Elektronika i Telekomunikacja (wrzesień 2025)

Języki Programowania

1. **Typy danych i operatory.**
   - 1.1. **C++**
     - 1.1.1. Typy podstawowe: int, float, double, char, short, long, unsigned, bool.
     - 1.1.2. Tablice jedno- i wielowymiarowe, struktury, unie, enumeracje.
     - 1.1.3. Operatory arytmetyczne, logiczne, bitowe, warunkowe, hierarchia i łączność.
   - 1.2. **Python**
     - 1.2.1. Typy podstawowe: int, float, str, bool, complex.
     - 1.2.2. Kolekcje: list, tuple, set, dict.
     - 1.2.3. Operatory arytmetyczne, logiczne, porównania, operatory na kolekcjach.
2. **Instrukcje sterujące i obsługa wyjątków.**
   - 2.1. **C++**
     - 2.1.1. Instrukcje: if, switch-case, pętle for, while, do-while.
     - 2.1.2. Obsługa wyjątków: throw, try-catch.
   - 2.2. **Python**
     - 2.2.1. Instrukcje: if, elif, else, pętle for, while.
     - 2.2.2. Obsługa wyjątków: try-except-finally, raise.
3. **Funkcje i programowanie funkcyjne.**
   - 3.1. **C++**
     - 3.1.1. Funkcje, argumenty, zwracanie wartości.
     - 3.1.2. Wskaźniki na funkcje, callbacki.
     - 3.1.3. Rekurencja.
     - 3.1.4. Lambdy, std::function.
   - 3.2. **Python**
     - 3.2.1. Definicje funkcji (def), argumenty pozycyjne i nazwane, wartości domyślne.
     - 3.2.2. Funkcje jako obiekty, dekoratory.
     - 3.2.3. Rekurencja.
     - 3.2.4. Programowanie funkcyjne: map, filter, reduce, lambda.
4. **Wskaźniki, referencje i zarządzanie pamięcią.**
   - 4.1. **C++**
     - 4.1.1. Wskaźniki, referencje, lvalue reference.
     - 4.1.2. Dynamiczna alokacja (new, delete, malloc, free).
     - 4.1.3. Bufory, pule pamięci, operacje na pamięci (memset, memcpy).
   - 4.2. **Python**
     - 4.2.1. Brak wskaźników jawnych – wszystko jest referencją do obiektu.
     - 4.2.2. Automatyczne zarządzanie pamięcią (GC – garbage collector).
     - 4.2.3. id(), is, semantyka kopiowania (copy, deepcopy).
5. **Programowanie obiektowe.**
   - 5.1. **C++**
     - 5.1.1. Klasa, obiekt, hermetyzacja.
     - 5.1.2. Dziedziczenie i polimorfizm.
     - 5.1.3. Przeciążanie operatorów.
     - 5.1.4. Konstruktor, destruktor.
     - 5.1.5. SOLID w C++.

5.2. **Python**

5.2.1. Klasa, obiekt, atrybuty i metody.

5.2.2. Dziedziczenie i polimorfizm (także wielokrotne).

5.2.3. Metody specjalne (__init__, __str__, __add__ itd.).

5.2.4. Właściwości (@property) i hermetyzacja umowna.

5.2.5. SOLID w Pythonie.

6. Szablony i generyczność vs. dynamiczność.

6.1. **C++**

6.1.1. Szablony funkcji i klas.

6.1.2. Programowanie generyczne (STL oparty na szablonach).

6.2. **Python**

6.2.1. Typowanie dynamiczne.

6.2.2. Adnotacje typów (typing).

6.2.3. Generics (List[T], Dict[K,V]).

6.2.4. Duck typing i elastyczność zamiast szablonów.

7. Biblioteki standardowe i kolekcje.

7.1. **C++ – STL**

7.1.1. Kontenery: std::vector, std::string, std::array, std::list, std::map.

7.1.2. Algorytmy: std::sort, std::find, std::for_each, std::transform, std::copy, std::max_element, std::min_element, std::count.

7.2. **Python**

7.2.1. Kolekcje: list, dict, set, tuple.

7.2.2. Biblioteka standardowa: math, itertools, functools, collections.

7.2.3. Wbudowane funkcje (len, max, min, sum, sorted).

8. Współbieżność i proces budowania kodu.

8.1. **C++**

8.1.1. Proces budowania kodu: preprocessing, kompilacja, linkowanie, optymalizacja.

8.1.2. Programowanie wielowątkowe: std::thread, std::mutex, std::lock, sekcje krytyczne, operacje atomowe.

8.1.3. Synchronizacja i komunikacja między wątkami.

8.2. **Python**

8.2.1. Brak procesu kompilacji (interpreter, bajtkod).

8.2.2. Wątki: threading, Global Interpreter Lock (GIL).

8.2.3. Współbieżność: multiprocessing, asyncio.

8.2.4. Synchronizacja: Lock, Semaphore, Queue.