Структуры

Семестр 1
Семинар 6

Структуры данных

```
struct name_of_structure {
    int i;
    char c;
};
```

- новый тип struct name_of_structure
  - name_of_structure - имя структуры
  - i, c - поля структуры
- слово struct
  - входит в имя типа (C)

    ```
    struct name_of_structure example_structure;
    ```
  - можно опустить (C++)

    ```
    name_of_structure example_structure;
    ```

Обращение к полю структуры

```
struct name_of_structure {
    int i;
    char c;
} a, b;
    int main () {
    name_of_structure  x, y;
    y.c = 'z';
    x.i = 7;
    y.i = x.i + 3;
    x = y; //побитовое копирование
    printf ("%d  %c \n", x.i, x.c);
}
```

Декларация и инициализация

```
struct Point {
        int x;
        int y;
 } p1, p2;
Point maxp = {640, 480};


struct Rectange {
        Point lt;               // left-top
        Point rb;               // right-bottom
} rect = { {0, 0}, {640, 480} };
rect.lt.x = 0;
```

Структуры и функции

```
void incr (Point p) {
    p.x++;
}
int main ( ) {
    int i;        Point p;    p.x=1;
    for (i=0; i < 3; i++) {
        incr (p);
        printf (“p.x=%d\n”, p.x);
    }
    return 0;
}
```

Структуры и функции

```
void incr (Point * px) {
    px -> x ++; //(*px).x++
}
int main ( ) {
    int i;        Point p = {1, 'b'};
    for (i=0; i < 3; i++) {
          incr (&p);
          printf (“p=%d\n”, p.x);
    }
    return 0;
}
```

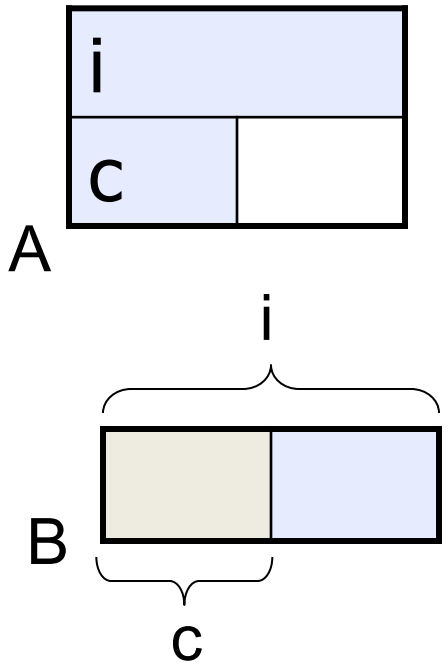
Массивы структур

```
Point arr [3];
Point b [ ] = {
        {0, 0},
        {640, 480}
};

b[1].x = 1024;
```

union

- struct A {
  int i;
  char c;
  };
- union B {
  int i;
  char c;
  };



все поля начинаются с одного места в памяти

Методы структуры

```
struct Point {
        float x;
        float y;
        float abs()
        {
                return sqrt(x*x + y*y);
        }
 } p1, p2;
…
float f = p1.abs();
```

Перегрузка функций

```
double sqrt ( double x );
//Функция корня для чисел с плавающей точкой
int sqrt ( int x );
//Функция корня для целых чисел


...
sqrt(1.5);
//В этом случае вызовется функция чисел с
//плавающей точкой
sqrt(7);
//А в этом уже для целых чисел
```

Переопределение операторов

бинарных

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

```
…
Point p1 = {3, 4}, p2 = {3, 1}, p3;
p3.x = p1.x + p2.x;
p3.y = p1.y + p2.y;
```

Переопределение операторов

бинарных

```
struct Point {
        float x;
        float y;
};
Point operator+(Point a, Point b) {
        Point tmp = {a.x + b.x, a.y + b.y};
        return tmp;
}
…
Point p1 = {3, 4}, p2 = {3, 1};
Point s = p1 + p2;
```

Переопределение операторов

унарных

```
struct Point {
        float x;
        float y;
} p1;
Point operator-(Point a)
{
        Point _a = {-a.x, -a.y};
        return _a;
}
…
Point s = -p1;
```

Переопределение операторов

унарных

```
struct Point {
        float x;
        float y;
        Point operator-()
        {
                Point _a = {-this->x, -this->y};
                return _a;
        }
} p1;
…
Point s = -p1;
```

Переопределение операторов

сравнения

```
struct Point {
        float x;
        float y;
};
int operator==(Point a, Point b)
{
        return a.x == b.x && a.y == b.y;
}
…
Point p1 = {3, 4}, p2 = {3, 1};
if (p1 == p2)
```

Переопределение операторов

какие можно?

```
+ - * / %                    //Арифметические операторы
+= -= *= /= %=
+a -a                        //Операторы знака
++a a++ --a a--              //Префиксные и постфиксные
                             //инкремент и декремент
&& || !                      //Логические операторы
& | ~ ^
&= |= ^=
<< >> <<= >>=                //Битовый сдвиг
=                            //Оператор присваивания
== !=                        //Операторы сравнения
< > >= <=


&a *a a-> a->*               //Адресация
() []
(type)                       //Приведение типа
```

Переопределение операторов

инкремента/декремента

```
//префиксный
Point & operator ++() {
        ...
        return *this;
}
//постфиксный
Point & operator ++( int n ) {
        Point t = *this;
        ++(*this);
        return t;
}
```

Переопределение операторов

общие правила и рекомендации

- Унарные операторы(+=, -=, *= и т.д.) - внутри класса/структуры.
- Бинарные операторы(+, -, * и т.д.) (можно на основе унарных) – снаружи класса/структуры.
- Для логических операторов можно использовать bool.

- Нельзя
  - определять новые операторы
  - менять приоритет операторов

Задачи

- Календарь. С клавиатуры вводится целое число N и N имен с датами рождения. Имя – строка, длина не более 10 символов; дата – два целых числа. После этого вводится одно целое число M. Вывести фамилии всех, кто родился в месяце M.

Пример ввода:	4 Alex 10 31 Petr 1 1 July 7 7 Vlad 7 22 7	3 Yury 6 22 Mary 11 15 Lev 2 29 9
Пример вывода:	July Vlad	---

- Поиск в календаре даты по фамилии.

Задачи

- Комплексные числа. Создать тип данных, в котором будет храниться комплексное число. Написать функции, которые буду реализовывать:
  - сложение/вычитание
  - умножение
  - деление
  - поиск модуля числа
- Векторы. Создать тип данных, в котором будет храниться двухмерный/трехмерный вектор. Написать функции, которые буду реализовывать:
  - сложение/вычитание
  - умножение
  - деление
  - поиск модуля
  - скалярное произведение
  - векторное произведение (для 3Д)