

Лекция 1,2

Описание и вызов функций. Прототипы функций. Параметры-значения

Описание функций

Функция - это отдельный фрагмент программы, представляющий собой самостоятельный программный блок, возвращающий значение, полученное в ходе реализации алгоритма, представленного внутри функции.

Синтаксис описания функции:

<тип возвращаемого значения> <Имя функции> (Список аргументов)

{

<блок описания локальных переменных>;

<Тело функции (набор операторов, реализующий алгоритм)>

return <возвращаемое значение>;

}

- <Тип возвращаемого значения> - определяет вид возвращаемого значения (объем памяти, диапазон значений);

* Лекция 1, 2

Описание функций

- <Имя функции> - Определяет как обратиться к функции;
- Список аргументов - список входных параметров функции.

Такие параметры называются **формальными параметрами**. Они резервируют места, в которые при вызове функции будут помещены аргументы (**фактические параметры**).

Список параметров задается следующим образом:

(<Тип параметра1> <Имя параметра 1>, ..., <Тип параметра N> <Имя параметра N>)

Параметры, описанные в заголовке при описании функции считаются **формальными** (они определяют порядок и типы аргументов на входе функции), а те параметры, которые подставляются на их место при вызове функции (реально использоваться), называются **фактическими**.

При вызове функции необходимо полное соответствие фактических параметров формальным **по типу и порядку следования**.

* Лекция 1, 2

Описание функций

Вид заголовка функции:

- `int Func1(int a, char c);`
- `float Func2 (unsigned char cc);`
- `double Summa (float a, float b, unsigned int k);`
- `unsigned long Factorial(unsigned char n);`

* Функции языка C (C++)

Описание функций

- <Локальные переменные> - переменные (области памяти), необходимые для реализации алгоритма функции. Они создаются в функции и видимы только в ее области (за ее пределами невидимы - исчезают). Исключения составляют статические переменные `static`, которые передают свой адрес).

Локальные переменные - принято делать **автоматическими**. При вызове функции под них выделяются области памяти в **СТЕКЕ**. После завершения работы функции области памяти в стеке очищаются и доступ к этим переменным невозможен.

- <Тело функции> - это набор операторов, реализующих некоторый алгоритм, закрепленный за функцией. Тело функции - это почти то же самое, что и тело главной программы. Функция, получившая набор фактических параметров, соответствующих порядку и типу формальных, реализует набор операторов.

* Функции языка C (C++)

Описание функций

Пример функции с локальными переменными:

```
double Summa(float a, float b)                int f()
{                                              {
    // локальные переменные
    double s;                                char i,j,k;
    int i;
    // реализация алгоритма
    <операторы функции>
    // возвращаемое значение
    return s;                                return k;
}
```

* Функции языка C (C++)

Вызов функции

Функции вызываются в выражениях основного блока программы.

```
void main()
{
    double Res;
    int Res1;
    float f=2.5;
    .
    .
    .
    Res=Summa(f,4.9);
    Res1=f();
}
```

* Функции языка C (C++)

Описание функций

- Возвращающий оператор return состоит из ключевого слова return и возвращающего в программу значения (переменная, выражение, константное значение). Функция, возвращающая значение должна обязательно вызываться в выражении (значение, возвращаемое функцией поступает в переменную, находящуюся слева в выражении). Если функция не возвращает значения, то она может вызываться обособленно (вне выражения). В этом случае функция реализуется как процедура.

В определении функции после оператора `return` могут следовать другие операторы, но выполняться они не будут.

На выполнении оператора `return` обращение к функции заканчивается.

* Функции языка C (C++)

Если функция не возвращает значения то на месте типа возвращаемого значения указывают тип

`void`

В этом случае функция используется как процедура. Она вызывается не в выражении, а обособленно.

Например, функция

```
void func(char p)
```

```
{
```

```
·
```

```
·
```

```
·
```

```
Return; // может отсутствовать
```

```
}
```

Вызовется в главном блоке `main()` следующим образом:

```
main()
```

```
{
```

```
    func(5);
```

```
}
```

* Функции языка C (C++)

ПРОТОТИП ФУНКЦИИ

Прототип функции - это заголовок пользовательской функции, который содержит всю необходимую информацию для обращения к функции. Обычно прототип функции должен находиться перед главной частью программы. Прототип сообщает компилятору, выполняющему синтаксический анализ текста программы о том, какие функции она использует.

