ЎЗБЕКИСТОН РЕСПУБЛИКАСИ

ОЛИЙ ВА ЎРТА МАХСУС ТАЪЛИМ ВАЗИРЛИГИ

ОЛИЙ ТАЪЛИМ ТИЗИМИ ПЕДАГОГ ВА РАҲБАР КАДРЛАРИНИ ҚАЙТА ТАЙЁРЛАШ ВА УЛАРНИНГ МАЛАКАСИНИ ОШИРИШНИ ТАШКИЛ ЭТИШ БОШ ИЛМИЙ - МЕТОДИК МАРКАЗИ

ТОШКЕНТ АХБОРОТ ТЕХНОЛОГИЯЛАРИ УНИВЕРСИТЕТИ ҲУЗУРИДАГИ ПЕДАГОГ КАДРЛАРНИ ҚАЙТА ТАЙЁРЛАШ ВА УЛАРНИНГ МАЛАКАСИНИ ОШИРИШ ТАРМОҚ МАРКАЗИ

“Тасдиқлайман”

Тармоқ маркази директори

_____ Х.М.Холмедов

“_____”_____ 2015 йил

“ДАСТУРЛАШ АСОСЛАРИ” МОДУЛИ БЎЙИЧА

Ў Қ У В –У С Л У Б И Й М А Ж М У А

Тузувчи: **Қ.С.Рахманов**

Тошкент – 2015

МУНДАРИЖА

ИШЧИ ДАСТУР

I. Модулнинг мақсади ва вазифалари

“Дастурлаш асослари” модулининг мақсади: Педагог кадрларни қайта тайёрлаш ва малака ошириш курс тингловчиларини C/C++ тилида дастур тузиш, маълумотларнинг турли таркибий тузилмалари, алгоритмлаш ва дастурлаш асослари ва C/C++ дастурлаш тилининг асослари ва дастурий муҳитларни тадбиқ қилишкўникмаларни шакллантириш.

“Дастурлаш асослари” модулининг вазифалари:

- C/C++ дастурлаш тилининг асосий конструкциялари, маълумотлар тоифалари ва тузилмаларини қўллаш.
- C/C++ дастурлаш тилида аниқ алгоритмларини,тилнинг статик ва динамик имкониятларни тадбиқ этиш.
- C/C++ дастурлаш тилининг график имкониятлари.
- Объектга йўналтирилган дастурлаш асосларини қўллаш.
- Мураккаб дастурий тизимларни ташкил этиши ва қўйилган мақсадларга эришиш.
- Масалаларни ечиш ва натижавий маҳсулот сифати муҳимлигини тушуниш ҳамданатижаларни таҳлил қилиш.

Модулни ўзлаштиришга қўйиладиган талаблар

“Дастурлаш асослари” модулини ўзлаштириш жараёнида амалга ошириладиган масалалар доирасида тингловчилар:

- ахборот, информатика ва унинг таркибий қисмлари ҳақида тушунчага эга бўлиши;
- C/C++ тилида дастур тузиши;
- дастурлашда маълумотларнинг турли таркибий тузилмаларини тадбиқ эта олиши;
- алгоритмлаш ва дастурлаш асосларини билиши;
- C/C++ дастурлаш тилининг асосларини ва дастурий муҳитларни тадбиқ қилиши;
- тилнинг содда ва мураккаб тузилмаларини қўллай олиши;
- алгоритмларни баҳолаш, қўйилган масалани ечиш алгоритмини

танлаш, танловни асослаш, алгоритмни тадбиқ этиши керак.

Модулнинг ўқув режадаги бошқа модуллар билан боғлиқлиги ва узвийлиги

Модул мазмуни ўқув режадаги “Интернет технологиялари ва Web ресурслар”, “Мультимедия технологиялари асослари”, “Масофавий таълим технологияси”,“Дастурлаш технологияси”, “Компьютер тизимлари ва тармоқлари”, “Ахборот хавфсизлиги”, “Объектга йўналтирилган дастурлаш асослари”, “Компьютер графикаси ва дизайн”, “Маълумотлар базасини бошқариш тизимлари” ва “Web дастурлаш” ўқув модулларига эса – асос бўлиб хизмат қилади.

