



MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI



DASTURLASH 1

SWD1316

02

MAVZU

ALGORITMLASH VA DASTURLASHNING ASOSIY TUSHUNCHALARI.

```
children = []  
for n, child in enumerate(ast[1:]):  
    children.append(dotwrite(child))  
print ' %s -> {' % node  
for name in children:  
    print '%s' % name,
```



**SHOKIROV SHODMON
SHOYIMOVICH**








Informatika asoslari kafedrası
katta o'qituvchisi



Ma'ruza rejasi



-  Binar, unar va ternar operatorlar
-  Statik operator (sizeof)
-  Kompilyator va uning turlari
-  Kiritish va chiqarish operatorlari
-  Preprotssessor deriktivalari

Binar, unar va ternar operator



Binar amallar additiv ya'ni $+$ qo'shish va $-$ ayirish amallariga, hamda multiplikativ, ya'ni $*$ ko'paytirish, $/$ bo'lish va $\%$ modul olish amallariga ajratiladi.

Butun sonni butun songa bo'lganda natija butun songacha yaxlitlanadi.

Misol uchun, $20/3 = 6$; $(-20)/3 = -6$; $20/(-3) = -6$.

Unar amallarga ishorani o'zgartiruvchi unar minus $-$ va unar plyus $+$ amallari kiradi. Bundan tashqari inkrement $++$ va dekrement $--$ amallari ham unar amallarga kiradi.

Sizeof amali



sizeof amali operand sifatida ko'rsatilgan ob'ektning baytlarda xotiradagi hajmini hisoblash uchun ishlatiladi.

Bu amalning ikki ko'rinishi mavjud:

sizeof ifoda; sizeof (tip) ;

Shuni ta'kidlab o'tish lozimki, sizeof funksiyasi preprotssessor qayta ishlash jarayonida bajariladi, shuning uchun dastur bajarilish jarayonida vaqt talab etmaydi.

Misol uchun:

`sizeof 3.14 = 8`

`sizeof 3.14f = 4`

`sizeof 3.14L = 10`

`sizeof(char) = 1`

`sizeof(double) = 8.`

Kompilyator va uning turlari;



Kompilyator bu – dastur tuzish uchun, ya’ni kodlarning qonun qoida bo’yicha terilganligini nazorat qiluvchi va dasturning natijasini chiqaruvchi amaliy dasturdir.

Kompilyator turlari:

1. Dev;
2. CodeBlocks;
3. Visual Studio;
4. Borland C++Builder;
5. EmbarCadero;



Kiritish chiqarish operatorlari



- Chiqarish operatorining C tilidagi ko'rinishi mavjud.

Printf("%d",6*7);

Consol rejimida
ma'lumotlarni
ekranga chiqaradi

Oqimdan keyin o'zgaruvchi yoki
raqamlar arifmetik amallar bilan
yozilsa uni hisoblab chiqaradi

cout << "Natija" << 6*7 << **endl**;

