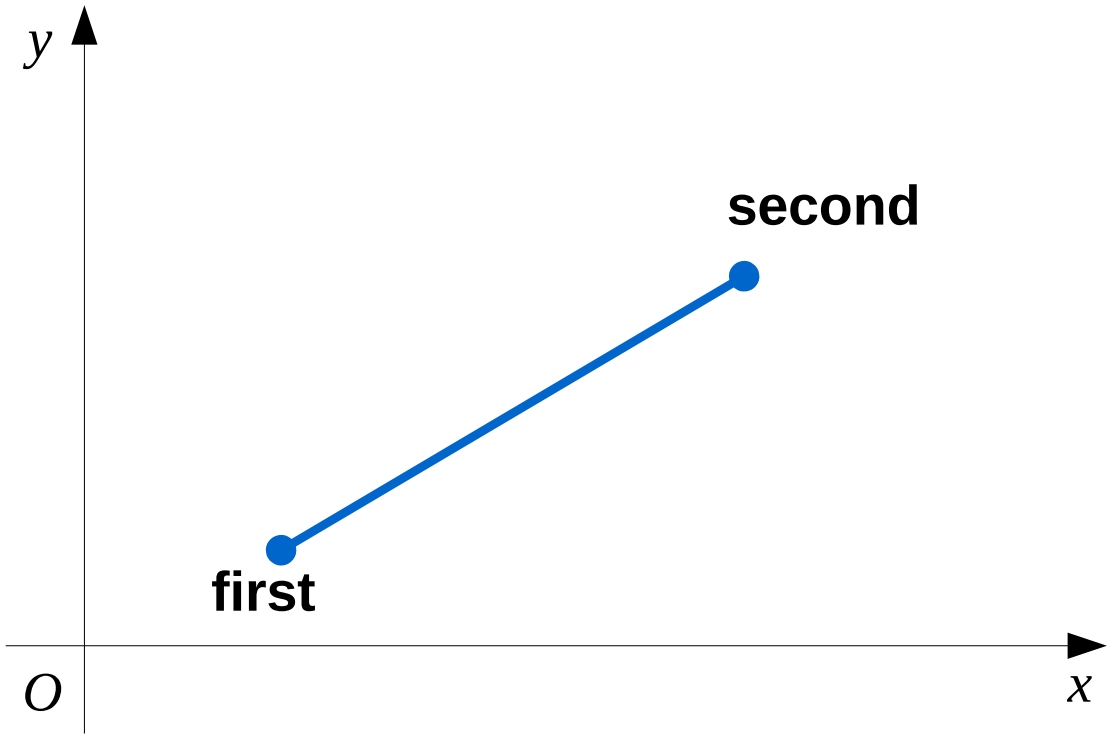
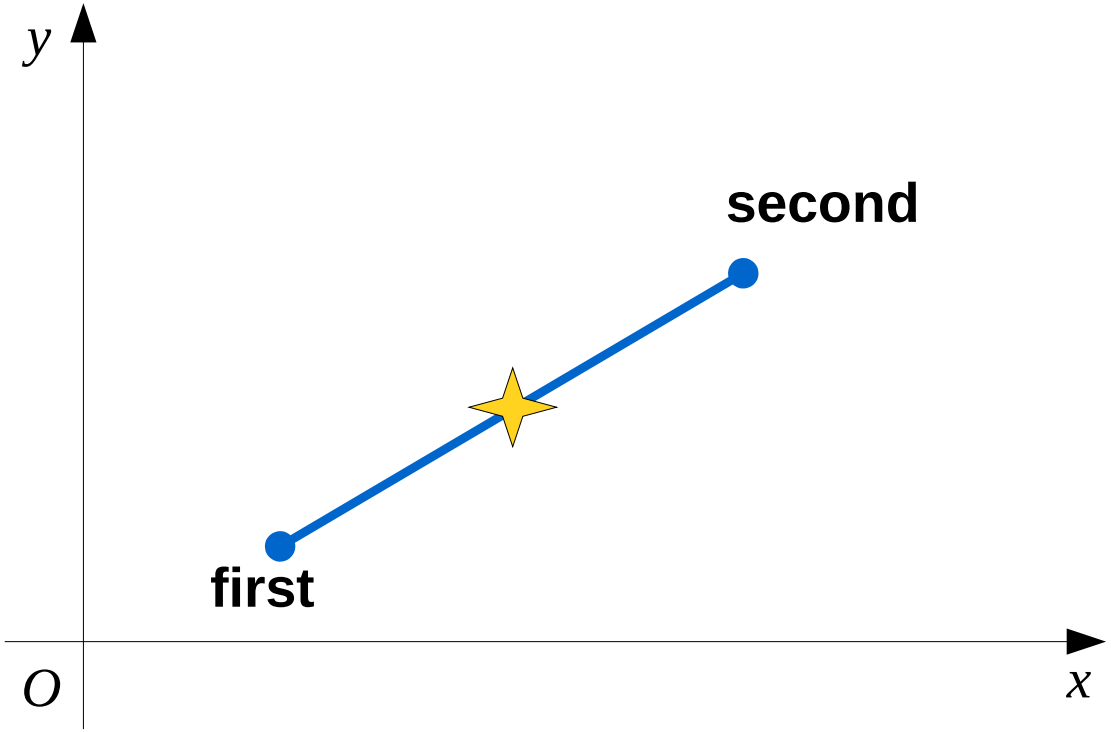
Использование структурного типа данных в функциях

