第13回
計算機基礎実習II

座席指定はありません。

計算機基礎実習II 2018 のウェブページから、以下の課題に自力で取り組んで下さい。

第12回の復習課題(rev12)

第13回の基本課題(base13)

第12回課題の解答例

ex12–2.c

自然数 n の値を入力し、つづけて n 次元ベクトルの要素を実数値として繰り返し入力すると、それまでに入力された n 次元ベクトルの和を出力するプログラム

```
#include <stdio.h>

int main(void) {
  int n, i;

  do {
    printf("n? ");
    scanf("%d", &n);
  } while (n < 1);

  double a[n];
  for (i=0; i<n; i++) a[i] = 0;
```

```
  while (1) {
    int end = 1;
    printf("elements? ");
    for (i=0; i<n; i++) {
      double xi;
      scanf("%lf", &xi);
      if (xi != 0) end = 0;
      a[i] += xi;
    }

    if (end) break;

    printf("(");
    for (i=0; i<n; i++) {
      printf("%f", a[i]);
      if (i < n-1) printf(", ");
    }
    printf(")\n");
  }
  return 0;
}
```

ex12-3.c

stdlib.h で宣言されるC言語の標準ライブラリの関数 rand() は、返り値として 0〜RAND_MAX の範囲の整数値を疑似乱数として返す関数

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

void cnt(void);
int count = 0;                // ブロック{ } によるスコープを持たないグローバル変数

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

  while (1) {
    if (n > rand()) {
      printf("cnt() は %d 回呼び出されました。\n", count); // count はグローバル変数
      exit(1);
    }
    else
      cnt();                  // グローバル変数 count に1加算
  }
  return 0;
}

void cnt(void) { count++; }   // count はグローバル変数（cnt()内で宣言されていない）
```

ex12-4.c

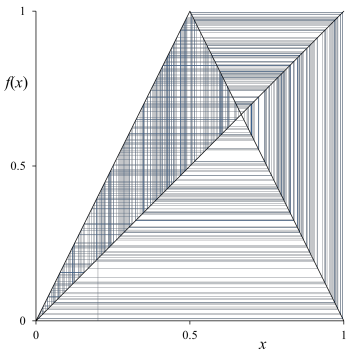
```
#include <stdio.h>
#define N 20
double tent(double);

int main(void) {
  double x;

  do {
    printf("x0? ");
    scanf("%lf", &x);
  } while (x < 0 || 1 < x);

  int i=0;
  while (i<N) {
    printf("x%d: %.20f\n", i, x);
    i++;
    x = tent(x);
  }
  return 0;
}

double tent(double x) {
  if (x < 1/2.)
    return 2*x;
  else
    return 2*(1-x);
}
```



ex12-5.c

ex12-3.c

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

void cnt(void);
int count = 0;

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

  while (1) {
    if (n > rand()) {
      printf("cnt() は %d 回...",
          count);
      exit(1);
    }
    else
      cnt();
  }
  return 0;
}

void cnt(void) { count++; }
```

グローバル変数を使わずに同じ動作を...

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

int cnt(int);

int main(void) {
  int n, count = 0;
  printf("n? ");
  scanf("%d", &n);

  while (1) {
    if (n > rand()) {
      printf("cnt() は %d 回...",
          count);
      exit(1);
    }
    else
      count = cnt(count);
  }
  return 0;
}

int cnt(int count) { return ++count; }
```

ex12-6.c

配列 v[] の要素に s が含まれていればその要素番号をすべて出力し、
また、s が含まれていなければ s がない旨を出力する

```
#include <stdio.h>
#define N 20

int main(void) {
  int v[] = {77, 37, 29, 54, 73, 64, 43, 53, 5, 52, 59, 49, 2, 93, 9, 20, 52, 75, 34, 99};
  int i, s;

  printf("s? ");
  scanf("%d", &s);

  int cnt = 0;
  for(i=0; i<N; i++) {
    if (v[i] == s) {
      cnt++;
      printf("%d は %d 番目に見つかりました\n", s, i);
    }
  }
  if (cnt == 0)
    printf("%d は見つかりませんでした\n", s);

  return 0;
}
```

ex12-7.c

```
#include <stdio.h>

int main(void) {
  int n, i, j;
  do {
    printf("n? ");
    scanf("%d", &n);
  } while (n < 1);

  int p[n];
  p[0] = 1;
  for (i=1; i<n; i++) p[i] = 0;    // p[] を 0 で初期化

  for (i=0; i<n; i++) {
    for (j=0; j<n-i; j++)          // 位置決めの空白表示
      printf("   ");

    for (j=0; j<=i; j++)           // i 列目を表示
      printf("%6d", p[j]);

    for (j=i+1; j>0; j--)          // i+1 行目を後ろから計算
      p[j] = p[j-1] + p[j];

    printf("\n");
  }
  return 0;
}
```

第13回の内容

配列変数（再）

関数（再）

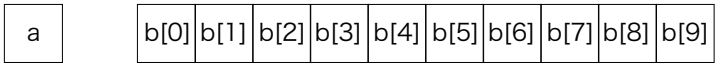
配列の応用（ヒストグラム）

配列変数とは（再）

