



ТИПІЗАЦІЯ МОВ ПРОГРАМУВАННЯ

Питання 3.3

Типи даних vs структури даних

- **Тип даних** — характеристика, яку явно чи неявно надано об'єкту (змінній, функції, полю запису, константі, масиву тощо).
 - Тип даних визначає множину припустимих значень, формат їхнього збереження, розмір виділеної пам'яті та набір операцій, які можна робити над даними.
 - Біти, байти та слова називають *машинними типами даних*.
- **Абстрактний тип даних (АТД)** — це математична модель для типів даних, де тип даних визначається поведінкою (семантикою) з точки зору користувача даних.
 - в термінах можливих значень, можливих операцій над даними цього типу і поведінки цих операцій.
 - АТД визначає набір функцій, незалежних від конкретної реалізації типу, для оперування його значеннями.
 - Конкретні реалізації АТД називаються *структурами даних*.
- **Типізація мови програмування** – спосіб перевірки типів.
 - **Типізовані мови**
 - C/C++
 - Python, PHP, Ruby
 - Lua, JavaScript, Action Script
 - **Нетипізовані мови**
 - Асемблер
 - Forth
 - Brainfuck

Типи даних у мові C/C++

■ Вбудовані типи даних (Pre-defined DataTypes)

- *Примітивні (фундаментальні) типи даних*: ціле число (integer), символ (character), булеве значення (boolean), дробове число (floating point, double floating point), порожній тип (void), широкий символ (wide character)
- + *Специфікатори/Модифікатори типів даних* (Signed, Unsigned, Short, Long)

■ Похідні типи даних (Derived Data Types, *тема 3*)

- *Вказівник (pointer)* – адреса іншої змінної у віртуальній пам'яті
- *Масив (array)* – набір однотипних значень
- *Функція (function type)* – іменований блок коду
- *Користувацькі (абстрактні) типи даних (User-defined DataTypes)*
 - Структура (structure) – тип, що є набором з кількох різних типів
 - Об'єднання (union)
 - Перелічення (enumeration)
 - typedef
 - Клас (class) – тільки в C++

Демонстраційна програма



predefined_types.c

```
1 #include <stdio.h>
2 int main()
3 {
4     int a = 1;
5     char b = 'G';
6     double c = 3.14;
7     printf("Hello World!\n");
8
9
10    printf("Hello! I am a character. My value is %c and "
11           "my size is %lu byte.\n", b, sizeof(char));
12
13
14    printf("Hello! I am an integer. My value is %d and "
15           "my size is %lu bytes.\n", a, sizeof(int));
16
17
18    printf("Hello! I am a double floating point variable."
19           " My value is %lf and my size is %lu bytes.\n", c, sizeof(double));
20
21    printf("Bye! See you soon. :)\n");
22
23    return 0;
24 }
```

C:\Users\spuasson\Desktop\typeDemo.exe

```
Hello World!
Hello! I am a character. My value is G and my size is 1 byte.
Hello! I am an integer. My value is 1 and my size is 4 bytes.
Hello! I am a double floating point variable. My value is 3.140000 and my size is 8 bytes.
Bye! See you soon. :)
```

Типи даних на 32-бітному компіляторі gcc

Data Type	Memory (bytes)	Range	Format Specifier
short int	2	-32,768 to 32,767	%hd
unsigned short int	2	0 to 65,535	%hu
unsigned int	4	0 to 4,294,967,295	%u
int	4	-2,147,483,648 to 2,147,483,647	%d
long int	4	-2,147,483,648 to 2,147,483,647	%ld
unsigned long int	4	0 to 4,294,967,295	%lu
long long int	8	-(2 ⁶³) to (2 ⁶³)-1	%lld
unsigned long long int	8	0 to 18,446,744,073,709,551,615	%llu
signed char	1	-128 to 127	%c
unsigned char	1	0 to 255	%c
float	4		%f
double	8		%lf
long double	12		%Lf

- Мова передбачає 3 комплексні типи: `float _Complex`, `double _Complex` та `long double _Complex`.
 - Дійсні та комплексні типи чисел загалом називають плаваючими типами (floating types).