Постановка задачи

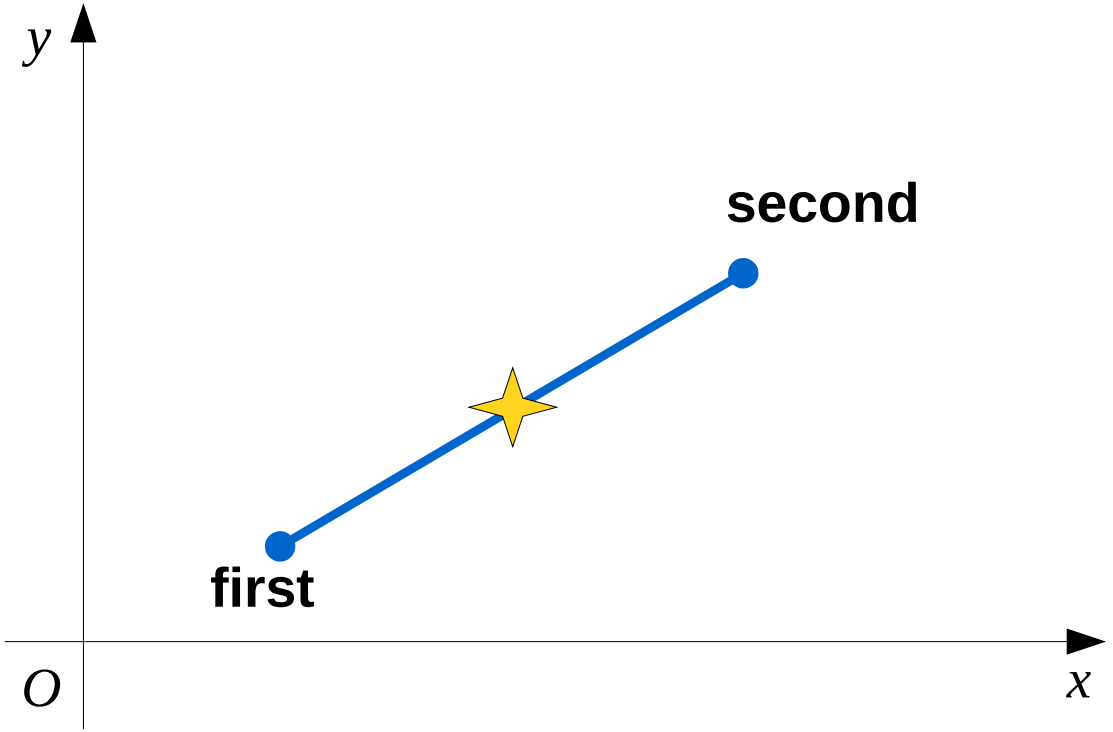


Постановка задачи



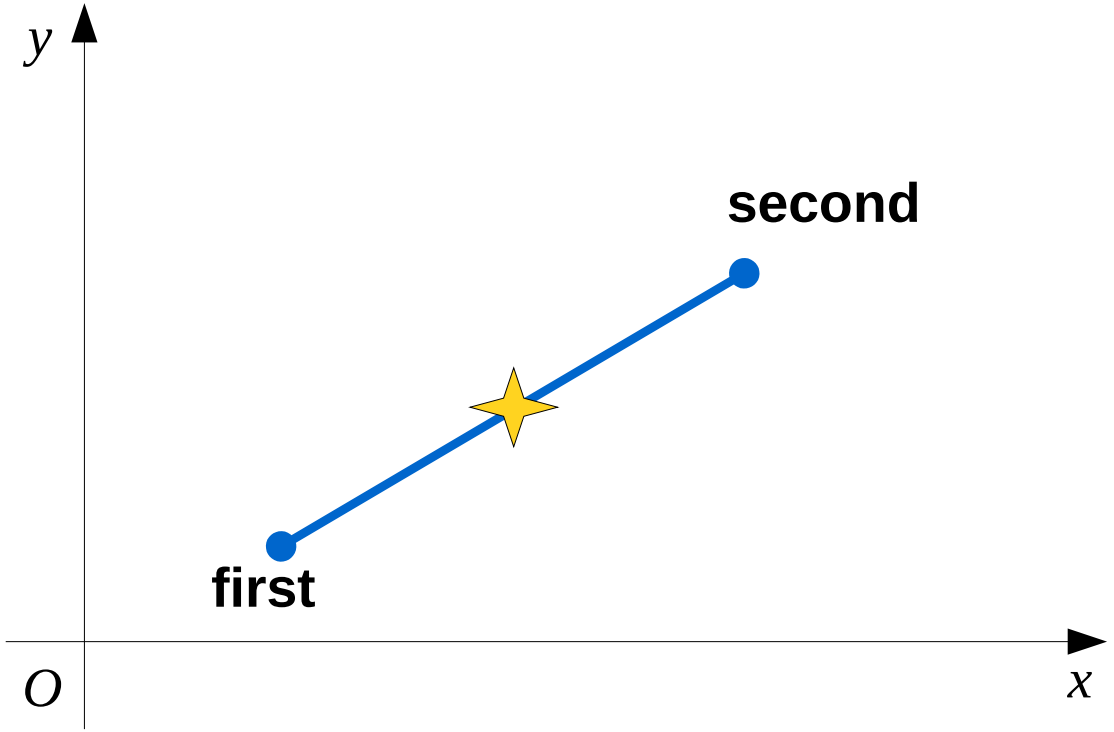
- ✔ Разработать программу расчёта середины заданного отрезка на декартовой плоскости.

Постановка задачи



- ✔ Разработать программу расчёта середины заданного отрезка на декартовой плоскости.
- ✔ Координаты отрезка запрашиваются у пользователя
- ✔ Найденное значение должно выдаваться на стандартное устройство вывода.

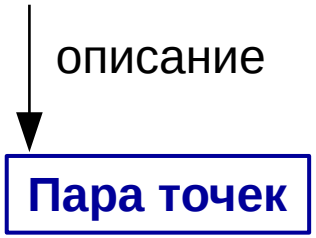
Постановка задачи



- ✔ Разработать программу расчёта середины заданного отрезка на декартовой плоскости.
- ✔ Координаты отрезка запрашиваются у пользователя
- ✔ Найденное значение должно выдаваться на стандартное устройство вывода.

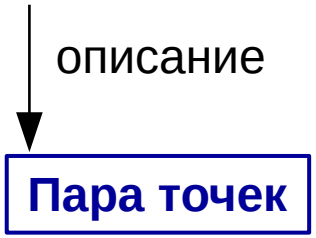
Используемые понятия

Отрезок

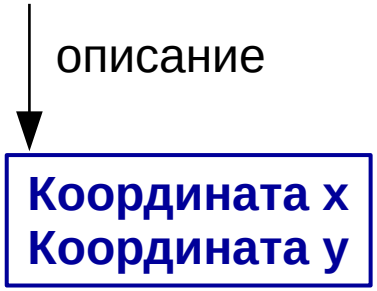


Используемые понятия

Отрезок



Точка на плоскости



Используемые понятия

Отрезок

описание

Пара точек

Точка на плоскости

представление

Структурный тип данных

описание

Координата x
Координата y

описание

Числовое значение x
Числовое значение y

Используемые понятия

Отрезок → представление → **Структурный тип данных**

описание → **Пара точек**

описание → **Первая точка first**
Вторая точка second

Точка на плоскости → представление → **Структурный тип данных**

описание → **Координата x**
Координата y

описание → **Числовое значение x**
Числовое значение y

Структурный тип данных

Объявление типа данных:

```
struct имяТипа
 {
       Тип1    имяПоля1;
       Тип2    имяПоля2;
       . . .
       ТипN    имяПоляN;
 };
```

