

Работа на ЭВМ и программирование (группа 114)

Занятие 4 (часть 1)

Контактная информация

- Шундеев Александр Сергеевич
- alex.shundeev@gmail.com
- <http://group112.github.io/sem1.html>

Электронная почта

- Тема письма

- 114 Фамилия Имя Отчество
- 114 Фамилия Имя

- Пример

- 114 Иванов Иван Иванович
- 114 Иванов Иван

Стиль оформления программы

Стиль Олмана и стиль Кернигана и Ритчи

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

Оформление блока

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

Открывающая скобка {

```
int foo(int x, double y)
{ <пусто>
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    { <пусто>
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) { <пусто>
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) { <пусто>
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

Закрывающая скобка }

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```


Закрывающая скобка } (неправильно)

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

Отдельные строки для операторов

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

Отдельные строки для операторов (неправильно)

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z); v ++;

    if(z)
    {
        u = abc(u, v); zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z); v ++;

    if(z) {
        u = abc(u, v); zoo();
    }

    return z + x;
}
```

Отдельные строки для операторов (неправильно)

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    } return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    } return z + x;
}
```

Отступы внутри блока

```
{  
    определение1  
    определение2  
    оператор1  
    оператор2  
    оператор3  
}
```

```
... {  
    определение1  
    определение2  
    оператор1  
    оператор2  
    оператор3  
}
```

Отступы внутри блока

```
{  
    определение1  
    определение2  
    ← оператор1  
    оператор2  
    оператор3  
}
```

```
... {  
    определение1  
    определение2  
    ← оператор1  
    оператор2  
    оператор3  
}
```

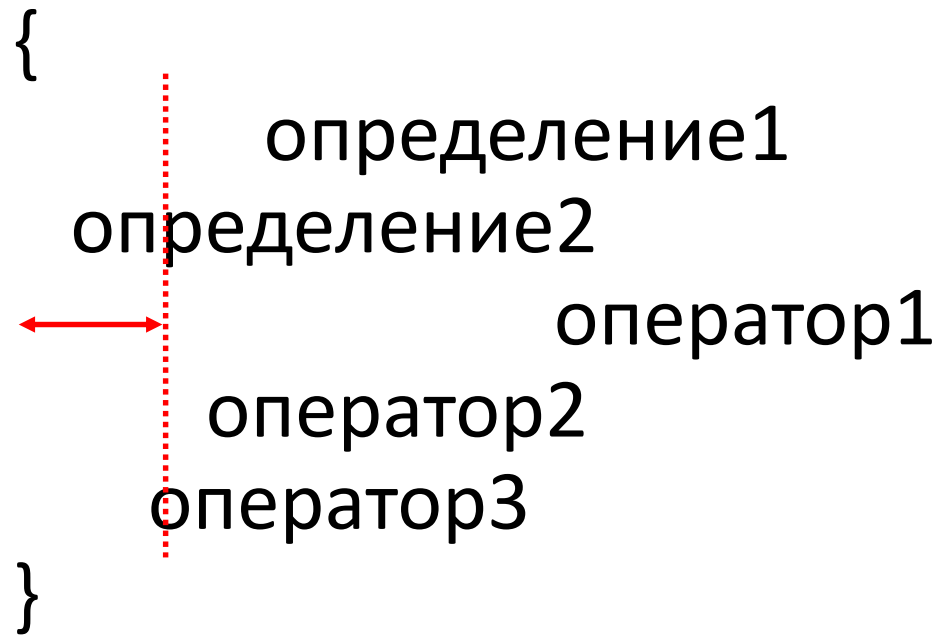
Отступ

2 пробела

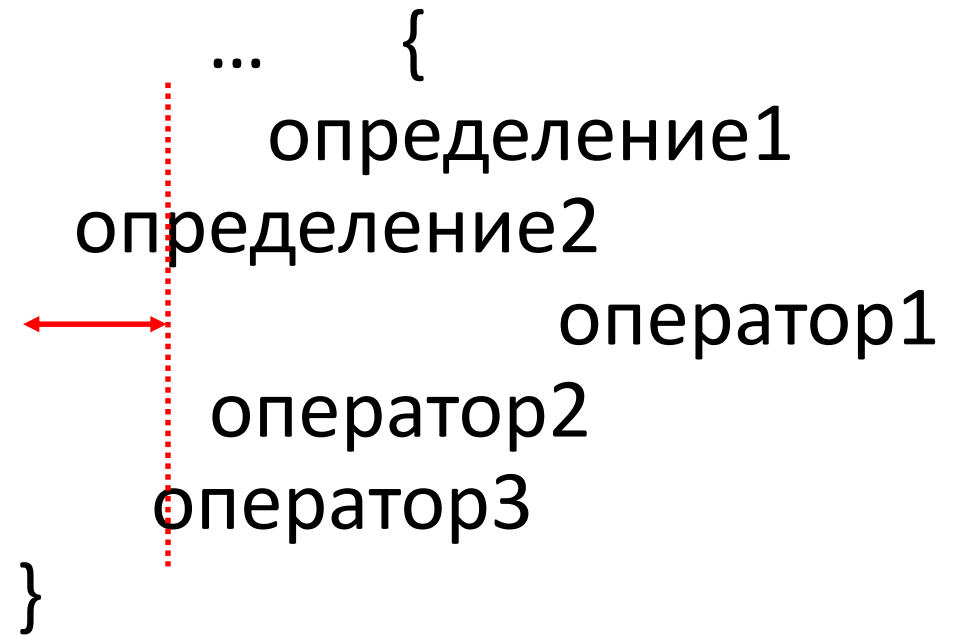
4 пробела (рекомендую)

Отступы внутри блока (неправильно)

