3.2 DASTURLASH TILI ELEMENTLARI. MUTAXASSISLIK MASALALARINI DASTURLASHDA CHIZIQLI ALGORITMLARDAN FOYDALANISH

Ma’lumki, dastur mashina kodlarining shunday ketma-ketligi bo`lib, aniq bir hisoblash vositasini amal qilishini boshqaradi. Dastur vositasini yaratish jarayonini osonlashtirishi uchun bir qatory dasturlash tillari yaratilgan. Barcha dasturlash tillarini ikkita toifaga ajratish mumkin:

- quyi darajadagi dasturlash tillari;
- yuqori darajadagi dasturlash tillari.

Quyi darajadagi dasturlash tiliga Assembler tili kiradi. Bu til nisbatan qisqa va tezkor bajariluvchi kodlarni yaratish imkoniyatini beradi. Bunga qarama-qarshi ravishda yuqori bosqich tillari yaratilganki, ularda tabiiy tilning (ingliz tilining) cheklangan ko`rinishidan foydalangan holda dastur tuziladi. Yuqori bosqich tillaridagi operatorlar, berilganlarning turlari, o`zgaruvchilar va dastur yozishning turli usullari tilning ifodalash imkoniyatini oshiradi va dasturni «sodda» bo`lishini ta’minlaydi. Yuqori bosqich tillariga Fortran, Pl/1, Prolog, Lisp, Basic, Pascal, C va boshqa tillarni misol keltirish mumkin. Kompyuter arxitekturasini takomillashuvi, kompyuter tarmog`ining rivojlanishi mos ravishda yuqori bosqich tillarini yangi variantlarni yuzaga kelishiga, yangi tillarni paydo bo`lishiga, ayrim tillarni yo`qolib ketishiga olib keldi. Hozirda keng tarqalgan tillarga misol sifatida Object pascal, C, C++, Php, Java, Asp tillarni keltirish mumkin. Xususan, C tilining takommillashgan varianti sifatida C++ tilini olishimiz mumkin. 1972 yilda Denis Ritch va Brayan Kernegi tomonidan C tili yaratildi. 1980 yilda B’yarn Straustrop C tilining avlodi C++ tilini yaratdi, natijada strukturali va obyektga yo`naltirilgan dasturlash texnologiyasiga tayangan holda dastur yaratish imkoniyati paydo bo`ldi.

1. C++ dasturlash tili va Dev-C++ dasturlash muhiti.

C++ dasturlash tilining nomi C dasturlash tilidan kelib chiqqan bo`lib, ++ belgisi inkrement amali, ya`ni o'zgaruvchining qiymatini bittaga oshish amalidan olingan. C ++ dasturlash tili turli xil amaliy dasturlarni yaratish, operatsion

tizimlarni, qurilma drayverlarini, shuningdek video o'yinlarni va boshqalarni yaratish uchun keng qo'llaniladi.

C ++ dasturlash tili 1980-yillarning boshlarida Bell Laboratories firmasi xodimi Byorn Stroustrup tomonidan yaratilgan

1.2.1. C++ dasturlash tili va va uning kompilyasiyasi Dev-C++ dasturlash muhiti.

C++ dasturlash tilining nomi C dasturlash tilidan kelib chiqqan bo`lib,  ++ belgisi inkrement amali, ya`ni o'zgaruvchining qiymatini bittaga oshish amalidan olingan. C ++ dasturlash tili turli xil amaliy dasturlarni yaratish, operatsion tizimlarni, qurilma drayverlarini, shuningdek video o'yinlarni va boshqalarni yaratish uchun keng qo'llaniladi. C ++ dasturlash tili 1980-yillarning boshlarida Bell Laboratories firmasi xodimi Byorn Stroustrup tomonidan yaratilgan. C++ tilida dastur yaratish bir nechta bosqichlardan iborat bo`ladi. Dastlab, matn tahririda (odatda dasturlash muhitining tahririda) dastur matni teriladi, bu faylning kengaytmasi «.cpp» bo`ladi, keyingi bosqichda dastur matn yozilgan fayl kompilyatorga uzatiladi, agarda dasturda xatoliklar bo`lmasa, kompilator «.obj» kengaytmali obyekt modul faylini hosil qiladi. Oxirgi qadamda komponovka (yig`uvchi) yordamida «.exe» kengaytmali bajariluvchi fayl - dastur hosil bo`ladi. Bosqichlarda yuzaga keluvchi fayllarning nomlari boshlang`ich matn fayl nomi bilan bir xil bo`ladi. Kompilatsiya jarayonining o`zi ham ikkita bosqichdan tashkil topadi. Boshida preprosessor ishlaydi, u matndagi kompilatsiya direktivalarini bajaradi, xususan #include direktivasi bo`yicha ko`rsatilgan kutubxonalardan C++ tilida yozilgan modullarni dastur tarkibiga kiritadi. Shundan so`ng kengaytirilgan dastur matni kompilatorga uzatiladi. Kompilator o`zi ham dastur bo`lib, uning uchun kiruvchi ma’lumot bo`lib C++ tilida yozilgan dastur matni hisoblanadi. Kompilator dastur matnini leksema (atomar) elementlarga ajratadi va uni leksik, keyinchalik sintaksis tahlil qiladi. Leksik tahlil jarayonida u matnni leksamalarga ajratish uchun «probel ajratuvchisini» ishlatadi. Probel ajratuvchisiga – probel belgisi (« »), « \t » –

tabulyasiya belgisi, « \n » – keyingi qatorga o`tish belgisi, boshqa ajratuvchilar va izohlar (kommentariylar) kiradi.

Dastur matni tushunarli bo`lishi uchun izohlar ishlatiladi. Izohlar kompilator tomonidan «o`tkazib» yuboriladi va ular dastur amal qilishiga hech qanday ta’sir qilmaydi.

C++ tilida izohlar ikki ko`rinishda yozilishi mumkin. Birinchisida «/*» dan boshlanib va «*/»belgilari bilan tugagan barcha belgilar ketma-ketligi izoh hisoblanadi, ikkinchisi «satriy izoh» deb nomlanadi va u «//»belgilardan boshlangan va satr oxirigacha yozilgan belgilar ketma-ketligi bo`ladi. Izohning birinchi ko`rinishida yozilgan izohlar bir necha satr bo`lishi va ulardan keyin C++ operatorlari davom etish mumkin.

Misol.

```
int main()
{
// bu qator izoh hisoblandi
int a=0; // int d;
int c;
/* int b=15 */
/* – izoh boshlanishi
a=c;
izoh tugashi */
return 0;
}
```

Dasturda d, b o`zgaruvchilar e’lonlari inobatga olinmaydi va a=c amali bajarilmaydi.

Quyida C++ tilidagi sodda dastur matni keltirilgan.

```
# include <iostream.h>              // sarlavha faylni qo`shish
int main ()              // bosh funksiya tavsiyi
{          // blok boshlanishi
cout << — salom olam!\n||;          // satrni chop etish
return 0;          // funksiya qaytaradigan qiymat
}                                    // blok tugashi
```

Dastur bajarilishi natijasida ekranga «salom olam!» satri chop etiladi.

Dasturning 1–satrida #include preprosessor direktivasi bo`lib, dastur kodiga oqimli o`qish/yozishfunksiyalari va uning o`zgaruvchilari e'loni joylashgan iostream.h sarlavha faylini qo`shadi. Keyingi qatorlarda dasturning yagona, asosiy funksiyasi – main() funksiyasi tavsifi keltirilgan. Shuni qayd etish kerakki, C++ dastursida albatta main() funksiyasi bo`lishi shart va dastur shu funksiyani bajarish bilan o`z ishini boshlaydi.

Dastur tanasida konsol rejimida belgilar ketma–ketligini chiqarish amali qo`llanilgan. Ma'lumotlarni standart oqimga chiqarish (ekranga) uchun quyidagi format ishlatilgan:

cout <<ifoda;

Bu yerda ifoda sifatida o`zgaruvchi yoki sintaksisi to`g`ri yozilgan va qandaydir qiymat qabul qiluvchi til ifodasi kelishi mumkin.

Masalan:

```
int uzg=324;
cout << uzg; // butun son chop etiladi
```

Berilganlarni standart oqimdan (odatda klaviaturadan) o`qish quyidagi formatda amalga oshiriladi:

cin >> o`zgaruvchi;

Bu yerda o`zgaruvchi qiymat qabul qiluvchi hisoblanadi.

Misol.

```
int yosh;
cout <<«yoshingizni kiriting»;
cin >>yosh;
```

Butun turdagi yosh o`zgaruvchisi kiritilgan qiymatni o`zlashtiradi. Kiritilgan qiymatni o`zgaruvchi turiga mos kelishini tekshirish mas'uliyati dastur tuzuvchi zimmasiga yuklanadi.

Bir paytning o`zida probel «□»vositasida bir nechta va har xil turdagi qiymatlarni kiritish mumkin. Qiymat kiritish «enter» tugmasini bosish bilan tugaydi. Agar kiritilgan qiymatlar soni o`zgaruvchilar sonidan ko`p bo`lsa, «ortiqcha» qiymatlar bufer xotirada saqlanib qoladi.

```
#include <iostream.h>
int main ()
{
int x, y;
float z;
cin >>x>>y>>z;
cout <<«O`qilgan qiymatlar\n»;
cout << x<<y<<z
return 0;
}
```

O`zgaruvchilarga qiymatlar kiritish uchun klaviatura orqali

10 20 3.14 <enter>

harakati amalga oshiriladi.

