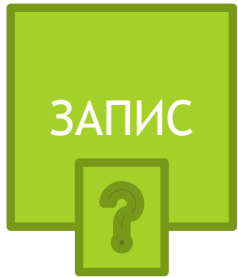


ЗАПИС (СТРУКТУРА)

гл.ас. д-р. Нора Ангелова

ЗАПИС (СТРУКТУРА)



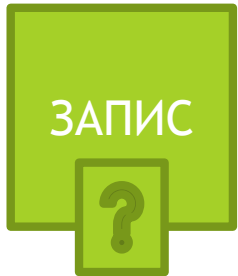
⦿ Логическо описание

Съставна, статична, хетерогенна структура от данни.

Определя се като крайна редица от фиксиран брой елементи, които могат да са от различни типове.

Достъпът до всеки елемент е пряк и се осъществява чрез име, наречено **поле** на записа.

ЗАПИС (СТРУКТУРА)



- ⦿ **Физическо представяне**

Полетата на записа се представят последователно в паметта.

ЗАПИС (СТРУКТУРА)

⦿ Дефиниция

```
struct <име_на_структура> {  
    <дефиниция_на_полета>;  
    {<дефиниция_на_полета>;} опц  
}[<променлива>] опц;
```

<дефиниция_на_полета> ::= <тип> <име_поле>
{, <име_поле>} опц

<име_на_структура>, <име_поле> ::= <идентификатор>

<променлива> ::= <идентификатор>

<тип> ::= <име_на_тип> | <дефиниция_на_тип>

ЗАПИС (СТРУКТУРА)

● Множество от стойности

Всички крайни редици от по толкова елемента, колкото са полетата му, като всеки елемент е от тип, съвместим с типа на съответното поле на записа.

Пример:

```
struct structName {  
    int name1;  
    double name2;  
};
```

Всички двойки от вида:
{int, double}

ЗАПИС (СТРУКТУРА)

- Дефиниране на променлива от тип запис (структура)

$\langle \text{деф_променлива_от_тип_структура} \rangle ::=$
 $\langle \text{име_на_структура} \rangle \langle \text{променлива} \rangle$
 $[= \{ \langle \text{редица_от_изрази} \rangle \}]_{\text{опц}}$
 $\{ , \langle \text{променлива} \rangle [= \{ \langle \text{редица_от_изрази} \rangle \}]_{\text{опц}} \}_{\text{опц}}$

$\langle \text{редица_от_изрази} \rangle ::=$
 $\langle \text{израз} \rangle \mid \langle \text{израз} \rangle , \langle \text{редица_от_изрази} \rangle$

Примери:

```
structName obj1 = {10, 14.5};
```

```
structName obj2, obj3 = {5, 1.5};
```

ЗАПИС (СТРУКТУРА)

Местоположението на дефиницията определя областта на името на записа - съответно за всички функции след дефиницията на структурата, в рамките на функцията и в рамките на блока.

ЗАПИС (СТРУКТУРА)

◉ Достъп

Достъпът до полетата на структура е пряк.

Може да се осъществи чрез променлива от типа на структурата.

Променлива и името на полето се разделят с оператора точка.

Пример:

```
structName obj1;  
cout << obj1.name1 << obj2.name2;
```

* *name1* и *name2* се наричат полета на променливата от тип структура или член-данни на структурата.

ЗАПИС (СТРУКТУРА)

⦿ **Операции:**

- Над член-данните;
- Над променливи от тип запис (структура) - присвояване на друга променлива или израз;

ЗАПИС (СТРУКТУРА)

◉ Памет

Дефиницията на запис НЕ предизвиква отделянето на памет за съхраняване на полетата ѝ.

Памет се заделя при създаването на променлива от тип запис (структура).

ЗАПИС (СТРУКТУРА)

Представяне в паметта

Дефиницията на променлива от тип структура предизвиква заделяне на памет за всяко поле на променливата.

Пример:

Очакваме заделената памет за структурата да изглежда по следния начин:

```
struct example {  
    char a;  
    short int b;  
    int c;  
    char d;  
};
```

Size of 1 block = 1 byte

Size of 1 row = 4 byte

a	b	b	c
c	c	c	d

ЗАПИС (СТРУКТУРА)

Представяне в паметта

Процесорът не може да достъпи паметта - използва се равен размер на думите (4B - max size на полето).

Всяко поле трябва да се разположи в паметта на адрес, който е равен на число кратно на размера на полето.

Пример:

```
struct example {  
    char a;  
    short int b;  
    int c;  
    char d;  
};
```

Реално представяне

Size of 1 block = 1 byte

Size of 1 row = 4 byte

a	padding	b	b
c	c	c	c
d	padding	padding	padding

ЗАПИС (СТРУКТУРА)

◎ Представяне в паметта

```
struct X {  
    short s; /* 2 bytes */ + /* 2 padding bytes */  
    int i;   /* 4 bytes */  
    char c;  /* 1 byte   */ + /* 3 padding bytes */  
};
```

```
struct Z {  
    int i;   /* 4 bytes */  
    short s; /* 2 bytes */  
    char c;  /* 1 byte   */ + /* 1 padding byte */  
};
```

```
const int sizeX = sizeof(struct X); /* = 12 */  
const int sizeZ = sizeof(struct Z); /* = 8  */
```

ЗАПИС (СТРУКТУРА)

От направения анализ могат да се направят следните заключения:

- Полетата се сортират по тяхната големина в низходящ ред.
- Отстъпите (padding) са позволени между данните за отделните полета и след последното поле.

ЗАПИС (СТРУКТУРА)

- Имената на полетата в рамките на една дефиниция на структурата трябва да са различни идентификатори.

Пример:

```
struct structName {  
    int name1;      ←  
    double name2;  ←  
};
```

ЗАПИС (СТРУКТУРА)

- Възможно е име на запис, на негово поле и на произволна променлива в програмата да е един и същ идентификатор.

X НЕ ГО ИЗПОЛЗВАЙТЕ!

ЗАПИС (СТРУКТУРА)

- Възможно е влягане на структури, т.е.
поле на структура може да е от тип структура=

Пример:

```
struct student {  
    //...  
};
```

```
struct classRoom {  
    student maria;  
};
```

ЗАПИС (СТРУКТУРА)

- Не е възможно поле на структурата да е от тип, съвпадащ с името на структурата.

Пример:

```
struct student {  
    student maria;  
};
```

* *Какъв би бил размерът на student?*

ЗАПИС (СТРУКТУРА)

- Възможно е поле на структурата да е от тип указател към името на структурата.

Пример:

```
struct student {  
    student * maria;  
};
```

** Какъв е размерът на student?*

ЗАПИС (СТРУКТУРА)

- Ако две структури трябва да се обръщат една към друга, е необходимо пред дефинициите им да се постави декларацията на втората по ред структура.

Пример:

```
struct studentList;  
struct student {  
    // ...  
    studentList * friends;  
};
```

```
struct studentList {  
    // ...  
    student st;  
};
```

ЗАПИС (СТРУКТУРА)

- Дефиниция на указател към запис

`<указател_към_структура> ::=`

`<име_на_структура>* <променлива_указател>`
`[=&{<променлива>}]опц;`

`<променлива> ::= <име_на_структура>`

Примери:

```
structName obj1;
```

```
structName * stPointer = &obj1;
```

ЗАПИС (СТРУКТУРА)

- **Достъп до полетата на структурата чрез указател към запис**

Примери:

```
structName obj1;  
structName* stPointer = &obj1;
```

```
(*stPointer).name1; // Достъп до name1  
stPointer->name1;    // Достъп до name1
```

КРАЙ

?