```
{
    определение1
    определение2
    оператор1
    оператор2
    оператор3
}
```



```
... {
    определение1
    определение2
    оператор1
    оператор2
    оператор3
}
```



Отступы внутри блока

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

Определения переменных и операторы выделены разными цветами.

Отступы внутри блока

```
int foo(int x, double y)
{
    double u, v;
    int z;
    z = bar(u, v, z);
    v ++;
    if(z)
    {
        u = abc(u, v);
        zoo();
    }
    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;
    z = bar(u, v, z);
    v ++;
    if(z) {
        u = abc(u, v);
        zoo();
    }
    return z + x;
}
```

Определения переменных и операторы выделены разными цветами.

Отступы внутри блока (неправильно)

```
int foo(int x, double y)
{
    double u, v;
    int z;
    z = bar(u, v, z);
    v ++;
    if(z)
    {
        u = abc(u, v);
        zoo();
    }
    return z + x;
}
```

```
int foo(int x, double y) {
    double u, v;
    int z;
    z = bar(u, v, z);
    v ++;
    if(z) {
        u = abc(u, v);
        zoo();
    }
    return z + x;
}
```

Определения переменных и операторы выделены разными цветами.

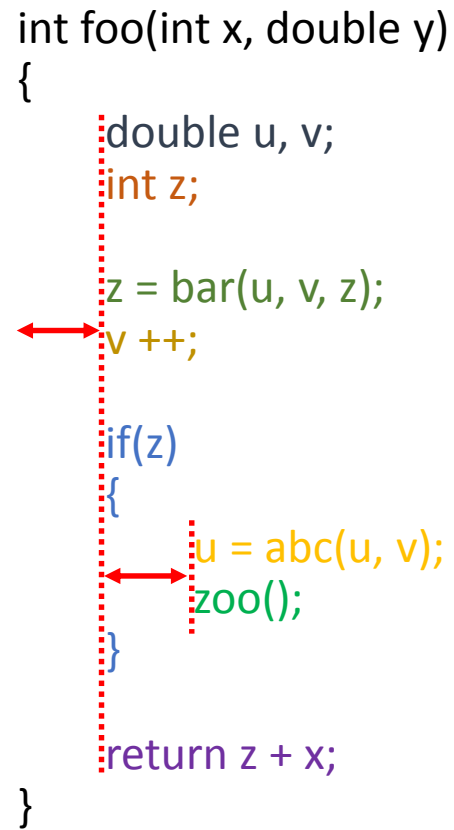
Отступы внутри блока

```
int foo(int x, double y)
{
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z)
    {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```