1.2.2. C++ tili alfaviti va leksemalari. Identifikatorlar va kalit so`zlar

C++ tili alfaviti va leksemlariga quyidagilar kiradi:

Katta va kichik lotin alfaviti harflari	A,B,..,Z,a,b,…,z
Arab raqamlari	0,1,2,3,4,5,6,7,8,9
Maxsus belgilar	“ , {} | [] () + - / % \ ; ‘ . : ? < = > _ ! & * # ~ ^
Ko`rinmas belgilar	probel, tabulyatsiya, yangi qatorga o`tish belgilari

Dasturlash tilining muhim tayanch tushunchalaridan biri – **identifikator** tushunchasidir. Identifikator deganda katta va kichik lotin harflari, raqamlar va tag chiziq « _ » belgilaridan tashkil topgan va raqamdan boshlanmaydigan belgilar ketma–ketligi tushuniladi.

Identifikatorlarda harflarning registrlari (katta yoki kichikligi) hisobga olinadi. masalan, RUN, Run, run bu har xil identifikatorlardir. Identifikator uzunligiga chegara qo`yilmagan, lekin ular faqat boshidagi 32 belgisi bilan farqlanadi.

Identifikatorlar kalit so`zlarni, o`zgaruvchilarni, funksiyalar, nishonlarni va boshqa obyektlarni nomlashda ishlatiladi.

C++ tilining kalit so`zlariga quyidagilar kiradi:

asm	class	double	for	Long	public	static	typename
auto	const	else	friend	mutable	register	struct	union

break	continue	enum	goto	new	return	swith	unsigned
case	default	explicit	if	operator	short	this	virtual
catch	delete	extern	inline	private	signet	throw	void
char	do	float	int	protected	sizeof	typedef	while

Yuqorida keltirilgan identifikatorlarni boshqa maqsadda ishlatish mumkin emas.

Prosessor registrlarini belgilash uchun quyidagi so`zlar ishlatiladi:

_AH	_BH	_CH	_DH	_CS	_GS	_ESI	_ES
_AL	_BL	_CL	_DL	_ESP	_DI	_BP	_SS
_AX	_BX	_CX	_DX	_EBP	_EDI	_SP	_FLAGS
_EAX	_EBX	_ECX	_EDX	_FS	_SI	_DS	

Bulardan tashqari «__» (ikkita pastki chiziq) belgilaridan boshlangan identifikatorlar kutubxona fayllari uchun mo`ljallangan. Shu sababli «_» va «__» belgilardan identifikatorlarning birinchi belgisi sifatida foydalanish tavsiya etilmaydi. Identifikator belgilari orasida probel ishlatish mumkin emas, zarur bo`lganda uning o`rniga«_» ishlatish mumkin: cilindr_radiusi, ailana_diametiri.

3.2.3.C++ tilida o`zgarmaslar *(Constants)*

O`zgarmas bu o`zgartirish mumkin bo`lmagan qiymatdir. C++ tilida besh turdagi o`zgarmaslardan foydalanish mumkin: butun, haqiqiy, belgi, simvol va sanovchi o`zgarmaslar.

***1. Butun o`zgarmaslar.***Butun sonlar o`nlik, sakkizlik yoki o`n oltilik sanoq sistemalarida berilishi mumkin. O`nlik sanoq sistemasida butun sonlar 0-9 raqamlari ketma-ketligidan iborat bo`lib, birinchi raqami 0 bo`lishi kerak emas. Sakkizlik sanoq sistemasida butun sonlar 0 bilan boshlanuvchi 0-7 raqamlaridan iborat ketma -ketlikdir. O`n oltilik sanoq sistemasida butun son 0x yoki 0X bilan boshlanuvchi 0-9 raqamlari va a-f yoki A-F harflaridan iborat ketma- ketlikdir.

Masalan 15 va 22 o`nlik sonlari sakkizlikda 017 va 026, o`n oltilikda 0xF va 0x16 shaklda tasvirlanadi.

*Uzun butun o`zgarmaslar.* Oxiriga l yoki L (long) harflari quyilgan o`nlik, sakkizlik yoki o`n oltilik butun son.

*Ishorasiz (unsigned) butun o`zgarmaslar.* Oxiriga u yoki U harflari qo`yilgan o`nlik, sakkizlik yoki o`n oltilik oddiy yoki uzun butun son.

***2. Haqiqiy o`zgarmaslar.*** Olti qismdan iborat bo`lishi mumkin: butun qism, o`nli nuqta belgisi, kasr qism, E yoki e eksponenta belgisi, o`n daraja ko`rsatkichi, qo`shimcha belgilar(F yoki f va L yoki l).

*Uzun haqiqiy o`zgarmaslar.*Oxiriga F yoki f va L yoki l suffikslari qo`yilgan haqiqiy son.

Eksponensial shakldagi o`zgarmaslarga misollar: 1e2; 5e+3; .27e5; 31.4e-l; 3.14F; 1.12e-12.

***3. Belgi o`zgarmaslar.*** Bittalik qavslarga olingan bitta belgi bo`lib ular shar kalit so`zi bilan aniqlanadi. Misol uchun ‘x’; ’*’; ’\’; ’0’; ’n’ ; ’@’ ; ’R’. Belgi o`zgarmaslar kompyuter xotirasidan bir bayt joy egallaydi.

Escape belgi ifodasi	Ichki kodi (16 sanoq sistemasida)	Nomi	Belgiga mos amallar
\a	0x07	bel (audible bell)	Tovush signali
\b	0x08	Bs (bascspase)	Kursorni bir qadam orqaga qaytarish
\f	0x0C	Ff (form feed)	Sahifani o`tkazish
\n	0x0A	lf (line feed)	Qatorni o`tkazish
\r	0x0D	Cr (carriage return)	Kursorni ayni qator boshiga qaytarish
\t	0x09	Ht (horizontal tab)	Kursorni tabulatsiyaning keyingi joyiga qaytarish
\v	0x0B	Vt (vertical tab)	Vertikal tabulatsiya(pastga)
\\	0x5C	\ (bacslash)	Teskari chiziq
\’	0x27	‘ (single out)	Apostrof (oddiy qavs)
\”	0x22	“ (double quote)	Ikkilik qavs
\?	0x3F	? (question mark)	Savol belgisi

Ayrim belgi o`zgarmaslar «\» belgisi bilan boshlanadi. Bu o`zgarmaslar birinchidan, grafik ko`rinishga ega bo`lmagan o`zgarmaslarni belgilaydi, ikkinchidan maxsus vazifalar yuklangan belgilarni - apostrof (‘), savol (?), teskari yon chiziq (\), ikkita qo`shtirnoq (“) belgilarni chop etishda foydalaniladi. Undan

tashqari, bu belgi orqali belgini ko`rinishini emas, balki oshkor ravishda uning ASCII kodini sakkizlik yoki o`noltilik shaklda yozish mumkin. Bunday belgilar bilan boshlangan belgilar escape ketma-ketliklar deb ataladi.

***4. Satrli o`zgarmas.*** Ikkita qo`shtirnoq (“ “) ichiga olingan belgilar ketma-ketligi satrli o`zgarmas deb ataladi. Satr - bu belgilar massivi.

Misol uchun «Men satrli o`zgarmasman».

Satr ichida escape ketma-ketligi ham ishlatilishi mumkin. Bu holda ketma-ketlik apostrofsiz yoziladi. Probel ajratgichi bilan ajratilgan ketma-ket satrlar bir-biriga ulanadi.

Satrlar orasiga escape simvollar ham kirishi mumkin. Bu simvollar oldiga \ belgisi quyiladi.

Misol uchun : “\n Bu satr \n uch qatorga \n joylashadi”.

Satr simvollari xotirada ketma-ket joylashtiriladi va har bir satrli o`zgarmas oxiriga avtomatik ravishda kompilator tomonidan ‘\0’ simvoli qo`shiladi. Shunday satrning xotiradagi hajmi simvollar sonicha baytga tengdir.

Ketma-ket kelgan va bo`shliq, tabulatsiya yoki satr oxiri belgisi bilan ajratilgan satrlar kompilatsiya davrida bitta satrga aylantiriladi.

Misol.

“Amaliy”

“informatika ”

satrlari bitta

“Amaliy informatika”

satri deb qaraladi.

Bu qoidaga bir necha qatorga yozilgan satrlar ham bo`ysinadi.

Misol.

“Toshkent”
“irrigatsiya”
”va”
“qishloq”
“xo’jaligini”
“ mexanisatsiyalash”
“ muhandislari”

”instituti”

satrlari bitta

“Toshkent irrigatsiya va qishloq xo’jaligini mexanisatsiyalash muhandislari instituti”

satriga mos.

Agar satrda ‘\’ belgisi uchrasa va bu belgidan so`ng *to* ‘\n’ satr oxiri belgisigacha bo`shliq belgisi kelsa bu bo`shliq belgilari ‘\’ va ‘\n’ belgisi bilan birga satrdan o`chiriladi. Satrning o`zi keyingi satrda kelgan satr bilan qo`shiladi.

