



# Структуровані типи даних: масиви, рядки, записи

# План

1. Поняття масиву (`array`): опис типу, дії над елементами масиву, головні прийоми і типові алгоритми опрацювання масивів.
2. Поняття двовимірного масиву: створення і виведення двовимірних масивів, типові задачі, які приводять до використання масивів.
3. Рядки (`string`) та дії з ними. Функції та процедури для дій з рядками.
4. Поняття про запис (`record`), його опис. Команда приєднання (`with`).

Типи даних, які конструюються з окремих частин, елементів, компонентів, тобто які мають внутрішню структуру, називаються ***складними*** або ***структурованими***.

Наприклад, масиви, рядки, записи, множини.

# Приклад

Дата	1.01	2.01	3.01	...	31.01
t, °C	-5	-10.5	-14.2	...	-4.3

# Приклад

-5    - 10.5    -14.2    ...    - 4.3

$t[1] = -5;$

$t[2] = -10.5;$

$t[3] = -14.2;$

$t[31] = -4.3;$

**Масив (*array*)** – це скінчений набір елементів одного (базового) типу, які зберігаються в послідовно розташованих комірках оперативної пам'яті і мають спільну назву.

Масив даних характеризується **іменем, кількістю елементів і типом елементів.** Ім'я масиву надає користувач. Елементи визначають тип масиву.

**Кількість елементів (розмір)** масиву найчастіше задають у вигляді діапазону або назви деякого перерахованого типу даних.

Кожний елемент масиву можна відшукати, знаючи його номер та ім'я масиву.

Номер елемента масиву називається **індексом**. Причому індексом можуть бути тільки елементи перерахованого типу.

Причому не тільки константи, а й довільні вирази перерахованого типу.

Наприклад,  $a[5]$ ,  $a[i+1]$ .

# Опис типу масив у розділі `type`

*type* <назва типу> = *array* [розмір] *of*  
<назва базового типу>;

Наприклад,

*type temp = array [1..28] of real;*



# Опис типу масив у розділі `var`

`var` <назва типу>: *array* [розмір] *of* <назва базового типу>;

Наприклад,

`var temp: array [1..28] of real;`

# Приклад

2003

2004

2005

2006

2007

2008

2009

142	147	151	149	154	150	148
-----	-----	-----	-----	-----	-----	-----

```
var Students: array [2003..2009] of integer;
```

```
Students [2005]=151; - 3-й елемент масиву
```

```
Students [2007]=154; - 5-й елемент масиву
```

# Зауваження

- Нумерувати елементи масиву можна з нуля або будь-якого іншого цілого числа.
- Значення індексу не завжди збігається з номером елемента. Щоб вони збігалися, нумерацію елементів потрібно починати з одиниці.

## З елементами масиву можна виконувати такі дії:

- введення даних (цикл);
- надання елементам масиву певних значень (операція присвоєння);
- виведення даних (операції `write`, `writeln`);
- інші операції, які визначені над даними відповідного типу.

# Етапи опрацювання масивів

1. Заповнення масиву даними (введення даних у масив).
2. Перетворення масиву (дії з даними).
3. Виведення масиву або окремих даних на екран.

# Двовимірні масиви



# Приклади двовимірних масивів

Фрагмент учнівського табеля за два семестри з 5 предметів:

	I	II
математика	8	9
рідна мова	9	10
історія	7	9
фізика	8	9
географія	10	9

Загальний вигляд масиву даних з  $n$  рядків та  $m$  стовпців (у математиці такий масив називається матрицею):

$$b_{11} \quad b_{12} \cdots b_{1m}$$

$$b_{21} \quad b_{22} \cdots b_{2m}$$

.....

$$b_{n1} \quad b_{n2} \cdots b_{nm}$$



# Оголошення двовимірного масиву

Якщо двовимірний масив має  $n$  рядків та  $m$  стовпців, то у розділі *type* його описують так:

```
type MyMassiv=array [1..n,1..m] of <базовий тип>;
```

або у розділі оголошення змінних:

```
var a: array [1..n,1..m] of <базовий тип>;
```

# Створення і виведення двовимірних масивів

Проте у пам'яті комп'ютера елементи масиву розташовані в послідовних комірках рядок за рядком, а саме:

$b[1,1], b[1,2], \dots, b[1,n], b[2,1], b[2,2], \dots, b[m,1], \dots, b[m,n]$ .

