



Примеры разработки программ

Алтайский государственный университет
Факультет математики и ИТ
Кафедра информатики
Барнаул 2014

Лекция 14

■ План

- Задача о структурах и функциях
- Задача об обработке текста



Несколько заданий для самопроверки

Задача 1

Допишите в следующей программе секцию инициализации переменной `st` так, чтобы программа порождала следующий

ВЫВОД:

```
2  Ы  Щ  1.400000
```

```
void main() {
    struct altai {
        char c;
        float d;
    };

    struct russia {
        int a[3];
        char b;
        struct altai barnaul;
    };

    struct russia st = // Инициализировать здесь

    printf("%d\t%c\t%c\t%f", st.a[1], st.b, st.barnaul.c, st.barnaul.d);
}
```

Задача 1

Допишите в следующей программе секцию инициализации переменной `st` так, чтобы программа порождала следующий

ВЫВОД:

```
2  Ы  Щ  1.400000
```

```
void main() {
    struct altai {
        char c;
        float d;
    };

    struct russia {
        int a[3];
        char b;
        struct altai barnaul;
    };

    struct russia st = { {1,2,3}, 'Ы', {'Щ', 1.4} };

    printf("%d\t%c\t%c\t%f", st.a[1], st.b, st.barnaul.c, st.barnaul.d);
}
```



Задача о структурах и функциях

- Постановка задачи
- Описание типа
- Описание функций
- Использование типа и функций
- Демо

Задача о структурах и функциях: постановка

- Описать тип `struct Quadric`, задающий квадратный трехчлен с коэффициентами a, b, c .
- Реализовать в виде отдельных функций следующие операции над переменными этого типа:
 - a) вычисление значения трехчлена для заданного значения переменной;
 - b) сложение/вычитание двух трехчленов;
 - c) умножение/деление трехчлена на действительное число;
 - d) вычисление i -го корня трехчлена (действительного или комплексного);
 - e) проверка равенства корней двух трехчленов;
 - f) вычисление значения абсциссы, соответствующего вершине параболы, описываемой трехчленом.

Задача о структурах и функциях: постановка (продолжение)

- С использованием описанных типа и функций разработать программу, которая для заданного набора из N трехчленов находит
 - 1) корни трехчлена, являющегося суммой заданных;
 - 2) все пары трехчленов с совпадающими комплексными корнями среди трехчленов с положительной вершиной абсциссы вершины определяемой ими параболы.

Квадратный трехчлен

- Квадратный трехчлен $q(x) = ax^2 + bx + c$ полностью определяется своими коэффициентами $a, b, c \in \mathbb{R}$

```
/* Тип, описывающий трехчлен */
```

```
struct Quadric {
```

```
    double a;
```

```
    double b;
```

```
    double c;
```

```
};
```

```
...
```

```
/* Переменные - трехчлены */
```

```
struct Quadric q={1,-2,4}; /* инициализация q */
```

```
struct Quadric p;
```

```
p.a = -1; /* так можно обращаться к коэффициентам p */
```

```
p.b = 4;
```

```
p.c = 2;
```

```
...
```

Описание структуры –
описание нового типа с
именем struct Quadric

Описание переменных
несколько громоздко

Квадратный трехчлен

- Квадратный трехчлен $q(x) = ax^2 + bx + c$ полностью определяется своими коэффициентами $a, b, c \in \mathbb{R}$

```
/* Тип, описывающий трехчлен */
```

```
typedef struct Quadric_t {
```

```
    double a;
```

```
    double b;
```

```
    double c;
```

```
} Quadric;
```

```
...
```

```
/* Переменные - трехчлены */
```

```
Quadric q={1,-2,4}; /* инициализация q */
```

```
Quadric p;
```

```
p.a = -1; /* так можно обращаться к коэффициентам p */
```

```
p.b = 4;
```

```
p.c = 2;
```

```
...
```

Описание структуры
совмещено с объявлением
синонима имени нового
типа

Описание переменных
становится лаконичнее

Квадратный трехчлен

- Вычисление значения трехчлена для некоторого x_0 :

$$q(x_0) = a x_0^2 + b x_0 + c$$

```
...
/* Вычисление значения трехчлена для x0 */
double value(Quadric q, double x0) {
    return q.a*x0*x0 + q.b*x0 + q.c;
}

void main() {
    Quadric p={1,-2,4};           /* инициализация p      */
    double x0=0.1;               /* инициализация x0    */
    double v=value(p,x0);        /* вычисление v=p(x0)  */
    printf("p(%ld)=%ld", x0, v); /* вывод x0 и v      */
}
```

