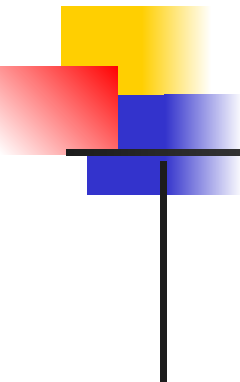


Программирование на языке Си

Лекция 9

Валиханов М.М., 2014г

- 
- Язык Си (Си++) является высокоуровневым искусственным языком для разработки программ.
 - Си-подобные языки используются для программирования различного вида аппаратуры.
 - Главное:

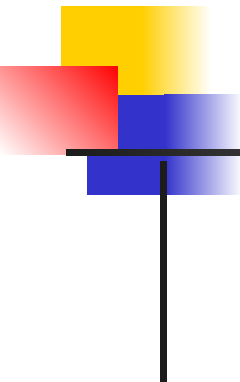
Развитие ПО опережает
развитие аппаратуры.

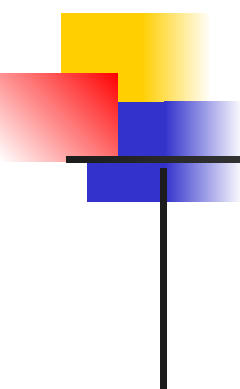
Стиль программирования

- организация программного кода в соответствии с определенными правилами.

```
int a=0,w,x=0;
while (a==0)
{ w=w+1;
x--;
}
```

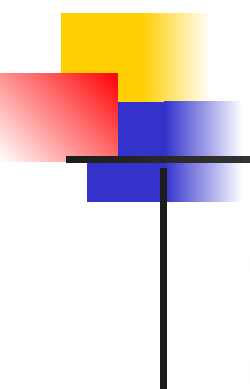
```
int a=0;
int w;
int x=0;
while (a==0)
{
    w=w+1; //Error
    x--;
}
```

- 
- Стиль программирования вырабатывается по мере получения опыта и зависит от самодисциплины программиста.
 - Единого стиля нет. Существуют рекомендации, которым следует придерживаться, так как они были выработаны в ходе программирования миллионов строк кода миллионами людей.



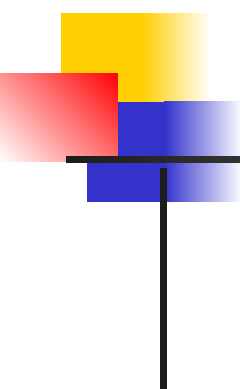
Основная цель стиля программирования

- Это организация кода:
 - легче искать ошибки;
 - легче и быстрее разбираться в своей и чужой программе;
 - «красота программного кода».



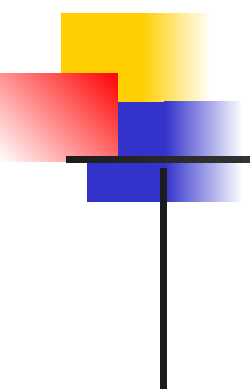
Рекомендации

- корректность, ясность и простота;
- имена по Венгерской нотации;
- комментируйте и документируйте код;
- не допускайте дублирования кода;
- инициализировать переменные перед их использованием;
- не используйте неизвестные аргументы в функциях (троеточия);
- устраняйте все ошибки и **замечания**;
- одна функция – одна задача.



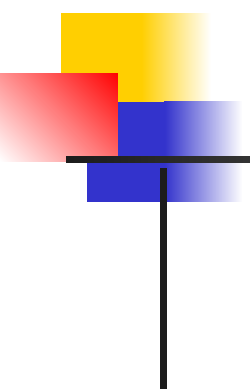
Структуры в Си

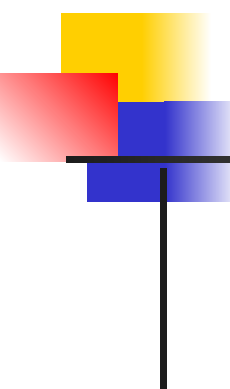
- это сложный тип данных представляющий собой упорядоченное в памяти множество элементов различного типа, **или** тип данных, позволяющий включать другие типы данных, в том числе структуры.



Синтаксис :

```
struct [имя типа]
{
    поле_1;
    поле_2;
    ...
    поле_N;
} [список переменных];
```

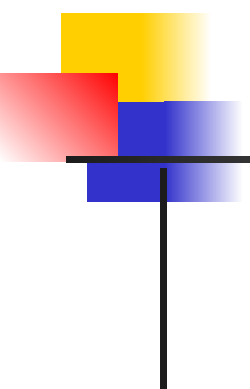
- 
- Структура представляет собой **объект** с набором различных параметров – полей.
 - Объектами могут быть различные предметы из реального или программного мира.
 - Обычно структуры записывают в отдельный модуль «.h» файл.
 - Например, любой автомобиль можно описать с помощью набора определенных свойств (параметров, полей).