Misol.

“Toshkent \
“irrigatsiya\
”va\
“qishloq \
“xo’jaligini\
“ mexanisatsiyalash\
“ muhandislari \
”instituti”

satrlari bitta

“Toshkent irrigatsiya va qishloq xo’jaligini mexanisatsiyalash muhandislari instituti”

satrga mos.

***5. Sanovchi o`zgarmas.*** Sanovchi o`zgarmaslar enum xizmatchi so`zi yordamida kiritilib, int tipidagi sonlarga qulay so`zlarni mos qo`yish uchun ishlatiladi.

Misol. enum{one=1,two=2,three=3}.

Agar son qiymatlari ko`rsatilmagan bo`lsa eng chapki so`zga 0 qiymati berilib qolganlariga tartib bo`yicha o`suvchi sonlar mos quyiladi.

Misol. Enum{zero,one,two}.

Bu misolda avtomatik ravishda o`zgarmaslar quyidagi qiymatlarni qabul qiladi: zero=0, one=1, two=2.

O`zgarmaslar aralash ko`rinishda kiritilishi ham mumkin.

Misol. Enum(zero,one,for=4,five,seeks}.

Bu misolda o`zgarmaslar avtomatik ravishda quyidagi qiymatlarni qabul qiladi: Zero=0, one=1, for=4; five=5, seeks=6.

Misol. Enum BOOLEAN {NO, YES};

O`zgarmaslar qiymatlari: NO=0, YES=1;

***Nomlangan o`zgarmaslar.*** C++ tilida o`zgaruvchilardan tashqari nomlangan o`zgarmaslar kiritilishi mumkin. Bu o`zgarmaslar qiymatlarini dasturda o`zgartirish mumkin emas. O`zgarmaslar nomlari dasturchi tomonidan kiritilgan va xizmatchi so`zlardan farqli bo`lgan identifikatorlar bo`lishi mumkin. Odatda nom sifatida katta lotin harflari va ostiga chizish belgilari kombinatsiyasidan iborat identifikatorlar ishlatiladi. Nomlangan o`zgarmaslar quyidagi shaklda kiritiladi:

Const tip o`zgarmas_nomi=o`zgarmas_qiymati.

Misol.

```
Const double EULER=2.718282;
Const long M=99999999;
Const R=765;
```

Bu misolda o`zgarmas tipi ko`rsatilmagan, bu o`zgarmas int tipiga tegishli deb hisoblanadi.

**Null *ko`rsatkich.*** NULL - ko`rsatkich yagona arifmetik bo`lmagan o`zgarmasdir. Konkret realizatsiyalarda null ko`rsatkich 0 yoki 0L yoki nomlangan o`zgarmas NULL orqali tasvirlanishi mumkin. Snuni aytish lozimki bu o`zgarmas qiymati 0 bo`lishi yoki ‘0’ simvoli kodiga mos kelishi shart emas.

3.2.4. O`zgaruvchilar va berilganlar turlari

Dastur bajarilishi paytida qandaydir berilganlarni saqlab turish uchun o`zgaruvchilar va o`zgarmaslarlardan foydalaniladi. O`zgaruvchi– dastur obyekti bo`lib, xotiradagi bir nechta yacheykalarni egallaydi va berilganlarni saqlash uchun xizmat qiladi. O`zgaruvchi nomga, o`lchamga va boshqa atributlarga – o`zgarish sohasi, amal qilish vaqti va boshqa xususiyatlarga ega bo`ladi. O`zgaruvchilarni ishlatish uchun ular e’lon qilinishi kerak. E’lon natijasida o`zgaruvchi xotiradan qandaydir soha zaxiralanadi, soha o`lchami esa o`zgaruvchining aniq turiga bog`liq

bo`ladi. Shuni qayd etish zarurki, bitta tur uchun turli apparat platyormalarda turlicha joyajratilishi mumkin.

O`zgaruvchie'loni uning turini aniqlovchi kalit so`zi bilan boshlanadi va « = » belgisi orqali boshlang`ich qiymat beriladi (shart emas). Bitta kalit so`z bilan bir nechta o`zgaruvchilarni e'lon qilish mumkin. Buning uchun o`zgaruvchilar bir – biridan « , » belgisi bilan ajratiladi. E'lonlar «_; » belgisi bilan tugaydi. O`zgaruvchi nomi 255 belgidan oshmasligi kerak.

Quyidagi jadvalda C++ tilining tayanch turlari, ularning baytlardagi lchamlari va qiymatlarining chegaralari keltirilgan:

Tipi	Hajmi, Bayt	Qiymatlar chegarasi
bool	1	true yoki false
unsigned short int	2	0..65535
short int	2	–32768..32767
unsigned long int	4	0..42949667295
long int	4	–2147483648..2147483647
int (16 razryadli)	2	–32768..32767
int (32 razryadli)	4	–2147483648..2147483647
unsigned int (16 razryadli)	2	0..65535
unsigned int (32 razryadli)	4	0..42949667295
char	1	0..255
float	4	1.2e–38..3.4e38
double	8	2.2e–308..1.8e308
long double (32 razryadli)	10	3.4e-4932..-3.4e4932
void	2 yoki 4	

Mantiqiy turlar. Bu turdagi o`zgaruvchi bool kalit so`zi bilan e'lon qilinadi. Bu turdagi o`zgaruvchi 1 bayt joy egallaydi va 0 (false, yolg`on) yoki 0 dan farqli qiymat (true, rost) qiymat qabul qiladi. Mantiqiy turdagi o`zgaruvchilar qiymatlar o`rtasidagi munosabatlarni ifodalaydigan mulohazalarni rost (true) yoki yolg`on (false) ekanligini tavsiylashda qo`llaniladi. Mantiqiy tur qiymatlari ustida matniqiy ko`paytirish, qo`shish va inkor amallarini qo`llash orqali murakkab mantiqiy ifodalarni qurish mumkin.

Butun son turi. Butun son qiymatlarni qabul qiladigan o`zgaruvchilar int (butun), short (qisqa) va long (uzun) kalit so`zlar bilan aniqlanadi. O`zgaruvchi

qiymatlari ishorali bo`lishi yoki unsigned kalit so`zi bilan ishorasiz son bo`lishi mumkin.

Belgi turi. Belgi turidagi o`zgaruvchilar char kalit so`zi bilan beriladi va ular o`zida ASCII kodi haqidagi ma'lumotlarni saqlaydi. Belgi turidagi qiymatlar nisbatan murakkab bo`lgan tuzilmalar – satrlar, belgilar massivlari va hakozalarni hosil qilishda ishlatiladi.

Haqiqiy son turi. Haqiqiy sonlar ***float*** kalit so`zi bilan e'lon qilinadi. Bu turdagi o`zgaruvchi uchun xotirada 4 bayt joyajratiladi va (ishora, tartib, mantissa) qolipida sonni saqlaydi. Agar kasrli son juda katta (kichik) qiymatlarni qabul qiladigan bo`lsa, u xotiradi 8 yoki 10 baytda ikkilangan aniqlik ko`rinishida saqlanadi va mos ravishda double va long double kalit so`zlari bilan e'lon qilinadi. Oxirgi holat 32-razryadli apparat platyormalari uchun o`rinli.

Void turi. Void turidagi dastur obyekti hech qanday qiymatga ega bo`lmaydi va bu turdan qurilmani til sintaksisiga mos kelishini ta'minlash uchun ishlatiladi. Masalan, C++ tili sintaksisi funksiya qiymat qaytarishini talab qiladi. Agar funksiya qiymat qaytarmaydigan bo`lsa u void kalit so`zi bilan e'lon qilinadi.

Misollar.

```
int a=0, a=1;
float abc = 17.5;
double ildiz;
bool ok=true;
char letter = _z‘;

Void Mening_funksiyam (); /* funksiya qaytaradigan qiymat inobatga olinmaydi */
```

3.2.5. Turni boshqa turga keltirish

C++ tilida bir turni boshqa turga keltirishning oshkor va oshkormas yo`llari mavjud.

Umuman olganda, turni boshqa turga oshkormas keltirish ifodada har xil turdagi o`zgaruvchilar qatnashgan hollarda amal qiladi (aralash turlar arifmetikasi). Agar turga keltirish tayanch turlar bilan bog`liq holda bajarilsa, xatoliklar yuzaga

kelishi mumkin, masalan, natijaning xotirada egallagan joyi, uni o`zlashtiradigan o`zgaruvchi uchun ajratilgan joydan katta bo`lsa. Bunda qiymatli razryadlarni yo`qotish holi yuz beradi.

Oshkor ravishda turga keltirishda o`zgaruvchi oldiga qavs ichida boshqa tur nomi yoziladi:

```
#include <iostream.h>
int main()
{
int integer_1= 54;
int integer_2;
float floating = 15.854;
integer_1 = (int) floating; // oshkor keltirish;
integer_2 = floating; // oshkormas keltirish;
cout <<  «yangi integer(oshkar): «<< integer_1<<»\n»;
cout <<«yangi integer(oshkarmas):« —<< integer_2<<»\n»;
return 0;
}
```

Dastur bajarilish natijasi quyidagi ko`rinishida bo`ladi:

Yangi integer(oshkar): 15

Yangi integer(oshkarmas): 15

3.2.6.Arifmetik amallar.

