C++ TILDE FUNKSIYALAR. MUTAXASSISLIK MASALALARINI YECHISHDA FUNKSIYALARDAN FOYDALANISH

Foydalanuvchi funksiyalari dasturchiga murakkab dasturni soddaroq bo'laklarga bo'lish hamda har bir bo'lakni bir-biridan ajralgan holda tuzish imkoniyatini beradi. Shu yol bilan dasturchi o'z ishini birmuncha osonlashtiradi.

Bu kabi holatlarda dasturchilar ishini engillashtirish maqsadida C++ dasturlash tilida andazalar qo'llash usuli kiritilgandir.

- **Standart Template Library**(qisqacha, STL) - standart andozalar kutubxonasi bo'lib, u konteynerlar, iteratorlar va algoritmlarni o'z ichiga oladi.

Standart andozalar kutubxonasi

Konteynerlar – bir xil turdagi qiymatlarni o'zida saqlaydigan maxsus tuzilma bo'lib, o'zida saqlayotgan qiymatlar ustida o'ziga xos amallar bajarish, tayyor algoritmlardan foydalanish imkoniyatini beradi.

Har bir konteyner o'zida **a'zo** funksiyalarini saqlaydi.

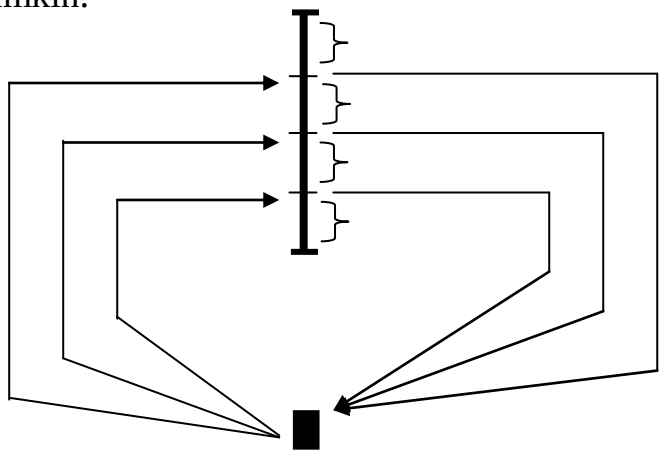
3.6.1. Funksiyalar

Dastur ta'minotini yaratish amalda murakkab jarayon hisoblanadi. Dastur tuzuvchi dastur kompleksini bir butunlikdagi va uning har bir bo`lagining ikir-chigirliklari va sezilmas farqlarini hisobga olishi kerak bo`ladi. Dasturlashga tizimli yondoshuv shundan iboratki, dastur tuzuvchi oldiga qo`yilgan masala oldindan ikkita – uchta nisbatan kichik masala ostilarga bo`linadi. O`z navbatida bu masala ostilari ham yana kichik masala ostilariga bo`linadi. bu jarayon toki mayda masalalar oddiy standart proxeduralar yordamida yechish mumkin bo`lguncha davom etadi. shunday qilib, masala dekompozixiyasi amalga oshiriladi.

Ikkinchi tomondan, dasturlashda shunday holatlar kuzatiladiki, unda dasturning turli joylarida mazmunan bir xil algoritmlarni bajarishga to`g`ri keladi. Bu algorimlar asosiy yechilayotgan masaladan ajratib olingan qandaydir masala ostini yechishga mo`ljallangan bo`lib, yetarlicha mustaqil qiymatga (natijaga) egadir.

Bunday hollarda dasturni ixcham va samarali qilish uchun C++ tilida dastur bo`lagini alohida ajratib olib, uni funksiya ko`rinishida aniqlashga imkon mavjud. Funksiya C++ tilida masala yechishdagi kalit elementlaridan biri hisoblanadi.

Dasturda funksiyaga bo`ladigan murojaatni sxematik ravishda quyidagicha ifodalash mumkin.