Пример объявления структуры

```
struct _Auto
{
    int year;
    int color;
    char title[100];
}; ←!!!
```

```
void main(void)
{
    _Auto myAuto;
}
```

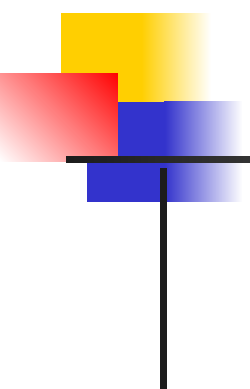


Обращение к полям

- выполняется с использованием оператора точки «.» для записи и извлечения значения;

```
myAuto.year = 2014;  
myAuto.color = 0;    //0-черный  
strcpy(myAuto.title, "Volga");
```

```
printf("%d", myAuto.year);
```



Размер структуры

- Используется функция `sizeof()`:

Синтаксис :

`sizeof`(имя структуры)

Например,

```
int sizeAuto = sizeof(myAuto);  
//sizeAuto=108 байт
```



Копирование структур

- Функция `memcpy` (небезопасная) или `memmove` (безопасная).

Синтаксис :

`memcpy` (адрес назначения, адрес источника, размер);

`memmove` (адрес назначения, адрес источника, размер);

```
_Auto      myAuto2;  
memcpy(&myAuto2, &myAuto, sizeof(_Auto));  
memmove(&myAuto2, &myAuto, sizeof(_Auto));
```



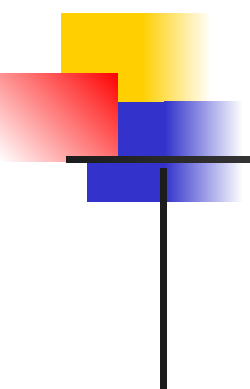
Массив структур

```
struct _Auto
{
    unsigned char actual;
    int year;
    int color;
```

```
const int NAUTO=10;
_Auto    allAuto[NAUTO];
int      iAuto;
for ( iAuto=0; iAuto<NAUTO; iAuto++ ) {
    if (allAuto[iAuto].actual) {
        printf ("%d %d %s",allAuto[iAuto].year,
            allAuto[iAuto].color,
            allAuto[iAuto].title);
    }
}
```

Функции в структурах

```
struct _Auto
{
    unsigned char actual;
    int year;
    int color;
    char title[100];
    _Auto() //Функция установки полей
    {
        //по умолчанию - конструктор
        actual = 1;
        year = 2001;
        color = 0;
        strcpy(title, «NoName»);
    }
};
```



Поля структур могут быть

- Открытого типа – public;
- Закрытого типа – private.

Инкапсуляция – разделение прав доступа к полям структуры.

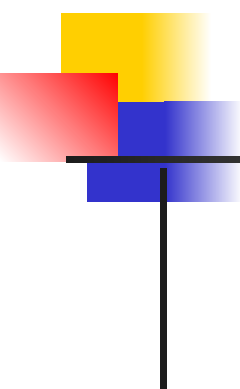
В полях с Private напрямую извне изменить или прочитать данные из полей нельзя, только через функции структуры.

```
struct _Auto
{
public:
    const int nTitle = 100;
    _Auto()
    {
        year = 2001;
        color = 0;
        strcpy(title, "sf");
    }
    void setYear( int year_)
    {
        year = year_;
    }
    int getYear(void)
    {
        return year;
    }
private:
    int year;
    int color;
    int title[nTitle];
};
```

В **public** указываются, например, константы, а также функции которые могут вызываться извне. Конструктор в **public** указывать обязательно.

```
//Пример
_Auto myAuto;
myAuto.setYear(2010);
```

Блок **private** рекомендуется указывать после **public**



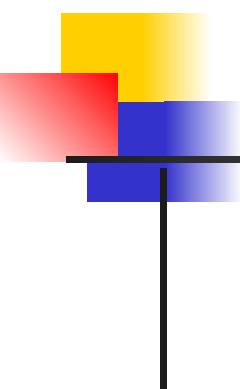
Объединения - union

- Объединения - это объект, позволяющий нескольким переменным различных типов занимать один участок памяти.
- Все элементы объединения начинаются с одного адреса.
- Размер объединения равен максимальному размеру типу данных.

Разделение целого числа на байты

4 3 2 1

```
void main(void)
{
    _Int32ForByte l4b;
    l4b.int32=0x04030201; //0x - 16ричная запись
    printf("%d %d %d %d", l4b.b[3], l4b.b[2],
    l4b.b[1],l4b.b[0]);
}
```



Другие виды типов данных

- Перечисления - enum.
- Битовые поля структур - int:1.