Ko`p dasturlar ishlashi davomida arifmetik amallarni bajaradi. C++ dagi matematik amallar quyidagi jadvalda berilgan. Ular ikkita operand bilan ishlatiladi.

C++ dagi amal	Arifmetik operator	Algebraik ifoda	C++ dagi ifodasi
Qo`shish	+	h+19	h+19
Ayirish	-	f-u	f-u
Ko`paytirish	*	sl	s*l
Bo`lish	/	v/d	v/d
Modul olish	%	k mod 4	k%4

Bularning ba’zi birlarinig xususiyatlarini ko`rib chiqaylik. Butun sonli bo`lishda, ya’ni bo`luvchi ham, bo`linuvchi ham butun son bo`lganda, javob butun son bo`ladi. Javob yaxlitlanmaydi, kasr qismi tashlab yuborilib, butun qismining o`zi qoladi.

Modul operatori (%) butun songa bo`lishdan kelib chiqadigan qoldiqni beradi. x%y ifodasi x ni y ga bo`lgandan keyin chiqadigan qoldiqni beradi. Demak,

7%4 bizga 3 javobini beradi. % operatori faqat butun sonlar bilan ishlaydi. Vergulli (real) sonlar bilan ishlash uchun "math.h" kutubxonasidagi fmod funksiyasini qo`llash kerak.

C++ da qavslarning ma’nosi xuddi algebradagi kabi bo`ladi. Undan tashqari boshqa algebraik ifodalarning bajarilish ketma-ketligi ham odatdagidek. Oldin ko`paytirish, bo`lish va modul olish operatorlari bajariladi. Agar bir necha operator ketma-ket kelsa, ular chapdan o`nga qarab bajariladi. Bu operatorlardan keyin esa qo`shish va ayirish ijro etiladi.

Misol. k = m * 5 + 7 % n / (9 + x);

Birinchi bo`lib m*5 hisoblanadi. Keyin 7%n topiladi va qoldiq (9 + x) ga bo`linadi. Chiqqan javob esa m*5 ning javobiga qo`shiladi. Qisqasi, amallar matematikadagi kabi. Lekin biz o`qishni osonlashtirish uchun va xato qilish ehtimolini kamaytirish maqsadida qavslarni kengroq ishlatishimiz mumkin.

Yuqoridagi misol quyidagi ko`rinishga ega bo`ladi.

k = (m * 5) + ((7 % n) / (9 + x));

Amallar jadvali.

Arifmetik amallar	Razryadli amallar	Nisbat amallari	Mantiqiy amallar
+ qo`shish	& va	= = teng	&& va
- ayirish	| yoki	!= teng emas	|| yoki
* ko`paytirish	^ inkor	> katta	! inkor
/ bo`lish	<< chapga surish	>= katta yoki teng	
% modul olish	>> o`ngga surish	< kichik	
- unar minus	~ inkor	<= kichik yoki teng	
+ unar plyus			
++ oshirish			
-- kamaytirish			

Imlo amallar	Qiymat berish va shartli amallar	Tipli amallar	Adresli amallar
() – doirali qavs	= - oddiy qiymat berish	(tip) – tipni o`zgartirish	& - adresni aniqlash
[] – kavadrat qavs	or= - murakkab qiymat berish	sizeof – hajmni hisoblash	* - adres bo`yicha qiymat aniqlash

			yoki joylash
, - vergul	? – shartli amal		

Arifmetik amallar.Amallar odatda unar ya’ni bitta operandga qo`llaniladigan amallarga va binar ya’ni ikki operandga qo`llaniladigan amallarga ajratiladi. Binar amallar additiv ya’ni + qo`shuv va – ayirish amallariga hamda multiplikativ ya’ni * ko`paytirish, / bo`lish va % modul olish amallariga ajratiladi.

Additiv amallarining ustivorligi multiplikativ amallarining ustivorligidan pastroqdir. Butun sonni butun songa bo`lganda natija butun songacha yaxlitlanadi.

Misol. 20/3=6; (-20)/3=-6; 20/(-3)=-6.

Modul amali butun sonni butun songa bo`lishdan hosil bo`ladigan qoldiqga tengdir. Agar modul amali musbat operandlarga qo`llanilsa, natija ham musbat bo`ladi, aks holda natija ishorasi kompilatorga bog`liq bo`ladi.

Binar arifmetik amallar bajarilganda tiplarni keltirish quyidagi qoidalar asosida amalga oshiriladi:

short va char tiplari int tipiga keltiriladi;

Agar operandlar biri long tipiga tegishli bo`lsa ikkinchi operand ham long tipiga keltiriladi va natija ham long tipiga tegishli bo`ladi;

Agar operandlar biri float tipiga tegishli bo`lsa ikkinchi operand ham float tipiga keltiriladi va natija ham float tipiga tegishli bo`ladi;

Agar operandlar biri double tipiga tegishli bo`lsa ikkinchi operand ham double tipiga keltiriladi va natija ham double tipiga tegishli bo`ladi;

Agar operandlar biri long double tipiga tegishli bo`lsa ikkinchi operand ham long double tipiga keltiriladi va natija ham long double tipiga tegishli bo`ladi;

Unar amallarga ishorani o`zgartiruvchi unar minus – va unar + amallari kiradi. Bundan tashqari ++ va -- amallari ham unar amallarga kiradi. ++ unar amali qiymatni 1 ga oshirishni ko`rsatadi. Amalni prefiks ya’ni ++i ko`rinishda ishlatish oldin o`zgaruvchi qiymatini oshirib so`ngra foydalanish lozimligini, postfiks ya’ni i++ ko`rinishda ishlatish oldin o`zgaruvchi qiymatidan foydalanib so`ngra oshirish kerakligini ko`rsatadi.

Misol. i qiymati 2 ga teng bo`lsin, u holda 3+(++i) ifoda qiymati 6 ga, 3+i++ ifoda qiymati 5 ga teng bo`ladi. Ikkala holda ham i qiymati 3 ga teng bo`ladi.

- unar amali qiymatni 1 ga kamaytirishni ko`rsatadi. Bu amal ham prefiks va postfiks ko`rinishda ishlatilishi mumkin. Bu ikki amalni faqat o`zgaruvchilarga qo`llash mumkindir. Unar amallarning ustivorligi binar amallardan yuqoridir.

Razryadli amallar.Razryadli amallar natijasi butun sonlarni ikkilik ko`rinishlarining har bir razryadiga mos mantiqiy amallarni qo`llashdan hosil bo`ladi.

Masalan 5 kodi 101 ga teng va 6 kodi 110 ga teng.

6&5 qiymati 4 ga ya’ni 100 ga teng.

6|5 qiyjmati 7 ga ya’ni 111 ga teng.

6^5 qiymati 3 ga ya’ni 011 ga teng.

~6 qiymati 4 ga ya’ni 010 ga teng.

Bu misollarda amallar ustivorligi oshib borishi tartibida berilgan. Bu amallardan tashqari M<<N chapga razryadli siljitish va M>>N o`ngga razryadli siljitish amallari qo`llaniladi. Siljitish M butun sonning razryadli ko`rinishiga qo`llaniladi. N nechta pozitsiyaga siljitish kerakligini ko`rsatadi. Chapga N pozitsiyaga surish bu operand qiymatini ikkining N chi darajasiga ko`paytirishga mos keladi.

Misol uchun 5<<2=20. Bu amalning bitli ko`rinishi: 101<<2=10100.
Agar operand musbat bo`lsa N pozitsiyaga o`ngga surish chap operandni ikkining N chi darajasiga bo`lib kasr qismini tashlab yuborishga mosdir. Misol uchun 5>>2=1. Bu amalning bitli ko`rinishi 101>>2=001=1. Agarda operand qiymati manfiy bo`lsa ikki variant mavjuddir: arifmetik siljitishda bo`shatilayotgan razryadlar ishora razryadi qiymati bilan to`ldiriladi, mantiqiy siljitishda bo`shatilayotgan razryadlar 0 lar bilan to`ldiriladi.

Razryadli surish amallarining ustivorligi o`zaro teng, razryadli inkor amalidan past, qolgan razryadli amallardan yuqoridir. Razryadli inkor amali unar qolgan amallar binar amallarga kiradi.

Nisbat amallari. Nisbat amallari qiymatlari 1 ga teng, agar nisbat bajarilsa va aksincha 0 ga tengdir. Nisbat amallari arifmetik tipdagi operandlarga yoki ko`rsatkichlarga qo`llaniladi.

Misollar:

1!=0 qiymati 1 ga teng;

1==0 qiymati 0 ga teng;

3>=3 qiymati 1 ga teng;

3>3 qiymati 0 ga teng;

2<=2 qiymati 1 ga teng;

2<2 qiymati 0 ga teng.

Katta >, kichik <, katta yoki teng >=, kichik yoki teng <= amallarining ustivorligi bir xildir.

Teng == va teng emas != amallarining ustivorligi o`zaro teng va qolgan amallardan pastdir.