```
int main() {
```

```
  int a;
```
a という名前の整数型の変数を1つ用意

```
  int b[10];
    :
```
b[0], b[1], … , b[9] という名前の整数型の変数を合計10こ用意



b という整数型の変数は存在しない。b[0] や b[9] が一つの変数。

b[n] と宣言すると、b[0] から b[n-1] までが使える。b[n] は使えない。

配列変数の [ ] の中の何番目かを表す0以上の整数を「添字」と呼ぶ。

配列変数の参照・代入

添字が付く以外は、通常の変数と同じ

```
int a[10];

a[0] = 0;
a[1] = 1;
a[2] = a[0] + a[1];
```

ただし、添字を整数で表現できるので

```
double x[10];
int i = 0;

x[i] = 0.1;              // x[0] = 0.1
i++;                     // i = 1
x[i] = 0.2;              // x[1] = 0.2
x[i*3] = x[i] * x[0];    // x[3] = x[1] * x[0]
```

配列変数とforループ

配列 a[] の要素番号を順番に増減し、
配列の要素 a[0], a[1], a[2], ..., a[9] を順に参照する。

```
int i;
int a[] = {10, 9, 8, 7, 6, 5, 4, 3, 2, 1};

for (i=0; i<10; i++)
   printf("%d ", a[i]);

printf("\n\n");

for (i=9; i>=0; i--)
   printf("%d ", a[i]);
```

```
10 9 8 7 6 5 4 3 2 1

1 2 3 4 5 6 7 8 9 10
```

引数がなく、返り値もない func という名の関数

```
#include <stdio.h>

void func();

int main() {
  func();
  return 0;
}

void func() {
  printf("Hello\n");
}
```

void 型の関数 func() のプロトタイプ宣言

関数 func() の呼び出し（使う）

関数 func() の定義

プロトタイプ宣言：
関数の名前や返り値の型(あるいは引数)などを宣言（他の関数などで利用できるように教える）する

関数定義：
実際にその関数がどんな仕事をするのかをプログラムコードとして記述する

main() 0

func()

引数がなく、int型を返す ten という名の関数

```
#include <stdio.h>

int ten();

int main() {
  printf("%d\n", ten());
  return 0;
}

int ten() {
  return 10;
}
```

int 型の関数 ten() のプロトタイプ宣言

関数 ten() の呼び出しと返り値の利用

関数 ten() の定義

関数を評価する（呼び出す）と定義された型の値を（返り値として）もつ

返り値の値は、関数定義の return 文で与えられる

関数の型と rerurn 文で返す式（値）の型は一致

main() 0

10

ten()

int型の引数1つと、int型の返り値がある square という名の関数

```
#include <stdio.h>

int square(int);
// int square(int b);

int main(void) {
  printf("%d\n", square(3));
  return 0;
}

int square(int a) {
  return a*a;
}
```

int 型の関数 square() のプロトタイプ宣言

関数 square() の呼び出し

関数 square() の定義

関数に渡される値（や変数）を引数と呼ぶ: square(3)

関数 square() は整数型の引数を1つもつ: int square(int)

関数に引数として渡された値は、その関数の実行時に（引数として）定義された変数に代入される: int a = 3;

main() 0

3 9

square()

ex13-1.c

0から100の値が、配列 data[] の要素にそれぞれいくつ含まれているかを出力するプログラム

```
#include <stdio.h>
#define N 50
#define MAX 101        // 0 〜 100 なので101個

int main(void) {
  int data[] = {61, 32, 97, 47, ..., 36, 46, 46};
  int cnt[MAX] = {0};  // 個数を数える配列を0で初期化

  int i;
  for (i=0; i<50; i++)
    cnt[data[i]]++;    // 整数値data[i]をそのまま要素番号として使う

  for (i=0; i<MAX; i++)
    printf("%d: %d\n", i, cnt[i]);

  return 0;
}
```