Демонстрація long як типу та специфікатора типу (type specifier)

```
#include <stdio.h>
```

```
int main() {  
    int a;  
    long b;  
    long long c;
```

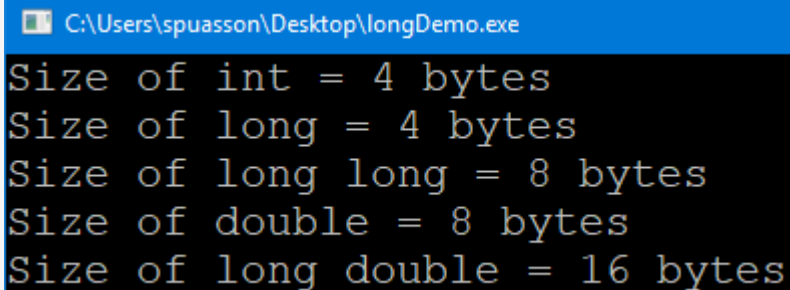
```
    double e;  
    long double f;
```

```
    printf("Size of int = %ld bytes \n", sizeof(a));  
    printf("Size of long = %ld bytes\n", sizeof(b));  
    printf("Size of long long = %ld bytes\n", sizeof(c));
```

```
    printf("Size of double = %ld bytes\n", sizeof(e));  
    printf("Size of long double = %ld bytes\n", sizeof(f));
```

```
    return 0;
```

```
}
```



```
C:\Users\spuasson\Desktop\longDemo.exe  
Size of int = 4 bytes  
Size of long = 4 bytes  
Size of long long = 8 bytes  
Size of double = 8 bytes  
Size of long double = 16 bytes
```

- Ключове слово long не можна використовувати з типами float та char.