# Опрацювання двовимірних масивів

- Двовимірні масиви створюють і виводять на екран у вигляді таблиці за допомогою вкладених циклів *for*.
- Опрацьовують двовимірні масиви поелементно за допомогою алгоритмічної конструкції “вкладені цикли”. Наприклад, ввести дані з клавіатури у двовимірний масив з  $n$  рядків та  $m$  стовпців можна так:

```
write ('Ведіть дані в масив:');  
for i:=1 to n do  
  for j:=1 to m do  
    begin  
      write (' b [', i, ', ', j, ']=');  
      read ( b [i, j] );  
    end;
```

# Типові задачі, що ведуть до використання масивів

- визначення суми елементів масиву (наприклад, обсягу випущеної продукції протягом року; кількості балів, набраних студентом за семестр з деякої дисципліни; вартості продуктів у магазині тощо). Для цього потрібно додати всі елементи деякого числового масиву;
- визначення середнього значення (наприклад, середньодобової  $t^\circ$  протягом місяця; рейтингу студента; середньомісячної зарплатні тощо). Для цього потрібно додати всі значення елементів масиву і розділити результат на кількість елементів масиву;
- визначення максимального або мінімального елементу масиву (методом перегляду і порівняння всіх елементів);
- визначення номера максимального або мінімального елементу масиву;
- пошук у масиві даних за певною ознакою;
- впорядкування елементів масиву за певною ознакою тощо

# Тип даних String

- Дане типу рядок (***string***) - це послідовність довільних символів, тобто елементів типу *char*.
- Рядкові дані записують в одинарних лапках.

**Наприклад:**

'Україна'; '235'; '?1abc';

" – порожній рядок нульової довжини;

' ' – рядок, що містить один символ (пропуск).

# Опис змінних рядкового типу

- у розділі опису констант.

Наприклад: *const* s='Bye!';

- у розділі опису типів. Наприклад:

*type* t=*string*[10];

*var* top: t;

- у розділі змінних:

*var* <список змінних>: *string* [n], де n – довжина рядка, яку можна не зазначати.

# Операції, визначені над рядковими змінними

- з'єднання (+);
- порівняння (<, <=, >, >=, =, <>);
- введення-виведення;
- рядкові функції та процедури.



- Порівняння рядків здійснюється зліва направо до перших різних символів, причому `'A' < 'B'`, `'B' < 'C'` тощо. “Більшим” вважається символ, який розташований в алфавіті правіше.
- Функція `ord` дає числовий код символу. Наприклад, `ord('B')=66`; `ord('A')=65`.
- Функція `chr` виконує зворотну дію: `chr(66)='B'`.

# Приклад

Нехай  $a1 = \text{'New'}$ ,  $a2 = \text{'Year'}$ .

Тоді  $a := a1 + a2 = \text{'New Year'}$ .

Доступ до  $i$ -го символу текстового даного з іменем  $a$  можна отримати за допомогою виразу  $a[i]$ . Наприклад:

$a[1] = \text{'N'}$ ,  $a[4] = \text{' '}$ .

# Стандартні функції для дій з рядками

***length*** ( $r$ ) - визначає кількість символів у рядку  $r$ ;

***copy*** ( $r, m, n$ ) – дає  $n$  символів рядка  $r$ , починаючи з символу з номером  $m$ ;

***concat*** ( $r_1, r_2, \dots, r_n$ ) – з'єднує рядки  $r_1, r_2, \dots, r_n$  в один рядок;

***pos*** ( $r_1, r_2$ ) – визначає номер символу, з якого починається входження рядка  $r_1$  у рядок  $r_2$ .

# Приклади

**Функція**

**Значення**

*length* ("")

0

*length* ('')

1

*length* ('інформатика')

11

*сору* ('інформатика', 1, 6)

'інформ'

*pos* ('т', 'інформатика')

8

*concat* ('20', '10')

'2010'