Mantiqiy amallar.C ++ tilida mantiqiy tip mavjud emas. Shuning uchun mantiqiy amallar butun sonlarga qo`llanadi. Bu amallarning natijalari qo`yidagicha aniqlanadi:

x||y amali 1 ga teng agar x>0 yoki y>0 bo`lsa, aksincha 0 ga teng;

x&&y amali 1 ga teng agar x>0 va y>0 bo`lsa, aksincha 0 ga teng;

!x amali 1 ga teng agar x>0 bo`lsa, aksincha 0 ga teng.

Bu misollarda amallar ustivorligi oshib borish tartibida berilgandir.

Inkor ! amali unar qolganlari binar amallardir. Bu amallardan tashqari quyidagi amallar ham mavjud.

Qiymat berish amali. Qiymat berish amali = binar amal bo`lib chap operandi odatda o`ng o`zgaruvchi ifodaga teng bo`ladi.

Misol uchun: Z=4.7+3.34

Bu qiymati 8.04 ga teng ifodadir. Bu qiymat Z o`zgaruvchiga ham beriladi. Bu ifoda oxiriga nuqta vergul (;) belgisi qo`yilganda operatorga aylanadi: Z=4.7+3.34;

Bitta ifodada bir necha qiymat berish amallari qo`llanilishi mumkin. Misol: C=y=f=4.2+2.8;

Bundan tashqari C ++ tili da murakkab qiymat berish amali mavjud bo`lib, umumiy ko`rinishi quyidagicha:

O`zgaruvchi_nomi amal= ifoda;

Bu yerda amal quyidagiardan *, /, %, +, -, &, ^, |, <<, >> biri bo`ladi.

Misol:

X+=4 ifoda x=x+4 ifodaga ekvivalent;

X*=a ifoda x=x*a ifodaga ekvivalent;

X/=a+b ifoda x=x/(a+b) ifodaga ekvivalent;

X>>=4 ifoda x=x>>4 ifodaga ekvivalent.

Imlo belgilari amal sifatida. C ++ tilida ba'zi bir imlo belgilari ham amal sifatida ishlatilishi mumkin. Bu belgilardan oddiy () va kvadrat [] qavslardir. Oddiy qavslar binar amal deb qaralib ifodalarda yoki funksiyaga murojaat qilishda foydalaniladi. Funksiyaga murojaat qilish qo`yidagi shaklda amalga oshiriladi:

<funksiya nomi> (<argumentlar ruyxati>).

Misol: sin(x) yoki max(a,b).

Kvadrat qavslardan massivlarga murojaat qilishda foydalaniladi. Bu murojaat quyidagicha amalga oshiriladi:

<massiv nomi>[<indeks>].

Misol: a[5] yoki b[n][m].

Vergul simvolini ajratuvchi belgi deb ham qarash mumkin amal sifatida ham qarash mumkin. Vergul bilan ajratilgan amallar ketma-ketligi bir amal deb qaralib, chapdan o`ngga hisoblanadi va oxirgi ifoda qiymati natija deb qaraladi.

Misol: d=4, d+2 amali natijasi 8 ga teng.

Shartli amal. Shartli amal ternar amal deyiladi va uchta operanddan iborat bo`ladi:

<1-ifoda>?<2-ifoda>:<3-ifoda>

Shartli amal bajarilganda avval 1- ifoda hisoblanadi. Agar 1-ifoda qiymati 0 dan farqli bo`lsa 2- ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi, aks holda 3-ifoda hisoblanadi va qiymati natija sifatida qabul qilinadi.

Misol uchun modulni hisoblash: x<0?-x:x yoki ikkita son kichigini hisoblash a<b?a:b .

Shuni aytish lozimki shartli ifodadan har qanday ifoda sifatida foydalanish mumkin. Agar F - FLOAT tipga va N – INT tipga tegishli bo`lsa, (N > 0) ? F : N ifoda N musbat yoki manfiyligidan qat'iy nazar DOUBLE tipiga tegishli bo`ladi. Shartli ifodada birinchi ifodani qavsga olish shart emas.

Tiplar bilan ishlovchi amallar. Tiplarni o`zgartirish amali quyidagi ko`rinishga ega:

(tip_nomi) operand;

Bu amal operandlar qiymatini ko`rsatilgan tipga keltirish uchun ishlatiladi. Operand sifatida kostanta, o`zgaruvchi yoki qavslarga olinga ifoda kelishi mumkin. Misol uchun (long) 6 amali o`zgarmas qiymatini o`zgartirmagan holda operativ xotirada egallagan baytlar sonini oshiradi. Bu misolda o`zgarmas tipi o`zgarmagan bo`lsa, (double) 6 yoki (float) 6 amali o`zgarmas ichki ko`rinishini ham o`zgartiradi. Katta butun sonlar haqiqiy tipga keltirilganda sonning aniqligi yuqolishi mumkin. sizeof amali operand sifatida ko`rsatilgan obyektning baytlarda xotiradagi hajmini hisoblash uchun ishlatiladi. Bu amalning ikki ko`rinishi mavjud: sizeof ifoda sizeof (tip).

Misol:
Sizeof 3.14=8;
Sizeof 3.14f=4;
Sizeof 3.14L=10;
Sizeof(char)=1;
Sizeof(double)=8.

3.2.7. Amallar ustivorligi

№	Amallar	Yo`nalish
1.	() [] - > :: .	Chapdan o`ngga
2.	*! ~ + - ++ -- & * (tip) sizeof new*	O`ngdan chapga

	delete tip()	
3.	. * - > *	Chapdan o`ngga
4.	* / % (multiplikativ binar amallar)	Chapdan o`ngga
5.	+ - (additiv binar amallar)	Chapdan o`ngga
6.	<<>>	Chapdan o`ngga
7.	<<= >= >	Chapdan o`ngga
8.	= !=	Chapdan o`ngga
9.	&	Chapdan o`ngga
10.	^	Chapdan o`ngga
11.	|	Chapdan o`ngga
12.	&&	Chapdan o`ngga
13.	||	Chapdan o`ngga
14.	*? :* (shartli amal)	Chapdan o`ngga
15.	= *= /= %= += -= &= ^= |= <<= >>=	Chapdan o`ngga
16.	, (vergul amali)	Chapdan o`ngga

3.2.8. Dastur tuzilishi

C++ dasturlash tilida dastur quyidagi tarkibda tashkil topadi:

- Direktivalar – funksiyalar kutubxonasini chaqirish. Ular maxsus include katalogida joylashgan va kengaytmali fayllar bo`ladi.
- C++ tilida masalaning qo`yilishiga qarab kerakli kutubxonalar chaqiriladi. Bu esa dasturning xotirada egallaydigan joyini minimallashtiradi.

Sodda dastur tuzilishi. Dastur preprocessor komandalari va bir necha funksiyalardan iborat bo`lishi mumkin. Bu funksiyalar orasida *main* nomli asosiy funksiya bo`lishi shart. Agar asosiy funksiyadan boshqa funksiyalar ishlatilmasa dastur quyidagi ko`rinishda tuziladi:

Preprocessor_komandalari

Void main()

{

Dastur tanasi.

}

Preprocessor direktivalari kompilatsiya jarayonidan oldin preprocessor tomonidan bajariladi. Natijada dastur matni preprocessor direktivalari asosida o`zgartiriladi. Preprocessor komandalaridan ikkitasini ko`rib chiqamiz.

 # 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.

Misol tariqasida C ++ tilida tuzilgan birinchi dasturni keltiramiz:

```
#include <iostream.h>
void main()
{
Cout << “\n Assalomu-alaykum! \n”;
}
```

Bu dastur ekranga Assalomu-alaykum! Jumlasini chiqaradi. Define direktivasi yordamida bu dasturni quyidagicha yozish mumkin:

```
#include <iostream.h>
#define pr Cout << “\n Salom, do`stim! \n”
#define begin
{
#define end
 }
void main()
begin
pr;
end
```

Define direktivasidan nomlangan o`zgarmaslar kiritish uchun foydalanish mumkin.

Misol:

#define EULER 2.718282

Agar dasturda quyidagi matn mavjud bo`lsin:

Double mix=EULER

D=alfa*EULER

Preprocessor bu matnda har bir EULER o`zgarmasni uning qiymati bilan almashtiradi, va natijada quyidagi matn hosil bo`ladi.

Double mix=2.718282

D=alfa*2.718282

Dastur matni va preprocessor. C ++ tilida matnli fayl shaklida tayyorlangan dastur uchta qayta ishlash bosqichlaridan o`tadi. Matnni preprocessor direktivalari asosida o`zgartilishi. Bu jarayon natijasi Yana matnli fayl bo`lib preprocessor tomonidan bajariladi.

Kompilyasiya. Bu jarayon natijasi mashina kodiga o`tkazilgan obyektli fayl bo`lib, kompilator tomonidan bajariladi.

**Bog`lash**. Bu jarayon natijasi to`la mashina kodiga o`tkazilgan bajariluvchi fayl bo`lib, bog`lagich(komponovchik) tomonidan bajariladi. Preprocessor vazifasi dastur matnini preprocessor direktivalari asosida o`zgartirishdir.