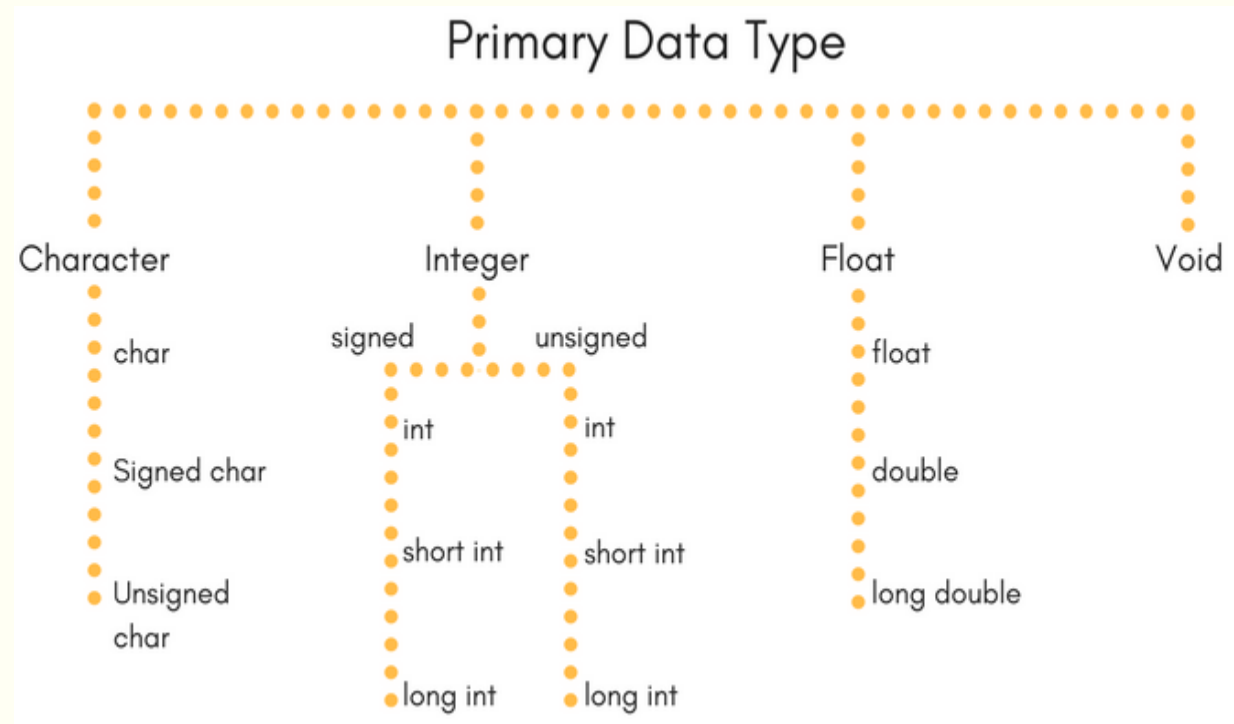
long_type_demo.c

Числові типи

Python

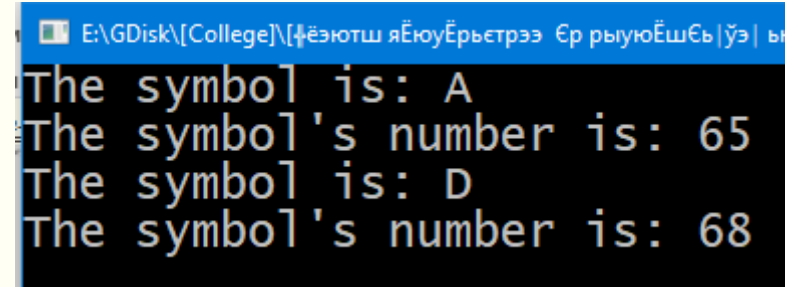
- три числових типи: integers, floating point number і complex number.
 - Логічний тип даних є підтипом цілих чисел.
 - Цілі числа мають необмежену точність
 - Дробові числа зазвичай реалізовано за допомогою double в C;
 - У комплексних числах – реальна та уявна частина (дробові числа).
 - Для їх отримання з комплексного числа z , використовуйте $z.real$ і $z.imag$.
 - Стандартна бібліотека включає додаткові числові типи:
 - [fractions](#) – зберігає раціональні числа,
 - [decimal](#) – зберігає числа з плаваючою крапкою з визначеною користувачем точністю.

C



Символьний тип char

```
1 #include <stdio.h>
2 #include <conio.h>
3
4 int main(void) {
5     char c = 'A';
6     int shift = 3;
7     printf("The symbol is: %c \n", c);
8     printf("The symbol's number is: %d \n", c);
9
10    char shifted_c = c + shift;
11    printf("The symbol is: %c \n", shifted_c);
12    printf("The symbol's number is: %d \n", shifted_c);
13
14    getch();
15    return 0;
16 }
```



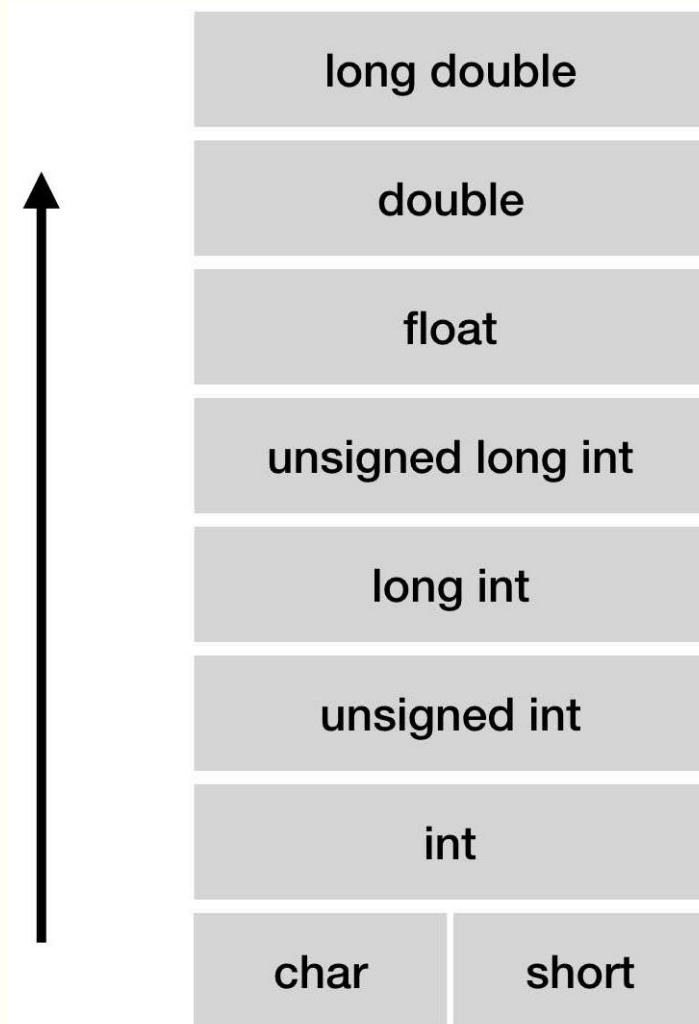
```
E:\GDisk\College\...
The symbol is: A
The symbol's number is: 65
The symbol is: D
The symbol's number is: 68
```