Chiqarish
oqimi

Izoh yozish.
Qo'shtirnoq ichidagi
yozuv ekranga chiqadi

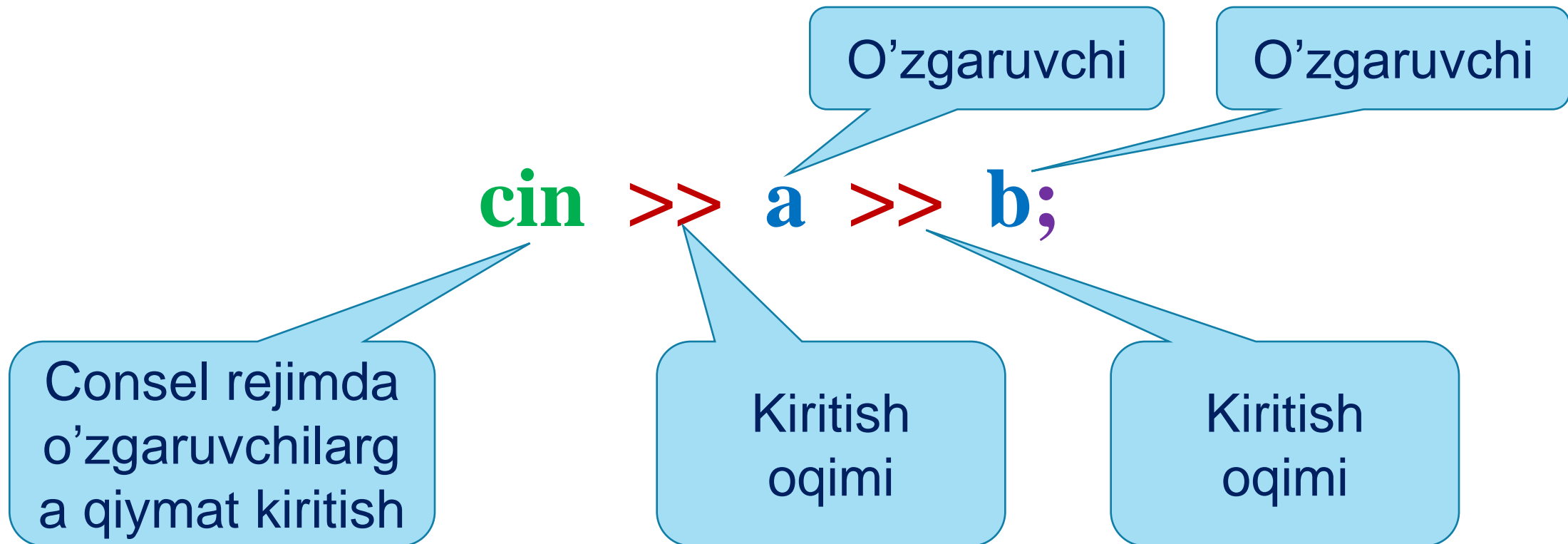
Kursorni bitta
qator pastga
tushirish