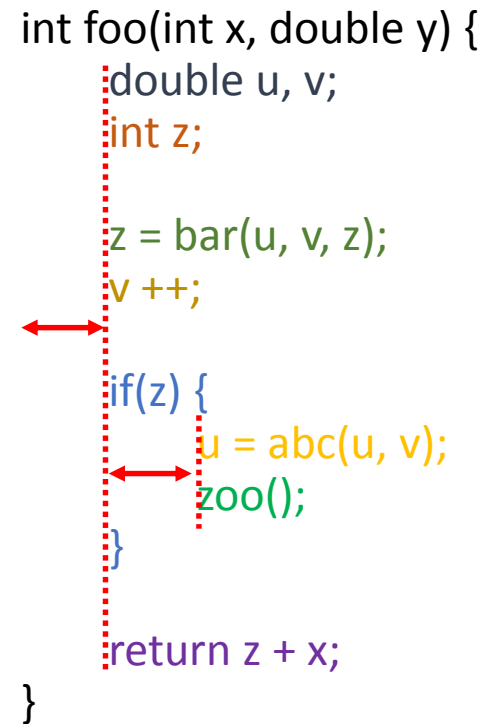


```
int foo(int x, double y) {
    double u, v;
    int z;

    z = bar(u, v, z);
    v ++;

    if(z) {
        u = abc(u, v);
        zoo();
    }

    return z + x;
}
```



Определения переменных и операторы выделены разными цветами.

Оператор if

```
if(...)
  оператор1
```

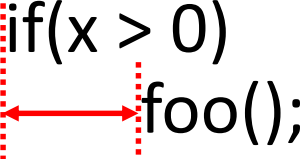
```
if(...)
{
  оператор1
  оператор2
  оператор3
}
```

```
if(...)
  оператор1
```

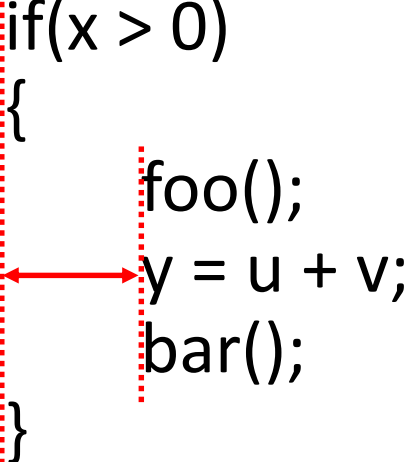
```
if(...) {
  оператор1
  оператор2
  оператор3
}
```

Оператор if (пример)

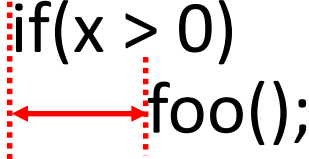
```
if(x > 0)
    foo();
```



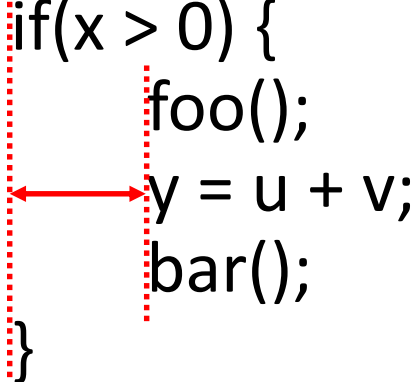
```
if(x > 0)
{
    foo();
    y = u + v;
    bar();
}
```



```
if(x > 0)
    foo();
```



```
if(x > 0) {
    foo();
    y = u + v;
    bar();
}
```



Оператор if-else

```
if(...)
↔ оператор1
else
↔ оператор2
```

```
if(...)
{
↔ оператор1
↔ оператор2
}
else
{
↔ оператор3
↔ оператор4
}
```

```
if(...)
↔ оператор1
else
↔ оператор2
```

```
if(...) {
↔ оператор1
↔ оператор2
} else {
↔ оператор3
↔ оператор4
}
```

Оператор for

```
for(...;...;...)  
↔ оператор1
```

```
for(...;...;...)  
{  
↔ оператор1  
  оператор2  
  оператор3  
}
```

```
for(...;...;...)  
↔ оператор1
```

```
for(...;...;...) {  
↔ оператор1  
  оператор2  
  оператор3  
}
```


Структура программы

```
#include <stdio.h>
<пустая строка>
int foo(FILE *fi, double *p);
<пустая строка>
int main(void)
{
    ...
    foo(fi, &x);
    ...
    return 0;
}
<пустая строка>
int foo(FILE *fi, double *p)
{
    ...
}
```

```
#include <stdio.h>
<пустая строка>
int foo(FILE *fi, double *p);
<пустая строка>
int main(void) {
    ...
    foo(fi, &x);
    ...
    return 0;
}
<пустая строка>
int foo(FILE *fi, double *p) {
    ...
}
```