Пример.

```
#include<>
#include<>
// прототипы функций
<тип возвращаемого значения 1> <Имя функции 1>(список аргументов1);
<тип возвращаемого значения 2> <Имя функции 2>();
void main()
{
    <Тело программы, использующая описанные в прототипах функции>
}
```

Где описаны сами функции????

* Функции языка С (С++)

```
#include<>
```

```
#include<>
```

```
// прототипы функций
```

```
<тип возвращаемого значения 1> <Имя функции 1>(список аргументов1);
```

```
<тип возвращаемого значения 2> <Имя функции 2>();
```

```
void main()
```

```
{
```

```
    <Тело главной программы, использующая описанные в прототипах функции>
```

```
}
```

```
// описание функций
```

```
<тип возвращаемого значения 1> <Имя функции 1>(список аргументов1)
```

```
{
```

```
    <тело функции1>;
```

```
    return возвращаемое значение;
```

```
}
```

```
<тип возвращаемого значения 2> <Имя функции 2>()
```

```
{
```

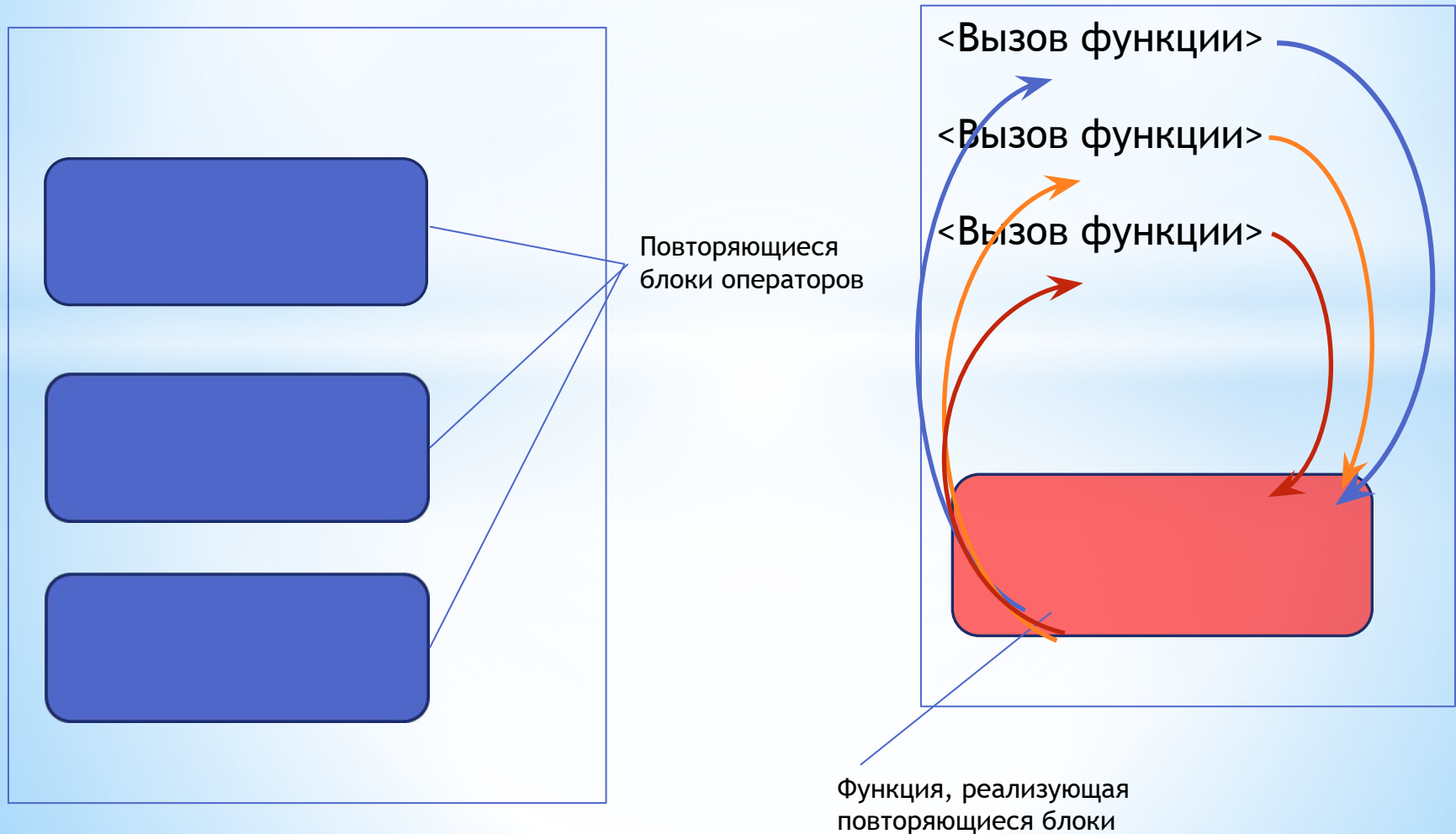
```
    <тело функции 2>;
```

```
    return возвращаемое значение;
```

```
}
```

* Функции языка С (С++)

Помимо выше отмеченного, использование подпрограмм (функций) позволяет сократить код программы, организовать повторяющиеся участки кода программы в виде одного блока с организацией обращения к нему.



