Житомирський державний технологічний університет

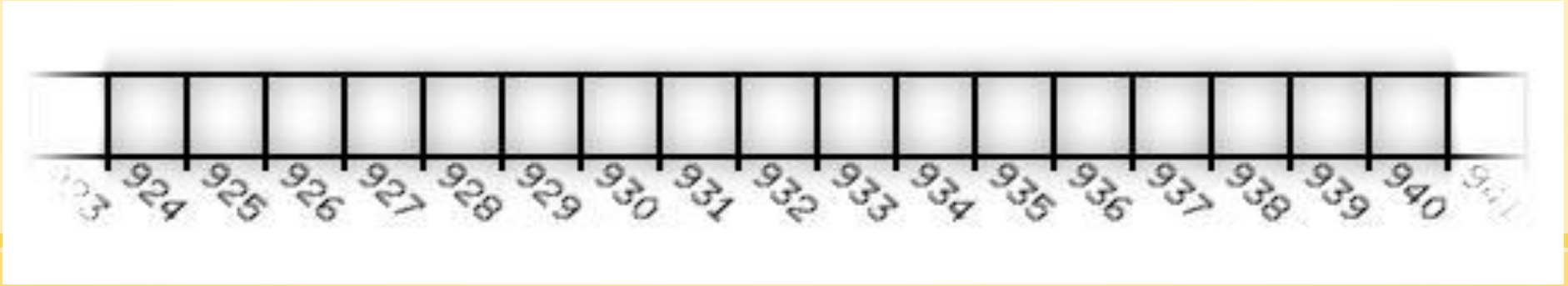
Лекція 18.
Покажчики у мові С.

Морозов А.В., к.т.н., доц., декан ФІКТ
morozov.andriy@gmail.com

Згадаємо теорію…

Змінна – це область пам’яті, яка має ім’я і в якій зберігається значення певного типу даних

Будь-яке значення змінної зберігається у **пам’яті**.



Пам’ять під **локальні** змінні виділяється при запуску функції і звільняється при завершенні функції.

Пам’ять під **глобальні** змінні виділяється при запуску функції **main** і звільняється при завершенні функції **main**.

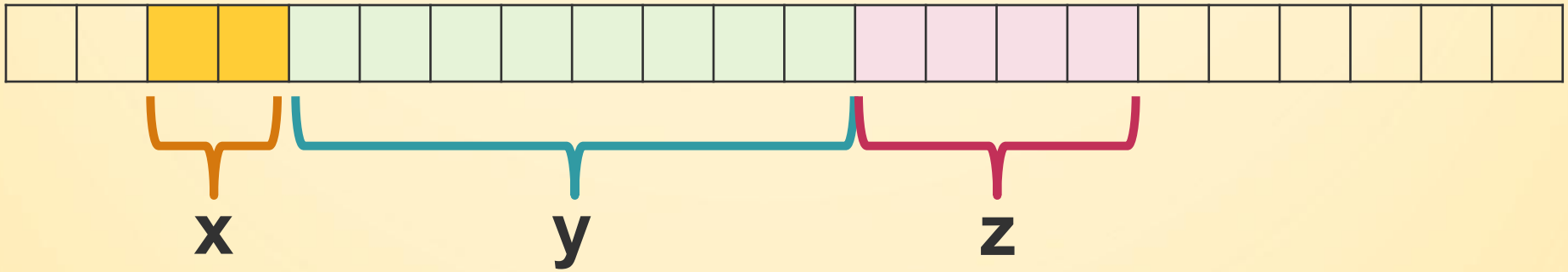
Виділення пам’яті під змінну – це закріплення за змінною конкретних комірок пам’яті.

Для того, щоб створити змінну, її потрібно оголосити, вказавши тип даних:

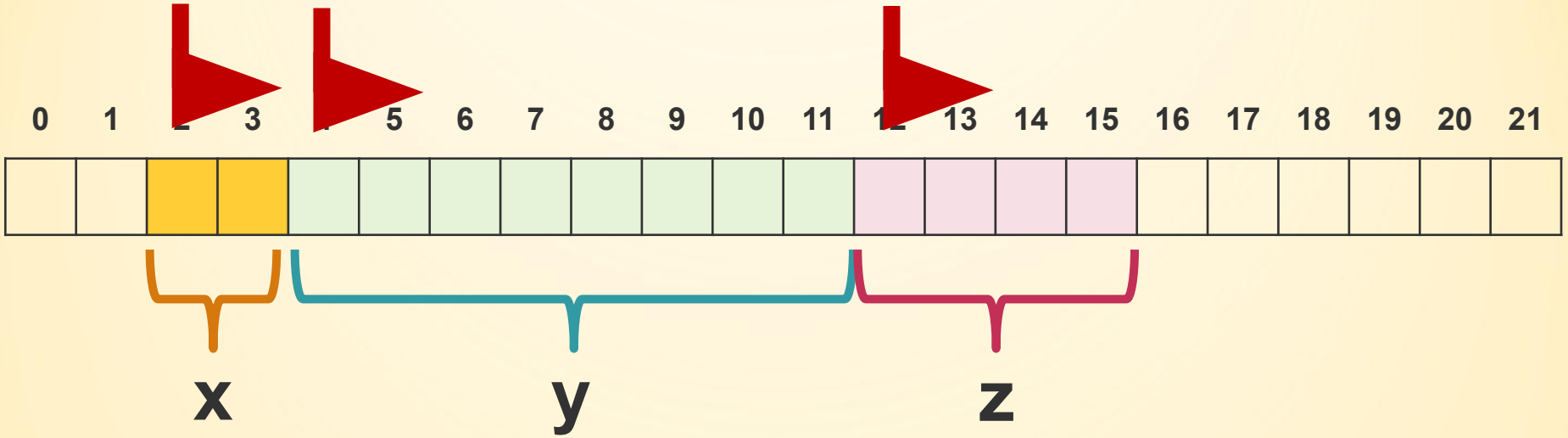
тип даних | ім’я змінної ;

Приклад:

```
short x;
double y;
int z;
```



Тип даних визначає скільки байтів пам’яті буде виділено під змінну.



Кожна комірка у пам’яті має унікальний номер.

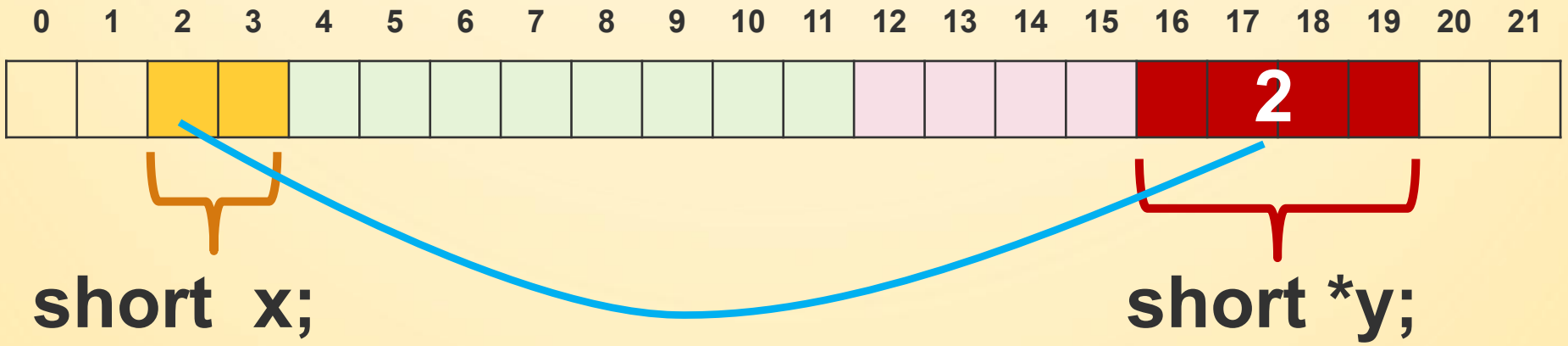
Номер комірки пам’яті, де зберігається значення змінної називають адресою змінної.

Якщо змінна займає кілька байтів, то адреса вказує на першу (початкову) комірку.

У мові С є можливість оголосити змінну, яка міститиме номер комірки пам’яті, де зберігатиметься значення певного типу.

Така змінна називається **покажчиком.**

Покажчик – це змінна, значенням якої є адреса пам’яті, де зберігається значення певного типу.



