

Раздел «Язык Си» . Coffe-structF_contC :

- Структуры.
 - Задача о точке.
 - Как описать структуру и объявить структурные переменные.
 - Функция с параметром-структурой по-значению
 - Функция с адресом параметром-структурой
 - Функция, возвращающая значение-структуру.
 - Структуры в структуре.
 - Задача о длине отрезка.
 - Функция печати для отрезка.
 - Функция для получения данных об отрезке.
 - 🍌 Задачи
 - Задача S1. Перемещение точки.
 - Задача S2. Отображение на ось Y.
 - Задача S4. Расстояние между точками.
 - Задача S5. Проекция на ось X
 - Задача S6. Поворот на 90°
 - Задача S7. Корни квадратного уравнения
 - Задача S8. Треугольник.

Структуры.

Задача о точке.

Необходимо написать функцию `mirror(int *x1, int *y1)`, которая получает точку точку зеркально отраженную относительно оси **OY**.

Эту задачу можно решить используя указатели координат новой точки (файл `point1.c`):

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

// функция получения отображенной точки
// Чтобы записать значения в x1, y1 нужны их адреса
// Поэтому используем УКАЗАТЕЛИ
// Функция НИЧЕГО не возвращает
void mirrorP(int *x1, int *y1){
    *x1 = -*x1;
};

// Проверим работу функции

int main(){
    int x,y,x1,y1;
    scanf("%d%d",&x, &y);

    // Чтобы записать значения в переменные x1, y1 передаем
    // их адреса
    mirrorP(&x1,&y1);
    printf("(%d,%d)\n",x1,y1);
    return 0;
}
```

Компиляция и запуск:

```
>gcc point1.c -o point
>./point
```

Поиск

Поиск

Раздел «Язык Си»

Главная
Зачем учить C?
Определения

Инструменты:

Поиск
Изменения
Index
Статистика

Разделы

Информация
Алгоритмы
Язык Си
Язык Ruby
Язык
Ассемблера
El Judge
Парадигмы
Образование
Сети
Objective C

Logon>>

```
10 12
-10 12
>
```

Но часто хочется чтобы функция ВОЗВРАЩАЛА две и более переменных. Да и точка – это один объект, а не две переменные.


ДАННЫЕ в программе должны быть организованы как можно ближе к привычному представлению об объектах и сущностях.

Попробуем решить эту задачу по-другому.

Соберем вместе в один объект все координаты одной точки. Имена этих координат (**x** и **y**) сохраняются. Поэтому к координатам будем обращаться как обычно: **x**, **y**. Для этого в C есть **СТРУКТУРЫ**.

Структура – это НОВЫЙ тип данных, придуманный программистом. Сначала ее нужно ОПИСАТЬ (как проект). Описанная структура – это еще не переменные. Переменные появятся в функциях, когда мы их объявим.

Итак, структура для точки. Все нужные переменные помещаются в одно место в памяти, которому можно дать имя или обратиться по адресу.

struct Point p1 

Как описать структуру и объявить структурные переменные.

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

// Описываем структуру Point
struct Point{
    int x; // для координаты x
    int y; // для координаты y
};

int main(){

    // объявляем структурные переменные:
    // точки a, b и c
    struct Point a, b, c;

    // Как записать и получить
    // значения координат точки a
    a.x = 10;
    a.y = 12;

    // Как записать и получить
    // значения координат точки b
    b.x = 10;
    b.y = 12;

    // Как скопировать всю точку b в точку c:
    c = b;

    // Как напечатать значения координат
    // точки c
    printf("c: (%d,%d)\n", c.x, c.y);
    return 0;
}
```

Функция с параметром-структурой по-значению

Но структуры особенно удобны для работы с функциями. Напишем функцию печати координат точки.