* Функции языка C (C++)

Пример 1. По ходу реализации алгоритма требуется 3 раза считать сумму элементов последовательности от 1 до n (n - изменяется от 2 до 25).

Реализация без функций:

```
void main()
{
    unsigned char i;
    unsigned char Res;
```

```
    Res=0;
    for (i=1;i<=10;i++)
        Res=Res+i;
```

```
    Res=0;
    for (i=1;i<=5;i++)
        Res=Res+i;
```

```
    Res=0;
    for (i=1;i<=20;i++)
        Res=Res+i;
```

```
}
```

Реализация с помощью функции:

```
unsigned char Sum(unsigned char n);
void main()
```

```
{
    unsigned char Res;
```

```
    Res=Sum(10);
```

```
    Res=Sum(5);
```

```
    Res=Sum(20);
```

```
}
```

```
unsigned char Sum(unsigned char n)
```

```
{
```

```
    unsigned char i, s;
```

```
    s=0;
```

```
    for (i=1;i<=n; i++)
```

```
        s=s+i;
```

```
    return s;
```

```
}
```

* Функции языка C (C++)

Пример 2. По ходу реализации алгоритма требуется 2 раза вывести в текстовом окне, с заданными координатами и цветом, сумму двух дробных чисел.

Реализация с помощью функции:

Реализация без функций:

```
#include<conio.h>
#include<stdio.h>
void main()
{
    textbackground(0);
    clrscr();

    textbackground(1);
    window(10,10,20,20);
    clrscr();
    printf(“%6.2f”,23.5+89.6);
    getch();

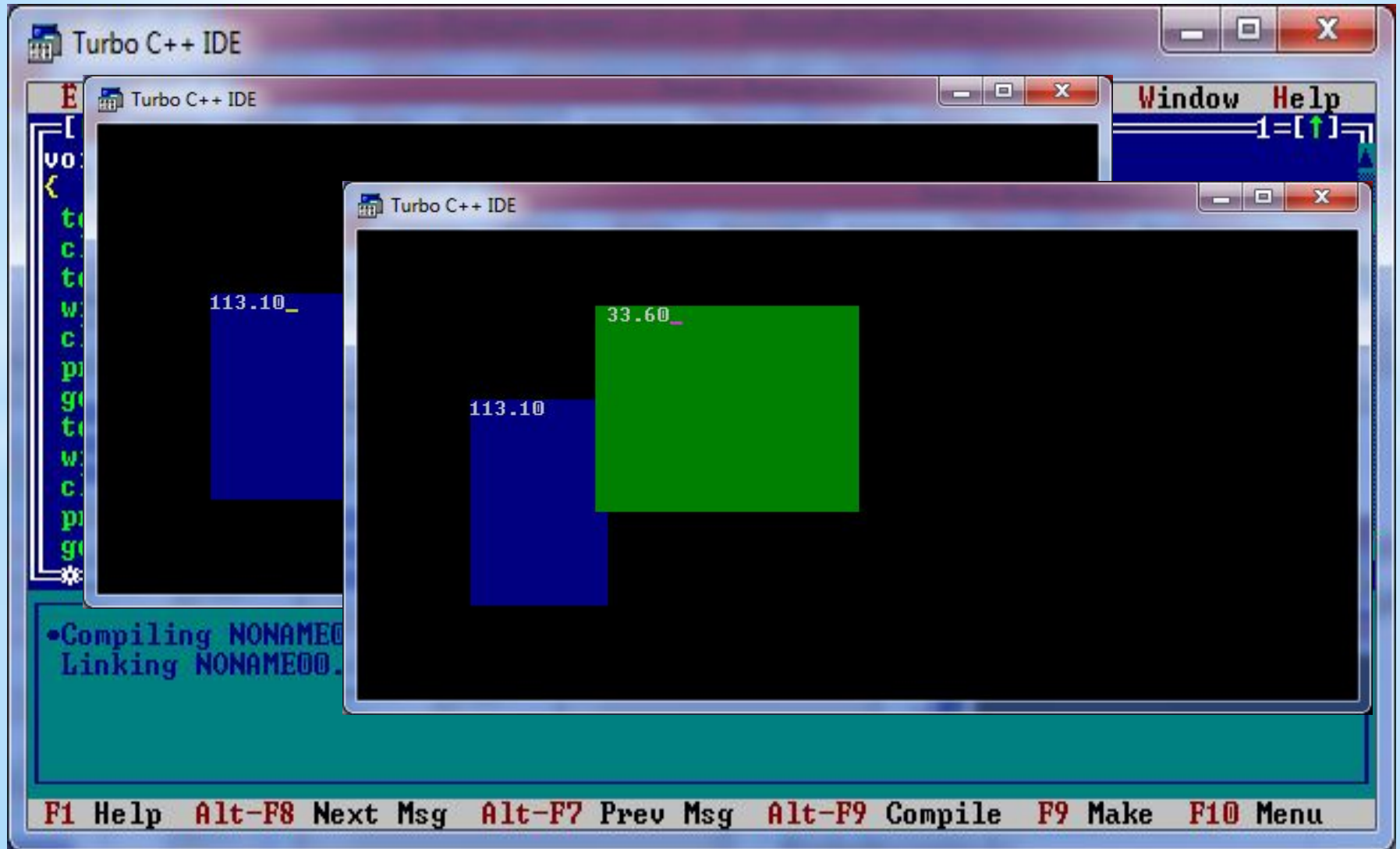
    textbackground(2);
    window(20,5,40,15);
    clrscr();
    printf(“%6.2f”,3.5+30.1);
    getch();
}
```

```
#include<conio.h>
#include<stdio.h>
void okno_sum(float a, float b, unsigned char color,
unsigned char x1, unsigned char y1, unsigned char x2,
unsigned char y2);
void main()
{
    textbackground(0);
    clrscr();
    okno_sum(23.5,89.6,1,10,10,20,20);
    okno_sum(3.5,30.1,2,20,5,40,15);
}

void okno_sum(float a, float b, unsigned char color,
unsigned char x1, unsigned char y1, unsigned char x2,
unsigned char y2);
{
    textbackground(color);
    window(x1,y1,x2,y2);
    clrscr();
    printf(“%6.2f”,a+b);
    getch();
}
```

