СТРУКТУРЫ

Структура – это совокупность логически связанных данных, возможно, различных типов, сгруппированных под одним именем для удобства дальнейшей обработки.

```
struct [<НАЗВАНИЕ_СТРУКТУРЫ>]
{
   <Тип_данных> <имя_поля1>;
   . . .
   <Тип_данных> <имя_поляN>;
} [<Объявления переменных через запятую>];
```

```
<НАЗВАНИЕ_СТРУКТУРЫ> <Названия переменных через запятую>;
```

```
<Размер> = sizeof <Переменная>;
<Размер> = sizeof (<НАЗВАНИЕ_СТРУКТУРЫ>);
```

```
using TMarks = int[5];
struct BIRTH_DAY
{
     int day, month, year;
};
struct STUDENT
{
     std::string FIO;
     BIRTH_DAY birth_day;
     short course, group;
     TMarks marks;
} Student;
```

```
STUDENT new_Student={"Иванов И.И.",17,05,1998,1,2,4,5,4,5,4};
```

Обращение к полям структуры

```
<Имя_объекта>.<имя_члена>
```

```
cout << "Ф.И.О.: " << Student.FIO << endl;
cout << "Дата рождения: " << Student.birth_day.day << '.'
                          << Student.birth_day.month << '.'
                          << Student.birth_day.year << endl;
```

Одну структуру можно присвоить другой структуре с помощью оператора =. В этом случае копируются значения всех полей структуры.

Ввод / вывод структуры

```
void Read (STUDENT &Student)
{
     cin >> Student.FIO;
     cin >> Student.birth_day.day;
     cin >> Student.birth_day.month;
     cin >> Student.birth_day.year;
     cin >> Student.course;
     cin >> Student.group;
     for (short i=0; i<5; i++)
          cin >> Student.marks[i];
}

void Print (STUDENT Student)
{
     cout << Student.FIO << endl;
     cout << Student.birth_day.day << '.' << Student.birth_day.month
<< '.' << Student.birth_day.year << endl;
     cout << Student.course << endl;
     cout << Student.group << endl;
     cout << "Оценки:" << endl;
     for (short i=0; i<5; i++)
          cout << Student.marks[i] << ' ';
     cout << endl;
     cout << "------------------------\n";
}
```

Объявление указателя на структуру производится также как и на любой другой тип данных. Для получения адреса структуры используется оператор &, а для доступа к полю структуры вместо точки применяется оператор –>

```
STUDENT *pStudent = &Student; // указатель на существующую структуру

std::cout << pStudent->FIO << '\n';
std::cout << (*pStudent).birth_day.day << '\n';

STUDENT *pStudent2 = new STUDENT; // указатель на новую структуру

delete pStudent2;
```

Массив структур

```
STUDENT Faculty[3]; // статический массив
cout << Faculty[0].FIO << endl;

STUDENT *pFaculty = new STUDENT[3]; // динамический массив
cout << pFaculty[0].group << endl;

delete [] pFaculty;
```