«Середина отрезка» (1)

```
//center.cpp
//расчёт середины отрезка на декартовой плоскости
```

«Середина отрезка» (1)

```
//center.cpp
//расчёт середины отрезка на декартовой плоскости
#include <iostream>
using namespace std;
```

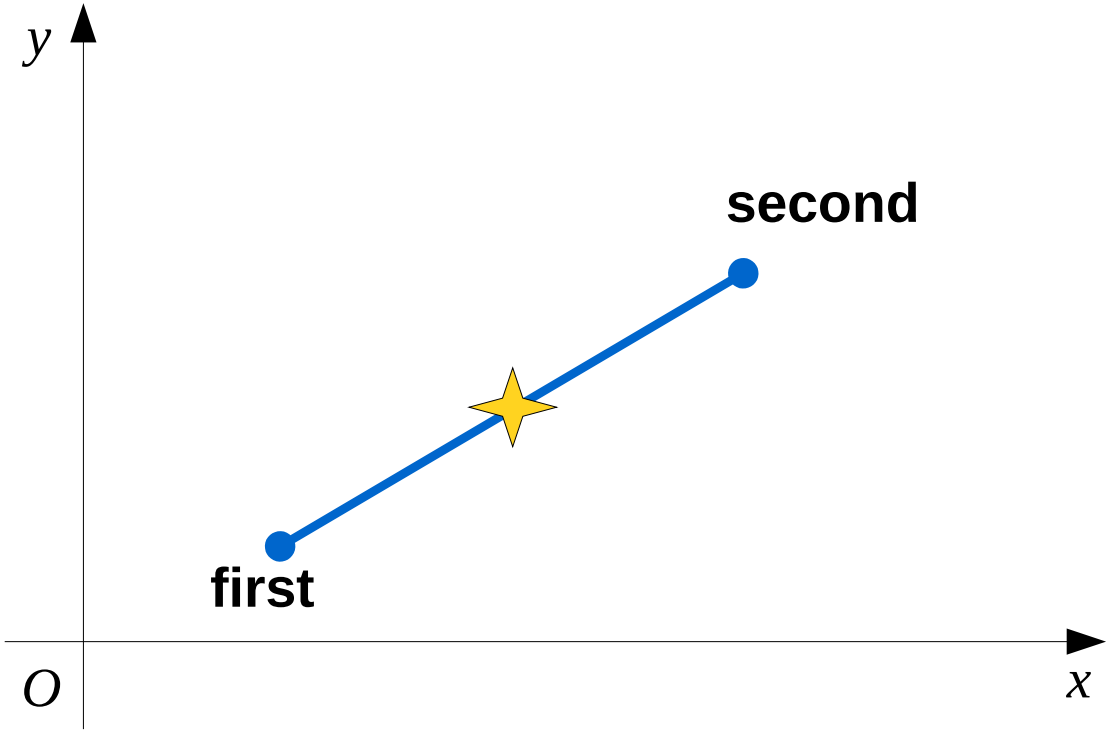
«Середина отрезка» (1)

```
//center.cpp
//расчёт середины отрезка на декартовой плоскости
#include <iostream>
using namespace std;

struct Point {
   double x;
   double y;
};
```

«Середина отрезка» (1)

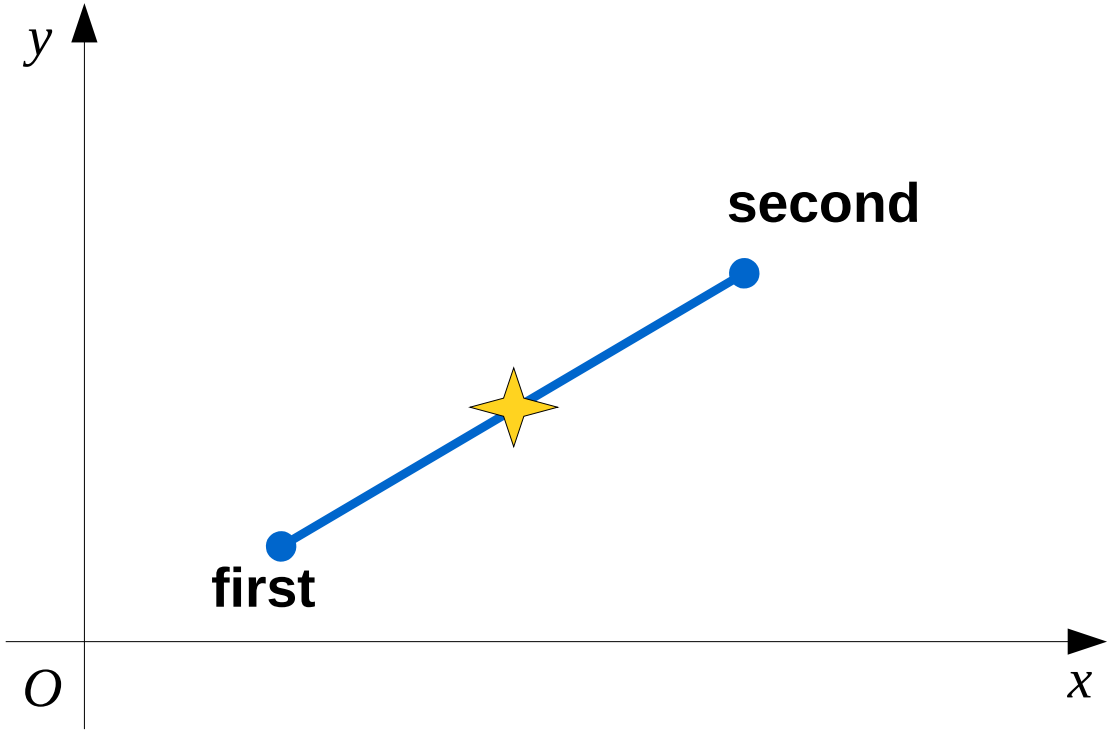
```
//center.cpp
//расчёт середины отрезка на декартовой плоскости
#include <iostream>
using namespace std;

struct Point {
   double x;
   double y;
};

struct Section {
    Point first;
    Point second;
};
```

А что с ними делать?