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

// Описываем структуру Point
struct Point{
    // поля или атрибуты структуры
    int x; // для координаты x
    int y; // для координаты y
};

// функция получения отображенной точки
void mirrorP(int x, int y, int
*x1, int *y1){
    *y1 = y;
    *x1 = -x;
};

//функция writeP будет работать с
// переменной типа struct Point a
// Эта переменная передается "по-значению",
// то есть копируется.
// будет создана новая локальная
// переменная типа struct Point в функции
// с именем a.
// В нее будут скопированы значения полей
// переменной-аргумента
void writeP(struct Point a){
    printf("c:(%d,%d)\n",a.x,a.y);
};
// Проверим работу функции

int main(){
    // объявляем структурные переменные:
    // точки a, b и c
    struct Point a, b,c;

    // Как записать и получить
    // значения координат точки b
    b.x = 10;
    b.y = 12;

    // Как скопировать всю точку b в точку c:
    c = b;

    // Как напечатать значения координат
    // точки c
    // вызов функции
    writeP(c);
    return 0;
}
```

Функция с адресом параметром-структурой

Теперь напишем функцию получения значений (ввод с клавиатуры)

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

// Описываем структуру Point
struct Point{
    // поля или атрибуты структуры
    int x; // для координаты x
```

```

    int y; // для координаты y
};

// функция получения отображенной точки

void mirrorP(int x, int y, int
*x1, int *y1){
    *y1 = y;
    *x1 = -x;
};

// Функция будет использовать адрес
// переменной типа struct Point чтобы
// записать значения именной в эту
// переменную
void getPoint(struct Point *a){
    int x,y;
    // считываем значения для x и y
    scanf("%d%d",&x,&y);
    // присваиваем значения полям
    // через указатель на
    // структурную переменную
    a->x = x;
    a->y = y;
};

void writeP(struct Point a){
    printf("c:(%d,%d)\n",a.x,a.y);
};
// Проверим работу функции

int main(){
    // объявляем структурные переменные:
    // точки a, b и c
    struct Point a, b,c;

    // получаем значения с клавиатуры.
    getPoint(&b);
    // Как скопировать всю точку b в точку c:
    c = b;

    // Как напечатать значения координат
    // точки c
    // вызов функции
    writeP(c);
    return 0;
}

```

Функция, возвращающая значение-структуру.

Функции в C могут возвращать структурные переменные. Напишем функцию для получения НОВОЙ точки – отображения относительно оси **OY**.

```

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

// Описываем структуру Point

struct Point{
    // поля или атрибуты структуры
    int x; // для координаты x
    int y; // для координаты y
};

// функция получения отображенной точки

struct Point mirrorP(struct Point a){

```

```

// чтобы вернуть переменную, ее нужно иметь
// объявляем локальную переменную struct Point
struct Point tmp;
// присваиваем значения полям временной переменной
tmp.y = a.y;
tmp.x = -a.x;
// все поля заполнены
// возвращаем готовую переменную
return tmp;
};

// Функция будет использовать адрес
// переменной типа struct Point чтобы
// записать значения именной в эту
// переменную
void getPoint(struct Point *a){
    int x,y;
    // считываем значения для x и y
    scanf("%d%d",&x,&y);
    // присваиваем значения полям
    // через указатель на
    // структурную переменную
    a->x = x;
    a->y = y;
};

void writeP(struct Point a){
    printf("c:(%d,%d)\n",a.x,a.y);
};
// Проверим работу функции

int main(){
    // объявляем структурные переменные:
    // точки a, b и c
    struct Point a, b,c;

    // получаем значения с клавиатуры.
    getPoint(&b);

    // получаем отображение в точку c
    // точка b остается неизменной
    c = mirrorP(b);

    // Как напечатать значения координат
    // точки c
    // вызов функции
    writeP(c);
    return 0;
}

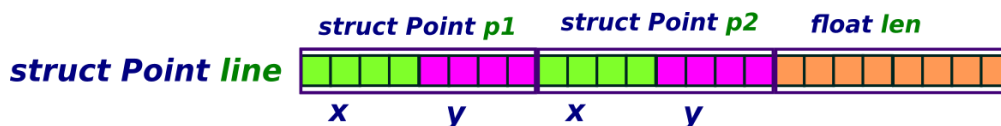
```

Структуры в структуре.

Задача о длине отрезка.

Необходимо написать функции, которые работают с отрезками.

