Методические указания для практической работы №2

Программирование на Python

1. Новые концепции

1.1 Ввод данных

в C:

```
int x;
scanf("%d", &x);
```

В Python:

```
x: int = int(input("Введите число: "))
```

- `input()` всегда возвращает строку.
- Чтобы получить число, используем `int(...)`.

1.2 Генерация случайных чисел

В C:

```
#include <stdlib.h>
rand() % 100;
```

В Python:

```
import random
number: int = random.randint(1, 100)  # случайное число от 1 до 100
```

1.3 Условные операторы

В C:

```
if (x > y) {
    printf("x больше");
}
```

В Python:

```
if x > y:
    print("x больше")
```

1.4 Циклы

В C:

```
for (int i = 0; i < 5; i++) {
    printf("%d\n", i);
}
```

В Python:

```
for i in range(5):
    print(i)
```

Бесконечный цикл с условием выхода:

```
while True:
    # действия
    if условие:
        break
```

1.5 Обработка ошибок

Чтобы программа не падала, если пользователь ввёл текст вместо числа, используем конструкцию `try ... except`:

```
try:
    guess: int = int(input("Введите число: "))
except ValueError:
    print("Ошибка: нужно ввести число!")
```

1.6 Измерение времени

```
import time

start: float = time.time()
# ... игра ...
end: float = time.time()
print(f"Время игры: {end - start:.2f} секунд")
```

1.7 Возврат нескольких значений

Пример:

```
def get_point() -> tuple[int, int]:
    return 3, 5

x, y = get_point()
print(x, y)
```

---

2. Пошаговое выполнение задания

1. **Минимальная версия**
   - Сгенерировать случайное число.
   - Дать пользователю несколько попыток угадать.
   - После каждой попытки сообщать «больше» или «меньше».
2. **Выбор уровня сложности**
   - Перед началом игры спросить у пользователя, какой уровень выбрать.
   - Для разных уровней используйте разные диапазоны чисел и количество попыток.
3. **Детективные сообщения**
   - Замените стандартные фразы («больше», «меньше») на сюжетные («свидетель утверждает, что код больше» и т. п.).
4. **Подсказки**

- Реализуйте функцию, которая по секретному числу возвращает подсказку (например: чётность, делимость, диапазон).
- Подсказки выдаются не каждый раз, а ограниченно.

5. **Статистика**
   - Подсчитывайте количество сыгранных игр, побед, среднее время и среднее число попыток.
   - Выводите статистику после каждой игры.
6. **Многоразовая игра**
   - После завершения партии спросите, хочет ли игрок сыграть ещё.
   - Если ответ «нет» — завершите расследование.

---

3. Возможные ошибки и их решения

1. **ValueError** – если вместо числа введён текст → обработать через `try/except`.
2. **ZeroDivisionError** – избегать деления на ноль при подсчёте статистики.