Типове данни

- Още на първата консултация направихме едно условно разграничение на типовете данни:
  - примитивни
  - съставни

- Вече знаем кои са примитивните

- Тогава какво остава за съставни?

Съставни типове данни

- От самото име следва, че са съставени от нещо, но от какво

- Всеки съставен вид данни се състои или надгражда други типове данни, без значение дали са примитивни, или съставни

$\Rightarrow$Пойнтъри и масиви

Структура

- Добре дошли в курса по ООП (spoiler alert)
- Представя обединение от данни
- Винаги има фиксиран брой елементи
- Елементите могат да са от различни типове
- По подразбиране предоставя директен достъп до всеки елемент

Структура

- Подобно на функциите, при структурата има 2 ключови момента
- Декларация+дефиниция
- Извикване на дефинираното

Структура

- Декларация

struct <име>;

- Дефиниция + декларация

```
struct <име>
{
        [<тяло>]
};
```

Структура

- Също както при функции, съществува и така наречения forward declaration:

```
struct example;

……

struct example
{
};
```

Структура

- Нека разгледаме какво се случва при дефиниция

```
struct example
{
        //казахме, че тук може да има някакво тяло

};  //не забравяйте ; след края на scope на декларация на структура
```

Структура

- Нека разгледаме какво се случва при дефиниция

```
struct example
{
        //в тялото се записват така наречените член-данни и функции
        //сега ще говорим само за член-данни, а след време и за член-функции
        //нека example е съставена от 1 променлива от тип int
        int member1;
};  //не забравяйте ; след края на scope на декларация на структура
```

Структура

- Нека разгледаме какво се случва при дефиниция

```
struct example
{
        int member1, member2, member3;
        char member4;
        double members5To10[5];
};  //не забравяйте ; след края на scope на декларация на структура
```

Структура

- Нека първо преминем към втората част – извикване на структура, а после ще се върнем към дефиниране, за да разширим наученото

- Извикване на структура

<Име на структурата/тип> <наименование>;

- Примери:

  example a;

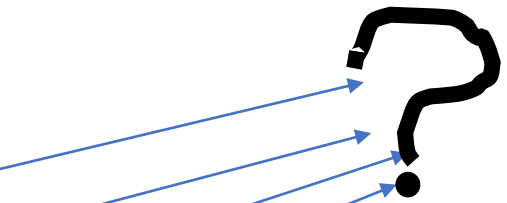
  example b;

Какво означава неинициализиран обект?

Структура

- Нека разгледаме какво се случва при дефиниция

```
struct example
{
        int member1, member2, member3;
        char member4;
        double members5To10[5];
};  //не забравяйте ; след края на scope на декларация на структура
```



Структура

- Извикване на структура

<Име на структурата/тип> <наименование>;

- При създаване на обект можем и да инициализираме член-данните по 3 начина, като по-използваният ще остане за курса по ООП

<Име на структурата/тип> <наименование> = { <стойност_за_член_1>, <стойност_за_член_2>,….};

- По този начин не може да се пропускат членове и може да се спре по всяко време

Структура

- Нека използваме за пример

```
struct example{
        int member1, member2, member3;
        char member4;
        double members5To10[5];
};
……….
example a = {1};                    //даваме стойност 1 на member1
example b = {1,2,3};                //даваме стойности на member1-3
example c = {1,2,3,'!'};            //даваме стойности на member1-4
example d = {1,2,3,'!',1,2,3,4,5}   //даваме стойности на member1-5
```

Структура

- Нека използваме за пример

```
struct example{
        int member1, member2, member3;
        char member4;
        double members5To10[5];
};
……….
example a = {1};
```

Знаем, че member1 има стойност 1, но как да я използваме?

Структура

- Нека пак имаме структурата example

example a = {1};

- Как да достъпим member1?

- Достъпът в този случай се осъществява с оператор .(точка) + име на член

- std::cout<<a.member1; //извежда 1

Структура

- Какво означава „в този случай“?
- Има случаи, в които обръщението към член данна не се осъществява с оператор .(точка)
- Има случаи, в които нямаме достъп до дадена член данна
- Надявам се някой ден да се реванширам и да ги обясня в презентация на тема ООП

Структура

- Какво ни дава достъпът до дадена член-данна?

- Достъп до информацията, която съдържа

- Възможност да променяме стойността ѝ ако не е константна

- Пример

a.member1 = 10;

Структура

- Как една член данна може да е константна

- Има 2 начина

1. Самата член данна да е константна
   - const int member11
2. Цялата структура да е създадена константна
   - const example b; //лош пример, ще има фойерверки

- Какви правила трябваше да спазваме при работа с константи?

Структура

- Когато създаваме константа винаги трябва да задаваме някаква стойност при инициализация.
- Структурите имат начин за справяне с този проблем, освен използването на { } при инициализация
- Спомняте ли си параметрите по подразбиране?
- Също като тях, членовете на структурата могат да имат стойности по подразбиране

Структура

- За разлика от параметрите по подразбиране на функциите, при структурите не е необходимо само първите n на брой да имат стойности по подразбиране
- Пример:

```
struct example{
        int member1 = 2, member2 = 6, member3 = 534;
        char member4;
        double members5To10[5] = {1,2,3,4,5};
};
```

Структура

- Пример:

```
struct example{
        int member1 = 2, member2 = 6, member3 = 534;
        char member4 = ‘?’;
        double members5To10[5] = {1,2,3,4,5};
};
example a;                          //example a = {2, 6, 534, ’?’, 1, 2, 3, 4, 5};
example b = {5,4};                  //example b = {5, 4, 534, ’?’, 1, 2, 3, 4, 5};
example c = {12,34,1, ‘%’, 5};      //example c = {12,34,1, ‘%’, 5, 2, 3, 4, 5};
```