У отрезка есть две точки – концы отрезка и длина (дробное число). Для точек-концов них мы уже умеем описывать структуру. Рассмотрим как может выглядеть структура для отрезка.



```

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

```

```
// Описываем структуру Point
// typedef позволяет дать новое имя любому
// типу
// теперь можно писать просто Point a
typedef struct Pnt{
// поля или атрибуты структуры
    int x; // для координаты x
    int y; // для координаты y
} Point;

// Описываем структуру Line

typedef struct Ln{
// поля или атрибуты структуры Line
// координаты концов - это точки
    Point a,b;
// длина отрезка - дробное число
    float distance;
} Line;

/*=====
Все функции для работы с точками будут нужны!!!
=====*/
// функция получения отображенной точки
// теперь можно писать просто Point

Point mirrorP(Point a){
// чтобы вернуть переменную, ее нужно иметь
// объявляем локальную переменную struct Point
    Point tmp;
// присваиваем значения полям временной переменной
    tmp.y = a.y;
    tmp.x = -a.x;
// все поля заполнены
// возвращаем готовую переменную
    return tmp;
};

// Функция будет использовать адрес
// переменной типа struct Point чтобы
// записать значения именной в эту
// переменную
void getPoint(Point *a){
    int x,y;
// считываем значения для x и y
    scanf("%d%d",&x,&y);
// присваиваем значения полям
// через указатель на
// структурную переменную
    a->x = x;
    a->y = y;
};

void writeP(Point a){
    printf("c:(%d,%d)\n",a.x,a.y);
};
// Проверим работу функции

int main(){
// объявляем структурные переменные:
// отрезок line1
    Line line1,line2;

// Как записать значения координат в отрезке
    line1.a.x = 0;
    line1.a.y = 3;
```

```

    line1.b.x = 4;
    line1.b.y = 0;
    line1.distance = 5; // проверьте!!

// как скопировать один отрезок в другой
line2 = line1;

// Как напечатать все про отрезок:

writeP(line2.a);
writeP(line2.b);
printf(" длина: %0.2f ", line2.distance);
return 0;
}

```

Функция печати для отрезка.

```

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

// Описываем структуру Point
// typedef позволяет дать новое имя любому
// типу
// теперь можно писать просто Point а
typedef struct Pnt{
    // поля или атрибуты структуры
    int x; // для координаты x
    int y; // для координаты y
} Point;

// Описываем структуру Line

typedef struct Ln{
    // поля или атрибуты структуры Line
    // координаты концов - это точки
    Point a,b;
    // длина отрезка - дробное число
    float distance;
} Line;

/*=====
Все функции для работы с точками будут нужны!!!
=====*/

// функция получения отображенной точки
// теперь можно писать просто Point

Point mirrorP(Point a){
    // чтобы вернуть переменную, ее нужно иметь
    // объявляем локальную переменную struct Point
    Point tmp;
    // присваиваем значения полям временной переменной
    tmp.y = a.y;
    tmp.x = -a.x;
    // все поля заполнены
    // возвращаем готовую переменную
    return tmp;
};

// Функция будет использовать адрес
// переменной типа Point чтобы
// записать значения именной в эту
// переменную
void getPoint(Point *a){
    int x,y;
    // считываем значения для x и y
    scanf("%d%d",&x,&y);
    // присваиваем значения полям

```

```

// через указатель на
// структурную переменную
a->x = x;
a->y = y;
};

void writeP(Point a){
    printf("c:(%d,%d)",a.x,a.y);
};
// Проверим работу функции

/*=====
    Функции работы с отрезками будут использовать
    функции для работы с точками, поэтому они должны
    быть написаны ПОСЛЕ функций работы с точками
=====*/

/*=====
    Функции для работы с отрезками
=====*/

// Функция печати (ничего не возвращает)
void writeL(Line lin){
    //Используем функции для печати точек
    writeP(lin.a);
    writeP(lin.b);
    printf(" длина: %0.2f\n",lin.distance);
}

int main(){
    // объявляем структурные переменные:
    // отрезок line1
    Line line1,line2;

    // Как записать значения координат в отрезке
    line1.a.x = 0;
    line1.a.y = 3;
    line1.b.x = 4;
    line1.b.y = 0;
    line1.distance = 5;// проверьте!!

    // как скопировать один отрезок в другой
    line2 = line1;

    // Как напечатать все про отрезок:

    writeL(line2);
    return 0;
}

```

Функция для получения данных об отрезке.

```

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

// Описываем структуру Point
// typedef позволяет дать новое имя любому
// типу
// теперь можно писать просто Point а
typedef struct Pnt{
    // поля или атрибуты структуры
    int x; // для координаты x
    int y; // для координаты y
} Point;

// Описываем структуру Line