char_demo.c

- Тип void містить порожній набір значень.
 - Тип має відомий фіксований розмір, якщо він не є неповним та не є масивом змінної довжини.

Неявне перетворення типів (implicit type conversion)



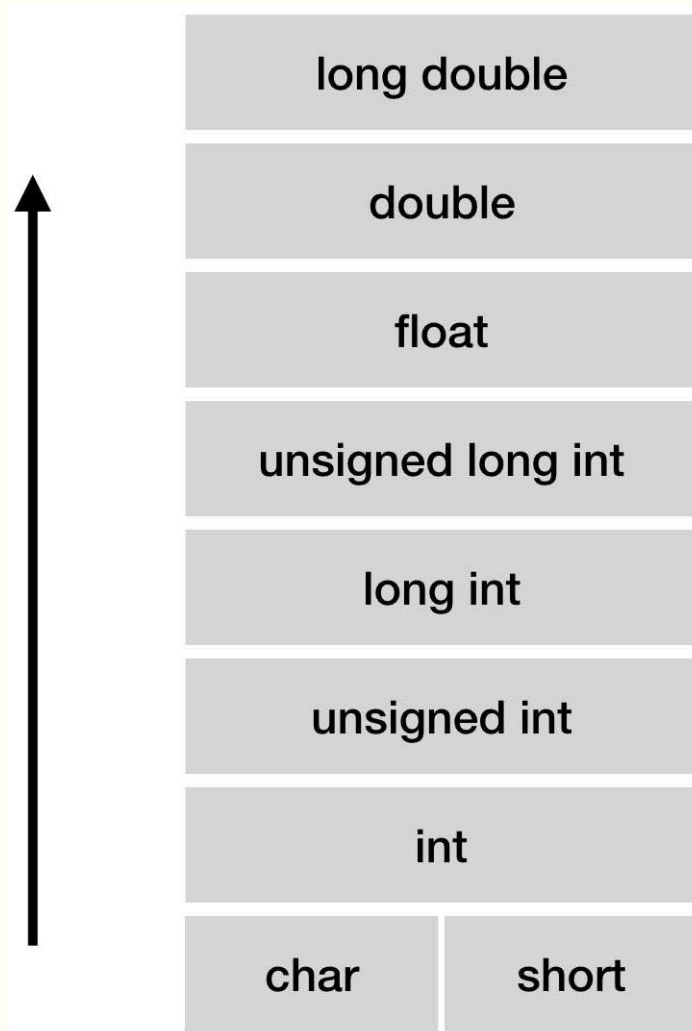
■ Неявне (автоматичне) перетворення типів

```
1 #include <stdio.h>
2 #include <conio.h>
3
4 int main(void) {
5     int x1 = 10, x2 = 17;    // integer x
6     char y = 'a';          // character c
7     x1 = x1 + y;            // y implicitly converted to int.
8                             // ASCII value of 'a' is 97
9     float z = x1 + 1.0;     // x is implicitly converted to float
10    int z_int;
11    float z_float;
12
13    z_int = x1/x2;           // integer division
14    z_float = z/x2;         // float division
15    printf("x1 = %d, z = %f \n", x1, z);
16    printf("fractions: \nz_int = %d, z_float = %f \n", z_int, z_float);
17
18    getch();
19    return 0;
20 }
```

implicit_type_conversion.c

```
E:\GDisk\College\Дієютьш яЁюуЁрьстрээ Ёр рыуюЁшСь|Ўэ| ьютш)\_x
x1 = 107, z = 108.000000
fractions
z_int = 6, z_float = 6.352941
```