Модулнинг олий таълимдаги ўрни

Модулни ўзлаштириш орқали тингловчилар дастурлаш тиллари ва воситалари орқали дастурлар тузишга ва уларни амалга тадбиқ этиш бўйича касбий компетентликка эга бўладилар.

Модул бўйича соатлар тақсимоти:

№	Модул мавзулари	Тингловчининг ўқув юкламаси, соат					
		Ҳаммаси	Аудитория ўқув юкламаси				Мустақил таълим
			жами	жумладан			
				Назарий	Амалий машғулот	Кўчма машғулот	
1	Тил лексик асослари	6	6	2	4		
2	Операторлар ва функциялар	6	6	2	4		
3	Массивлар ва сатрлар	8	6	2	4		2
4	Структуралар ва файллар	8	6	2	4		2
	Жами:	28	24	8	16		4

НАЗАРИЙ МАШҒУЛОТЛАР МАЗМУНИ

1-мавзу. Тил лексик асослари (2 соат)

Режа:

1. Алфавит ва хизматчи сўзлар.
2. Ўзгарувчилар.
3. Константалар.
4. Амаллар.
5. C++ тилида дастур тузилиши
6. Маълумотларни киритиш ва чиқариш

2-мавзу. Операторлар ва функциялар (2 соат)

Режа:

1. Операторлар турлари.
2. Танлаш операторлари.
3. Цикл операторлари.
4. Ўтиш операторлари.
5. Фойдаланувчи функциялари
6. Рекурсия

3-мавзу. Массивлар ва сатрлар (2 соат)

Режа:

1. Бир ўлчовли массивлар.
2. Кўп ўлчовли массивлар.
3. Белгили ахборот ва сатрлар.
4. Сўзлар массивлари.
5. Излаш ва тартиблаш

4-мавзу. Структуралар ва функциялар (3 соат)

Режа:

1. Структура таърифи.
2. Структуралар ва массивлар
3. Структура хоссалари
4. Бирлашмалар
5. Файллар
6. Файлга кетма-кет мурожаат қилиш

АМАЛИЙ МАШҒУЛОТЛАР МАЗМУНИ

1-мавзу. Тил лексик асослари (4 соат)

Режа:

1. Алфавит ва хизматчи сўзлар.
2. Ўзгарувчилар ва константалар.
3. C++ тилида дастур тузилиши
6. Маълумотларни киритиш ва чиқариш

2-мавзу. Операторлар ва функциялар (4 соат)

Режа:

1. Танлаш, цикл ва ўтиш операторлари.
2. Фойдаланувчи функциялари
3. Рекурсия

3-мавзу. Массивлар ва сатрлар (4 соат)

Режа:

1. Бир, кўп ўлчовли ва сатрли массивлар.
2. Массивларда қидириш ва саралаш алгоритмлари

4-мавзу. Структуралар ва файллар (4 соат)

Режа:

1. Структура ва унинг хоссалари
2. Бирлашмалар
3. Файллар

КЎЧМА МАШҒУЛОТЛАР МАЗМУНИ

Кўчма машғулот ўқув модулида режалаштирилмаган.

МУСТАҚИЛ ТАЪЛИМ МАВЗУЛАРИ

1. Чизиқли масалаларни ишлаш (acm.tuit.uz).
2. Такрорланувчи масалаларни ишлаш (acm.tuit.uz).
3. Тармоқланувчи масалаларни ишлаш (acm.tuit.uz).
4. Мантиқий масалаларни ишлаш (acm.tuit.uz).
5. Массивларга доир масалаларни ишлаш (acm.tuit.uz).
6. Структурага доир масалаларни ишлаш (acm.tuit.uz).
7. Саралаш алгоритмларига доир масалаларни ишлаш (acm.tuit.uz).
8. Қидириш алгоритмларига доир масалаларни ишлаш (acm.tuit.uz).
9. Рекурсияга доир масалаларни ишлаш (acm.tuit.uz).
10. Стекка доир масалаларни ишлаш (acm.tuit.uz).
11. Навбатга доир масалаларни ишлаш (acm.tuit.uz).

12. Узун сонларга доир масалаларни ишлаш (acm.tuit.uz).
13. Сатрли массивларга доир масалаларни ишлаш (acm.tuit.uz).

