第12回
プログラミング及び実習 II

第11回課題の解答例

ex11-1：引数のある関数

```
#include <stdio.h>

void prnum(int);

int main(void) {
  int i;

  for (i=0; i<10; ++i)
    prnum(i);

  return 0;
}

void prnum(int num) {
    printf("%dです。\n", num);
    return;
}
```


ex11-2：2つの引数がある関数

```
#include <stdio.h>

int primef(int);

int main() {
    int n;
    scanf("%d", &n);

    if (n <= 1 || primef(n) == 0)
        printf("%dは素数ではありません\n", n);
    else
        printf("%dは素数です\n", n);
}

int primef(int n) {
    int found = 0;
    int k = 2;
    while (k < n) {
        if (n % k == 0)
            return 0;
        k = k+1;
    }
    return 1;
}
```

ex11-3：素数判定の関数

```
#include <stdio.h>

int primef(int);

int main() {
    int n;
    scanf("%d", &n);

    if (n <= 1 || primef(n) == 0)
        printf("%dは素数ではありません\n", n);
    else
        printf("%dは素数です\n", n);
}

int primef(int n) {
    int found = 0;
    int k = 2;
    while (k < n) {
        if (n % k == 0)
            return 0;
        k = k+1;
    }
    return 1;
}
```


ex11-4：配列変数の平均値

```
#include <stdio.h>

double average(int[], int);

int main() {
    int n;
    scanf("%d", &n);
    if (n <= 0) return 1;
    int a[n];
    for (int i=0; i<n; i++)
        scanf("%d", &a[i]);

    printf("%.3f\n", average(a, n));
    return 0;
}

double average(int a[], int n) {
    double sum = 0.;
    for (int i=0; i<n; i++)
        sum += a[i];
    return sum/n;
}
```


ex11-5：アルファベットの大文字小文字変換

```
#include <stdio.h>

char trcase(char);

int main() {
    char c;
    do {
        scanf("%c", &c);
        printf("%c",trcase(c));
    } while (('A'<=c && c<='Z') || ('a'<=c && c<='z'));

    return 0;
}

char trcase(char c) {
    if ('A'<=c && c<='Z')
        return c+('a'-'A');
    else if ('a'<=c && c<='z')
        return c-('a'-'A');
    else
        return c;
}
```


ex11-6：最頻値

```
#include <stdio.h>
#define NMAX 100

int mvalue(int[], int);

int main() {
    int n;
    scanf("%d", &n);
    if (n <= 0) return 1;

    int a[n];
    for (int i=0; i<n; i++)
        scanf("%d", &a[i]);

    printf("%d\n", mvalue(a, n));
    return 0;
}

int mvalue(int a[], int n) {
    int cnt[NMAX+1] = {0};
    int max = cnt[a[0]] = 1;
    int value = a[0];

    for (int i=1; i<n; i++) {
        cnt[a[i]]++;
        if (cnt[a[i]] >= max) {
            max = cnt[a[i]];
            value = a[i];
        }
    }

    return value;
}
```

ex11-7：4つの整数の最大値

```
#include <stdio.h>

double max(double, double);
double max4(double, double, double, double);

int main() {
    double a, b, c, d;
    do {
        scanf("%lf%lf%lf%lf", &a, &b, &c, &d);
        printf("%f\n", max4(a,b,c,d));
    } while (!(a==b && b==c && c==d));

    return 0;
}

double max(double a, double b) {
    return a>b ? a : b;
}
double max4(double a, double b, double c, double d) {
    double max1 = max(a, b);
    double max2 = max(c, d);
    return max1>max2 ? max1 : max2;
}
```


ex11-8：2科目の相関係数

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

double average(int[], int);
double corr(int[], int[], int);

int main() {
  int n;
  do {
    scanf("%d", &n);
  } while(n<=0);

  int a[n], b[n];
  for (int i=0; i<n; i++)
    scanf("%d%d", &a[i], &b[i]);

  printf("%.3f\n", corr(a, b, n));
  return 0;
}
```

```
double average(int a[], int n) {
    double sum = 0.;
    for (int i=0; i<n; i++)
        sum += a[i];
    return sum/n;
}

double corr(int a[], int b[], int n) {
    double avg_a = average(a, n);
    double avg_b = average(b, n);

    double cov, sd_a, sd_b;
    cov = sd_a = sd_b = 0.0;
    for (int i=0; i<n; i++) {
        cov += (a[i]-avg_a)*(b[i]-avg_b);
        sd_a += (a[i]-avg_a)*(a[i]-avg_a);
        sd_b += (b[i]-avg_b)*(b[i]-avg_b);
    }
    return cov/(sqrt(sd_a)*sqrt(sd_b));
}
```


第12回の内容

変数のスコープ

ローカル変数とグローバル変数

関数引数として渡された配列変数は変更できる

変数のスコープ

変数のスコープ ＝ 変数の有効範囲

変数は「いつ生成されて」「いつ破棄されるか」
＝ プログラムのどこで参照・代入できるか

変数が宣言されてから、宣言されたブロック{ } が閉じるまで