Явне перетворення типів (зведення типів, type casting)



```
1  #include <stdio.h>
2  #include <conio.h>
3
4  int main(void)
5  {
6      double x = 1.2;
7      // Explicit conversion from double to int
8      int sum = (int)x + 1;
9
10     printf("sum = %d", sum);
11     getch();
12     return 0;
13 }
```

E:\GDisk\[College]V
sum = 2



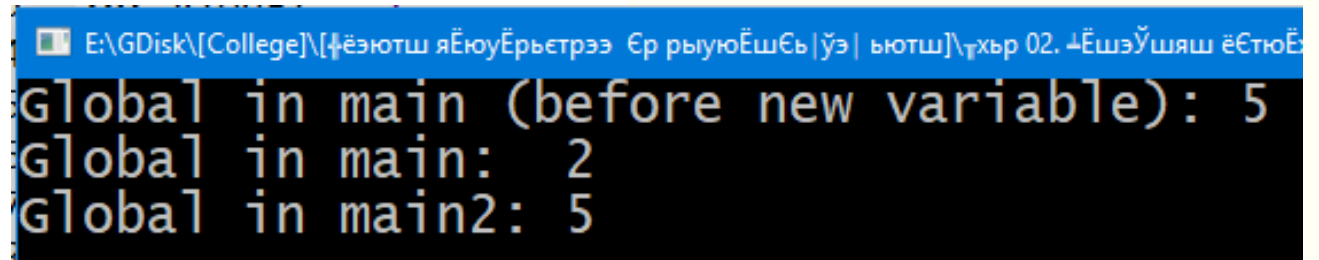
explicit_type_conversion.c

Який тип використати?

- Вага виловленої з річки риби?
- Кількість цукерок у пакеті?
- Пін-код на смартфоні?
- Баланс на рахунку карти?
- Номер банківської карти?
- Стать людини?
- Оцінка за предмет?
- Потужність блоку живлення?
- Назва рослини?
- Автомобільний номер?
- Температура повітря?
- Автор книги?
- Розміри пакунку?
- Номінал купюри?
- Задоволеність клієнта послугою?
- Кількість атомів у заготовці?
- Код банківської транзакції?
- $3\sqrt{-2}$

Локальні та глобальні змінні

```
1 #include <stdio.h>
2
3 int global = 5;
4
5
6 int main2(void) {
7     return global;
8 }
9
10 int main(void) {
11     printf("Global in main (before new variable): %d \n", global);
12
13     int global = 2;
14     printf("Global in main: %d \n", global);
15     printf("Global in main2: %d \n", main2());
16     return 0;
17 }
```



```
E:\GDisk\[College]\[фёзютш яёюёрьётрээ ёр рыуюёшёь|ўэ| ьютш]\тхър 02. 4ёшэўшяш ёётюё
Global in main (before new variable): 5
Global in main: 2
Global in main2: 5
```

- Локальні змінні оголошуються всередині функції, доступ до них мають оператори МП цієї функції, створюються при оголошенні/вході в блок та знищуються після виходу з нього.
- Глобальні змінні оголошуються ззовні будь-якої функції, доступні в усій програмі з моменту оголошення та існують весь час роботи програми.

