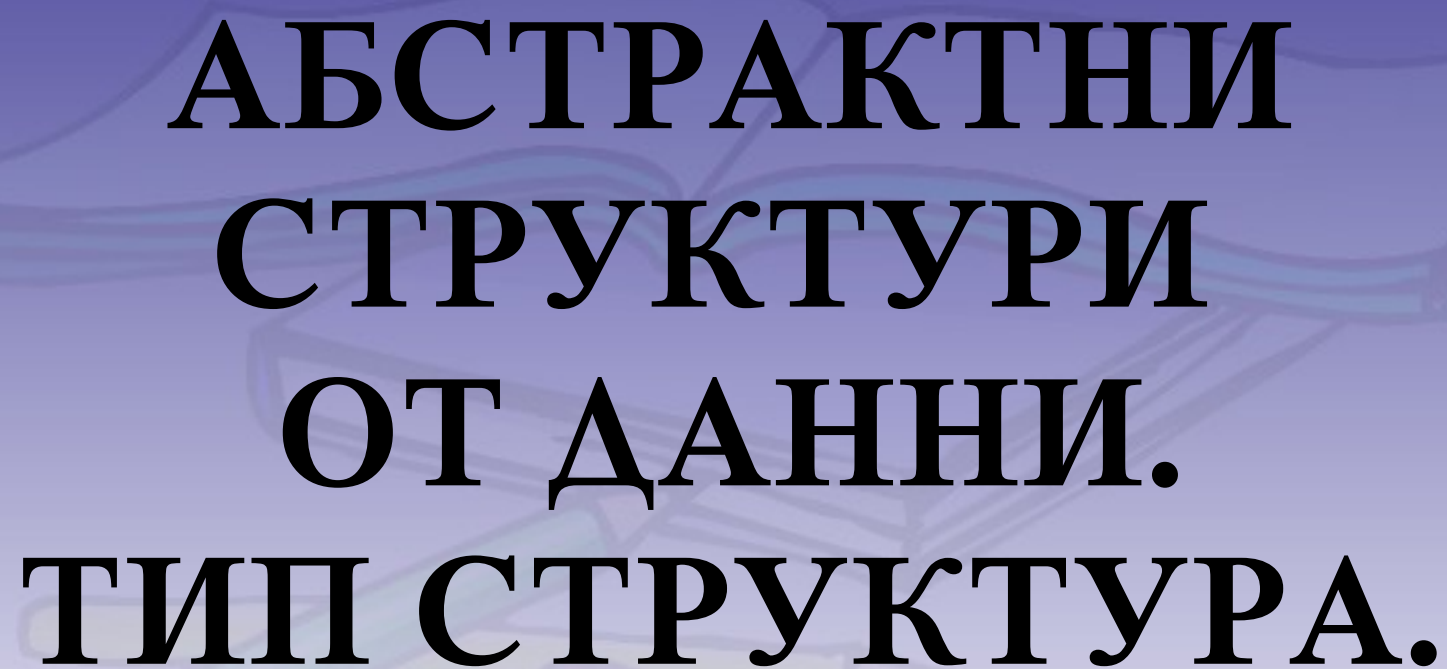


АБСТРАКТНИ СТРУКТУРИ ОТ ДАННИ. ТИП СТРУКТУРА.