Kiritish chiqarish operatorlari

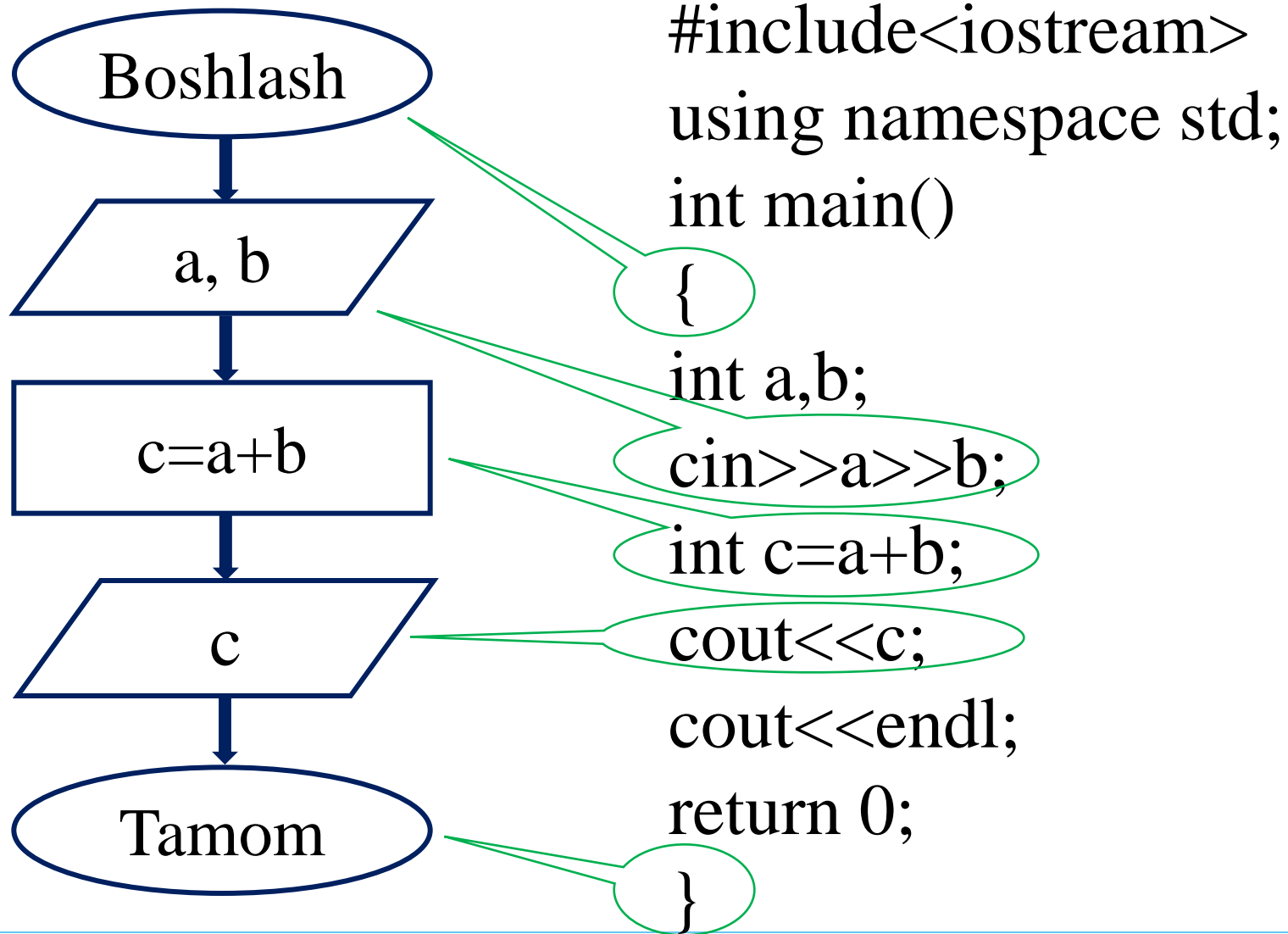


Kiritish operatorining C tilidagi ko'rinishi

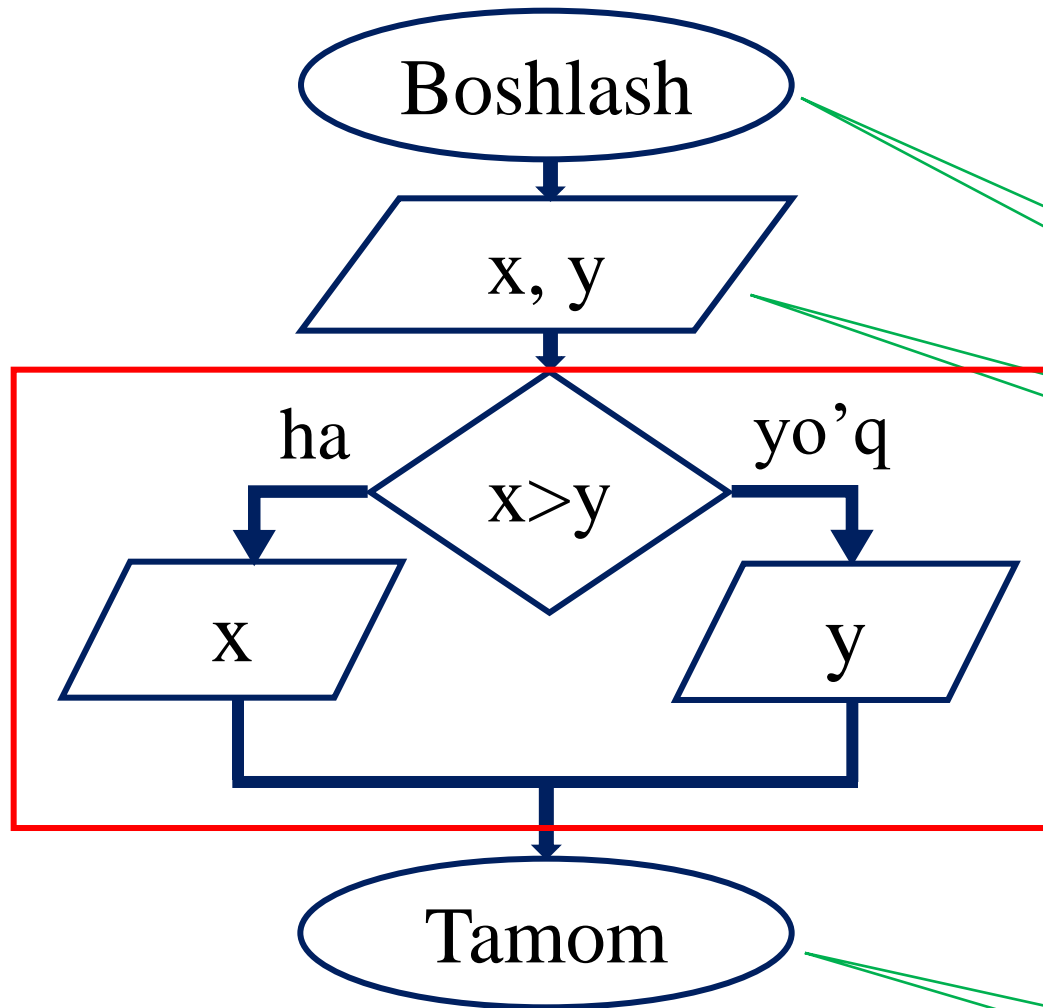
scanf("%d",&a);