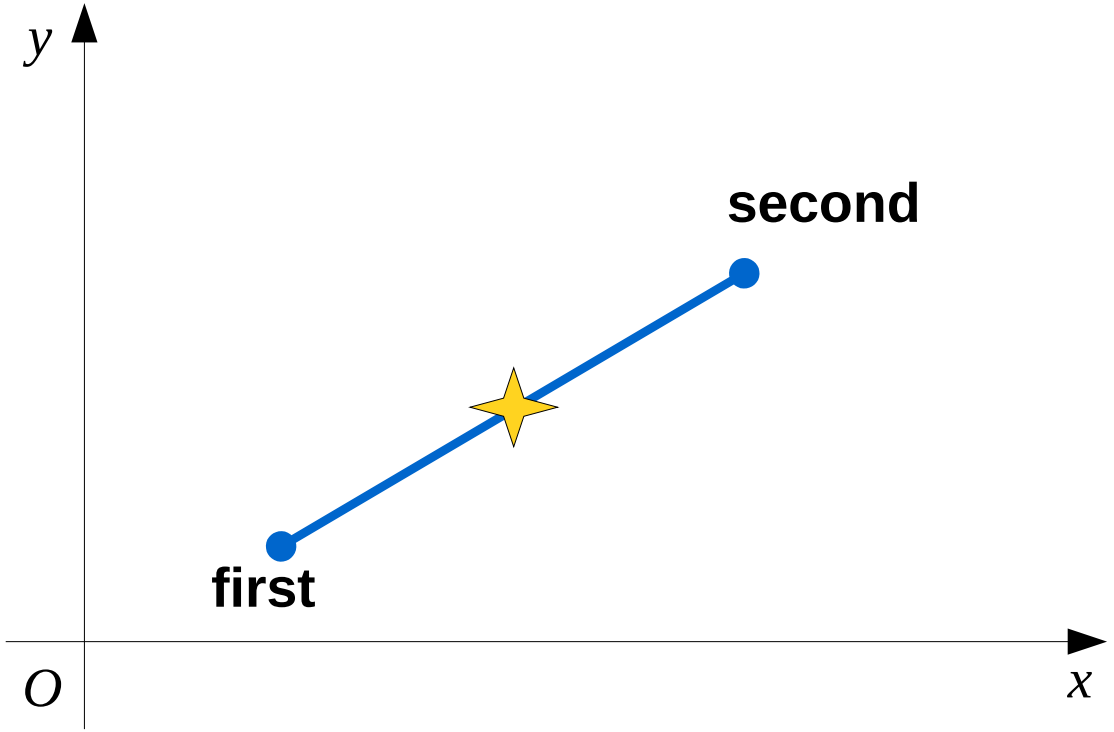
- ✔ Разработать программу расчёта середины заданного отрезка на декартовой плоскости.
- ✔ Координаты отрезка запрашиваются у пользователя
- ✔ Найденное значение должно выдаваться на стандартное устройство вывода.

А что с ними делать?



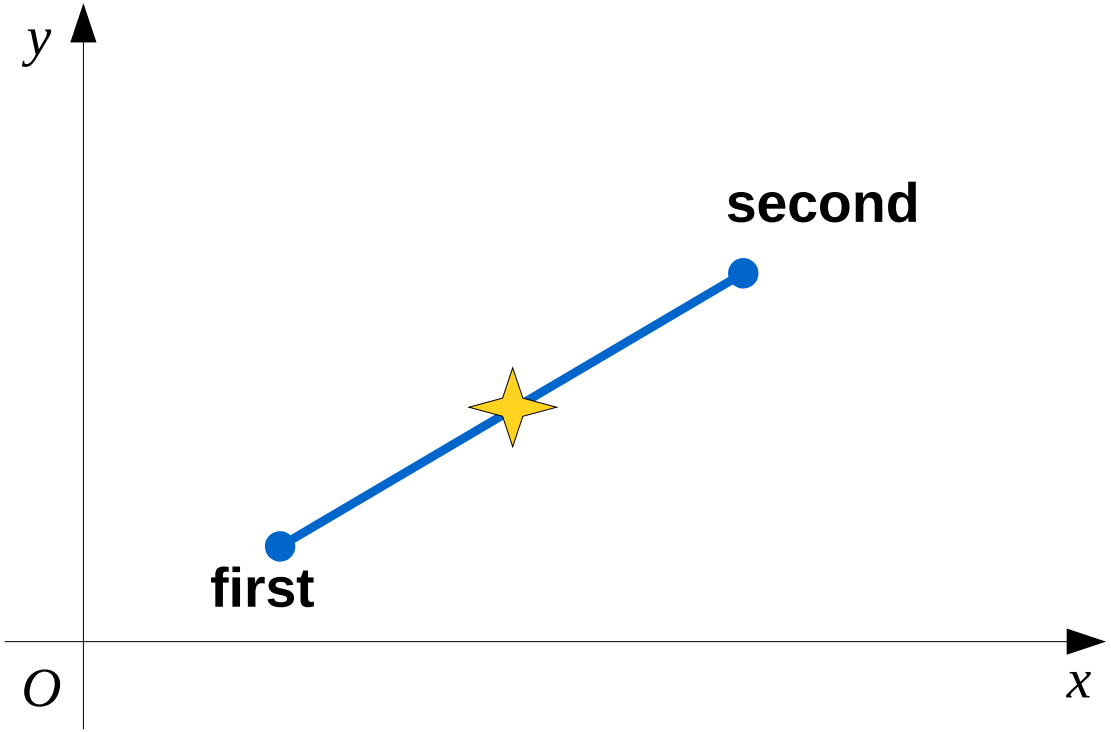
- ✔ Разработать программу расчёта середины заданного отрезка на декартовой плоскости.
- ✔ Координаты отрезка запрашиваются у пользователя
- ✔ Найденное значение должно выдаваться на стандартное устройство вывода.

А что с ними делать?



- ✔ Разработать программу расчёта середины заданного отрезка на декартовой плоскости.
- ✔ Координаты отрезка запрашиваются у пользователя
- ✔ Найденное значение должно выдаваться на стандартное устройство вывода.

А что с ними делать?



- ✔ Разработать программу расчёта середины ② заданного отрезка на декартовой плоскости.
- ✔ Координаты отрезка запрашиваются ① у пользователя
- ✔ Найденное значение должно выдаваться ③ на стандартное устройство вывода.