Квадратный трехчлен

- Сложение двух трехчленов

$$q(x) = a_q x^2 + b_q x + c_q \quad \text{и} \quad p(x) = a_p x^2 + b_p x + c_p :$$
$$p(x) + q(x) = (a_p + a_q)x^2 + (b_p + b_q)x + (c_p + c_q)$$

```
...
/* Сложение двух трехчленов */
Quadric sum(Quadric q, Quadric p) {
    Quadric r;
    r.a = q.a + p.a;
    r.b = q.b + p.b;
    r.c = q.c + p.c;
    return r;
}

void main() {
    Quadric u={1,-2,4}, v={1.5,0,-1}, w;
    w = sum(u,v);
}
```

Квадратный трехчлен

- Сложение двух трехчленов

$$q(x) = a_q x^2 + b_q x + c_q \quad \text{и} \quad p(x) = a_p x^2 + b_p x + c_p :$$
$$p(x) + q(x) = (a_p + a_q)x^2 + (b_p + b_q)x + (c_p + c_q)$$

```
...
/* Сложение двух трехчленов */
Quadric sum(Quadric q, Quadric p) {
    Quadric r={q.a + p.a, q.b + p.b, q.c + p.c};
    return r;
}

void main() {
    Quadric u={1,-2,4}, v={1.5,0,-1}, w;
    w = sum(u,v);
}
```

**Автоматические переменные можно инициализировать
неконстантными выражениями**

Квадратный трехчлен

- Умножение трехчлена $q(x) = ax^2 + bx + c$ на действительное число d :

$$dq(x) = dax^2 + dbx + dc$$

```
...
/* Умножение трехчлена на число */
Quadric prod(Quadric q, double d) {
    Quadric r={d*q.a, d*q.b, d*q.c};
    return r;
}

void main() {
    Quadric u={14,12,2009}, w;
    double z=13.13;
    w = prod(u,d);
}
```

Квадратный трехчлен

- Вычисление корней трехчлена $q(x) = ax^2 + bx + c$:
- Если $D = b^2 - 4ac \geq 0$, то

$$x_1 = \frac{-b - \sqrt{D}}{2a}$$

$$x_2 = \frac{-b + \sqrt{D}}{2a}$$

- иначе

$$x_1 = -\frac{b}{2a} - \frac{\sqrt{|D|}}{2a}i$$

$$x_2 = -\frac{b}{2a} + \frac{\sqrt{|D|}}{2a}i$$

Квадратный трехчлен

- Вычисление корней трехчлена $q(x) = ax^2 + bx + c$

```
typedef struct Complex_t {
    double re,im;
} Complex;

/* i-й корень трехчлена, i=1,2 */
Complex root(Quadric q, int i) {
    double D = q.b*q.b - 4*q.a*q.c; /* дискриминант */
    double s = (i==1)?-1:1;        /* знак */
    Complex c = {0,0};             /* корень */
    if (D >= 0)
        c.re = (-q.b+s*sqrt(D))/(2*q.a);
    else {
        c.re = -q.b/(2*q.a);
        c.im = s*sqrt(fabs(D))/(2*q.a);
    }
    return c;
}
```

Нужна
предварительная
проверка на
невыврожденность
трехчлена ($a \neq 0$)

Квадратный трехчлен

- Вычисление корней трехчлена $q(x) = ax^2 + bx + c$

```
typedef struct Complex_t {
    double re,im;
} Complex;

/* i-й корень трехчлена, i=1,2 */
Complex root(Quadric q, int i) {
    ...
}

void main() {
    Quadric u={14,12,2009};
    Complex x1=root(u,1), x2=root(u,2);
    printf("x1 = %lf + %lf*i\n", x1.re, x1.im);
    printf("x2 = %lf + %lf*i\n", x2.re, x2.im);
}
```

Квадратный трехчлен

- Проверка равенства корней двух трехчленов $q(x)$ и $p(x)$

```
...
#define EPS 1E-6

/* Проверка равенства двух комплексных чисел */
int is_equal(Complex z, Complex w) {
    return fabs(z.re-w.re)<EPS && fabs(z.im-w.im)<EPS;
}

/* Проверка равенства корней двух трехчленов */
int is_roots_equal(Quadric q, Quadric p) {
    return is_equal(root(q,1),root(p,1)) &&
           is_equal(root(q,2),root(p,2));
}
...
```