```



```
typedef struct Ln{
// поля или атрибуты структуры Line
// координаты концов - это точки
    Point a,b;
// длина отрезка - дробное число
    float distance;
} Line;

/*=====
Все функции для работы с точками будут нужны!!!
=====*/
// функция получения отображенной точки
// теперь можно писать просто Point

Point mirrorP(Point a){
// чтобы вернуть переменную, ее нужно иметь
// объявляем локальную переменную Point
    Point tmp;
// присваиваем значения полям временной переменной
    tmp.y = a.y;
    tmp.x = -a.x;
// все поля заполнены
// возвращаем готовую переменную
    return tmp;
};

// Функция будет использовать адрес
// переменной типа Point чтобы
// записать значения именной в эту
// переменную
void getPoint(Point *a){
    int x,y;
// считываем значения для x и y
    scanf("%d%d",&x,&y);
// присваиваем значения полям
// через указатель на
// структурную переменную
    a->x = x;
    a->y = y;
};

void writeP(Point a){
    printf("(%d,%d)",a.x,a.y);
};
// Проверим работу функции

/*=====
Функции работы с отрезками будут использовать
функции для работы с точками, поэтому они должны
быть написаны ПОСЛЕ функций работы с точками
=====*/

/*=====
Функции для работы с отрезками
=====*/

// Функция печати (ничего не возвращает)
void writeL(Line lin){
//Используем функции для печати точек
    writeP(lin.a);
    writeP(lin.b);
    printf("    длина: %0.2f\n",lin.distance);
}

// Функция для получения данных об отрезке
// Используем указатель
void getLine(Line * lin){
```

```

// getPoint использует АДРЕС точки
// Мы имеет адрес lin.
// lin->a - обращение к полю a,
// если lin - это адрес
// Затем вычисляем адрес поля a - &(lin->a)
getPoint(&(lin->a));
getPoint(&(lin->b));
// длину вычислить самостоятельно
lin->distance = 5.0;
};

int main(){
// объявляем структурные переменные:
// отрезок line1
Line line1,line2;

// Как записать значения координат в отрезке
getLine(&line1);

// Как напечатать все про отрезок:

writeL(line1);
return 0;
}

```

Задачи

Задача S1. Перемещение точки.

Написать функцию **void moveX(Point * a, int z)**, которая перемещает точку вдоль оси **OX** на **z**.

Задача S2. Отображение на ось Y.

Написать функцию **int toY(Point a)**, которая отображает точку на ось **OY** и получает координату на оси.

Задача S4. Расстояние между точками.

Написать функцию **float distP(Point a, Point b)**, которая вычисляет расстояние между точками. Используя эту функцию, вычислить расстояние и заполнить поле **distance** в функции **getLine()**.

Задача S5. Проекция на ось X

Написать функцию **Line lineToX(Line a)**, которая возвращает отрезок-проекцию на ось **OX**.

Задача S6. Поворот на 90°

Написать функцию **void rotate90(Line* lin)**, которая поворачивает отрезок по часовой стрелке на 90° относительно середины отрезка.

Задача S7. Корни квадратного уравнения

Корни квадратного уравнения описываются структурой

```

typedef struct Rt{
float x1, x2;
} Radix;

```

Квадратный трехчлен задается тремя параметрами: **a, b, c** ($ax^2+bx+c=0$). Написать функцию **int findRoots(Radix* kv, float a, float b, float c)**, которая выясняет есть ли решение этого уравнения в действительных числах, и, если есть, вычисляет корни этого уравнения.

В проверочной программе печатать количество различных корней и их значения.

Задача S8. Треугольник.



Треугольник задается тремя точками на плоскости. Все координаты – целые числа.

Треугольник описать как структуру, содержащую три точки.

Написать функции:

функция	назначение
void getTri(Tri* d)	получение координат треугольника
void writeTri(Tri d1)	печать координат треугольника
float squareTri(Tri tr1)	вычислить площадь
float perimTri(Tri tr1)	вычислить периметр
void rotate90(Tri * tr1)	повернуть треугольник против часовой стрелки на 90^0 относительно точки (0,0)

-- [TatyanaOvsyannikova2011](#) - 03 Oct 2017

Attachment	Action	Size	Date	Who	Comment
 structPoint.png	manage	6.7 K	03 Oct 2017 - 06:59	TatyanaOvsyannikova2011	
 structLine.png	manage	20.0 K	03 Oct 2017 - 07:44	TatyanaOvsyannikova2011	

(с) Материалы раздела "Язык Си" публикуются под лицензией [GNU Free Documentation License](#).