Define direktivasi dasturda bir jumlani ikkinchi jumla bilan almashtirish uchun ishlatiladi. Bu direktivadan foydalanishning sodda misollarini biz yuqorida ko`rib chiqdik. *Include* direktivasi ikki ko`rinishda ishlatilishi mumkin.

#include fayl nomi direktivasi dasturning shu direktiva urniga qaysi matnli fayllarni qo`shish kerakligini ko`rsatadi.

#include<fayl nomi> direktivasi dasturga kompilator standart bibliotekalariga mos keluvchi sarlavhali fayllar matnlarini qushish uchun mo`ljallangandir. Bu fayllarda funksiya prototipi, tiplar, o`zgaruvchilar, o`zgarmaslar ta’riflari yozilgan bo`ladi. Funksiya prototipi funksiya qaytaruvchi tip, funksiya nomi va funksiyaga uzatiluvchi tiplardan iborat bo`ladi. Misol uchun *cos* funksiyasi prototipi quyidagicha yozilishi mumkin: *double cos(double)*. Agar funksiya nomidan oldin *void* tipi ko`rsatilgan bo`lsa bu funksiya hech qanday qiymat qaytarmasligini ko`rsatadi. Shuni ta’kidlash lozimki bu direktiva dasturga

standart biblioteka qo`shilishiga olib kelmaydi. Standart funksiyalarning kodlari bog`lash ya'ni aloqalarni tahrirlash bosqichida, kompilyasiya bosqichidan so`ng amalga oshiriladi. Kompilyasiya bosqichida sintaksis xatolar tekshiriladi va dasturda bunday xatolar mavjud bo`lmasa, standart funksiyalar kodlarisiz mashina kodiga o`tkaziladi. Sarlavhali fayllarni dasturning ixtiyoriy joyida ulash mumkin bo`lsa ham, bu fayllar odatda dastur boshida qo`shish lozimdir. Shuning uchun bu fayllarga sarlavhali fayl (*header file*) nomi berilgandir.

Dasturda kiritish va chiqarish funksiyalaridan masalan *Cout*<< funksiyasidan foydalanish uchun *#include <iostream.h>* direktivasidan foydalanish lozim. Bu direktivada iostream.h sarlavhali fayl nomi quyidagilarni bildiradi:

st - standart (standartj), *i - input* (kiritish), *o - output* (chiqarish), *h – head* (sarlavha).

3.2.9. Mantiqiy solishtirish operatorlari

C++ bir necha solishtirish operatorlariga ega bo`lib ular quyidagilardan iborat.

Algebraik ifoda	C++ dagi operator	C++ dagi ifoda	Algebraik ma'nosi
=	==	x==y	x teng y ga
teng emas	!=	x!=y	x teng emas y ga
>	>	x>y	x katta y dan
<	<	x<y	x kichkina y dan
katta-teng	>=	x>=y	x katta yoki teng y ga
kichik-teng	<=	x<=y	x kichik yoki teng y ga

==, !=, >= va <= operatorlarni yozganda oraga bo`sh joy qo`yib ketish sintaksis xato hisoblanadi. Ya'ni kompilator dasturdagi xatoni ko`rsatib beradi va uni tuzatilishini talab qiladi. Ushbu ikki belgili operatorlarning belgilarining joyini almashtirish, masalan <= ni =< qilib yozish ko`p hollarda sintaksis xatolarga olib keladi. != ni =! deb yozganda sintaksis xato hisoblanib, bu mantiqiy xato bo`ladi. Mantiqiy xatolarni kompilator topa olmaydi. Lekin ular dastur ishlash algoritmini

o`zgartirib yuboradi. Bu kabi xatolarni topish esa ancha mashaqqatli ishdir (! operatori mantiqiy inkordir). Yana boshqa xatolardan biri tenglik operatori (==) va tenglashtirish, qiymat berish operatorlarini (=) bir-biri bilan almashtirib qo`yishdir. Bu ham juda ayanchli oqibatlarga olib keladi, chunki ushbu xato aksariyat hollarda mantiq xatolariga olib keladi. Yuqoridagi solishtirish operatorlarini ishlatadigan bir dasturni ko`raylik.

```
//Mantiqiy solishtirish operatorlari
# include <iostream.h>
int main()
{
int s1, s2;
cout << "Ikkita son kiriting: " << endl;
cin >> s1 >> s2; //Ikkita son olindi.
if (s1 == s2) cout << s1 << "teng" << s2 << "ga" << endl;
if (s1 < s2) cout << s1 << "kichik" << s2 << "dan" << endl;
if (s1 >= s2) cout << s1 << "katta yoki teng " << s2 << "ga" << endl;
if (s1 != s2) cout << s1 << "teng emas" << s2 << "ga" << endl;
return (0);
}
```

Ekranda:

```
Ikki sonni kiriting: 74 33
74 katta yoki teng 33 ga
74 teng emas 33 ga
```

Bu yerda bizga yangi bu C++ ning *if* (agar) strukturasidir. *if* ifodasi ma’lum bir shartning to`g`ri (*true*) yoki noto`g`ri (*false*) bo`lishiga qarab, dasturning u yoki bu blokini bajarishga imkon beradi. Agar shart to`g`ri bo`lsa, *if* dan so`ng keluvchi amal bajariladi. Agar shart bajarilmasa, u holda *if* tanasidagi ifoda bajarilmay, *if* dan so`ng keluvchi ifodalar ijrosi davom ettiriladi. Bu strukturaning ko`rinishi quyidagichadir:

if (shart) ifoda;

Shart qismi qavs ichida bo`lishi majburiydir. Eng oxirida keluvchi nuqta-vergul (;) shart qismidan keyin qo`yilsa (*if* (shart); ifoda;) mantiq xatosi vujudga keladi. Chunki, bunda *if* tanasi bo`sh qoladi. Ifoda qismi esa shartning to`g`ri-noto`g`ri bo`lishiga qaramay ijro qilaveradi.

C++ da bitta ifodani qo`yish mumkin bo`lgan joyga ifodalar guruhini ham qo`yish mumkin. Bu guruhni {} qavslar ichida yozish kerak.

Misol:
if (shart)
{
ifoda1;
ifoda2;
...
ifodaN;
}

Agar shart to`g`ri javobni bersa, ifodalar guruhi bajariladi, aks holda blokni yopuvchi qavslardan keyingi ifodalardan dastur ijrosi davom ettiriladi.

3.2.10. Ko`rsatkichlar (*Pointers*)

**Ko`rsatkichlar ta'rifi.** C va C++ tillarining asosiy xususiyatlaridan ko`rsatkichlarning keng qo`llanilishidir. Ko`rsatkichlar tipda o`zgarmas ko`rsatkichlar va o`zgaruvchi ko`rsatkichlarga ajratiladi. Ko`rsatkichlar qiymati konkret tipdagi obyektlar uchun xotirada ajratilgan adreslarga tengdir. Shuning uchun ko`rsatkichlar ta'riflanganda ularning adreslarini ko`rsatish shart. O`zgaruvchi ko`rsatkichlar qo`yidagicha ta'riflanadi.

<tip> * <ko`rsatkich nomi>

Misol: int* lp,lk.

Ko`rsatkichlarni ta'riflaganda initsializatsiya qilish mumkindir. Initsializatsiya quyidagi shaklda amalga oshiriladi:

<tip> * <ko`rsatkich nomi>=<o`zgarmas ifoda>

O`zgarmas ifoda sifatida qo`yidagilar kelishi mumkin.

- Xotira qismining aniq ko`rsatilgan adresi.

Misol uchun: char* comp=(char*) 0xF000FFFE;

Bu adresda kompyuter tipi shaklidagi ma'lumot saqlanadi.

- Qiymatga ega ko`rsatkich: char c1=comp;

- & simvoli yordamida aniqlangan obyekt adresi.

Misol uchun: char c='d'; char* pc=&c;

Borland kompilatorlarida maxsus *NULL* qiymat kiritilgan bo`lib, bu qiymatga ega ko`rsatkichlar bo`sh ko`rsatkichlar deyiladi. Bo`sh ko`rsatkichlar bilan xotirada hech qanday adres bog`lanmagan bo`ladi, lekin dasturda konkret obektlar adreslarini qiymat sifatida berish mumkin.

Char ca=’d’; char* pa(NULL); pa=&ca;

Ko`rsatkichlar ustida amallar. Yuqorida keltirilgan misollarda & adres olish amalidan keng foydalanilgan. Bu amal nomga va hotirada aniq adresga ega obyektlarga, misol uchun o`zgaruvchilarga qo`llaniladi. Bu amalni ifodalarga yoki nomsiz o`zgarmaslarga qo`llash mumkin emas. Ya’ni &3.14 yoki &(a+b) ifodalar xato hisoblanadi. Bundan tashqari ko`rsatkichlar bilan birga * adres bo`yicha qiymat olish yoki kiritish amali keng qo`llaniladi.

Misol uchun: Int i=5; int*pi=&I; int k=*pi; *pi=6.

Bu misolda *pi* ko`rsatkich i o`zgaruvchi bilan bog`lanadi. **pi*=6 amali i o`zgaruvchi qiymatini ham o`zgartiradi. O`zgarmas ko`rsatkich va o`zgarmasga ko`rsatkichlar. O`zgarmas ko`rsatkich quyidagicha ta’riflanadi:

<tip>* const<kursatkich nomi>=<o`zgarmas ifoda>

Misol uchun: char* const key_byte=(char*)0x0417.