Тавсия этилган адабиётлар рўйхати

Асосий адабиётлар

1. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013, 488 b.
2. Қосимов С.С. Ахборот технологиялари. Техника олий ўрта юртлари бакалавриат босқичи талабалари учун ўқув қўлланма сифатида тавсия этилган. Тошкент, “Алоқачи” нашриёти, 2006 й.
3. Aripov M., Begalov B., Begimqulov U., Mamarajabov M. Axborot texnologiyalari. Toshkent: Noshir, 2009. -368 b.
4. G’ulomov S.S., Begalov B.A. Informatika va axborot texnologiyalari. Toshkent, Fan, 2010,-686c.

Қўшимча адабиётлар

1. Меньшиков Ф. В. Олимпиадные задачи по программированию. - СПб.: Питер, 2006. - 315 с.
2. Кнут Д. Искусство программирования. Том 1-4., СПб. Вильямс 2007.
3. Холзнер С. Visual C++ 6. Учебный курс. — СПб.: Питер, 2007. - 570 с.
4. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. –СПб: ООО «ДиаСофтЮП», 2003.-560с.
5. Культин Н. Б. C/C++ в задачах и примерах. — СПб.: БХВ-Петербург, 2005. -288 с.
6. Подбельский В. В., Фомин С. С. Программирование на языке Си: Учеб. пособие. - 2-е доп. изд. - М.: Финансы и статистика, 2004. - 600 с
7. Долинский М. С. Решение сложных и олимпиадных задач по программированию: Учебное пособие. — СПб.: Питер, 2006. — 366 с.
8. Павловская Т.С., Щупак Ю.С. C/C++. Структурное программирование. Практикум. -СПб.: Питер, 2007-240с
9. Павловская Т.С., Щупак Ю.С. C++. Объектно- ориентированное программирование. Практикум.-СПб.: Питер, 2005-265с.
10. Романов Б.А. Практикум по программированию на C++: Учебное пособие. СПб.: BXB-Петербург, Новосибирск: Из-во НГТУ, 2006.- 432с.

Электрон дарсликлар, ўқув қўлланмалар ва Интернет ресурслар

1. www.ziyonet.uz - Ўзбекистан Республикаси ахборот-таълим портали.
2. Martijn Koster "Robots in the Web: threat or treat?". http://info.webcrawler.com/mak/projects/robots/threat-or-treat.html
3. acm.tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.
4. neerc.ifmo.ru – Дастурлаш бўйича жаҳон чемпионатининг Шимолий Шарқий Европа Минтақасини расмий сайти.

5. icpc.baylor.edu - Дастурлаш бўйича жаҳон чемпионатининг расмий сайти.
6. acm.timus.ru – дастурий ечим тўғрилигини автоматик тестловчи тизим.
7. algo.urgench-tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.
8. codeforces.com - дастурий ечим тўғрилигини автоматик тестловчи тизим.

МАЪРУЗА МАТНИ

1-мавзу. Тил лексик асослари

Режа:

1. Алфавит ва хизматчи сўзлар.
2. Ўзгарувчилар.
3. Константалар.
4. Амаллар.
5. C++ тилида дастур тузилиши.
6. Маълумотларни киритиш ва чиқариш.

Таянч иборалар: тил, тип, хизматчи сўзлар, ўзгарувчи, константа, дастурлаш, сатрлар, идентиифкатор, амал, литерал.

1.1. Алфавит ва хизматчи сўзлар

Алфавит. С тили алфавитига қуйидаги символлар киради.

- Катта ва кичик лотин алфавити ҳарфлари (A,B,..,Z,a,b,…,z)
- Рақамлар: 0,1,2,3,4,5,6,7,8,9
- Махсус символлар: “, {} | [] () + - / % \ ; ‘ . : ? < = > _ ! & * # ~ ^
- Кўринмайдиган символлар ("умумлашган бўшлиқ символлари"). Лексемаларни ўзаро ажратиш учун ишлатиладиган символлар (мисол учун бўшлиқ, табуляция, янги қаторга ўтиш белгилари).

Изоҳларда, сатрларда ва символли константаларда бошқа литераллар, масалан рус ҳарфлари ишлатилиши мумкин.

C++ тилида олти хил турдаги лексемалар ишлатилади: эркин танланадиган ва ишлатиладиган идентификаторлар, хизматчи сўзлар, константалар (константа сатрлар), амаллар (амаллар белгилари) ва ажратувчи белгилар.