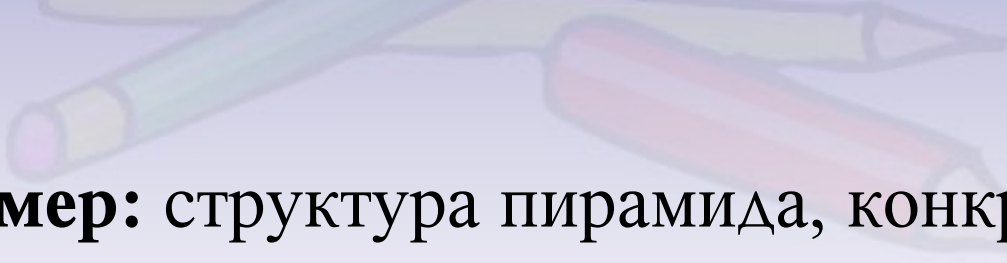
СТРУКТУРИ ДАННИ

- Много често, когато пишем програми ни се налага да работим с множество от обекти (данни). Понякога добавяме и премахваме елементи, друг път искаме да ги подредим или да обработваме данните по друг специфичен начин. Поради това има различни начини за структуриране на данните в зависимост от задачата.
- Тук на помощ ни идват **структурите данни** – множество от данни, организирани на основата на логически и математически закони. Много често избора на правилната структура прави програмата много по-ефективна – можем да спестим памет и време за изпълнение.



АБСТРАКТЕН ТИП ДАННИ

Най-общо **абстрактният тип данни (АТД)** дава определена дефиниция (абстракция) на конкретната структура, т.е., определя допустимите операции и свойства, без да се интересува от конкретната реализация. Това позволява един тип абстрактни данни да има различни реализации, респективно различна ефективност.




Пример: структура пирамида, конкретно реализирана чрез масив или чрез двоично дърво



СД В ПРОГРАМИРАНЕТО

Могат ясно да се различат няколко групи структури:

- **Линейни** – към тях спадат списъците, стековете и опашките
 - **Дървовидни** – различни типове дървета
 - **Речници** – хеш-таблици и множества
- 



СПИСЪЧНИ СТРУКТУРИ

Най-често срещаните и използвани са линейните (списъчни) структури. Те представляват абстракция на всякакви видове редици, поредици и други подобни от реалния свят.

Списък

Най-просто можем да си представим списъка като редица от елементи. Например покупките от магазина или задачите за деня представляват списъци. В списъка можем да четем всеки един елементите напр. покупките, както и да добавяме нови покупки в него. Можем спокойно да задраскваме (изтрием) покупки или да ги разместваме.

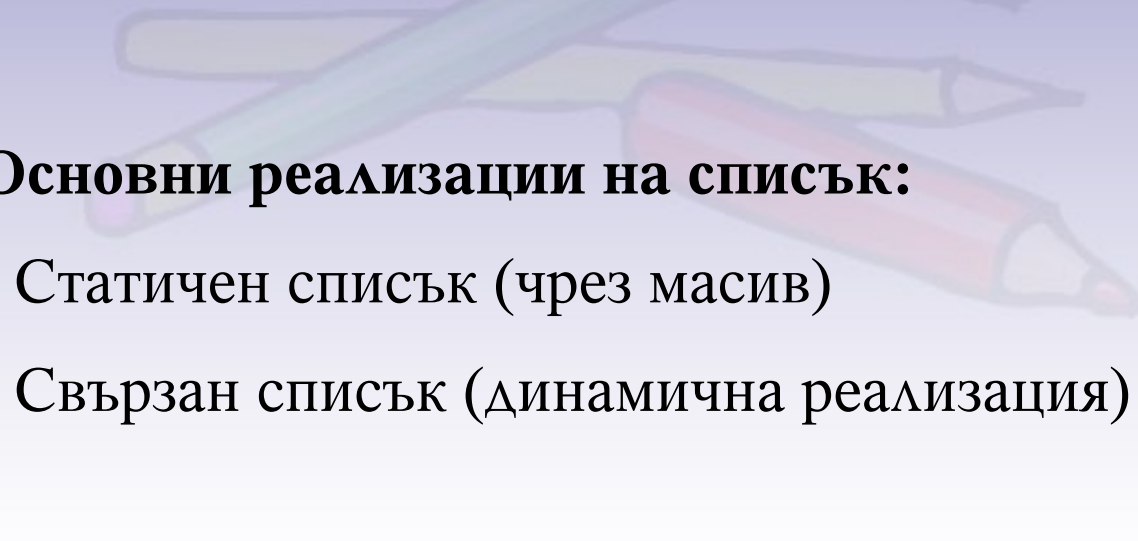


АСД “Списък”

Списък – линейна структура от данни, която съдържа поредица от елементи. Има свойството дължина, елементите му са наредени последователно.