Bu misolda o`zgarmas ko`rsatkich klaviatura holatini ko`rsatuvchi bayt bilan bog`langan. O`zgarmas ko`rsatkich qiymatini o`zgartirish mumkin emas lekin * amali yordamida xotiradagi ma’lumot qiymatini o`zgartirish mumkin. Misol uchun *key_byte=’Yo` amali 1047(0x0417) adres qiymati bilan birga klaviatura holatini ham o`zgartiradi.

O`zgarmasga ko`rsatkich quyidagicha ta’riflanadi:

<tip>*const**<ko`rsatkich nomi>=<o`zgarmas ifoda>. Misol uchun const int zero=0; *int*

*const** p=&zero;

Bu ko`rsatkichga * amalini qo`llash mumkin emas, lekin ko`rsatkichning qiymatini o`zgartirish mumkin. Qiymati o`zgarmaydigan o`zgarmasga ko`rsatkichlar quyidagicha kiritiladi:

<tip>*const** *const*<ko`rsatkich nomi>=<o`zgarmas ifoda>.

Misol: *const float* pi=3.141593; *float const* const* pp=π

3.2.11. Operatorlar va bloklar

Har qanday dastur funksiyalar ketma ketligidan iborat bo`ladi. Funksiyalar sarlavha va funksiya tanasidan iborat bo`ladi. Funksiya sarlavhasiga *void main()* ifoda misol bo`laoladi. Funksiya tanasi obyektlar ta'riflari va operatorlardan iborat bo`ladi. Har qanday operator nuqta-vergul belgisi bilan tugashi lozim. Quyidagi ifodalar X=0, yoki I++ operatorga aylanadi agar ulardan so`ng nuqtali vergul kelsa X = 0; I++;

Operatorlar bajariluvchi va bajarilmaydigan operatorlarga ajratiladi.

Bajarilmaydigan operator bu izoh operatoridir. Izoh operatori /* belgisi bilan boshlanib */ belgisi bilan tugaydi. Bu ikki simvol orasida ixtiyoriy jumla yozish mumkin. Kompilator bu jumlani tekshirib o`tirmaydi. Izoh
operatoridan dasturni tushunarli qilish maqsadida izohlar kiritish uchun foydalaniladi. Bajariluvchi operatorlar o`z navbatida ma'lumotlarni o`zgartiruvchi va boshqaruvchi operatorlarga ajratiladi. Ma'lumotlarni o`zgartiruvchi operatorlarga qiymat berish operatorlari va nuqta vergul bilan tugovchi ifodalar kiradi.

Misol:

I++;

X*=I;

I=x-4*I;

Boshqaruvchi operatorlar dasturni boshqaruvchi konstruksiyalar deb ataladi. Bu operatorlarga quyidagilar kiradi:

Qo`shma operatorlar;

Tanlash operatorlari;

Sikl operatorlari;

O`tish operatorlari;

**Qo`shma operatorlar.** Bir necha operatorlar { va } figurali qavslar yordamida qo`shma operatorlarga yoki bloklarga birlashtirilishi mumkin. Blok

yoki qo`shma operator sintaksis jihatdan bitta operatorga ekvivalentdir. Blokning qo`shma operatordan farqi shundaki blokda obyektlar ta’riflari mavjud bo`lishi mumkin. Quyidagi dastur qismi qo`shma operator:

```
{
n++;
summa+=(float)n;
}
```

Bu fragment bo`lsa blok:

```
{
int n=0;
n++;
summa+=(float)n;
}
```

Kiritish chiqarish operatorlari. Chiquvchi oqim *cout* kelishilgan bo`yicha ekranga mos keladi. Lekin maxsus operatorlar yordamida oqimni printer yoki faylga mos quyish mumkin. Misol uchun MS-DOS qo`yidagi komandasi FIRST.EXE dasturi chiqishni printerga yo`naltiradi:

```
S:\> FIRST > PRN <ENTER>
```

Quyidagi dastur 1001.SRR 1001 sonini ekranga chiqaradi:

```
#include <iostream.h>
void main(void)
{
cout << 1001;
}
```

Dastur bajarilishi natijasi :

```
S:\> 1001 <ENTER>
1001
```

Bir necha qiymatlarni chiqarish:

```
#include <iostream.h>
void main(void)
(
cout << 1 << 0 << 0 << 1;
}
```

Natija:

S:\> 1001TOO <ENTER>

1001

3.2.11. Operatorlar va bloklar

Har qanday dastur funksiyalar ketma ketligidan iborat bo`ladi. Funksiyalar sarlavha va funksiya tanasidan iborat bo`ladi. Funksiya sarlavhasiga *void main()* ifoda misol bo`laoladi. Funksiya tanasi obyektlar ta'riflari va operatorlardan iborat bo`ladi. Har qanday operator nuqta-vergul belgisi bilan tugashi lozim. Quyidagi ifodalar X=0, yoki I++ operatorga aylanadi agar ulardan so`ng nuqtali vergul kelsa X = 0; I++;

Operatorlar bajariluvchi va bajarilmaydigan operatorlarga ajratiladi.

Bajarilmaydigan operator bu izoh operatoridir. Izoh operatori /* belgisi bilan boshlanib */ belgisi bilan tugaydi. Bu ikki simvol orasida ixtiyoriy jumla yozish mumkin. Kompilator bu jumlani tekshirib o`tirmaydi. Izoh operatoridan dasturni tushunarli qilish maqsadida izohlar kiritish uchun foydalaniladi. Bajariluvchi operatorlar o`z navbatida ma'lumotlarni o`zgartiruvchi va boshqaruvchi operatorlarga ajratiladi. Ma'lumotlarni o`zgartiruvchi operatorlarga qiymat berish operatorlari va nuqta vergul bilan tugovchi ifodalar kiradi.

Misol:

```
I++;
X*=I;
I=x-4*I;
```

Boshqaruvchi operatorlar dasturni boshqaruvchi konstruksiyalar deb ataladi. Bu operatorlarga quyidagilar kiradi:

Qo`shma operatorlar;

Tanlash operatorlari;

Sikl operatorlari;

O`tish operatorlari;

**Qo`shma operatorlar.** Bir necha operatorlar { va } figurali qavslar yordamida qo`shma operatorlarga yoki bloklarga birlashtirilishi mumkin. Blok yoki qo`shma operator sintaksis jihatdan bitta operatorga ekvivalentdir. Blokning qo`shma operatordan farqi shundaki blokda obyektlar ta'riflari mavjud bo`lishi mumkin. Quyidagi dastur qismi qo`shma operator:

```
{
n++;
summa+=(float)n;
}
```

Bu fragment bo`lsa blok:

```
{
int n=0;
n++;
summa+=(float)n;
}
```

Kiritish chiqarish operatorlari. Chiquvchi oqim *cout* kelishilgan bo`yicha ekranga mos keladi. Lekin maxsus operatorlar yordamida oqimni printer yoki faylga mos quyish mumkin. Misol uchun MS-DOS qo`yidagi komandasi FIRST.EXE dasturi chiqishni printerga yo`naltiradi:

S:\> FIRST > PRN <ENTER>

Quyidagi dastur 1001.SRR 1001 sonini ekranga chiqaradi:

```
#include <iostream.h>
void main(void)
{
cout << 1001;
```

}

Dastur bajarilishi natijasi :

S:\> 1001 <ENTER>

1001

Bir necha qiymatlarni chiqarish:

```
#include <iostream.h>
void main(void)
(
cout << 1 << 0 << 0 << 1;
}
```

Natija:

S:\> 1001TOO <ENTER>

1001

Ma’lumotlarni kiritish operatori:

cin>>1-o`zgaruvchi>>2-o`zgaruvchi>>…>>n-o`zgaruvchi; yoki **cin>>**o`zgaruvchi nomi**;**

Misol: **cin**>>x>>y;

Ushbu buyruqdan keyin konsol oynasida x,y o`zgaruvchilarning qiymatlarini kiritish so`raladi.

Ma’lumotlarni chiqarish operatori:

Cout<<1-o`zgaruvchi<<2-o`zgaruvchi<<…<<n-o`zgaruvchi; yoki **cout<<**o`zgaruvchi nomi;

Misol**:** **cout**<<x<<y;

cout<<“x=”<<x <<“y=”<<y; Ushbu buyruqdan keyin konsol oynasida x,y o`zgaruvchilarning qiymatlari chiqadi.

O`zlashtirish operatori:

O`zgaruvchi=ifoda;

Misol: **y**=sqrt(x)+pow(2,x)

C++da dasturi tuzilishi:

 # include <iostream> *// sarlavha faylni qo'shish*

using namespace std; *// standart funksiya*

int main () *// bosh funksiya tavsifi*

{ *// blok boshlanishi*

dastur tanasi

} // blok tugashi

Main()funktsiyasi

Main()funktsiyasi - asosiy degan ma'noni anglatadi. Bu funksiya "{" belgisidan boshlanadi va dasturning asosini tashkil etuvchi o'zgaruvchilarning toifalari ko'rsatiladi. Dastur "}" belgisi bilan yakunlanishi shart. Agar dasturda qism dasturlardan foydalanilayotgan bo'lsa, ularning nomlari va haqiqiy parametrlari keltiriladi. So'ngra dasturning asosiy buyruqlari yoziladi. Agar buyruqlar murakkab bo'lsa, ular alohida "{}" belgilari orasiga olingan bo'lishi kerak

Chiziqli algoritmni dasturlashga misol keltiramiz.