Идентификатор. Идентификаторлар лотин ҳарфлари, остки чизиқ белгиси ва сонлар кетма-кетлигидан иборат бўлади. Идентификатор лотин ҳарфидан ёки остки чизиқ белгисидан бошланиши лозим.

Мисол учун:

A1, _MAX, adress_01, RIM, rim

Катта ва кичик ҳарфлар фарқланади, шунинг учун охирги икки идентификатор бир-биридан фарқ қилади.

Borland компиляторларидан фойдаланилганда номнинг биринчи 32 ҳарфи, баъзи компиляторларда 8 та ҳарфни инобатга олади. Бу ҳолда NUMBER_OF_TEST ва NUMBER_OF_ROOM идентификаторлари бир биридан фарқ қилмайди.

Хизматчи сўзлар. Тилда ишлатилувчи, яъни дастурчи томонидан ўзгарувчилар номлари сифатида ишлатиш мумкин бўлмаган идентификаторлар хизматчи сўзлар дейилади.

C++ тилида қуйидаги хизматчи сўзлар мавжуд:

Типлар номлари: char, short, int, unsigned, long, float, double.

Операторлар номлари: if, else, switch, case, while, do, for, default, break, continue, goto.

Хотира турлари: auto, register, static, extern.

Типлар билан ишлаш: typedef, sizeof.

Структура: struct,union.

Чиқиш: return, entry.

1.2. Ўзгарувчилар

Ўзгарувчилар объект сифатида. C++ тилининг асосий тушунчаларидан бири номланган хотира қисми – объект тушунчасидир. Объектнинг хусусий холи бу ўзгарувчидир. Ўзгарувчига қиймат берилганда унга ажратилган хотира қисмига шу қиймат коди ёзилади. Ўзгарувчи қийматига номи орқали мурожаат қилиш мумкин, хотира қисмига эса фақат адреси орқали мурожаат қилинади. Ўзгарувчи номи бу эркин киритиладиган идентификатордир. Ўзгарувчи номи сифатида хизматчи сўзларни ишлатиш мумкин эмас.

Ўзгарувчиларни таърифлаш. C тилида ўзгарувчини аниқлаш учун компьютерга унинг типи (масалан, int, char ёки float) ҳамда исми ҳақида маълумот берилади. Бу ахборот асосида компиляторга ўзгарувчи учун қанча жой ажратиш лозим ва бу ўзгарувчида қандай турдаги қиймат сақланиши мумкинлиги ҳақида маълумот аниқ бўлади. Ўзгарувчи номи идентификатор бўлиб, хизматчи сўзлардан фарқли бўлиши керак.

Ҳар бир ячейка бир байт ўлчовга эга. Агар ўзгарувчи учун кўрсатилган тип 4 байтни талаб қилса, унинг учун тўртта ячейка ажратилади. Айнан ўзгарувчини типига мувофиқ равишда компилятор бу ўзгарувчи учун қанча жой ажратиш кераклигини аниқлайди.

Компютерда қийматларни ифодалаш учун битлар ва байтлар қўлланилади ва хотира байтларда ҳисобланади.

Ўзгарувчилар типлари. Ўзгарувчиларнинг қуйидаги типлари мавжуд:

char – битта символ;
long char – узун символ;
int – бутун сон;
short ёки **short int** – қисқа бутун сон;
long ёки **long int** – узун бутун сон;
float ҳақиқий сон;
long float ёки **double** – иккиланган ҳақиқий сон;
long double – узун иккиланган ҳақиқий сон.

Бутун сонлар таърифланганда кўрилган типлар олдига unsigned (ишорасиз) таърифи қўшилиши мумкин. Бу таъриф қўшилган бутун сонлар устида амаллар mod 2^n арифметикасига асосланган дир. Бу ерда n сони int типи хотирада эгалловчи разрядлар сонидир. Агар ишорасиз k сони узунлиги int сони разрядлар сонидан узун бўлса, бу сон қиймати k mod 2^n га тенг бўлади. Ишорали, яъни signed типидаги сонларнинг энг катта разряди сон

ишорасини кўрсатиш учун ишлатилса unsigned (ишорасиз) типдаги сонларда бу разряд сонни тасвирлаш учун ишлатилади.