Списъкът позволява добавяне елементи на всяко едно място както и премахването им, както и последователното им обхождането. Както споменахме по-горе, един АД може да има няколко реализации.

Основни реализации на списък:

- Статичен списък (чрез масив)
 - Свързан списък (динамична реализация)
- 



АСД “Стек”

Да си представим няколко кубчета, които сме наредили едно върху друго. Можем да слагаме ново кубче на върха, както и да махаме най-горното кубче. Такава конструкция представлява **стек**. Стекът е често срещана и използвана структура от данни.

Абстрактна структура данни „стек”

Стека представлява структура от вида “последният влязъл – първи излиза” (LIFO – Last In First Out). Както се вижда от примера с кубчетата, елементите могат да се добавят и премахват само от върха на стека. Тук също съществуват различни реализации – динамична и статична.



АСД “Опашка”

Да си представим няколко пенсионери, чакащи пред пощенския клон, за да си получат пенсиите. Който е пръв на опашката, пръв си взема пенсията. Ако дойде нов пенсионер, се нарежда последен. Също така, чакащи документи за принтиране, чакащи процеси за достъп до общ ресурс и др., много удобно и естествено се моделират чрез опашки.

Абстрактна структура данни „опашка”

Абстрактната структура опашка изпълнява условието „първият влязъл, първи излиза” (FIFO – First In First Out). Добавените елементи се нареждат в края на опашката, а се взимат от началото ѝ. Както и при списъка, за структурата от данни опашка отново е възможна статична и динамична реализация.



ТИП СТРУКТУРА

Типът структура представлява **съставен тип** данни. За разлика от простите типове данни (int, double, char), както и от типа масив (съвкупност от еднотипни елементи), съставният тип данни представлява **съвкупност от полета от разнороден тип**. Удобно е да се използва за описание на различни обекти от реалния свят.

Пример: формално описание на книги в книжарница

Всяка книга има заглавие (символен низ), автор (символен низ), година на издаване (цяло число), цена (реално число)



ТИП СТРУКТУРА

Формално описание на тип структура в езика C++:

```
struct <име> {  
    <списък от полета>  
};
```

Пример (описание на книга):

```
struct Book {  
    char title[30];  
    char author[20];  
    int year;  
    double price;  
};
```



