# Конспект лекції: Типи даних в Arduino

**Тема:** Основи програмування мікроконтролерів. Типи даних в середовищі Arduino.
**Мета:** Ознайомити студентів з основними типами даних, що використовуються в мові програмування Arduino (C++), пояснити їх призначення, розмір у пам'яті, діапазон значень та продемонструвати практичне застосування на прикладах.

## 1. Вступ: Що таке типи даних і чому вони важливі?

**Тип даних** — це класифікація, яка визначає, якого виду значення може приймати змінна, і які операції можна над нею виконувати. Коли ми оголошуємо змінну, ми резервуємо для неї місце в пам'яті мікроконтролера.

На відміну від програмування для ПК, пам'ять мікроконтролерів (наприклад, ATmega328P в Arduino Uno має всього 2 КБ оперативної пам'яті) є вкрай обмеженим ресурсом. Тому правильний вибір типу даних є критично важливим для створення ефективних та стабільних програм. Використання `long` там, де достатньо `byte`, марнує дорогоцінну пам'ять.

[Зображення таблиці з типами даних Arduino]

## 2. Цілочисельні типи (Integer Types)

Ці типи використовуються для зберігання цілих чисел без дробової частини.

### `byte`

- **Опис:** Зберігає беззнакове число від 0 до 255.
- **Розмір:** 1 байт (8 біт).
- **Діапазон:** 0 до 255.
- **Застосування:** Ідеально підходить для зберігання значень з аналогових датчиків, які були перетворені в 8-бітний формат, для роботи з портами, або коли значення гарантовано не перевищує 255.

**Приклад:**
```
// Зберігаємо рівень яскравості для ШІМ-сигналу
byte pwmValue = 180;
// analogWrite працює зі значеннями 0-255
analogWrite(9, pwmValue);
```

-

### `int` (Integer)

- **Опис:** Найбільш поширений тип для зберігання цілих чисел зі знаком.

- **Розмір:** 2 байти (16 біт) на більшості плат Arduino (Uno, Nano, Mega). На платах з 32-бітним процесором (Due, ESP32) займає 4 байти.
- **Діапазон (для 2 байт):** -32,768 до 32,767.
- **Застосування:** Лічильники в циклах, результати простих математичних операцій, значення з датчиків.

**Приклад:**
```
int counter = 0;
for (int i = 0; i < 100; i++) {
  counter++;
}
Serial.println(counter); // Виведе 100
```

- 

### `unsigned int`

- **Опис:** Аналогічний до `int`, але зберігає тільки додатні значення.
- **Розмір:** 2 байти (16 біт).
- **Діапазон:** 0 до 65,535.
- **Застосування:** Коли значення ніколи не буває від'ємним, що дозволяє розширити діапазон додатних значень вдвічі.

**Приклад:**
```
// Зберігаємо частоту звукового сигналу
unsigned int frequency = 440; // Нота "Ля"
tone(8, frequency, 1000); // Пін 8, частота, тривалість 1с
```

- 

### `long`

- **Опис:** Призначений для зберігання дуже великих цілих чисел зі знаком.
- **Розмір:** 4 байти (32 біти).
- **Діапазон:** -2,147,483,648 до 2,147,483,647.
- **Застосування:** Робота з часом (функція `millis()`), складні математичні розрахунки, обробка даних GPS.

**Приклад:**
```
long veryLargeNumber = 123456789;
Serial.println(veryLargeNumber);
```

- 

### `unsigned long`

- **Опис:** Зберігає дуже великі беззнакові цілі числа.
- **Розмір:** 4 байти (32 біти).
- **Діапазон:** 0 до 4,294,967,295.

- **Застосування: Критично важливий для роботи з функцією `millis()`**, яка повертає кількість мілісекунд з моменту запуску плати.

**Приклад:**
```
unsigned long previousMillis = 0;
const long interval = 1000; // Інтервал в 1 секунду

void loop() {
  unsigned long currentMillis = millis();
  if (currentMillis - previousMillis >= interval) {
    previousMillis = currentMillis;
    Serial.println("Пройшла ще одна секунда");
  }
}
```

- 

## 3. Типи з плаваючою комою (Floating-Point Types)

Використовуються для зберігання чисел з дробовою частиною.

### `float`

- **Опис:** Зберігає дійсні числа з одинарною точністю.
- **Розмір:** 4 байти (32 біти).
- **Діапазон:** приблизно від -3.4028235E+38 до 3.4028235E+38.
- **Точність:** 6-7 значущих цифр після коми.
- **Застосування:** Робота зі значеннями датчиків, що повертають нецілі числа (температура, вологість), складні математичні розрахунки (тригонометрія, фільтрація сигналів).

**Приклад:**
```
// Розрахунок напруги з аналогового входу
int sensorValue = analogRead(A0);
// Перетворюємо значення 0-1023 у напругу 0-5В
float voltage = sensorValue * (5.0 / 1023.0);
Serial.print("Напруга: ");
Serial.println(voltage);
```

- 

### `double`

- **Опис:** Зберігає дійсні числа з подвійною точністю.
- **Важливо:** На 8-бітних платах Arduino (Uno, Nano, Mega) `double` має такий самий розмір і точність, як `float` (4 байти). На 32-бітних платах (Due) він займає 8 байт і надає значно вищу точність.