Константи та літерали

- **Константи** – фіксовані значення, які програма не може змінювати під час свого виконання.
 - Такі фіксовані значення також називають *літералами*.
- **Цілочисельні літерали**: 85 (десятковий), 0213 (вісімковий), 0x4b (шістнадцятковий), 30 (int), 30u (unsigned int), 30l (long), 30ul (unsigned long)
- **Літерали з плаваючою комою**: 3.14159, 314159E-5L
- **Символьні константи** записуються в одинарних лапках та зберігаються у змінних типу `char`.
 - 'x' (символ), '\t' (управляюча послідовність, escape sequence), '\u02C0' (універсальний символ).
- **Рядкові літерали** записуються в подвійних лапках `“”`.
 - Можна розбивати довгий рядок на кілька рядків за допомогою рядкових літералів та пробілів.

Оголошення та використання констант

Використовуючи директиву препроцесора #define.

```
#include <stdio.h>
```

```
#define LENGTH 10  
#define WIDTH 5  
#define NEWLINE '\n'
```

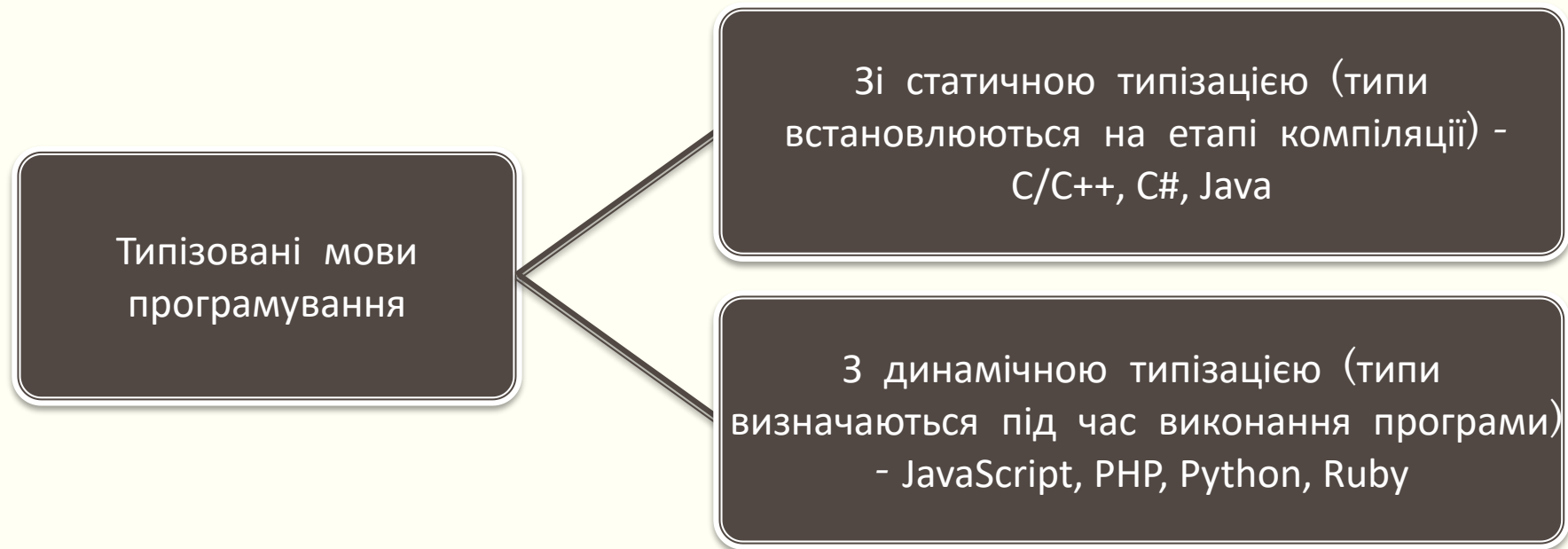
```
int main() {  
    int area;  
    area = LENGTH * WIDTH;  
    printf("value of area : %d", area);  
    printf("%c", NEWLINE);  
  
    return 0;  
}
```

Використовуючи ключове слово const.

```
#include <stdio.h>
```

```
int main() {  
    const int LENGTH = 10;  
    const int WIDTH = 5;  
    const char NEWLINE = '\n';  
    int area;  
  
    area = LENGTH * WIDTH;  
    printf("value of area : %d", area);  
    printf("%c", NEWLINE);  
  
    return 0;  
}
```

Типізовані мови програмування



- **Статична типізація:** компілятор ще до запуску програми виправляє (повідомляє) помилки при невідповідності типів.
- **Динамічна типізація:** дуже важливо приділяти особливу увагу перевіркам та перехопленню помилок.