1-misol. Quyidagi ifodaning qiymatini hisoblang:

$Z = \frac{cos^2(x+a)}{tg(bx^2+a)}$ bu yerda ***a*** = -3,15; ***b***=4,33; ***x*** - ixtiyoriy son.

Dastur kodi:

```
#include <iostream>
#include <math.h>
using namespace std;
int  main( )
{ double a,b,x,z;
cout<< "a,b,x larning qiymatlarini kiriting:\n";
cin>>a>>b>>x;
z=pow(cos(3*x+a),2)/tan(b*x*x+a);
cout <<"z="<<z;
}
```

1-Masala: Qumning tabiiy namligi quyidagi ifoda orqali aniqlanadi:

$$W_T = \frac{m_1 - m_2}{m_2} * 100\%$$

Bu yerda: $\mathbf{m_1}$-tabiiy nam qumning og'irligi, g;

$\mathbf{m_2}$-quritilgan qumning og'irligi, g

Agar tabiiy nam qumning og'irligi 1000 g , quritilgan qumning og'irligi 987 g bo'lsa, qumning namligini aniqlashga dastur tuzing.

Dastur kodi:

```
# include <iostream>
using namespace std;
int main()
{
int   m1, m2, Wt;
cout<<"tabiiy nam qumning og'irligini kiriting:"<<endl;
cin>>m1;
cout<<"Quritilgan qumning og'irligini kiriting:"<<endl;
cin>>m2;
Wt=(m1+m2)/m2;
cout<<"Qumning tabiiy namligi="<<Wt<<endl;
}
```

2-Masala: Elektr zanjiridagi iste'molchi qarshiligi R=8 om, tok kuchi I=2 amper, tok manbaining ichki qarshiligi r=1 om bo`lsa, tok manbai klemmalaridagi kuchlanishni, tok manbaining ichki kuchlanishini va elektr yurutuvchi kuchni hisoblash dasturini tuzing.

Dastur kodi:

```
# include <iostream>
using namespace std;
int main()
{
float  R,r,I,U,u,E;
```

```
cout<<"Istemolchi qarshiligini kiriting:"<<endl;
cin>>R;
cout<<"Tok manbai ichki  qarshiligini kiriting:"<<endl;
cin>>r;
cout<<"Zanjirdagi tok kuchini kiriting:\n";
cin>>I;
U=I*R;
u=I*r;
E=I*(R+r);
cout<<"Tok manbai klemmalaridagi kuchlanish="<<U<<endl;
cout<<"Tok manbaidagi ichki kuchlanish="<<u<<endl;
cout<<"Zanjirdagi elektr yurutuvchi kuch="<<E<<endl;
}
```

Chiziqli algoritmlarni dasturlash

Topshiriq: Quyidagi chiziqli algiritmlarni hisoblash uchun C++ tilida dastur tuzing.

1-misol. $a = y + \left(\frac{x}{x} * x + \left|e^y + \frac{x^3}{3}\right|\right); \quad b = 1 + \frac{\cos(a-2)}{x^4 + \sqrt[3]{sin^2(x-y)}}$

Dastur matni

```
# include<iostream.h>
# include<stdio.h>
# include<math.h>
Using namespace std;
int main()
{
float a,b,x,y;
cin>>x>>y;
```

a=y+(x/x*x+fabs(exp(y)+pow(x,3)/3)));

b=1+cos(a-2)/(pow(x,4)+pow(sin(x-y)* sin(x-y)/1./3);

Printf(“%.3f”,a);

cout<<endl;

Printf(“%.3f”,b);

//system(“pause”);

return 0;

}

Dasturni kiritib, qiymatlarni beramiz.

Kiritiladi **2 1.2** Javob **1.628**

Pi=3.1415 **0.115**

2-misol: $a = \frac{2*cos(x-\frac{Pi}{6})}{sin2x+sin^2(x-y)}$; $b = cos^2\left(arctg\frac{1}{a}\right) + \sqrt[3]{e^{x+y}}$

Dastur matni

 # include<iostream.h>

 # include<stdio.h>

 # include<math.h>

Using namespace std;

int main()

{

float a,b,x,y;

cin>>x>>y;

a=2*cos(x-Pi/6)/(sin(2*x)+sin(x-y)* sin(x-y));

b=cos(atan(1/a))* cos(atan(1/a))+pow(exp(x+y) ,1./3);

Printf(“%.3f”,a);

```
cout<<endl;
Printf("%.3f",b);
//system("pause");
return 0;
}
```

Dasturni kiritib, qiymatlarni beramiz.

Kiritiladi **1.6 -6.2** Javob **1.011**

Pi=3.1415 **0.721**

3-masala: Berilgan dastur matniga asosan algoritmning matematik ko'rinishini aniqlab bering.

```
# include<iostream.h>
# include<stdio.h>
# include<math.h>
Using namespace std;
int main()
{
float a,b,x,y;
cin>>x>>y;
a=(1+sin(x+y)* sin(x+y)/(2+fabs(x-2*x/(1+x*x*y*y))))+pow(x,1./3);
b=pow(1+a*a*(x+y)/(exp(x)*x*x*y*y));
 Printf("%.3f",a);
cout<<endl;
Printf("%.3f",b);
//system("pause");
return 0;
}
```

Dasturni kiritib, qiymatlarni beramiz.

Kiritiladi **4** **3.4** Javob **1.891**

Pi=3.1415 **0.290**

4-masala: Berilgan dastur matniga asosan algoritmning matematik ko'rinishini aniqlab bering.

```
# include<iostream.h>
# include<stdio.h>
# include<math.h>
Using namespace std;
int main()
{
float a,b,x,y;
cin>>x>>y;
a=log(fabs(y-sqrt(x)*x+(exp(y)/(pow(cos(x),2)+y*y/4))));
b=x+tan(2*Pi/a)+(5*pow(10,-6)+pow(a*y,1./4));
 Printf("%.3f",a);
cout<<endl;
Printf("%.3f",b);
//system("pause");
return 0;
}
```

Dasturni kiritib, qiymatlarni beramiz.

Kiritiladi **4.5** **3.4** Javob **2.934**

Pi=3.1415 **5.227**

5-masala: A va B ning yig'indisini hisoblashga dastur:

```
# include<iostream>

using namespace std;

int  main()

{

        long a,b,c;

        cin>>a>>b;

        c=a+b;

        cout<<c;

        return 0;

}
```

6-masala: A va B ning ko'paytmasini hisoblashga dastur:

```
# include <iostream>
using namespace std;
int main()
{
        int a,b,c;
        cin>>a>>b>>c;
        if (a*b==c)  {cout<<"yes";}
        else {cout<<"no";}
return 0;
}
```

7-masala: Butun sonlar summasini hisoblashga dastur:

```
# include <iostream>
using namespace std;
int main()
{
        int n,i,s;
        s=0; i=1;
        cin>>n;
        if (i<=n)   {s=s+i;  i=i+1;};
        cout<<s;
return 0;
```

}

Tayanch soz va iboralar

Dastur, dasturlash tili, C++ tili, kompilyatsiya, ma’lumotlani kiritish, ma’lumot, operand, o’zgarmas, haqiqiy.

Mavzuga oid savol va topshiriqlar

1. C++ tili qachon kim tomonidan yaratilgan?
2. Al–Xorazmiy kim?
3. Qanaqa dasturlash tillarini bilasiz?
4. Dasturlash tili deganda nimani tushunasiz?
5. Algoritm deb nimaga aytiladi?
6. Algoritm turlari?
7. Algoritmning grafik usulda ifodasi nima deb ataladi?
8. Blok-sxema nima?
9. Chiziqli algoritmni dasturlashda qanday operatorlardan foydalaniladi? (**cin**, **cout**, **o’zlashtirish operator**)
10. Dastur umumiy ko’rinishini aytib bering.
11. Dasturning tarkibiy qismi?
12. C++ tilida yozilgan dasturlar ishga tushirishdan oldin kompilyatsiya qaysi tugma yordamida qilinishi kerak?
13. F10 – tugma nima vazifani bajaradi?
14. F11 - tugma nima vazifani bajaradi?
15. C++ tilining alfaviti nimalardan tashkil topgan?
16. Algoritm va uning turlari.
17. Dastur strukturasi.
18. Dastur kompilatsiyasi qanday amalga oshiriladi?
19. C++ tili alfaviti va leksemlariga nimalar kiradi?
20. O`zgarmaslarning qanday turlari bor va ular qanday aniqlanadi?
21. O`zgaruvchilar va ularning turlari.
22. O`zgaruvchilar turlarini boshqa turglarga o’zgartirish.

23. Dasturlashda qanday amallardan foydalaniladi?
24. Amal ustuvorligi deganda nimani tushinasiz?
25. Dastur strukturasi qanday bo`limlardan iborat?
26. Mantiqiy amallardan qachon foydalaniladi?