Квадратный трехчлен

- Вычисление абсциссы x_0 вершины параболы, описываемой трехчленом $q(x) = ax^2 + bx + c$:

$$x_0 = -b/2a$$

```
...
#define EPS 1E-6

/* Проверка равенства двух комплексных чисел */
double parabola_x(Quadric q) {
    return -q.b/(2*q.a);
}
```

Нужна
предварительная
проверка на
невырожденность
трехчлена ($a \neq 0$)

Квадратный трехчлен

- Для набора из N трехчленов найти корни трехчлена, являющегося суммой исходных.

```
...
void main() {
    int i, N;
    Quadric *q, s={0,0,0};
    Complex x1, x2;

    /* Ввод N */
    printf("N=");
    scanf("%d", &N);

    /* Распределение памяти под массив */
    q=(Quadric *)malloc(N*sizeof(Quadric));
    if(q==NULL) {
        printf("Ошибка выделения памяти\n");
        return;
    }
    ...
}
```

Квадратный трехчлен

- Для набора из N трехчленов найти корни трехчлена, являющегося суммой исходных.

```
...
void main() {
    int i, N;
    Quadric *q, s={0,0,0};
    Complex x1, x2;
    /* Ввод N и распределение памяти под массив */
    ...
    /* Ввод коэффициентов трехчленов */
    for(i=0; i<N; i++) {
        printf("a,b,c %d:");
        scanf("%lf%lf%lf",&q[i].a, &q[i].b, &q[i].c);
    }

    /* Суммирование трехчленов */
    for(i=0; i<N; i++)
        s=sum(s,q[i]);
    ...
}
```

Квадратный трехчлен

- Для набора из N трехчленов найти корни трехчлена, являющегося суммой исходных.

```
...
void main() {
    int i, N;
    Quadric *q, s={0,0,0};
    /* Ввод N и распределение памяти под массив */
    ...
    /* Ввод коэффициентов трехчленов */
    ...
    /* Суммирование трехчленов */
    ...
    /* Вычисление корней */
    x1=root(s,1);
    x2=root(s,2);
    printf("x1 = %lf + %lf*i\n", x1.re, x1.im);
    printf("x2 = %lf + %lf*i\n", x2.re, x2.im);
    free(q); /* Освобождение памяти */
}
```

Нужна предварительная проверка на невырожденность трехчлена $s(x)$ ($a \neq 0$)

Квадратный трехчлен

- Для набора из N трехчленов найти все пары трехчленов с совпадающими комплексными корнями и положительной абсциссой вершины определяемой ими параболы.

```
...
void out(Quadric q) {
    printf("%lf*x^2 + %lf*x + %lf", q.a, q.b, q.c);
}

void main() {
    int i, j, N;
    Quadric *q;
    /* Ввод N, выделение памяти и ввод коэффициентов */
    ...
}
```

Квадратный трехчлен

- Для набора из N трехчленов найти все пары трехчленов с совпадающими комплексными корнями и положительной абсциссой вершины определяемой ими параболы.

```
...
void main() {
    ...
    for(i=0; i<N-1; i++)
        for(j=i+1; j<N; j++) {
            if(parabola_x(q[i])<=0) continue;
            if(is_roots_equal(q[i],q[j])) {
                printf("q[%d] и q[%d]: (" ,i,j);
                out(q[i]);
                printf(") и (");
                out(q[j]);
                printf(")\n");
            }
        }
    }
}
```


Квадратный трехчлен

ДЕМО





Задача об обработке текста

- Постановка задачи
- Алгоритм (менее эффективный)
- Алгоритм (более эффективный)
- Демо

Задача об обработке текста

- В заданном тексте найти строки-палиндромы, т.е. строки одинаково читающиеся слева направо и справа налево. Например, «КУЛИНАР, ХРАНИ ЛУК» или «ЛЕША НА ПОЛКЕ КЛОПА НАШЕЛ».