Алгоритм решения задачи

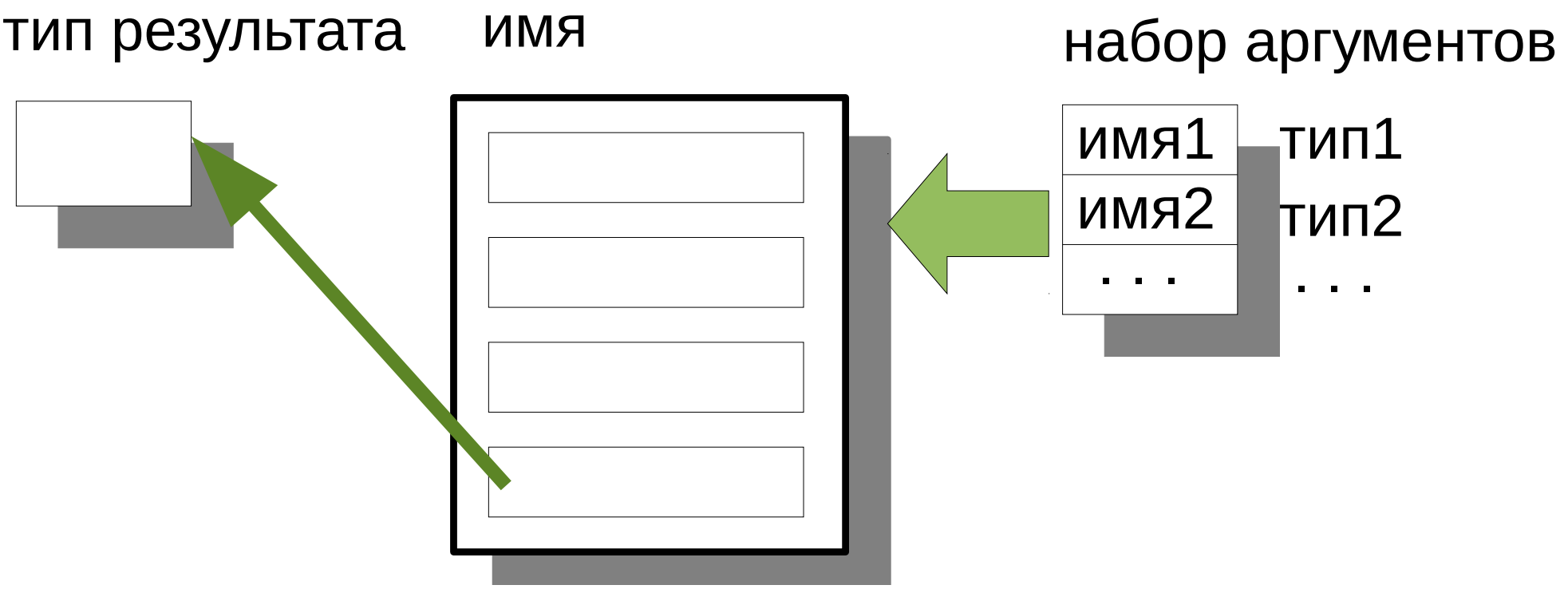
1. Ввод координат отрезка
2. Расчет центра отрезка
3. Выдача найденного центра