Статична та динамічна типізація

Мова С (статична типізація)

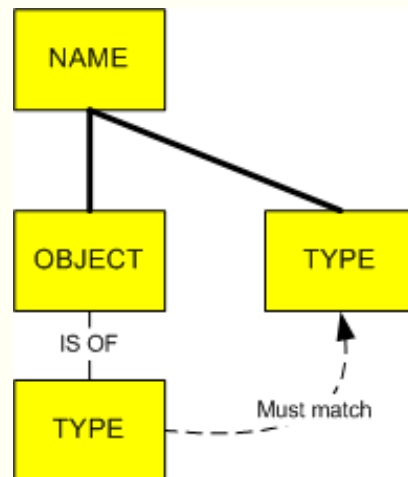
```
1 #include <stdio.h>
2
3 int main(void) {
4     double data;
5     data = 5.0;
6     data = "Hello World!";
7     printf("%s", data);
8     return 0;
9 }
```

In function 'main':

[Error] incompatible types when assigning to type 'double' from type 'char *'

```
1 #include <stdio.h>
2
3 int main(void) {
4     int data;
5     data = 5;
6     data = "Hello World!";
7     printf("%s", data);
8     return 0;
9 }
```

[Warning] assignment makes integer from pointer without a cast



```
E:\GDisk\[College]\[Робота]
Hello world!
```

Python (динамічна типізація)

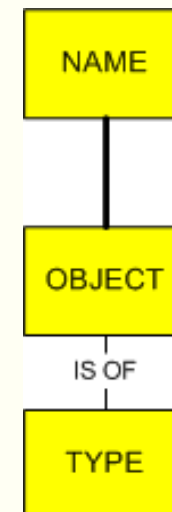
```
data = 5.0
data = "Hello World!"
print(data)
```

Вивід:

```
Hello World!
>>> |
```

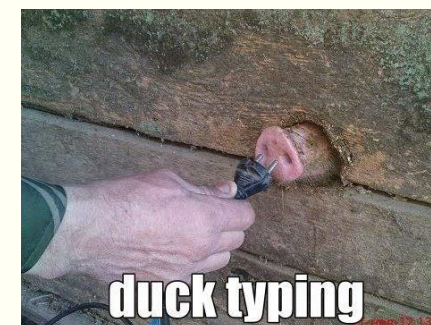


dynamic_typing.py

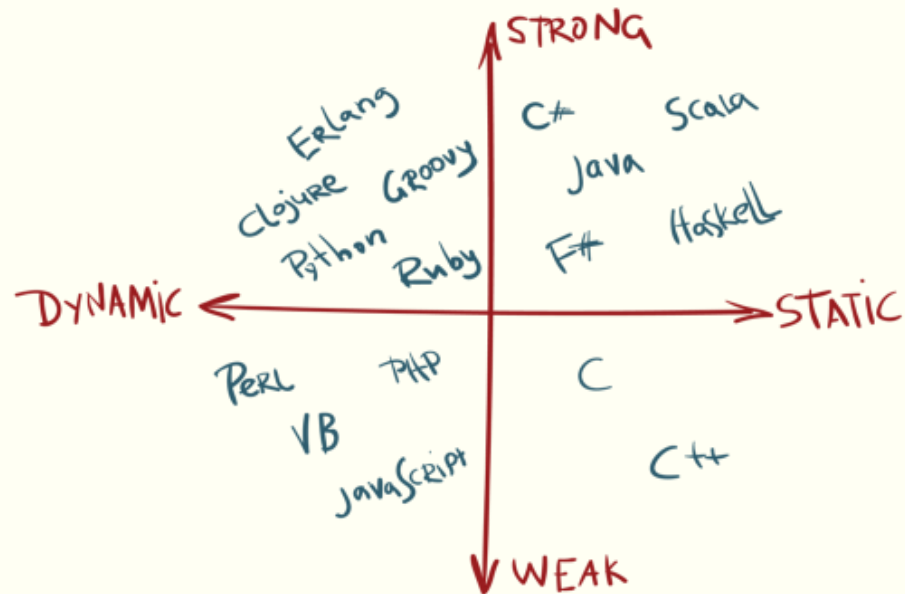


Для Python характерна качина типізація (duck typing)