Задача об обработке текста

- Алгоритм (менее эффективный)
 - Запросить имя файла
 - Открыть файл
 - Пока файл не окончился повторять
 - Читать из файла очередную строку
 - Сделать копию строки
 - Удалить в строке все символы-разделители
 - Если строка симметрична относительно центра, то вывести ее сохраненную копию
 - Закрывать файл

Задача об обработке текста

- Алгоритм (более эффективный)
 - Запросить имя файла
 - Открыть файл
 - Пока файл не окончился повторять
 - Читать из файла очередную строку
 - Указатель **A** установить на первую букву в строке
 - Указатель **B** установить на последнюю букву в строке
 - Пока (**A** < **B** не равны) и (совпадают символы, на которые они указывают) повторять
 - Сместить **A** на следующую (слева-направо) букву
 - Сместить **B** на следующую (справа-налево) букву
 - Если **A** >= **B**, то вывести строку.
 - Закрывать файл

Задача об обработке текста

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

void main() {
    FILE *f;
    char filename[40];          // Имя входного файла
    char s[80];                // Исследуемая строка
    char t[]=" ,. ; : ! ? \n \r"; // Игнорируемые символы
    char *b, *e;               // Указатели на тек. символы
    int isSymmetric;           // Результат проверки

    printf("Input file name:"); // Ввод имени файла
    gets(filename);

    f=fopen(filename,"rt");     // Открытие файла
    if(f==NULL) {
        printf("File open error: %s\n",filename);
        return;
    }
    ...
}
```

Задача об обработке текста

```
void main() {
    ...
    while( fgets(s,80,f) ){ // Чтение очередной строки
        b=s; e=s+strlen(s); // Инициализация указателей
        isSymmetric = 1; // Инициализация результата

        while(b<e && isSymmetric){
            /* Поиск неигнорируемых символов */
            while( b<e && strchr(t,*b) ) b++;
            while( b<e && strchr(t,*e) ) e--;
            /* Проверка совпадения символов и сдвиг указателей
*/
            if ( b<e && *b==*e ) { b++; e--; }
            else isSymmetric = 0;
        }

        if (isSymmetric) // Вывод строки-палиндрома
printf("%s\n", s);
    }
    fclose(f);
}
```

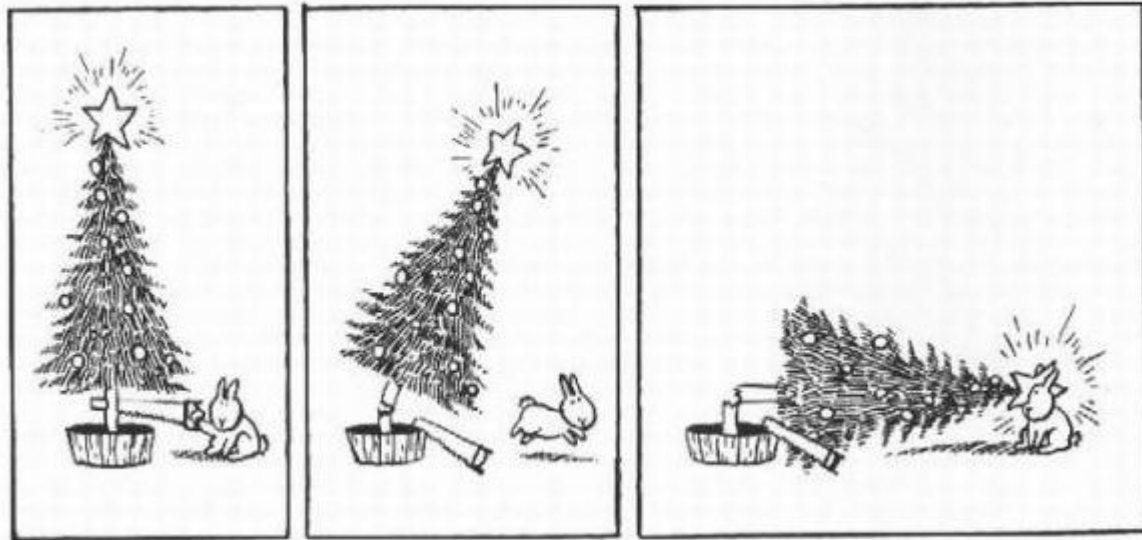
Поиск палиндромов

ДЕМО



Вопросы?

- Задача о структурах и функциях
 - Постановка задачи
 - Описание типа
 - Описание функций
 - Использование типа и функций
 - Демо
- Задача об обработке текста
 - Постановка задачи
 - Алгоритм (менее эффективный)
 - Алгоритм (более эффективный)
 - Демо



Э. Райли Кролик-самоубийца