Funksiyalar parametrlari va argumentlari

Dasturda ishlatiladigan har qanday funksiya unda e'lon qilinishi kerak. Odatda funksiyalar e'loni sarlavha fayllarda e'lon qilinadi va #include direktivasi yordamida dastur matniga qo`shiladi. Funksiya e'lonini funksiya prototipi tavsiylaydi (ayrim hollarda signatura deyiladi). Funksiya prototipi quyidagi ko`rinishda bo`ladi:

<qaytaruvchi qiymat turi><funksiya nomi>(<parametrlar ro`yxati >);

Bu yerda*<qaytaruvchi qiymat turi>* – funksiya ishlashi natijasida u tomonidan qaytaradigan qiymatning turi. Agar qaytariladigan qiymat turi ko`rsatilmagan bo`lsa, kelishuv bo`yicha funksiya qaytaradigan qiymat turi int hisoblanadi, *<parametrlar ro`yxati >* – vergul bilan ajratilgan funksiya parametrlarining turi va nomlari ro`yxati. Parametr nomini yozmasa ham bo`ladi. Ro`yxat bo`sh bo`lishi ham mumkin. Funksiya prototiplariga misollar:

```
int almashsin(int, int);
double max(double x, double y);
void func();
```

Funksiya prototipi tushirib qoldirilishi mumkin, agar matnda funksiya aniqlanishi uni chaqiradigan funksiyalardan oldin kelgan bo`lsa. Lekin bu holat yaxshi uslub hisoblanmaydi, ayniqsa o`zaro bir–biriga murojaat qiluvchi funksiyalarni e'lon qilishda muammolar yuzaga kelishi mumkin.

Funksiya aniqlanishi – uning sarlavhasi va figurali qavsga (‘{‘,’}‘) olingan qanday amaliy mazmunga ega tanadan iborat bo`ladi. Agar funksiya qaytaruvchi turi void turidan farqli bo`lsa, uning tanasida albatta mos turdagi parametrga ega return operatori bo`lishi shart. Agar funksiyaning qiymati dasturda ishlatilmaydigan bo`lsa, funksiyadan chiqish uchun parametrsiz return operatori ishlatilishi mumkin yoki umuman return ishlatilmasligi mumkin. Oxirgi holda funksiyadan qaytish oxirgi apiluvchi qavsga etib kelganda ro`y beradi. Dasturda biror modulda albatta funksiyaaniqlanishi bo`lishi kerak va u yagonadir, funksiya e’loni esa bir necha marta yozilishi mumkin shu funksiyani ishlatadigan modullarda. Funksiya aniqlanishdagi sarlavhada barcha parametrlarni ismi bo`lishi shart, prototipta esa shart emas.

Odatda dasturda funksiya ma’lum bir ishni amalga oshirish uchun u chaqirilishi kerak. Funksiyaga murojaat qilganda u qo`yilgan masalani yechadi va o`z ishini tugatishida qandaydir qiymatni natijasifatida qaytaradi.

Funksiyani chaqirish uchun uning nomi va undan keyin qavs ichida argumentlar ro`yxati beriladi:

<funksiya nomi>(<argument1>, <argument2>,…, <argumentn >);

Bu yerda har bir *<argument>* – funksiya tanasiga uzatiladigan va keyinchalik hisoblash jarayonida ishlatiladigan o`zgaruvchi, ifoda yoki o`zgarmasdir. Argumentlar ro`yxati bo`sh bo`lishi mumkin.

Funksiyalar ham o`z tanasida boshqa funksiyalarni, o`zini ham chaqirishi mumkin. O`zini chaqiradigan funksiyalarga rekursiv funksiyalar deyiladi.

Oldingi bo`limlarda ta’kidlab o`tgandek, C++ tilidagi har qanday dasturda albatta main() bosh funksiyasi bo`lishi kerak. Ayni shu funksiyadan dastur bajarilishi boshlanadi.

Quyida funksiyalarni e’lon qilish, chaqirish va aniqlashga misollar keltirilgan:

```
// funksiyalar e’loni
int mening_funksiyam(int number, float point);
char belgini_uqish( );
void bitni_o`rnatish(short num);
```

```
void amal_yoq(int, char);
// funksiyalarni chaqirish
result = mening_funksiyam(varb1, 3.14);
symb = belgini_uqish( );
bitni_o`rnatish(3);
amal_yoq(2, smbl);
// funksiyalarni aniqlash
int mening_funksiyam (int number, float point)
{int x;
…
return x;
}
char belgini_uqish( )
{char symbol;
cin >> symbol;
return symbol;

};
void bitni_o`rnatish(short number)
{
global_bit = global_bit | number;
};
void amal_yoq(int x, char ch)
{ };
```

Funksiyaning dasturdagi o`rnini yanada tushunarli bo`lishi uchun son kvadratini hisoblash masalasida funksiyadan foydalanishni ko`raylik.