- **Застосування:** На 8-бітних платах немає сенсу використовувати `double` замість `float`. На 32-бітних — для наукових та інженерних розрахунків, що вимагають високої точності.

## 4. Символьний та логічний типи

### `char`

- **Опис:** Зберігає один символ (літеру, цифру, знак). Фактично, це цілочисельний тип, що зберігає ASCII-код символу.
- **Розмір:** 1 байт (8 біт).
- **Діапазон:** -128 до 127.
- **Застосування:** Обробка даних з послідовного порту (Serial), робота з текстовими дисплеями, зберігання окремих символів.

**Приклад:**
```
char myChar = 'A';
Serial.println(myChar);      // Виведе 'A'
Serial.println((int)myChar); // Виведе 65 (ASCII-код 'A')
```

- 

### `boolean` (або `bool`)

- **Опис:** Може приймати тільки два значення: `true` (істина) або `false` (хибність).
- **Розмір:** 1 байт.
- **Застосування:** Зберігання стану (увімкнено/вимкнено), перевірка умов в операторах `if`, `while`.

**Приклад:**
```
boolean lightIsOn = false;
int buttonState = digitalRead(2); // Читаємо стан кнопки

if (buttonState == HIGH) {
  lightIsOn = true;
}

if (lightIsOn) {
  digitalWrite(13, HIGH); // Вмикаємо світлодіод
}
```

- 

## 5. Складні та специфічні типи Arduino

### `String` (з великої літери)

- **Опис:** Це не базовий тип, а **об'єкт**, призначений для роботи з текстовими рядками. Надає багато зручних методів для конкатенації, пошуку, вирізання підрядків тощо.
- **Недоліки:** Використовує динамічну пам'ять, що може призвести до її фрагментації та нестабільної роботи програми при тривалому використанні, особливо на платах з малою кількістю ОЗП.
- **Застосування:** Обробка текстових команд, робота з даними з мережі (HTTP, MQTT).

**Приклад:**
```
String message = "Температура: ";
float temperature = 23.5;
message += String(temperature); // Конкатенація (додавання)
message += " C";
Serial.println(message); // Виведе "Температура: 23.5 C"
```

- 
- **Альтернатива:** Масиви типу `char` (C-style strings). Вони швидші, не фрагментують пам'ять, але менш зручні у використанні.

**Масиви (`array`)**

- **Опис:** Набір змінних одного типу, що зберігаються в пам'яті послідовно.
- **Застосування:** Зберігання набору показників з датчиків, пін-кодів, мелодій.

**Приклад:**
```
// Масив з номерами пінів, до яких підключені світлодіоди
int ledPins[] = {2, 3, 4, 5, 6, 7};

// Вмикаємо всі світлодіоди по черзі
for (int i = 0; i < 6; i++) {
  pinMode(ledPins[i], OUTPUT);
  digitalWrite(ledPins[i], HIGH);
  delay(100);
}
```

- 

**`void`**

- **Опис:** Спеціальний тип, що означає "ніщо".
- **Застосування:** Використовується для функцій, які не повертають жодного значення (`setup()`, `loop()`), а також для створення "універсальних" вказівників (поглиблена тема).

**Приклад:**
```
void printMessage() {
  Serial.println("Це функція, що нічого не повертає.");
}
```

- 

## 6. Висновки та практичні поради

1. **Економте пам'ять:** Завжди обирайте найменший тип даних, який може вмістити ваші значення. Якщо змінна зберігає номери пінів, `byte` є кращим вибором, ніж `int`.
2. **Час – це `unsigned long`:** Для будь-яких операцій, пов'язаних з вимірюванням часу (затримки без `delay()`, таймери), завжди використовуйте `unsigned long` та функцію `millis()`.
3. **Обережно з діленням цілих чисел:** Результат ділення двох цілих чисел завжди буде цілим числом (дробова частина відкидається). `5 / 2` дорівнює `2`, а не `2.5`. Щоб отримати точний результат, хоча б один з операндів має бути типу `float`: `5.0 / 2.0`.
4. **`String` vs `char[]`:** Для простих проектів `String` є зручним. Але для пристроїв, що мають працювати стабільно 24/7, краще вивчити та використовувати масиви `char`.

Правильне розуміння та використання типів даних — це фундамент для написання надійного та ефективного коду для будь-якого мікроконтролера.

## Головні правила для запам'ятовування

1. **Не знаєш, що обрати? Бери `int`.** Для більшості простих завдань його вистачить.
2. **Працюєш з дробовими числами (температура, напруга)? Бери `float`.** Не забувай писати `.0` у розрахунках (наприклад, `5.0 / 1023.0`).
3. **Вимірюєш час за допомогою `millis()`? ЗАВЖДИ бери `unsigned long`.** Це врятує тебе від багатьох дивних помилок.
4. **Потрібно зберегти стан "увімкнено/вимкнено"? `boolean` — твій найкращий друг.**
5. **Економ пам'ять.** Якщо число точно менше 255, використовуй `byte` замість `int`.