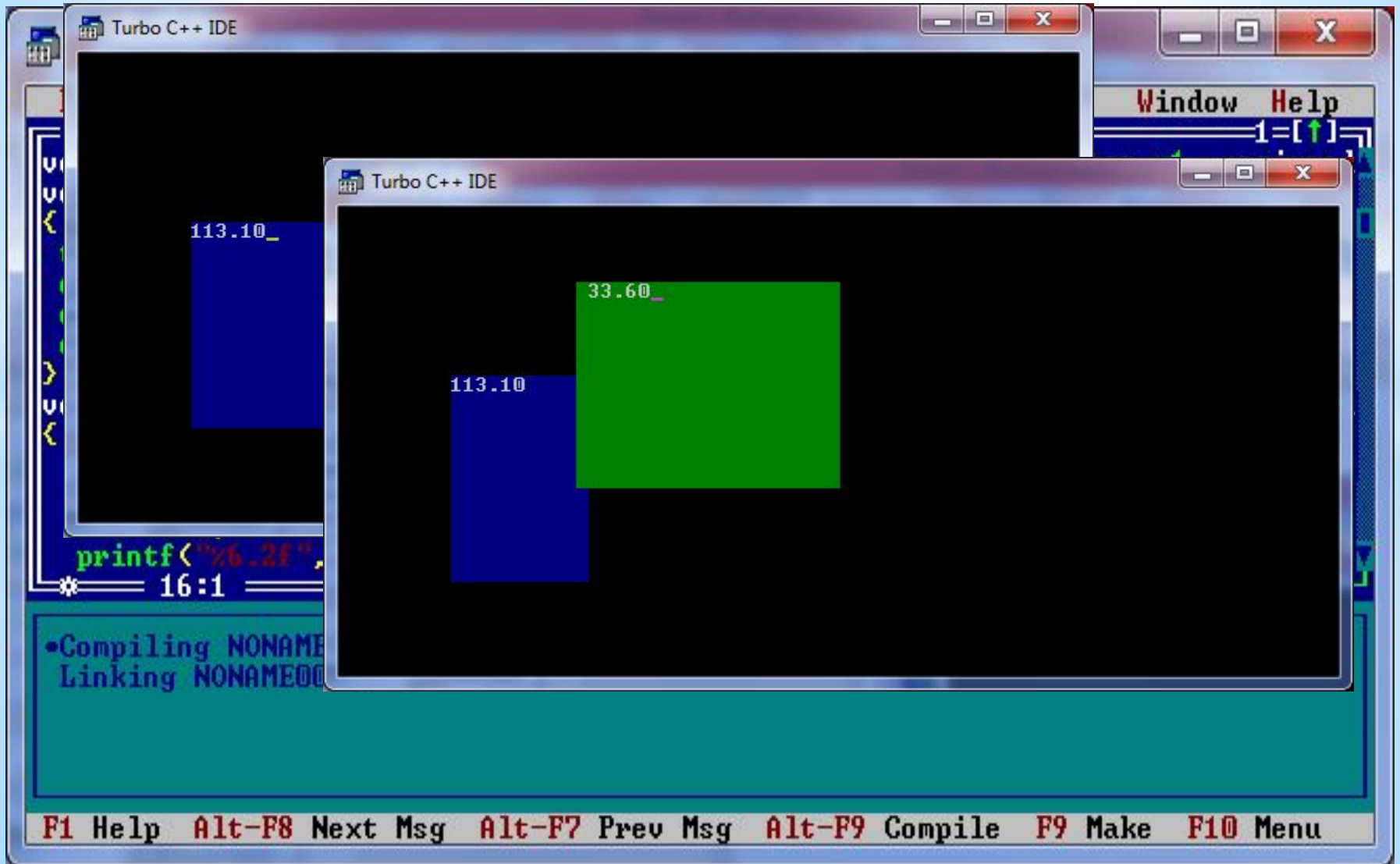
* Функции языка С (С++)