Funksiya prototipini sarlavha.h sarlavha faylida joylashtiramiz:

```
long son_kvadrati(int);
```

Asosiy dasturga ushbu sarlavha faylini qo`shishi orqali son_kvadrati( ) funksiya e’loni dastur matniga qo`shiladi:

```
#include ―sarlavha.h‖
int main( )
{
int o`zgaruvchi = 5;
cout << son_kvadrati(o`zgaruvchi);
return 0;
}
long son_kvadrati(int x)
{
return x*x;
}
```

Xuddi shu masalani sarlavha faylidan foydalanmagan holda funksiya e'lonini dastur matniga yozish orqali ham hal qilish mumkin:

```
long son_kvadrati(int);
int main( )
{
int o`zgaruvchi = 5;
cout << son_kvadrati(uzgaruvchi);
return 0;
}
long son_kvadrati(int x)
{
return x*x;
}
```

Dastur ishlashida o`zgarish bo`lmaydi va natijasifatida ekranga 25 sonini chop etadi.

3.2.20. Kelishuv bo`yicha argumentlar

C++ tilida funksiya chaqirilganda ayrim argumentlarni tushirib qoldirish mumkin. Bunga funksiya prototipida ushbu parametrlarni kelishuv bo`yicha qiymatini ko`rsatish orqali erishish mumkin. Masalan, quyida prototipi keltirilgan funksiya turli chaqirishga ega bo`lishi mumkin:

```
//funksiya prototipi
void butun_ko`rinishi(int i, bool bayroq = true, char belgi ='\n');
//funksiya chaqirishlari
butun_ko`rinish (1, false,'a');
butun_ko`rinishi(2, false);
butun_ko`rinishi(3);
```

Birinchi chaqiruvda barcha parametrlar mos argumentlar orqali qiymatlarini qabul qiladi, ikkinchi holda i parametiri 2 qiymatini, bayroq parametri false qiymatini va belgi o`zgaruvchisi kelishuv bo`yicha '\n' qiymatini qabul qiladi.

Kelishuv bo`yicha qiymat berishning bitta sharti bor – parametrlar ro`yxatida kelishuv bo`yicha qiymat berilgan parametrlardan keyingi parametrlar ham kelishuv bo`yicha qiymatga ega bo`lishlari shart. Yuqoridagi misolda i parametri

kelishuv bo`yicha qiymat qabul qilingan holda, bayroq yoki belgi parametrlari qiymatsiz bo`lishi mumkin emas. Misol tariqasida berilgan sonni ko`rsatilgan aniqlikda chop etuvchi dasturni ko`raylik. Qo`yilgan masalani yechishda sonni darajaga oshirish funksiyasi – pow() funksiyasi va suzuvchi nuqtali uzun sondan modul olish fabsl() funksiyasidan foydalaniladi. Bu funksiyalar prototipi «math.h» sarlavha faylida joylashgan:

```
#include <iostream.h>
# include <math.h>
void chop_qilish (double numb, double aniqlik = 1, bool bayroq = true);
int main( )
{
double mpi = -3.141592654;
chop_qilish(mpi, 4, false);
chop_qilish(mpi, 2);
chop_qilish(mpi);
return 0;
}
void chop_qilish(double numb, double aniqlik , bool bayroq )
{
if(!bayroq) numb = fabsl(numb);
numb = (int) (numb*pow(10, aniqlik));
numb = numb / pow(10, aniqlik);

cout << numb << _\n‘;
}
```

Dasturda sonni turli aniqlikda (aniqlik parametri qiymati orqali) chop etish uchun har xil variantlarda chop_qilish( ) funksiyasi chaqirilgan. Dastur ishlashi natijasida ekranda quyidagi sonlar chop etiladi:

3.1415

-3.14

-3.1

Parametrning kelishuvi bo`yicha beriladigan qiymati o`zgarmas, global o`zgaruvchi yoki qandaydir funksiya qaytaradigan qiymat bo`lishi mumkin.

3.2.21. Ko`rinish sohasi.

Lokal va global o`zgaruvchilar

*O`zgaruvchilar funksiya tanasi ichida yoki undan boshqa joyda e'lon qilinishi* mumkin. Funksiya ichida e'lon qilingan o`zgaruvchilarga lokal o`zgaruvchilar deb ataladi. Bunday o`zgaruvchilar dastur stekida joylashadi va faqat o`zi e'lon qilingan funksiya tanasida amal qiladi. Boshqaruv asosiy funksiyaga qaytishi bilan lokal o`zgaruvchilar uchun ajratilgan xotira bo`shatiladi (o`chiriladi).