Ўзгарувчиларни дастурнинг ихтиёрий қисмида таърифлаш ёки қайта таърифлаш мумкин.

Мисол учун:

int a, b1, ac; ёки

int a;

int b1;

int ac;

Ўзгарувчилар таърифланганда уларнинг қийматлари аниқланмаган бўлади. Лекин ўзгарувчиларни таърифлашда инициализация яъни бошланғич қийматларини кўрсатиш мумкин.

Мисол учун:

int i = 0;

char c = 'k';

Typedef таърифловчиси янги типларни киритишга имкон беради.

Мисол учун янги COD типини киритиш:

typedef unsigned char COD;

COD simbol;

Бутун сонлар ўлчами. Бир хил типдаги ўзгарувчилар учун турли компютерларда хотирадан турли ҳажмдаги жой ажратилиши мумкин. Лекин битта компютерда бир хил типдаги иккита ўзгарувчи бир хил миқдорда жой эгаллайди.

Масалан, char типли ўзгарувчи бир байт ҳажмни эгаллайди. Кўпгина компьютерларда short int (қисқа бутун) типи икки байт, long int типи эса 4 байт жой эгаллайди. Бутун қийматлар ўлчовини компьютер системаси ва ишлатиладиган компилятор аниқлайди. 32 – разрядли компьютерларда бутун ўзгарувчилар 4 байт жой эгаллайди.

1.3. Константалар

Константалар турлари. Константа бу ўзгартириш мумкин бўлмаган қийматдир. С тилида беш турдаги константалар ишлатилиши мумкин: символлар, бутун сонлар, ҳақиқий сонлар, сановчи константалар ва нуль кўрсаткич.

Белгили ўзгармаслар. Белгили ўзгармаслар одатда бир байт жойни эгаллайди ва бу 256 хил белгини сақлаш учун етарлидир. Char типи қийматларини 0..255 сонлар тўпламига ёки ASCII белгилар тўпламига интерпретация қилиш мумкин.

ASCII белгилари деганда компьютерларда қўлланиладиган стандарт белгилар тўплами тушунилади. ASCII - бу American Standard Code for Information Interchange (Американинг ахборот алмашиниши учун стандарт коди) деган маънони англатади.

Мисол учун ‘x’,’*’,’\012’,’\0’,’\n’- битта символли константа; ‘dd’,’\n\t’,’\x07\x07’ икки символли константалар.

С компиляторида текстларни форматловчи бир нечта махсус белгилардан фойдаланилади. (Улардан энг кўп тарқалгани жадвалда келтирилган).

Махсус белгилар ахборотларни экранга, файлга ва бошқа чиқариш қурилмаларига чиқаришда форматлаш учун қўлланилади.

Махсус '\' символидан бошланган символлар эскейп символлар дейилади. Символли константа қиймати символнинг компьютерда қабул қилинган сонли кодига тенгдир.

ESC (эскейп) символлар жадвали:

Ёзилиши	Ички коди	Символи(номи)	Маъноси
\a	0x07	bel (audible bell)	Товуш сигнали
\b	0x08	bs (bascspase)	Бир қадам қайтиш
\f	0x0C	ff (form feed)	Саҳифани ўтказиш
\n	0x0A	lf (line feed)	Қаторни ўтказиш
\r	0x0D	cr (carriage return)	Кареткани қайтариш
\t	0x09	ht (horizontal tab)	Горизонтал табуляция
\v	0x0B	vt (vertical tab)	Вертикал табуляция
\\	0x5C	\ (bacslash)	Тескари чизиқ
\'	0x27	‘ (single out)	Апостриф (оддий қавс)
\"	0x22	" (double quote)	Иккилик қавс
\?	0x3F	? (question mark)	Савол белгиси
\000	000	ихтиёрий (octal number)	Символ саккизлик коди
\xhh	0xhh	ихтиёрий (hex number)	Символ ўн олтилик коди

Маълумотларнинг бутун сон тури. Бутун сонлар ўнлик, саккизлик ёки ўн олтилик саноқ системаларида берилиши мумкин.

Ўнлик саноқ системасида бутун сонлар 0-9 рақамлари кетма-кетлигидан иборат бўлиб, биринчи рақами 0 бўлиши керак эмас.

