Coding style guide

В отличии от многих других языков, в языке C++ нет общепризнаного стиля написания кода. Можно основываться на рекомендациях, данных Бьёрном Страуструпом в основных рекомендациях *C++ Core Guidelines*. Но и там рекомендации по стилистики не носят обязательный характер. В результате разные программисты пишут на C++ в разных стилях. В этом курсе будет использоваться стиль описанный тут.

1 Стиль наименования идентификаторов

Самые распространённые стили наименования идентификаторов это:

`camelCase` `PascalCase` `snake_case`

В текущем семестре будет использоваться следующий стиль:

- `camelCase` – для имён переменных, полей структур и классов, функций, методов, пространств имён.
- `PascalCase` – для имён классов, структур, перечислений и элементов перечисления.
- `snake_case` – для имён файлов.

2 Стиль отступов

Используется стиль отступов с открывающейся скобкой на новой строке.

```
int printNumbers(int n)
{
    while (n > 0)
    {
        cout << i << " ";
        n--;
    }
}
```

```
int printNumbers(int n) {
    while (n > 0) {
        cout << i << " ";
        n--;
    }
}
```

3 Стиль отступов для однострочной области видимости

В случае если тело области видимости состоит из одной строки, то скобки нужно не писать.

```
int calculateSum(int n)
{
    int sum = 0;
    for (int i = 0; i < n; ++i)
        sum += i;

    return sum;
}
```

```
int calculateSum(int n)
{
    int sum = 0;
    for (int i = 0; i < n; ++i)
    {
        sum += i;
    }
    return sum;
}
```

Исключение – если одна из областей видимости оператора `if else` состоит из более чем одной строки, то все остальные области видимости этого операторы должны обрамляться скобочками.

```
if (a > b)
{
    c = a;
    cout << "a is greater than b\n";
}
else
{
    c = b;
}
```

```
if (a > b)
{
    c = a;
    cout << "a is greater than b\n";
}
else
    c = b;
```

4 Объявление указателей и ссылок

При объявлении указателей звёздочка обязательно ставится рядом с названием типа. Аналогично для положения амперсанда при объявлении ссылок.

```
int a = 100;
int* p = &a;
int& r = a;
```

```
int a = 100;
int *p = &a;
int &r = a;
```

```
int a = 100;
int * p = &a;
int & r = a;
```

5 Избегайте использование using-директив для полного включения пространств имён

```
#include <iostream>

int main()
{
    std::cout << "Hello" <<
    std::endl;
}
```

```
#include <iostream>
using std::cout, std::endl;

int main()
{
    cout << "Hello" << endl;
}
```

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello" << endl;
}
```

В заголовочных файлах вообще не стоит использовать никакие `using`-директивы.

6 Стиль наименования указателей

Имя указатель желательно начинать с символа `p`. Это относится как к обычным, так и к умным указателям.

```
int number = 10;
int* pNumber = &number;
auto pData = std::make_unique(20);
```

```
int number = 10;
int* x = &number;
auto y = std::make_unique(20);
```

7 Стиль наименования приватных полей класса

Приватные поля класса всегда начинаются с символа `m`

```
class String
{
public:
    // ...
private:
   char*  mpData;
   size_t mSize;
   size_t mCapacity;
};
```

```
class String
{
public:
    // ...
private:
   char*  pData;
   size_t size;
   size_t capacity;
};
```

8 Отступы

В качестве отступов используется 4 пробела, а не знак табуляции.

```
bool isPrime(int n)
{
    if (n <= 1)
        return false;

    for (int i = 2; i * i <= n; i++)
    {
        if (n % i == 0)
            return false;
    }
    return true;
}
```