Har bir o`zgaruvchi amal qilish sohasi va yashash vaqti xususiyatlari bilan xarakterlanadi.

O`zgaruvchi amal qilish sohasi deganda o`zgaruvchini ishlatish mumkin bo`lgan dastur sohasi (qismi) tushuniladi. Bu tushuncha bilan o`zgaruvchining ko`rinish sohasi uzviy bog`langan. O`zgaruvchi amal qilish sohasidan chiqqanda ko`rinmay qoladi. Ikkinchi tomondan, o`zgaruvchi amal qilish sohasida bo`lishi, lekin ko`rinmas bo`lishi mumkin. Oxirgi holda ko`rishga ruxsat beruvchi amallar yordamida ko`rinmas o`zgaruvchiga murojaat qilish mumkin bo`ladi.

O`zgaruvchining yashash vaqti deb u mavjud bo`lgan dastur bo`lagining bajarilish intervaliga aytiladi.

Lokal o`zgaruvchilar o`zlari e'lon qilgan funksiya yoki blok chegarasida kurinish sohasigaega. Blok ichidagi ichki bloklarda xuddi shu nomdagi o`zgaruvchi e'lon qilingan bo`lsa, ichki bloklarda lokal o`zgaruvchi amal qilmay qoladi. Lokal o`zgaruvchi yashash vaqti blok yoki funksiyani bajarish vaqti bilan aniqlanadi. Bu hol shuni anglatadiki, turli funksiyalarda bir-biriga umuman bog`liq bo`lmagan bir xil nomdagi lokal o`zgaruvchilarni ishlatish mumkin.

Quyidagi dasturda main() va sum() funksiyalarda bir xil nomdagi o`zgaruvchilarni ishlatish ko`rsatilgan. Dasturda ikkita sonning yig`indisi hisoblanadi va chop etiladi:

```
# include <iostream.h>
// funksiya prototipi
// int sum(int a; int b);
int main( )
{
// lokal o`zgaruvchilar
```

```
int x = r;
int y=4;
cout <<sum(x, y);
return 0;
}
int sum(int a, int b)
{
// lokal o`zgaruvchi
int x=a+b;
return x;
}
```

Global o`zgaruvchilar har qanday funksiyalardan tashqarida e’lon qilinadi va dastur bajarilishi tugaguncha amal qiladi. Bunday o`zgaruvchilarga dasturdan ixtiyoriy funksiyalardan murojat qilish mumkin. Funksiya ichidan global o`zgaruviga murojat qilish uchun funksiyada uning nomi bilan mos tushadigan lokal o`zgaruvchilar bo`lmasligi kerak. Agar global o`zgaruvchi e’lonida unga boshlang`ich qiymat berilmagan bo`lsa, ularning qiymati 0 hisoblanadi. Global o`zgaruvchining amal qilish sohasi uning ko`rinish sohasi bilan ustma-ust tushadi.

Shuni qayd qilish kerakki, dastur tuzuvchilar imkon qadar global o`zgaruvchilarni ishlatmaslikka harakat qiladi, chunki bunday o`zgaruvchilar qiymatini dasturning ixtiyoriy joyidan o`zgartirish imkoniyati mavjud va bu holat dastur ishlashida mazmuniy xatolarga olib kelishi mumkin. Bu fikrimizni tasdiqlovchi dasturni ko`raylik.

```
# include <iostream.h>
// global o`zgaruvchi e’loni
int test = 100;
void chop_qilish(void );
int main( )
{
//lokal o`zgaruvchi e’loni
int test =10;
//global o`zgaruvchi chop qilish funksiyasini chaqirish
chop_qilish( );
sout << «lokal o`zgaruvchi:»<< test<<'\n‘;

return 0;
}
void chop_qilish(void)
```

```
{
cout << «global o`zgaruvchi:»<<test<<’\n‘;
}
```