Алгоритм решения задачи

1. Ввод координат отрезка
2. Расчет центра отрезка
3. Выдача найденного центра

Функции

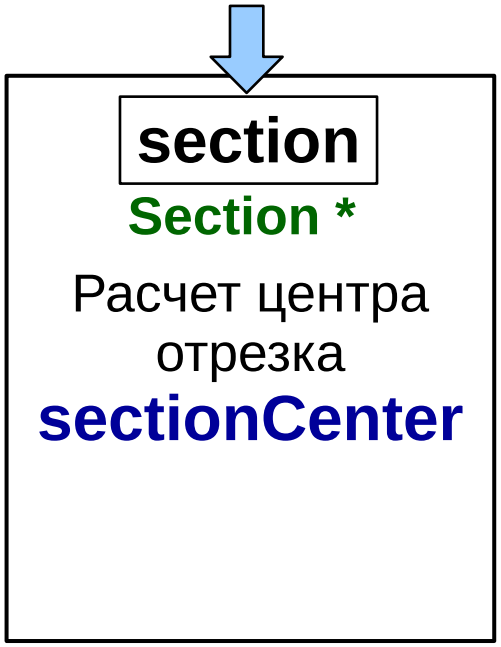
Структура функции



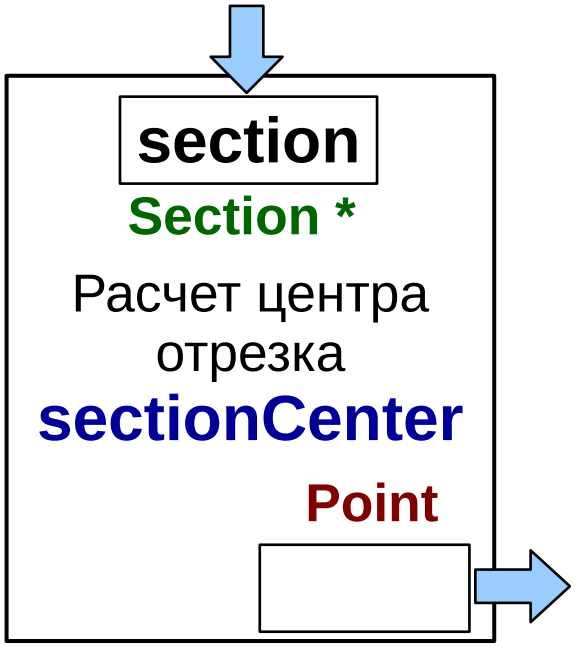
Проектирование функций

Расчет центра отрезка
sectionCenter

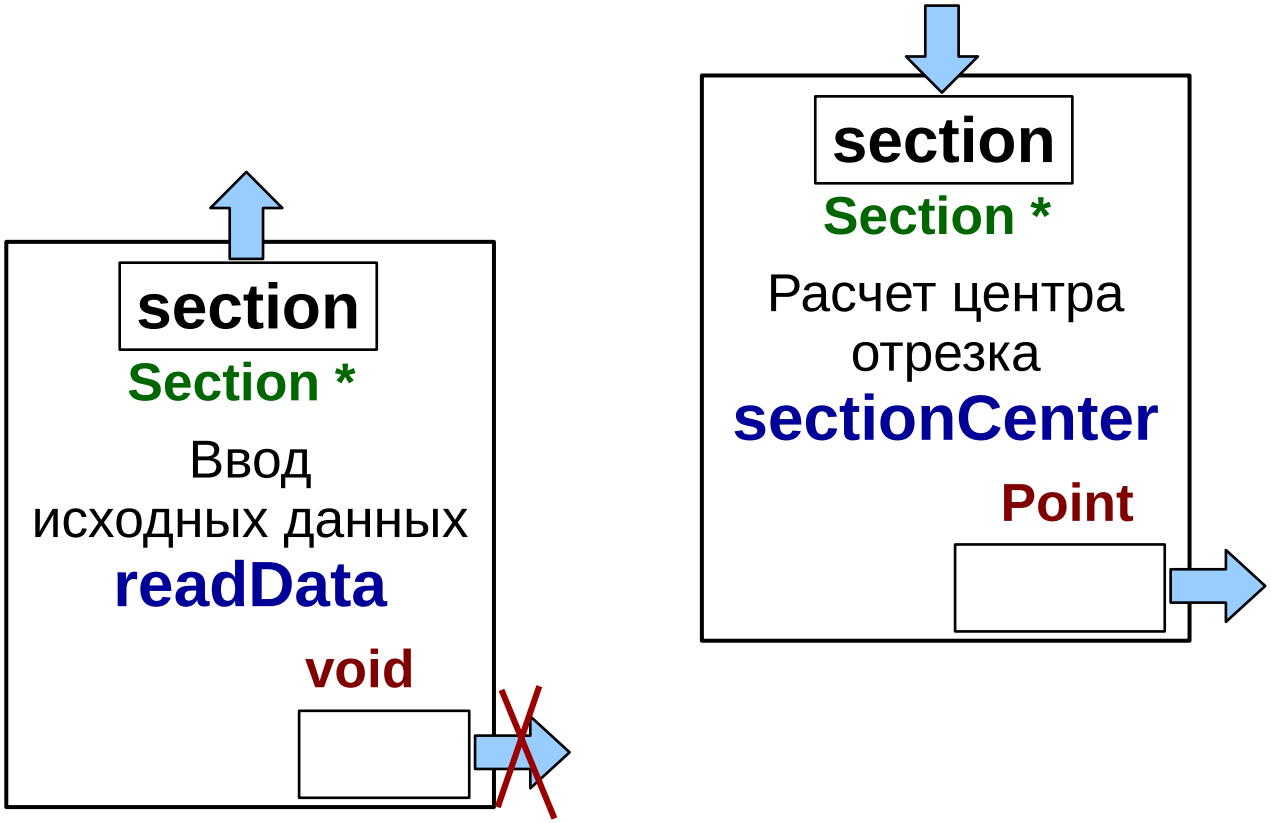
Проектирование функций



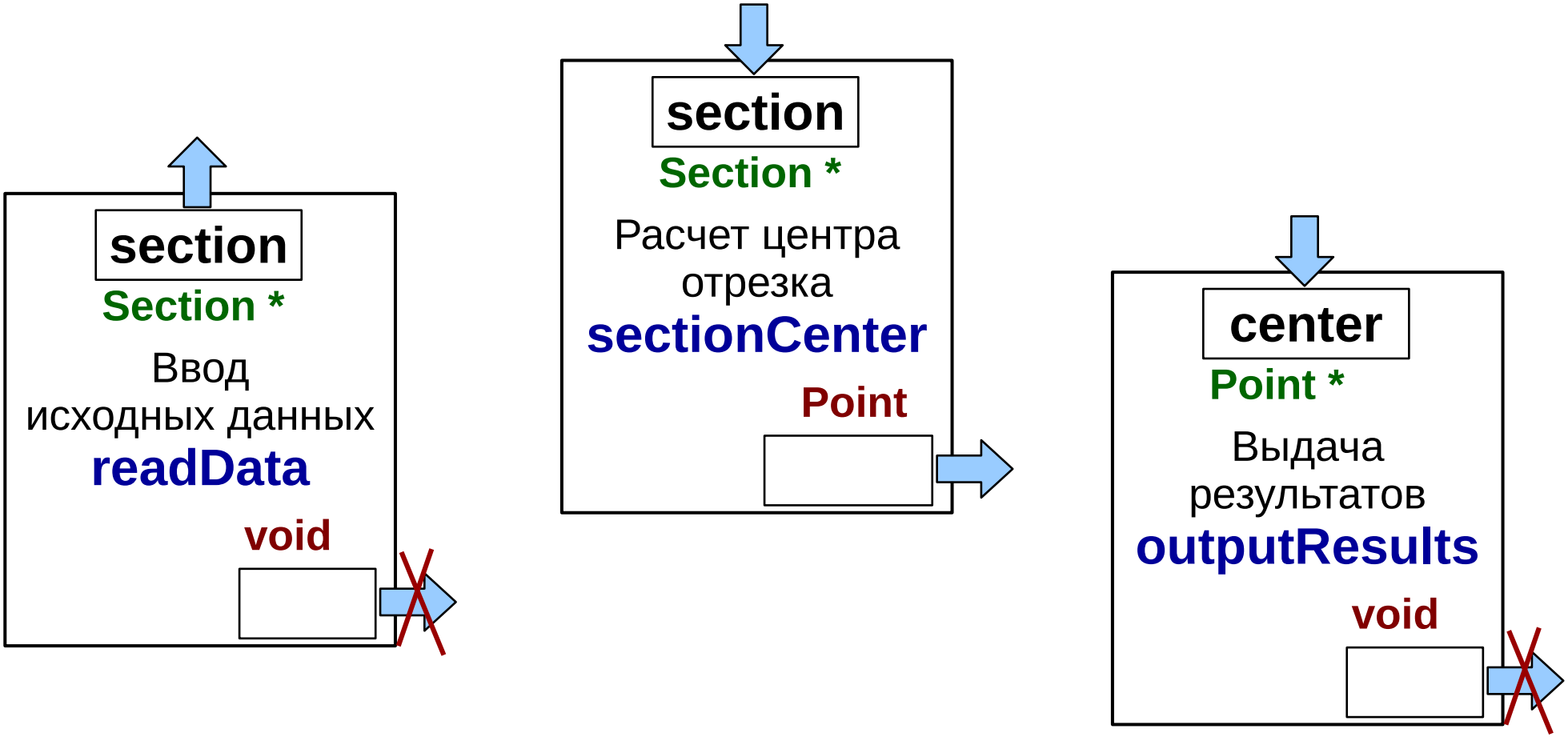
Проектирование функций



Проектирование функций



Проектирование функций

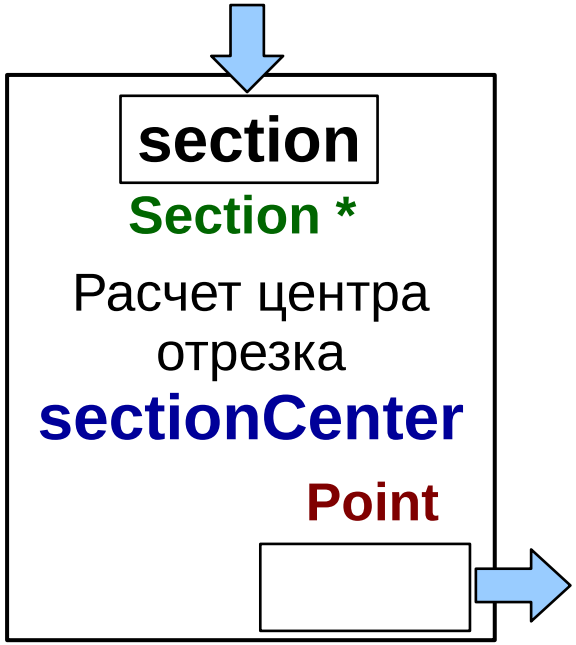


Описание функции

```
типРезультата имя(тип1 имя1, тип2 имя2, … типN имяN);
```

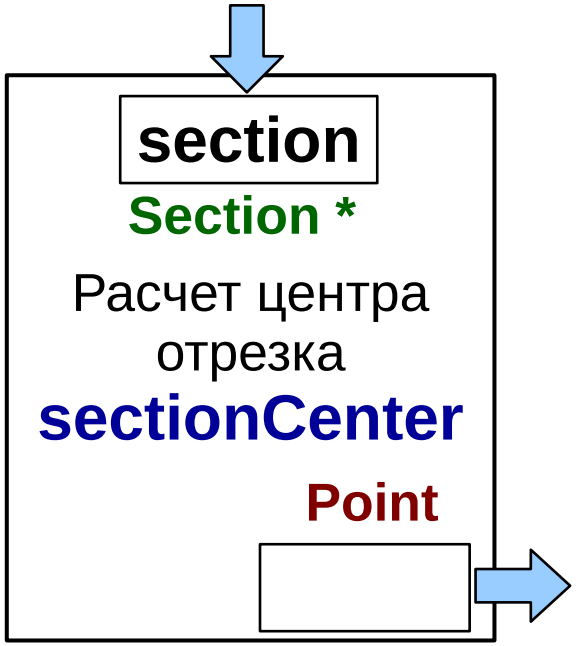
Описание функции

```
типРезультата имя(тип1 имя1, тип2 имя2, … типN имяN);
```



Описание функции

```
типРезультата имя(тип1 имя1, тип2 имя2, … типN имяN);
```




```
Point sectionCenter(Section* section);
```

тип результата — Выполняемое действие

имя

аргумент — Необходимая для выполнения информация

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
```

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
void readData(Section* section);
```

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
void readData(Section* section);
void outputResults(Point* center);
```

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
void readData(Section* section);
void outputResults(Point* center);

int main() {
```

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
void readData(Section* section);
void outputResults(Point* center);

int main() {

  Section section;
  readData(&section);
```

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
void readData(Section* section);
void outputResults(Point* center);