Реализация первого варианта



* Функции языка С (С++)

Реализация второго варианта



* Функции языка С (С++)

Параметры-значения

В приведенных выше примерах все параметры передаваемые в функции являются **параметрами-значениями**.

- `void okno_sum(float a, float b, unsigned char color, unsigned char x1, unsigned char y1, unsigned char x2, unsigned char y2);`
- `unsigned char Sum(unsigned n);`
- `int Func1(int a, char c);`
- `float Func2 (unsigned char);`
- `double Summa (float a, float b, unsigned c);`
- `unsigned long Factorial(unsigned char n);`

* Функции языка C (C++)

Параметры-значения

Данные параметры передают копии своих значений в функцию (память под копии выделяется в СТЕКЕ).

Параметры-значения можно изменять внутри функции, но эти изменения будут локальными (изменяться будут копии параметров в СТЕКЕ) - они не передадутся в основную программу (фактические параметры, указанные при вызове функции не изменят своих значений).

По завершению работы функции, выделенная в СТЕКЕ память под параметры-значения освободится.

Исходя из этого параметры-значения могут использоваться только как входные параметры функций.

* Функции языка С (С++)

Параметры-значения

Пример. Функция принимает, увеличивает и выводит входной параметр.

```
#include <iostream.h>
#include <conio.h>
// прототип
void f(int a);
void main()
{
    textbackground(0);
    clrscr();
    int b=9;
    f(b);
    cout<<b<<endl;
    getch();
}
//реализация
void f(int a)
{
    a++;
    cout<<a<<endl;
}

```



* Функции языка C (C++)

Комментарий:

```
#include <iostream.h>
#include<conio.h>
// прототип функции
void f(int a);
void main()
{
    textbackground(0);
    clrscr();
    int b=9;
    f(b);
    cout<<b<< endl;
    getch();
}
//реализация функции
void f(int a)
{
    a++;
    cout<<a<< endl;
}
```

При вызове функции f(b) в стеке выделяется память под значение фактического параметра b. В ячейке стека значение 9.

При выполнении оператора **a++** значение 9 увеличивается на 1 (в ячейке стека значение 10). Это значение выводится с помощью оператора cout.

Стек очищается (область памяти в которой содержится значение 10 становится недоступной). Работа функции завершена. Недоступен параметр a.

Оператор **cout<<b<<endl** выводит значение ячейки памяти, в которой хранится значение переменной b и не измененное в ходе работы функции f.

* Функции языка С (С++)

Результат реализации функции



The image shows two overlapping windows of the Turbo C++ IDE. The background window displays the number '10'. The foreground window displays the numbers '10' and '9' on separate lines, representing the output of a function call.

```
Turbo C++ IDE
10

Turbo C++ IDE
10
9
```