Chiziqli algoritmning blok sxemasi va dasturi



Tarmoqlanuvchi algoritmning blok sxemasi va dasturi



```
#include<iostream>
using namespace std;
int main()
```

```
{
```

```
int x,y;
```

```
cin>>x>>y;
```

```
if(x>y)
```

```
cout<<x<<endl;
```

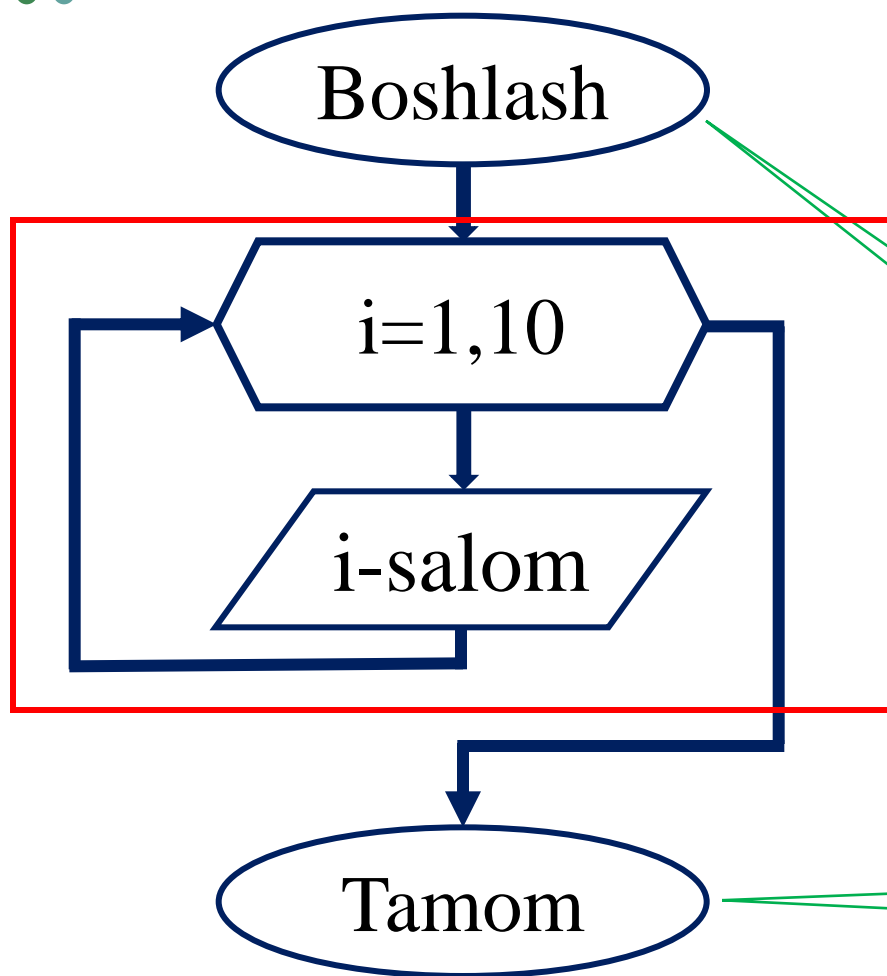
```
else
```

```
cout<<y<<endl;
```

```
return 0;
```

```
}
```