Методы

```
using TMarks = int[5];

struct BIRTH_DAY
{
    int day, month, year;
};

struct STUDENT
{
private:
    std::string FIO;
    BIRTH_DAY birth_day;
    short course;
    short group;
    TMarks marks;
public:
    // неинициализирующий конструктор
    STUDENT() {}
    // инициализирующий конструктор
    STUDENT(std::string FIO, BIRTH_DAY birth_day_, short course_,
short group, TMarks marks_) : group(group)
    {
        this->FIO = FIO;
        birth_day = birth_day_;
        course = course_;
        group = group_;
        for (short i = 0; i<5; i++)
            marks[i] = marks_[i];
    }
    void set_FIO(const std::string & FIO)
    {
        this->FIO = FIO;
    }
```

```
    std::string get_FIO()
    {
        return FIO;
    }
    // метод печати (реализация метода вне структуры)
    void print();
    // метод сравнения по полю FIO
    int compare(STUDENT *new_student);
};

void STUDENT::print()
{
    cout << "Ф.И.О.: " << FIO << endl;
    cout << "Дата рождения: " << birth_day.day << '.' <<
birth_day.month << '.' << birth_day.year << endl;
    cout << "Курс:   " << course << endl;
    cout << "Группа: " << group << endl;
    cout << "Оценки:" << endl;
    for (short i=0; i<5; i++)
        cout << marks[i] << ' ';
    cout << endl;
    cout << "-------------------------\n";
}

int STUDENT::compare(STUDENT *new_student)
{
    if (FIO==new_student->FIO)
        return 0;
    else
        if (FIO>new_student->.FIO)
            return 1;
        else
            return -1;
}
```

```
int main()
{
     SetConsoleOutputCP(1251);
     SetConsoleCP(1251);

     // объявление статической переменной
     // вызывается неинициализирующий конструктор
     STUDENT Student;

     // объявление указателя на структуру
     // вызывается инициализирующий конструктор
     BIRTH_DAY BD={11,07,1998};
     TMarks MK = {4,5,4,5,5};
     STUDENT *pStudent = new STUDENT("Иванов",BD,1,2,MK);

     pStudent->print();

     BIRTH_DAY BD2={22,12,1999};
     TMarks MK2 = {5,5,5,5,5};
     STUDENT *pStudent2 = new STUDENT("Петров",BD2,2,3,MK2);

     if (pStudent->compare(pStudent2)<0)
          cout << pStudent->FIO << '<' << pStudent2->FIO;
     else
          if (pStudent->compare(pStudent2)>0)
               cout << pStudent->FIO << '>' << pStudent2->FIO;
          else
               cout << pStudent->FIO << '=' << pStudent2->FIO;

     delete pStudent;
     delete pStudent2;

     return 0;
}
```

Результат:

```
Ф.И.О.: Иванов
Дата рождения: 11.7.1998
Курс:   1
Группа: 2
Оценки:
4 5 4 5 5
-------------------------
Иванов<Петров
```

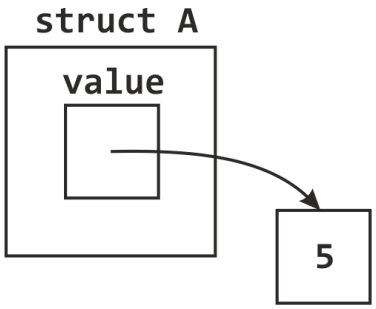
Передача и возврат структур в функцию (из функции)

```
#include <iostream>

struct A
{
	int* value;
	A(int x)
	{
		value = new int(x);
	};
	~A()
	{
		delete value;
	}
};
```

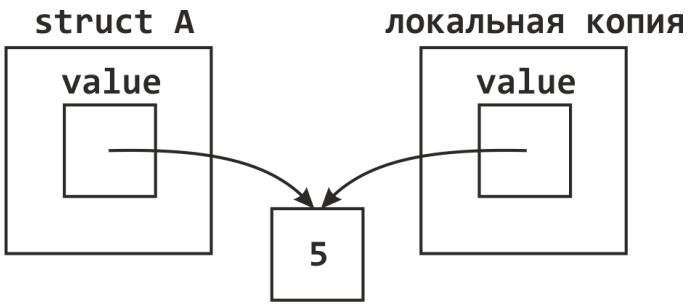
Создание структурной переменной:

```
A a(5);
```

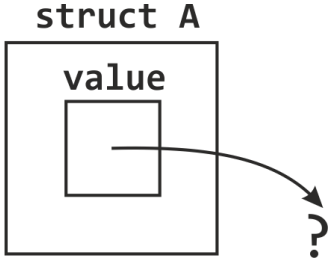


Создание локальной копии при передаче по значению:

```
void print_(A a)
```



При выходе из функции локальная копия деструктурируется (вызывается деструктор), память для параметра `value` освобождается:



При завершении работы функции `main()` происходит деструкция структурной переменной и попытка повторного освобождения памяти для параметра `value`, что приводит к аварийному завершению работы программы.

Если структура не должна изменяться внутри функции, то при передаче по ссылке используется спецификация класса памяти `const`.

```
void print_(const A &a)
{
    std::cout << *(a.value) << '\n';
}

int main()
{
    A a(5);
    print_(a);

    std::cin.get();
    return 0;
}
```

Лямбда функции в структурах

Члены структуры нельзя захватывать непосредственно. Если к ним необходим доступ из лямбда-функции, то необходимо захватывать указатель this, включив его в список захвата.

```
struct MyStruct
{
    int info;
    void change(int value)
    {
        info += value;
        // через лямбду
        auto Lambda = [this](int value) {info += value; };
    }
};
```