Boshda test global o`zgaruvchi 100 qiymati bilan e’lon qilinadi. Keyinchalik, main( ) funksiyasida test nomi bilan lokal o`zgaruvchisi 10 qiymati bilan e’lon qilinadi. Dasturda, chop_qilish( ) funksiyasiga murojaat qilinganida, asosiyfunksiya tanasidan vaqtincha chiqiladi va u yerda e’lon qilingan barcha lokal o`zgaruvchilarga murojaat qilish mumkin bo`lmaydi va chop_qilish () funksiyasida global o`zgaruvchi test qiymati chop etiladi. Asosiy dasturga qaytgandan keyin, lokal test o`zgaruvchisi global test o`zgaruvchisini «berkitadi» va «cout <<» qurilmasi bilan lokal test o`zgaruvchini qiymati chop etiladi. Dastur ishlashi natijasida ekranga quyidagi natijalar chop etiladi:

```
global o`zgaruvchi: 100
lokal o`zgaruvchi: 10
```

3.2.22. :: amali. Xotira sinflari

Yuqorida qayd qilingandek, lokal o`zgaruvchi e’loni xuddi shu nomdagi global o`zgaruvchini «berkitadi» va bu joydan global o`zgaruvchiga murojat qilish imkoni bo`lmay qoladi. C++ tilida bunday hollarda ham global o`zgaruvchiga murojat qilish imkoni mavjud va buning uchun ko`rinish sohasiga ruxsat berish amalidan foydalanish kerak bo`ladi. O`zgaruvchi oldigi ikkita nuqta («::») qo`yish zarur bo`ladi. Misol tariqasida quyidagi programani keltiramiz:

```
#inilude <iostream.h >
//global o`zgaruvchi e’loni
int uzg=5;
int main( )
{
//lokal o`zgaruvchi e’loni
int uzg=70;
//lokal o`zgaruvchini chop etish
cout << uzg << ’/n‘;
//global o`zgaruvchini chop etish
cout << ::uzg <<‘/n‘;
```

```
return 0;
}
```

Dastur ishlashi natijasida ekranga oldin 70 va keyin 5 sonlari chop etiladi.

O`zgaruvchilarning ko`rinish sohasi va amal qilish vaqtini aniqlovchi o`zgaruvchilar modifikatori mavjud:

Modifikator	Qo`llanishi	Amal qilish sohasi	Yashasha davri
auto	Lokal	blok	vaqtincha
register	Lokal	blok	vaqtincha
extern	Global	blok	vaqtincha
static	Lokal	blok	doimiy
	Global	fayl	doimiy
volatile	Global	fayl	doimiy

3.2.23. Joylashtiriladigan va rekursiv funksiyalar

Kompilator ishlashi natijasida har bir funksiya mashina kodi ko`rinishida bo`ladi. Agar dasturda funksiyaga murojaat buyrug`i bo`lsa, shu joyda funksiyani chaqirish kodi shakllanadi. Odatda funksiyani chaqirishni amalga oshirish qo`shimcha vaqt va xotira resurslarini talab qiladi. Shu sababli, kompilatorga, agar chaqiriladigan funksiya hajmi unchalik katta bo`lmagan hollarda, funksiyani chaqirish kodi o`rniga funksiya tanasini joylashtirishni ko`rsatma berish mumkin. Bu yo`l funksiya prototipini inline kalit so`zi bilan e’lon qilish orqali amalga oshiriladi. Natijada hajmi oshgan, lekin nisbatan tez bajariladigan dastur kodiga erishish mumkin.

Funksiya kodi joylashtiriladigan dasturga misol.

```
#include <iostream.h>
inline int summa(int,int);
int main()
{
int a=2,b=6,c=3;
char yangi_qator=‘\n‘;
cout<<summa(a,b)<<yangi_qator;
```

```
cout<<summa(a,c)<<yangi_qator;
cout<<summa(b,c)<<yangi_qator;
return 0;
}
int summa(int x,int y)
{
return x+y;
}
```

Keltirilgan dastur kodini hosil qilishda summa() funksiyasi chaqirilgan joylarga uning tanasi joylashtiriladi.

Rekursiya deb funksiya ichidan shu funksiyaning o`zini chaqirishiga aytiladi. Rekursiya ikki xil bo`ladi:

1) oddiy – agar funksiya o`z tanasida o`zini chaqirsa;

2) vositali – agar funksiya boshqa bir funksiyani chaqirsa, u esa o`z navbatida birinchi funksiyani chaqirsa.