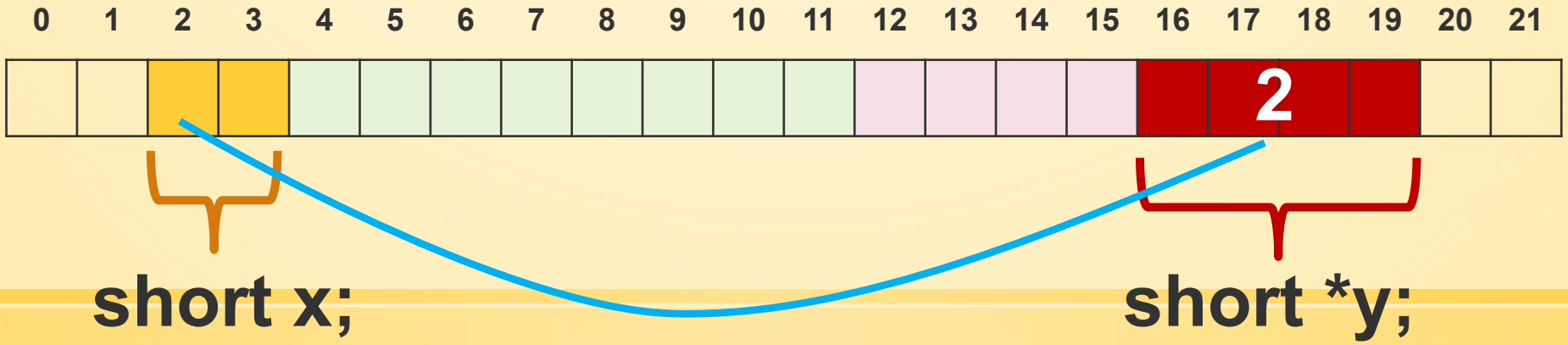
Щоб оголосити змінну типу покажчика на деякий тип даних, потрібно перед іменем змінної поставити зірочку:

тип даних * ім'я змінної ;

Приклад:

```
short *x;
double *y;
int *z;
```

Змінна-покажчик у пам'яті займає 4 байти, незалежно від того, на значення якого типу вона вказує.



В одному оголошенні можна вказувати звичайні змінні та змінні-покажчики:

```
short x, *px, y, *py;
double d, a, b = 0, *pd, *bp;
```

При запуску функції операційна система визначає вільні ділянки пам’яті і розміщує в них оголошені у функції змінні.

Тому при кожному виклику функції одна й та сама змінна може розміщуватися у різних комірках пам’яті.

Причина: при виході з функції усі локальні змінні видаляються, пам’ять очищується. При наступному виклику функції під локальні змінні знову виділяється пам’ять.

Для визначення адреси змінної у пам’яті застосовується операція «взяття адреси» **&**:

& | ім’я змінної

```
int x = 10;
int *px = &x;
```

```
double y = 10;
double *py = &x;
```

Помилка невідповідності типів. Неможливо записати у **py** адресу змінної типу **int**

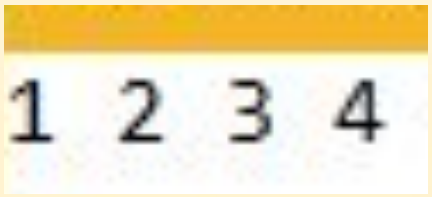
Особливості функції scanf

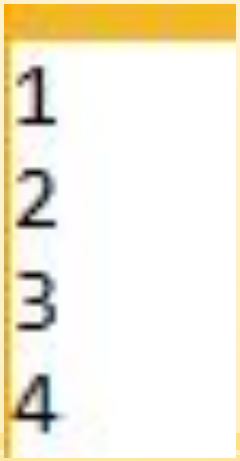
1) після рядку формату вказуються адреси змінних, в які записуються прочитані з клавіатури значення:

```
scanf("%d %d %lf %ld", &a, &b, &c, &d);
```

2) читання кількох значень:

```
int a, b;
double c;
long d;
scanf("%d %d %lf %ld", &a, &b, &c, &d);
```





3) у рядку формату між специфікаторами форматування можна вказувати роздільники:

```
int a, b;
double c;
long d;
scanf("%d, %d, %lf, %ld",
                        &a, &b, &c, &d);
```

```
1, 2, 3, 4
```

4) якщо значення, що вводилося не відповідало формату, то у змінну нічого не записується:

```
int x = 10;
scanf("%d", &x);
printf("x = %d\n", x);
```

```
asdasdasd
x = 10
```

Якщо значення змінної некоректно введене, то усі інші змінні прочитані не будуть:

```
int a, b, c, d;
int count = scanf("%d %d %d %d", &a,
                                 &b, &c, &d);
printf("Coorect values: %d\n", count);
```

```
1 asd 3 4
Coorect values: 1
```

5) функція scanf повертає кількість коректно прочитаних змінних:

```
int a, b, c, d;
int count = scanf("%d %d %d %d", &a,
                                 &b, &c, &d);
printf("Coorect values: %d\n", count);
```

```
1 2 3 asdasd
Coorect values: 3
```