# Стандартні процедури для дій з рядками

***insert*** ( $r1, r2, n$ ) – вставляє рядок  $r1$  у рядок  $r2$ , починаючи з позиції  $n$ ;

***delete*** ( $r, m, n$ ) – вилучає  $n$  символів з рядка  $r$ , починаючи з позиції  $m$ ;

***str*** ( $\langle\text{число}\rangle, r$ ) – переводить числове дане в дане  $r$  (типу рядок);

***val*** ( $r, s1, s2$ ) – переводить рядкове дане в числове (засилає у числову змінну  $s1$  числовий образ рядка  $r$ ;  $s2$  – індикатор помилки: якщо рядок  $r$  містить тільки числа, то процедура повертає значення  $s2$ , рівне 0; якщо рядок  $r$  містить крім цифр й інші символи, то  $s2$  отримує числове значення, рівне номеру першого символу, що не є цифрою).

# Приклади

Нехай  $a = \text{'інформатика'}$ .

**Процедура**

**Значення змінної  $a$**

*insert* ('наука ',  $a$ , 1)    'наука інформатика'

*delete* ( $a$ , 2, 4)            'іMATИКА'

*str* (100,  $a$ )                '100'

*val* ('23',  $a$ ,  $p$ )             $a = 23$ ,  $p = 0$

# Способи опрацювання даних типу `string`

- Опрацювати весь рядок як єдине ціле, застосовуючи до нього функції та процедури для дій з рядками;
- Розглядати рядок як масив, складений з елементів-символів, і опрацювати його за правилами роботи з елементами масиву.

# Записи

A 3D grid of dots on a blue background. The dots are arranged in a regular pattern, receding into the distance to create a perspective effect. The dots are light blue and connected by thin, light blue lines.



# Означення

**Запис (record)** – це структурований тип даних, призначений для збереження в оперативній пам'яті й опрацювання даних про властивості об'єкта.

Запис складається з полів. **Поля** – це дані різних типів. У полях розташовуються компоненти складеного даного.

# Опис типу даних record

Запис описують у розділі *type* чи *var* за допомогою такої конструкції:

```
type <назва типу запису>=record  
    <назва поля 1> : <тип поля 1>;  
    .....  
    <назва поля n> : <тип поля n>;  
end;
```

# Приклад 1

```
type Student = record  
  sname: string;  
  name: string;  
  kyrs: 1..5;  
  grupa: char;  
  born: 1980..1985;  
  serbal: real  
end;
```

# Приклад 2

```
type Address = record  
  city: string;  
  street: string;  
  house: integer;  
  flat: integer;  
end;
```

Кожне поле в запису можна вважати звичайною змінною, якій можна присвоїти ім'я, ввести або вивести її значення.

Звертатися до поля потрібно через складене ім'я так:

**<назва запису> . < назва поля>**

# Приклади

`Student.sname` – прізвище студента;

`Student.born` – рік народження студента;

`Address.flat` – номер квартири.

Конкретні записи типу Student оголошують у розділі *var* так:

```
var Student1, Student2, Student3: Student;
```

Наприклад:

Student1.name – ім'я 1-го студента;

Student2.serbal – середній бал 2-го студента.

Записи можуть бути елементами масивів.

Окремі поля записів також можуть бути записами. За рахунок цього в Pascal можна створювати складні структури даних.



# Приклад

```
type Anketa = record  
  facult: string;  
  group: 1..6;  
  fio: string;  
  address = record  
    k: longint;  
    city: string;  
    street: string;  
    house, flat: integer  
  end;  
  born = record  
    date: 1..31;  
    month: 1..12;  
    year: 1980..1985  
  end;  
end;  
var Student: array [1..30] of Anketa;
```

Звертатися до елементів таких складних структур потрібно за складеним іменем.

Наприклад:

```
Student [5].address.city:='Вінниця';
```

або

```
writeln (Student [5].address.city);
```

# Команда приєднання (with)

Команда приєднання дає змогу використовувати у програмі лише імена полів.

Загальний вигляд команди приєднання:

```
with <ім'я змінної типу запис> do <команда>;
```

# Приклад

```
with Student [5] do  
  begin  
    fio:= 'ІВАНОВ';  
    group:= 1;  
    with Address do  
      begin  
        city:='Вінниця';  
        house:= 38  
      end;  
    end;  
  end;
```