

第 14 の解答

【確認問題 14_1】

呼び出す関数へ渡すもの 変数のある場所 (例 &変数名)
呼び出された関数の受け取り方 . . . 変数のある場所を格納するポインタ

【確認問題 14_2】

*(配列へのポインタ + 要素番号)

【確認問題 14_3】

struct DATA *

【確認問題 14_4】

```
char *kKakunoBasyo(char *hairetuP)
{
    return hairetuP + 2;
}
```

【練習問題 14_1】

●理由

関数kNibaiから、main関数の変数sutiの値を書き換えられないためです。

つまり、関数kNibaiのみで使える引数sutiを2倍していますが、main関数の変数sutiに値を格納していないためです。

●修正した箇所をアンダラインで示します。

```
#include <stdio.h>
```

```
void kNibai(int *);          //関数のプロトタイプ宣言
```

```
int main(void)
```

```
{
```

```
    int sut_i= 100;
```

```
    kNibai(&suti);
```

```
//関数の呼び出し
```

```
    printf("%d¥n", sut_i);
```

```
//変数sutiが格納している値を表示
```

```
    return 0;
```

```
}
```

```
//関数kNibaiの処理
```

```
void kNibai(int *sut_i)
```

```
{
```

```
    *sut_i *= 2;
```

```
}
```

【練習問題14_2】

```
#include <stdio.h>
```

```
double kWarizan(int, int);          //関数のプロトタイプ宣言
```

```
int main(void)
```

```
{
```

```
    int suti1 = 10;
```

```
    int suti2 = 4;
```

```
    double kekka;
```

```
    kekka = kWarizan(suti1, suti2); //関数の呼び出し
```

```
    printf("%f¥n", kekka);          //変数kekkaが格納している値を表示
```

```
    return 0;
```

```
}
```

```
//関数kWarizanの処理
```

```
double kWarizan(int suti1, int suti2)
```

```
{
```

```
    double kekka;
```

```
    kekka = (double)suti1 / (double)suti2;
```

```
    return kekka;
```

```
}
```

【練習問題14_3】

```
#include <stdio.h>
```

```
void kKokan(int *, int *);          //関数のプロトタイプ宣言
```

```
int main(void)
```

```
{
```

```
    int suti1 = 100;
```

```
    int suti2 = 200;
```

```
    kKokan(&suti1, &suti2);
```

```
    printf("suti1は%d、suti2は%dです¥n", suti1, suti2);
```

```
    return 0;
```

```
}
```

```
//関数kKokanの処理
```

```
void kKokan(int *suti1, int *suti2)
```

```
{
```

```
    int taihi;
```

```
    taihi = *suti1;
```

```
    *suti1 = *suti2;
```

```
    *suti2 = taihi;
```

```
}
```