int main() {

  Section section;
  readData(&section);

  Point center;
  center = sectionCenter(&section);
```

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
void readData(Section* section);
void outputResults(Point* center);

int main() {

  Section section;
  readData(&section);

  Point center;
  center = sectionCenter(&section);

  outputResults(&center);
```

«Середина отрезка» (2)

```
Point sectionCenter(Section* section);
void readData(Section* section);
void outputResults(Point* center);

int main() {

  Section section;
  readData(&section);

  Point center;
  center = sectionCenter(&section);

  outputResults(&center);
  return 0;
}
```

Определение функции

```
типРезультата имя(тип1 имя1, тип2 имя2, … типN имяN) {

    //тело функции – последовательность операторов

}
```

«Середина отрезка» (3)

Заголовок — описание функции без **;**

```
типРезультата имя(тип1 имя1, тип2 имя2, … типN имяN) {

    //тело функции – последовательность операторов

}
```

```
Point sectionCenter(Section* section)
```

«Середина отрезка» (3)

```
типРезультата имя(тип1 имя1, тип2 имя2, … типN имяN) {
    //тело функции – последовательность операторов
}
```

```
Point sectionCenter(Section* section) {

    Point center;
    center.x = (section->first.x + section->second.x) / 2;
    center.y = (section->first.y + section->second.y) / 2;

    return center;

}
```

Тело функции

«Середина отрезка» (3)

```
типРезультата имя(тип1 имя1, тип2 имя2, … типN имяN) {

    //тело функции – последовательность операторов

}
```

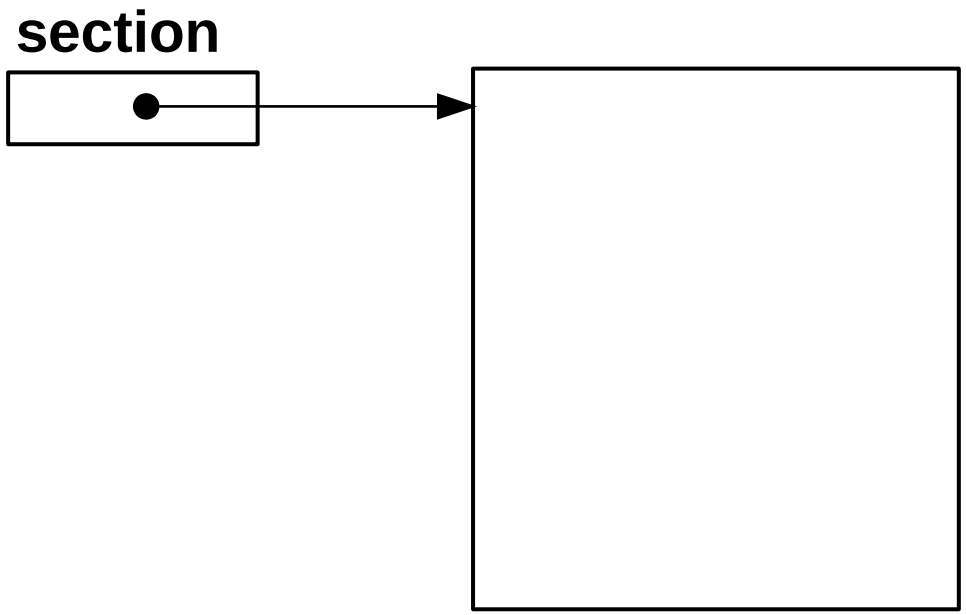
```
Point sectionCenter(Section* section) {