Odatda rekursiya matematikada keng qo`llaniladi. Chunki aksariyat matematik yormulalar rekursiv aniqlanadi. Misol tariqasida faktorialni hisoblash yormulasini

$$n! = \begin{cases} 1, & agar \quad n = 0 \\ n \cdot (n-1)! & agar \quad n > 0 \end{cases}$$

va sonning butun darajasini hisoblashni ko`rishimiz mumkin:

$$x^n = \begin{cases} 1, & agar \quad n = 0 \\ x \cdot x^{n-1} & agar \quad n > 0 \end{cases}$$

Ko`rinib turibdiki, navbatdagi qiymatni hisoblash uchun funksiyaning oldingi qiymati ma’lum bo`lishi kerak. C++ tilida rekursiya matematikadagi rekursiyaga o`xshash. Buni yuqoridagi misollar uchun tuzilgan funksiyalarda ko`rish mumkin. Faktorial uchun:

```
long f(int n)
{
if(!n) return 1;
else return n*f(n-1);
}
```

Berilgan haqiqiy x soning n – darajasini hisoblash funksiyasi:

```
double butun_daraja(double x, int n)
{
```

```
if(!n) return 1;
else return x*butun_daraja(x,n-1);
}
```

Agar faktorial funksiyasiga n>0 qiymat berilsa, quyidagi holat ro`y beradi: shart operatorining else shohidagi qiymat eslab qolinadi (n qiymati). Noma'lumlarni hisoblash uchun shu funksiyaning o`zi «oldingi» qiymat (n-1 qiymati) bilan chaqiriladi. O`z navbatida, bu qiymat ham eslab qolinadi (xotiraning boshqa joyida) va yana funksiya chaqiriladi va hokazo. Funksiya n=0 qiymat bilan chaqirilganda if operatorining sharti (!n) rost bo`ladi va return 1 amali bajarilib, ayni shu chaqiruv bo`yicha 1 qiymati qaytariladi. Shundan keyin «teskari» jarayon boshlanadi - xotiradagi saqlangan qiymatlar o`zaro ko`paytiriladi. Oxirgi qiymat – aniqlangandan keyin (1), u undan oldingi saqlangan qiymatga – 1 qiymatiga ko`paytirilashidan f(1) qiymati hisoblanadi, bu qiymat 2 qiymatiga ko`payishi bilan f(2) topiladi va hakoza. Jarayon fact(n) qiymatini hisoblashgacha «ko`tarilib» boradi.

Rekursiv funksiyalarni to`g`ri amal qilishi uchun rekursiv chaqirishlarning to`xtash sharti bo`ljshi kerak. Aks holda rekursiya to`xtamasligi va o`z navbatida funksiya ishi tugamasligi mumkin. Faktorial hisoblashida rekursiv tushishlarning to`xtash sharti funksiya parametri n qiymati 0 bo`lishidir (shart operatorining rost shoxi).

Har bir rekursiv murojaat qo`shimcha xotira talab qiladi – dasturning lokal obyektlari (o`zgaruvchilari) uchun har bir murojaatda stek deb nomlanuvchi xotira segmentidan yangidan joy ajratiladi. Masalan, rekursiv funksiyaga 100 marta murojaat bo`lsa, jami 100 lokal obyektlarning majmuasi uchun joyajratiladi. Shu sababli juda ko`p rekursiya bo`lganda, stek o`lchami cheklanganligi sababli (real rejimda 64 Kb o`lchamgacha) u to`lib ketishi mumkin va bu holda dastur o`z ishini «stek to`lib ketdi» xabari bilan to`xtatadi.

Rekursiya chiroyli, ixcham ko`ringani bilan xotirani tejash va hisoblash vaqtini qisqartirish nuqtayi-nazaridan imkon qadar uni iterativ hisoblash bilan

almashtirilgani ma’qul hisoblanadi. Masalan, x-haqiqiy sonini n butun darajasini hisoblashni quyidagi yechim varianti nisbatan kam resurs talab qiladi:

```
double butun_daraja(double x, int n)
{
double p=1;
for (int i=1; i<=n; i++) p*=x;
return p;
}
```

Lekin shunday masalalar borki, ularni yechishda rekursiya juda samarali, hattoki yagona usuldir. Xususan, grammatik tahlil masalalarida rekursiya qulay hisoblandi.

3.2.24. Qayta yuklanuvchi funksiyalar

Ayrim algoritmlar berilganlarning har xil turdagi qiymatlari uchun qo`llanishi mumkin. Masalan, ikkita sonning maksimumini topish algoritmida bu sonlar butun yoki haqiqiy turda bo`lishi mumkin. Bunday hollarda bu algoritmlar amalga oshirilgan funksiyalar nomlari bir xil bo`lgani ma’qul. Bir nechta funksiyani bir xil nomlash, lekin har xil turdagi parametrlar bilan ishlatish funksiyani qayta yuklash deyiladi.

