,
14                  &p->suugaku,
15                  ①                );
16   sum = p->kokugo + p->suugaku + ②          ;
17   printf("%f¥n",sum/3.0);
18 }
```

実行結果

```
% cc ptr511.c
% a.out
10 20 30
20.000000
```

問題 2

学生数n(最大100名)を読み込んだ後、国語、数学、英語の平均値を出力するプログラムを作成せよ。ただし、構造体とポインタ変数を使うこと。

```
 1  /* << ptr521.c >> */
 2  #include <stdio.h>
 3  #include <stdlib.h>
 4  int main () {
 5    int i,n,sum1,sum2,sum3;
 6    struct student {
 7      int kokugo;
 8      int suugaku;
 9      int eigo;
10    };
11    struct student *p[100];
12    scanf("%d",&n);
13    for( i=0; i<n; i++ ) {
14      p[i] = (struct student *)malloc(sizeof(struct student));
15      scanf("%d%d%d",&p[i]->kokugo,&p[i]->suugaku,
16                     ③                  );
17    }
18    sum1 = 0; sum2 = 0; sum3 = 0;
19    for( i=0; i<n; i++ ) {
20      sum1 = sum1 + p[i]->kokugo;
21      sum2 = sum2 + p[i]->suugaku;
22      sum3 = sum3 + ④                  ;
23    }
24    printf("国語の平均 : %f¥n",sum1/(float)n);
25    printf("数学の平均 : %f¥n",sum2/(float)n);
26    printf("英語の平均 : %f¥n",sum3/(float)n);
27  }
```

実行結果

```
% cc ptr521.c
% a.out
3
11 31 51
12 32 52
13 33 53
国語の平均 : 12.000000
数学の平均 : 32.000000
英語の平均 : 52.000000
```