Takrorlanuvchi algoritmning blok sxemasi va dasturi



```
#include<iostream>
using namespace std;
int main()
{
    int i;
    for(i=1;i<=10;i++)
        cout<<i<<" -salom"<<endl;
    return 0;
}
```

Preprotssessor direktivalari

- **Preprotssessor** direktivalari kompilyatsiya jarayonidan oldin preprotssessor tomonidan bajariladi. Natijada dastur matni preprotssessor direktivalari asosida o'zgartiriladi.

#include<fayl_nomi> bu direktiva standart bibliotekalardagi funksiyalarni dasturga joylash uchun foydalaniladi.

#define <almashtiruvchi ifoda> <almashinuvchi ifoda>

Bu direktiva bajarilganda dastur matnidagi almashtiruvchi ifodalar almashinuvchi ifodalarga almashtiriladi.

Preprotssessor direktivalari



Almashtiruvchi **define** direktivasidan nomlangan konstantalar kiritish uchun foydalanish mumkindir.

Misol uchun:

```
#define ZERO 0
```

Agar dasturda quyidagi matn mavjud bo'lsin:

```
int d = ZERO;
```

Preprotssessor bu matnda har bir **ZERO** konstantani uning qiymati bilan almashtiradi, va natijada quyidagi matn hosil bo'ladi.

```
int d = 0;
```

Misol



```
#include <stdio.h>
#define begin {
#define end }
#define pr printf ("\n Salom, Dunyo! \n");
void main()
begin
pr;
end
```

Preprocessorlarni boshqarish



- oldindan tayyorlangan simvollar ketma ketligi bilan identifikatorlarni almashtirish ;
- ko'rsatilgan fayldagi matnni dasturga ulash(bog'lash) ;
- dasturdan ba'zi qismlarni olib tashlash (shartli kompilyasiya) ;

Preprocessorlarni boshqarish



1. **#define** - makrosning aniqlanishi yoki preprocessorning identifikatori ;
2. **#include** - fayldan tekstni o'qish ;
3. **#undef** - identifikatorni va makrosni aniqlanishini bekor qilish;
4. **#if** - shart ifodani tekshirish;
5. **#ifdef** - identifikator aniqlanishini tekshirish;
6. **#else** - #if uchun alternativ tarmoqning boshlanishi;
7. **#line** - keyingi satr nomerini almashtirish;
8. **#error** - translatsiya xatosi haqida xabarni formatlashtirish;
9. **#pragma** – oldindan aniqlangan amallar;
10. **#** - bo'sh direktivalar.

XULOSA



Algoritm – bajariladigan ishning ketma-ketligi. Dastur esa – algoritmning bir ko'rinishidir.

Kompilyator – turli-xil dasturlash tillarida tuziladigan dastur kodlarini tekshiradigan amaliy dasturdir.

Preprotssessor – kompyuterning ichki xotiralari bilan ishlash imkoniyatlarini yaratadi.



MUHAMMAD AL-XORAZMIY NOMIDAGI TOSHKENT AXBOROT TEXNOLOGIYALARI UNIVERSITETI

```
def dotwrite(ast):  
    nodename = getNodeName()  
    label=symbol.sym_name.get(int(ast[0]),ast[0])  
    print '      %s [label="%s' % (nodename, label),  
    if isinstance(ast[1], str):  
        if ast[1].strip():  
            p  
        else:  
            p  
    else:  
        print  
        children = []  
        for n, child in enumerate(ast[1:]):  
            children.append(dotwrite(child))  
        print '      %s -> {' % nodename  
        for name in children:  
            print '%s' % name,
```

E'TIBORINGIZ UCHUN RAXMAT!



**SHOKIROV SHODMON
SHOYIMOVICH**



Informatika asoslari kafedrası
katta o'qituvchisi