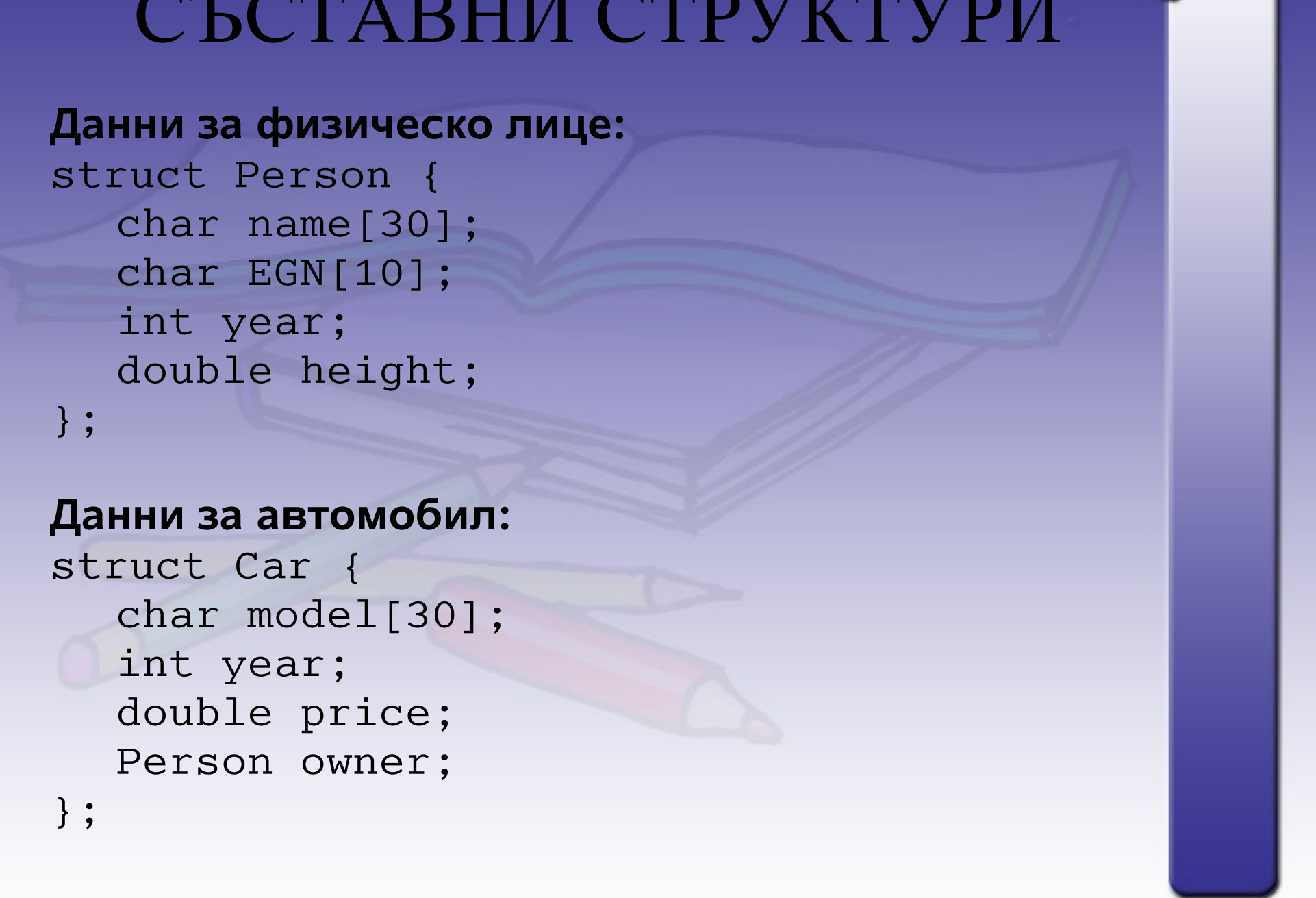
СЪСТАВНИ СТРУКТУРИ

Данни за физическо лице:

```
struct Person {  
    char name[30];  
    char EGN[10];  
    int year;  
    double height;  
};
```

Данни за автомобил:

```
struct Car {  
    char model[30];  
    int year;  
    double price;  
    Person owner;  
};
```





ПРОМЕНЛИВИ ОТ ТИП СТРУКТУРА

Деклариране на книга от тип *book*:

```
Book b1;
```

Деклариране на два автомобила от тип *car*:

```
Car c1, c2;
```

Деклариране на книги в книжарница (масив от тип *book*):

```
Book bookShop[100];
```

Указател към структура:

```
Book *books;
```

```
Car *cars;
```





ДОСТЪП ДО ОТДЕЛНИ ПОЛЕТА

Достъп до поле на променлива от структурен тип:

Извършва се чрез операция 'точка':

`b1.title; c1.year; c2.price`

Операцията 'точка' има висок приоритет и това налага прецизната ѝ употреба.

Обръщение към елемент на масив от структури:

`bookShop[i].title` – заглавието на *i*-та книга

`bookShop[j].price` – цената на *j*-та книга

Обръщение към съставни структури:

`c1.owner.EGN` – ЕГН на собственика на колата



ДОСТЪП ДО ОТДЕЛНИ ПОЛЕТА

Достъп до поле на променлива чрез указател към структурен тип:

Извършва се чрез операция 'стрелка':

`books->author` – автора на книгата, сочена от указателя `books`; еквивалентно на `(*books).author`

`cars->model` – модела на автомобила, сочен от указателя `cars`; еквивалентно на `(*cars).model`

Въпрос: кое от обръщенията е коректно:

`cars->(owner->name) ;`

`(cars->owner).name ;`