```
#include <stdio.h>

int main(void) {
  printf("a=%d\n", a);

  int a = 1;

  printf("a=%d\n", a);
  return 0;
}
```

変数 a が宣言されたブロック

変数 a の宣言

変数 a のスコープ（有効範

変数 a のスコープ外なのでエラー

スコープは変数ごとに決まる

変数が宣言されてから、宣言されたブロック{ } が閉じるまで

```
#include <stdio.h>

int main(void){
  int a = 1;
  printf("a=%d\n", a);

  if (a == 1) {
    int b = 10;
    printf("a=%d\n", ++a);
    printf("b=%d\n", b);
  }

  printf("a=%d\n", ++a);
  printf("b=%d\n", b);
  return 0;
}
```

変数 a の宣言

変数 b の宣言

変数 b のスコープ外なのでエラー

同じ名前で異なるスコープの変数

同名の変数もスコープが異なれば別の変数（関数もスコープを作る）

```
#include <stdio.h>

void inc(int);

int main(void){
  int a = 1;
  printf("a=%d\n", a);
  inc(a);
  printf("a=%d\n", a);
  return 0;
}

void inc(int a) {
  a++;
}
```

main()関数の a のスコープ

a=1

a=1

inc()関数の a のスコープ

main() から inc(1)で呼ばれて a=2 となるがスコープが終わって捨てられる

同じ名前で重複するスコープの変数

同名の変数は内側スコープの変数として扱われる

```
#include <stdio.h>

int main(void){
  int a = 1;
  printf("a=%d\n", a);

  if (a == 1) {
    int a = 10;
    printf("a=%d\n", ++a);
    {
        int a = 100;
        printf("a=%d\n", a);
    }
    printf("a=%d\n", ++a);
  }

  printf("a=%d\n", ++a);
  return 0;
}
```

外層の変数 a のスコープ

中層の変数 a のスコープ

内層の変数 a のスコープ

第12回の内容

変数のスコープ

ローカル変数とグローバル変数

関数引数として渡された配列変数は変更できる

ローカル変数とグローバル変数

ローカル変数 = ブロックによって制限された有効範囲をもつ
これまで扱ってきた変数はすべてローカル変数

グローバル変数 = すべてのスコープからアクセス可能
一見、便利そうだが、どこからでも変更できるのでバグの温床になりやすい（ご利用は計画的に）

```
#include <stdio.h>
void inc(void);
int a = 1;

int main(void) {
  printf("a=%d\n", a);
  a += 2;
  inc();
  printf("a=%d\n", a);
  return 0;
}

void inc(void) {
  a += 3;
}
```

グローバル変数 a の宣言

a=1

a=6

第12回の内容

変数のスコープ

ローカル変数とグローバル変数

関数引数として渡された配列変数は変更できる

配列変数の引数操作は呼び出し元に反映される

```
#include <stdio.h>

void s(int);

int main(void) {
  int a = 3;
  s(a);
  printf("%d\n", a);
  // 3 (4 ではない！)
  return 0;
}

void s(int a) {
  a++;
  return;
}
```

```
#include <stdio.h>

void s(int[], int);

int main(void) {
  int a[] = {3}; // a[0] = 3
  s(a, 1);
  printf("%d\n", a[0]);
  // 4 (3 ではない！)
  return 0;
}

void s(int a[], int n) {
  a[0]++;
  return;
}
```

一般変数の関数引数は、値渡し（値がコピーされるだけで関数ごと別の変数）

配列変数の関数引数は、配列変数自身が渡される（ポインタ渡し）

よって、関数引数として渡された配列変数は、関数内での変更がそのまま呼び出し元に反映される！！

関数引数の配列変数はその変更が呼び出し元に反映する

よって、例えばこんな配列を並び替える関数はちゃんとソートされる！

```
#include <stdio.h>

void sort(int[], int);

int main() {
  int a[10] = {1,3,5,7,9,8,6,4,2,0};
  sort(a, 10);
  // a → {0,1,2,3,4,5,6,7,8,9}
  return 0;
}

void sort(int x[], int n) {
  // 配列 x[] を小さい順に並び替える処理
  return;
}
```

関数に渡す配列変数を変更したくない場合は const（定数・読み取り専用）で宣言する。

```
#include <stdio.h>

void unsort(const int[], int);
```