Саккизлик саноқ системасида бутун сонлар 0 билан бошланувчи 0-7 рақамларидан иборат кетма-кетликдир.

Ўн олтилик саноқ системасида бутун сон 0x ёки 0X билан бошланувчи 0-9 рақамлари ва a-f ёки A-F ҳарфларидан иборат кетма-кетликдир.

Масалан, 15 ва 22 ўнлик сонлари саккизликда 017 ва 026, ўн олтиликда 0xF ва 0x16 шаклда тасвирланади.

Маълумотларнинг узун бутун сон тури.

Охирига l ёки L ҳарфлари қўйилган ўнлик, саккизлик ёки ўн олтилик бутун сон.

Маълумотларнинг ишорасиз (unsigned) бутун сон тури:

Охирига u ёки U ҳарфлари қўйилган ўнлик, саккизлик ёки ўн олтилик оддий ёки узун бутун сон.

Маълумотларнинг ҳақиқий сон тури. Маълумотларнинг ҳақиқий сон тури олти қисмдан иборат бўлиши мумкин: бутун қисм, нуқта, каср қисм, e ёки E белгиси, ўнлик даража ва F ёки f суффикслари.

Масалан : 66., .0, .12, 3.14F, 1.12e-12.

Маълумотларнинг узун ҳақиқий сон тури:

Охирига L ёки l суффикслари қўйилган ҳақиқий сон.

Масалан: 2E+6L;

Сановчи константа. Сановчи константалар enum хизматчи сўзи ёрдамида киритилиб, int типидаги сонларга қулай сўзларни мос қўйиш учун ишлатилади.

Мисол учун:

enum{one = 1,two = 2,three = 3};

Агар сон қийматлари кўрсатилмаган бўлса энг чапки сўзга 0 қиймати берилиб қолганларига тартиб бўйича ўсувчи сонлар мос қўйилади:

enum{zero,one,two};

Бу мисолда автоматик равишда константалар қуйидаги қийматларни қабул килади:

zero = 0, one = 1, two = 2;

Константалар аралаш кўринишда киритилиши ҳам мумкин:

enum(zero,one,for = 4,five,seeks}.

Бу мисолда автоматик равишда константалар қуйидаги қийматларни қабул килади:

zero = 0, one = 1, for = 4;five = 5,seeks = 6;

Яна бир мисол:

Enum BOOLEAN {NO, YES};

Константалар қийматлари:

NO = 0, YES = 1;

Нуль кўрсаткич. NULL- кўрсаткич ягона арифметик бўлмаган константадир. Конкрет реализацияларда null кўрсаткич 0 ёки 0L ёки номланган константа NULL орқали тасвирланиши мумкин. Шуни айтиш лозимки, бу константа қиймати 0 бўлиши ёки ‘0’ символи кодига мос келиши шарт эмас.

Мантиқий константа. Мантиқий константалар true(рост) ва false(ёлғон) қийматлардан иборат. С тилида бутун сонлар ва ифодалар мантиқий константалар сифатида қаралади. Ички кўриниши false – 0, ихтиёрий бошқа қиймат true деб қаралади.

Сатрли константа. Сатрли константалар С тили константаларига кирмайди, балки лексемалари алоҳида типи ҳисобланади. Шунинг учун адабиётларда сатрли константалар сатрли лексемалар деб ҳам аталади.

Сатрли константа бу иккилик қавсларга олинган ихтиёрий символлар кетма-кетлигидир. Мисол учун "Мен сатрли константаман".

Сатрлар орасига эскейп символлар ҳам кириши мумкин. Бу символлар олдига \ белгиси қўйилади. Мисол учун :

"\n Бу сатр \n уч қаторга \n жойлашади".

Сатр символлари хотирада кетма-кет жойлаштирилади ва ҳар бир сатрли константа охирига автоматик равишда компилятор томонидан '\0' символи қўшилади. Шундай сатрнинг хотирадаги ҳажми символлар сони +1 байтга тенгдир.

Кетма-кет келган ва бўшлиқ, табуляция ёки сатр охири белгиси билан ажратилган сатрлар компиляция даврида битта сатрга айлантирилади. Мисол учун:

"Салом" "Тошкент"

сатрлари битта сатр деб қаралади.