If it walks like a duck, swims like a duck, and quacks like a duck, then it probably is a duck



Сильна (strong) та слабка (weak) типізація



Типізовані мови програмування

Зі строгою (сильною) типізацією: не дозволяють змішувати у виразах різні типи та не будуть виконувати автоматично неявні перетворення типів

Зі слабкою типізацією: виконують багато неявних перетворень типів автоматично

■ Сильна типізація в Python:

```
temp = "Hello World!"  
temp = temp + 10;
```

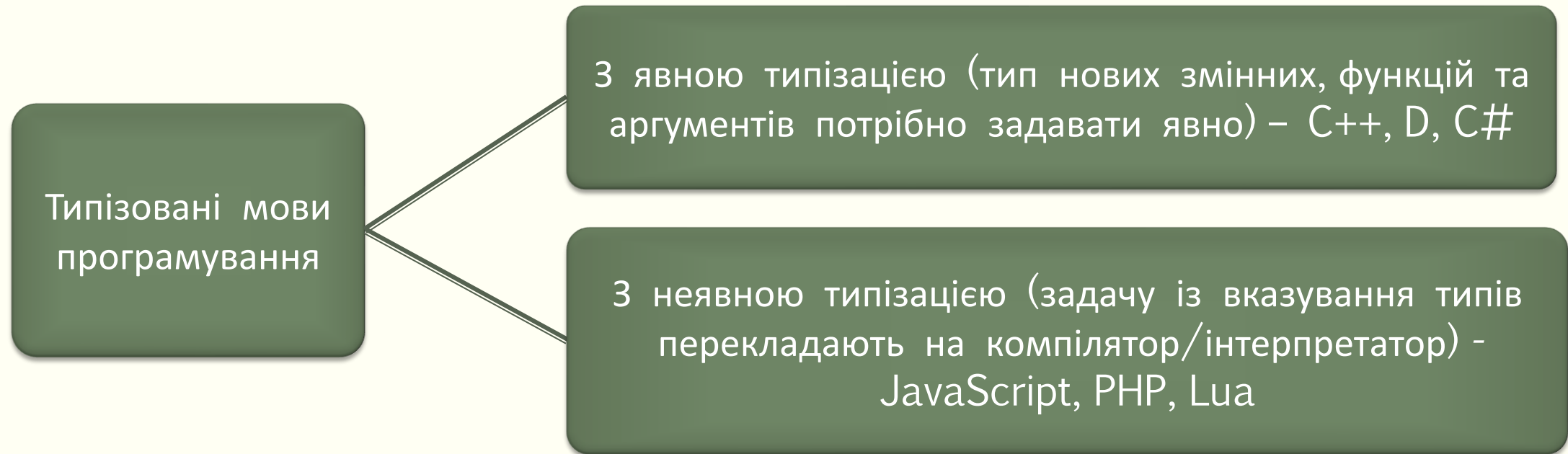
Traceback (most recent call last):

File "E:/GDisk/[College]/[Основи програмування та алгоритмічні мови]/Тема 02. Принципи створення та виконання програм/Code/strong_typing.py", line 2, in <module>

```
temp = temp + 10;
```

TypeError: can only concatenate str (not "int") to str

Явна та неявна типізація



- **C: статична слабка явна типізація** – компілятор перевіряє типи та у виразах, намагається їх сумістити, в коді типи записуються явно
- **Python: динамічна сильна неявна типізація** – інтерпретатор перевіряє типи в процесі виконання коду, при несумісності типів виникає помилка, явно типи в коді не записуються



ДЯКУЮ ЗА УВАГУ!

Наступне питання: організація коду за допомогою функцій

Теми для доповідей

- Вивід типів та алгоритм Гіндлі-Мілнера