    Point center;
    center.x = (section->first.x + section->second.x) / 2;
    center.y = (section->first.y + section->second.y) / 2;

    return center;

}
```

Доступ к полю координаты


```
center.x = (section->first.x + section->second.x) / 2;
```

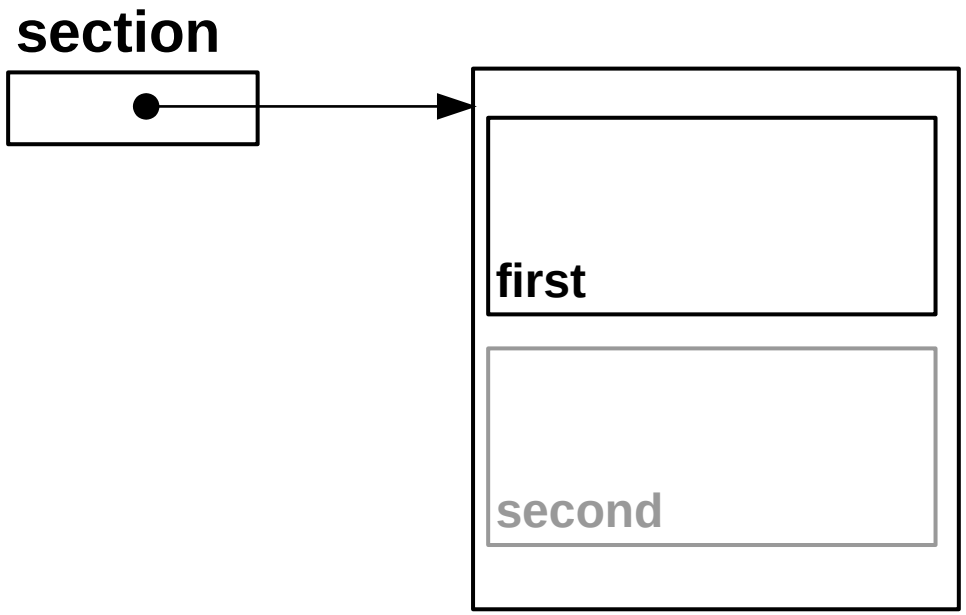
Доступ к полю координаты

```
center.x = (section->first.x + section->second.x) / 2;
```



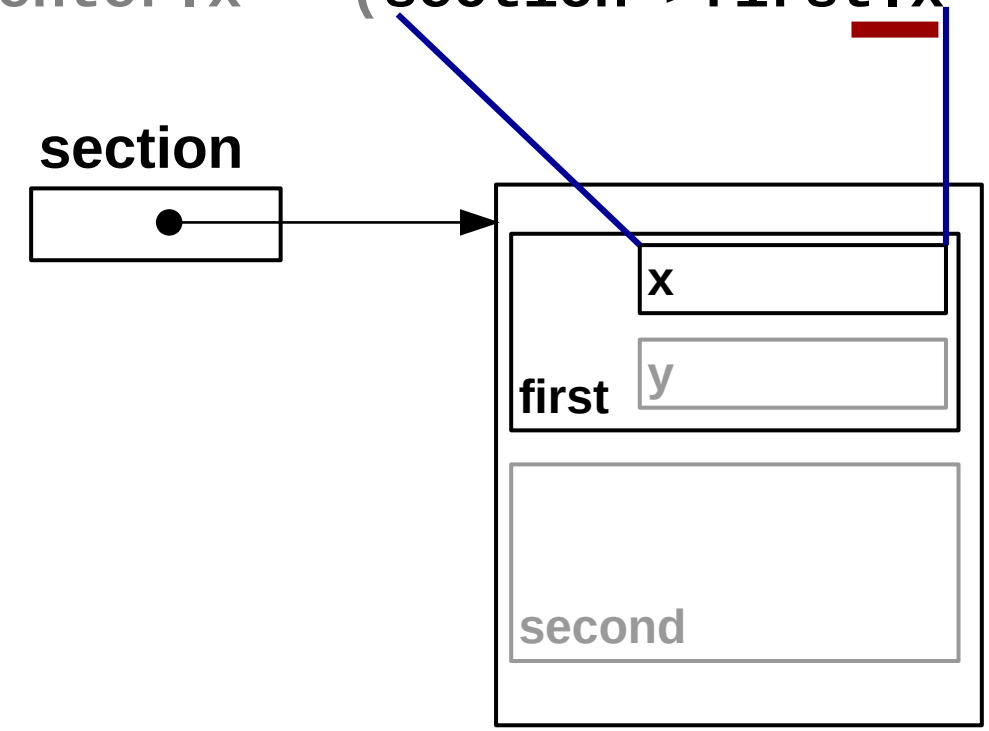
Доступ к полю координаты

```
center.x = (section->first.x + section->second.x) / 2;
```



Доступ к полю координаты

```
center.x = (section->first.x + section->second.x) / 2;
```



«Середина отрезка» (4)

```
void readData(Section* section) {
```

«Середина отрезка» (4)

```
void readData(Section* section) {

  cout << "введите координаты начала отрезка: ";
  cin >> section->first.x >> section->first.y;
```

«Середина отрезка» (4)

```
void readData(Section* section) {

  cout << "введите координаты начала отрезка: ";
  cin >> section->first.x >> section->first.y;
  cout << "введите координаты конца отрезка: ";
  cin >> section->second.x >> section->second.y;
  return;
}
```

«Середина отрезка» (4)

```
void readData(Section* section) {

  cout << "введите координаты начала отрезка: ";
  cin >> section->first.x >> section->first.y;
  cout << "введите координаты конца отрезка: ";
  cin >> section->second.x >> section->second.y;
  return;
}

void outputResults(Point* center) {
```

«Середина отрезка» (4)

```
void readData(Section* section) {

  cout << "введите координаты начала отрезка: ";
  cin >> section->first.x >> section->first.y;
  cout << "введите координаты конца отрезка: ";
  cin >> section->second.x >> section->second.y;
  return;
}

void outputResults(Point* center) {

  cout << "координаты центра: "
       << center->x << ' ' << center->y << endl;
  return;
}
```

Использование структурного типа данных в функциях