Kompilator parametrni turlariga va soniga qarab mos funksiyani chaqiradi. Bunday amalni «hal qilish amali» deyiladi va uning maqsadi parametrlarga ko`ra aynan (nisbatan) to`g`ri keladigan funksiyani chaqirishdir. Agar bunday funksiya topilmasa kompilator xatolik haqida xabar beradi. Funksiyani aniqlashda funksiya qaytaruvchi qiymat turining ahamiyati yo`q. Misol:

```
#include <iostream.h>

int max(int, int);
char max(char, char);
float max(float, float)
int max(int, int, int);
void main()
{
int a,int b, char c, char d,int k,float x,y;
cin >> a>>b>>k>>c>>d>>x>>y;
```

```
count << max(a,b)<<max(c,d)<<max(a,b,k)<<max(x,y);
}
int max(int i, int j) {return (i>j)? i: j;}
char max(char s1,char s2){return (s1>s2) ? s1: s2;}
float max(float x,float y){return (x>y)? x: y;}
int max(int i,int j,int k){return (i>j)? (i>k? i: k;): ((j>k)? j : k;);}
```

Agar funksiya chaqirilishida argument turi uning prototipidagi mos o`rindagi parametr turiga mos kelmasa, kompilator ularni parametr turiga keltirilishga harakat qiladi – bool va char turi int turiga, float turi *double* turiga va int turi double turiga o`tkaziladi.

Qayta yuklanuvchi funksiyalardan foydalanishda quyidagi qoidalarga rioya qilinishi kerak:

-qayta yuklanuvchi funksiyalar bitta ko`rinish sohasida bo`lishi kerak;

-qayta yuklanuvchi funksiyalarda kelishuv bo`yicha parametrlar ishlatilsa, bunday parametrlar barcha qayta yuklanuvchi funksiyalarda ham ishlatilishi va ular bir xil qiymatga ega bo`lish kerak;

- agar funksiyalar parametrlari turi faqat«const» va ‘&‘ belgilari bilan farq qiladigan bo`lsa, bu funksiyalar qayta yuklanmaydi.

3.2.25. Funksiya shablonlari (qoliblari)

Bazi bir algoritmlar berilganlar turiga bog`liq emas, masalan deylik tartiblash algotmini har xil turdagi berilganlar ustida bajarish mumkin.

Algoritmni har xil turdagi berilganlar bilan ishlatish uchun C++ tilida qolibli funksiyalar tushinchasi kiritilgan. Bu holda har xil turlar uchun alohida qayta yuklanuvchi funksiyalar tuzish shart emas, aksincha bitta funksiya tuzib va unda parametrlangan turni ishlatsa bo`ladi. Qolibli funksiyani shakli quyidagicha bo`ladi:

```
template <class type>
funksiya sarlavhasi
{funksiya tanasi}
```

Qolib sarlavhasi umumiy ko`rinishi:

template <class type1,class type2,…>
bu yerda type1, type2 turlar parametiri.

Misollar:

```
template <class type>
type abs(type x) {return x>0?x:-x;}
template <class t>
void swap (t*x,t*y) {t z=*x;*x=*y;*y=z;}
#include <iostream.h>
// qolib funksiya sarlavhasi
template <class d>
d sum(int,d*);
void main()
{
int a[]={1,0,6,2,4,10};
int n=6;
cout << sum(n,a);
float x[]={1.0,10.0,-2.3,0.0,2.1,-6.7,8.3}; n=7;
cout<<sum(n,x);
}
// qolib funksiyani aniqlanishi
template <class t>
t sum(int n,t *b );
{
t s=0;
for (int i=0;i<n;i++) s+=b[i];
return s;
}
```

Qolibli funksiyalar qoydalari:

1) qolibdagi tur parametrlar ismlari har xil bo`lishi kerak;

2) tur parametrlari funksiyani parametrlarini berilishida albatta ishlatilishi kerak.

Tayanch so’z va iboralar

Funksiya, *Lokal va global o`zgaruvchilar,* Amal qilish sohasi, Modifikator, Rekursiya, vositali, Standart andozalar kutubxonasi.

Mavzuga oid savol va topshiriqlar

1.