"Салом Тошкент"

Бу қоидага бир неча қаторга ёзилган сатрлар ҳам бўйсунади. Мисол учун:

"Ўзбекистонга "

"баҳор "

"келди"

қаторлари битта қаторга мос:

"Ўзбекистонга баҳор келди"

Агар сатрда '\' белгиси учраса ва бу белгидан сўнг то '\n' сатр охири белгисигача бўшлиқ белгиси келса бу бўшлиқ белгилари '\' ва '\n' белгиси билан бирга сатрдан ўчирилади. Сатрнинг ўзи кейинги сатрда келган сатр билан қўшилади.

"Ўзбекистонга \ "

" баҳор\

" келди"

қаторлари битта қаторга мос:

"Ўзбекистонга баҳор келди"

Номланган константалар. С тилида ўзгарувчилардан ташқари номланган константалар киритилиши мумкин. Бу константалар қийматларини дастурда ўзгартириш мумкин эмас. Константалар номлари дастурчи томонидан киритилган ва хизматчи сўзлардан фарқли бўлган идентификаторлар бўлиши мумкин. Одатда ном сифатида катта лотин ҳарфлари ва остига чизиш белгилари комбинациясидан иборат идентификаторлар ишлатилади. Номланган константалар қуйидаги шаклда киритилади:

const тип константа_номи = константа_қиймати.

Мисол учун:

const double EULER = 2.718282;

const long M = 99999999;

const R = 765;

Охирги мисолда константа типи кўрсатилмаган, бу константа int типига тегишли деб ҳисобланади.

1.4. Амаллар

Арифметик амаллар. Амаллар одатда унар, яъни битта операндга қўлланиладиган амалларга ва бинар, яъни икки операндга қўлланиладиган амалларга ажратилади.

Бинар амаллар аддитив яъни + қўшиш ва – айириш амалларига, ҳамда мултипликатив, яъни * кўпайтириш, / бўлиш ва % модуль олиш амалларига ажратилади.

Бутун сонни бутун сонга бўлганда натижа бутун сонгача яхлитланади. Мисол учун, 20/3 = 6; (-20)/3 = -6; 20/(-3) = -6.

Модуль амали бутун сонни бутун сонга бўлишдан ҳосил бўладиган қолдиққа тенгдир. Агар модуль амали мусбат операндларга қўлланилса, натижа ҳам мусбат бўлади, акс ҳолда натижа ишораси компиляторга боғлиқдир.

Унар амалларга ишорани ўзгартирувчи унар минус – ва унар плюс + амаллари киради. Бундан ташқари инкремент ++ ва декремент -- амаллари ҳам унар амалларга киради.

Инкремент ++ унар амали қийматни 1 га оширишни кўрсатади. Амални префикс, яъни ++i кўринишда ишлатиш олдин ўзгарувчи қийматини ошириб, сўнгра фойдаланиш лозимлигини, постфикс эса i++ кўринишда ишлатиш олдин ўзгарувчи қийматидан фойдаланиб, сўнгра ошириш кераклигини кўрсатади. Мисол учун, i нинг қиймати 2 га тенг бўлсин, у ҳолда 3+(++i) ифода қиймати 6 га, 3+i++ ифода қиймати 5 га тенг бўлади. Иккала ҳолда ҳам i нинг қиймати 3 га тенг бўлади.

Декремент -- унар амали қийматни 1 га камайтиришни кўрсатади. Бу амал ҳам префикс ва постфикс кўринишда ишлатилиши мумкин. Бу икки амални фақат ўзгарувчиларга қўллаш мумкин.

Амаллар устиворлиги. Мураккаб ифодаларда қайси амал биринчи навбатда бажарилиши оператор приоритетига боғлиқ.

Масалан: x = 5+3*8.

Кўпайтириш қўшишга нисбатан юқорироқ приоритетга эга. Шунинг учун бу ифода қиймати 29 га тенг бўлади.

Агарда иккита математик ифоданинг приоритети тенг бўлса, улар чапдан ўнгга қараб кетма-кет бажарилади.

Масалан: x = 5+3+8*9+6*4.

Бу ифодада биринчи кўпайтириш амаллари чапдан ўнгга қараб бажарилади 8*9 = 72 ва 6*4 = 24. Кейин қўшиш амаллари бажарилади. Натижада x = 104 қийматга эга бўлади.

Лекин, барча амаллар ҳам бу тартибга амал қилмайди. Масалан, ўзлаштириш амали ўнгдан чапга қараб бажарилади.

Аддитив амалларининг устиворлиги мультипликатив амалларининг устиворлигидан пастрокдир.

Унар амалларнинг устиворлиги бинар амаллардан юқоридир.

Разрядли амаллар. Разрядли амаллар натижаси бутун сонларни иккилик кўринишларининг ҳар бир разрядига мос мантиқий амалларни қўллашдан ҳосил бўлади. Масалан, 5 коди 101 га тенг ва 6 коди 110 га тенг.

6&5 қиймати 4 га, яъни 100 га тенг.

6|5 қиймати 7 га, яъни 111 га тенг.

6^5 қиймати 3 га, яъни 011 га тенг.

~6 қиймати 4 га, яъни 010 га тенг.

Бу мисолларда амаллар устиворлиги ошиб бориши тартибида берилгандир.

Бу амаллардан ташқари M<<N чапга разрядли силжитиш ва M>>N ўнгга разрядли силжитиш амаллари қўлланилади. Силжитиш M бутун соннинг разрядли кўринишига қўлланилади. N нечта позицияга силжитиш кераклигини кўрсатади.

Чапга N позицияга суриш бу операнд қийматини иккининг N чи даражасига кўпайтиришга мос келади. Мисол учун 5<<2 = 20. Бу амалнинг битли кўриниши: 101<<2 = 10100.

Агар операнд мусбат бўлса, N позицияга ўнгга суриш чап операндни иккининг N чи даражасига бўлиб каср қисмини ташлаб юборишга мосдир. Мисол учун 5>>2 = 1. Бу амалнинг битли кўриниши 101>>2 = 001 = 1. Агарда операнд қиймати манфий бўлса икки вариант мавжуддир: арифметик силжитишда бўшатилаётган разрядлар ишора разряди қиймати билан тўлдирилади, мантиқий силжитишда бўшатилаётган разрядлар нуллар билан тўлдирилади.

Разрядли суриш амалларининг устиворлиги ўзаро тенг, разрядли инкор амалидан паст, қолган разрядли амаллардан юқоридир. Разрядли инкор амали унар амалга қолган амаллар бинар амалларга киради.

Нисбат амаллари. Нисбат амаллари қийматлари 1 га тенг агар нисбат бажарилса ва аксинча 0 га тенгдир. Нисбат амаллари арифметик типдаги операндларга ёки кўрсаткичларга қўлланилади.

Мисоллар:

1! = 0 қиймати 1 га тенг;

1 == 0 қиймати 0 га тенг;

3> = 3 қиймати 1 га тенг;

3>3 қиймати 0 га тенг;

2< = 2 қиймати 1 га тенг;

2<2 қиймати 0 га тенг;

Катта >, кичик <, катта ёки тенг > = , кичик ёки тенг < = амалларининг устиворлиги бир хилдир.

Тенг = = ва тенг эмас ! = амалларининг устиворлиги ўзаро тенг ва қолган амаллардан пастдир.

Мантиқий амаллар. C тилида мантиқий тип йўқ. Шунинг учун мантиқий амаллар бутун сонларга қўлланади. Бу амалларнинг натижалари қуйидагича аниқланади:

x||y амали 1 га тенг агар x>0 ёки y>0 бўлса, аксинча 0 га тенг

x&&y амали 1 га тенг агар x>0 ва y>0 бўлса, аксинча 0 га тенг

!x амали 1 га тенг агар x>0 бўлса, аксинча 0 га тенг

Бу мисолларда амаллар устиворлиги ошиб бориш тартибида берилгандир.

Инкор ! амали унар қолганлари бинар амаллардир.

Қиймат бериш амали. Қиймат бериш амали = бинар амал бўлиб чап операнди одатда ўзгарувчи ўнг операнди эса ифодага тенг бўлади. Мисол учун

z = 4.7+3.34

Бу қиймати 8.04 га тенг ифодадир. Бу қиймат Z ўзгарувчига ҳам берилади.

Бу ифода охирига нуқта вергуль (;) белгиси қўйилганда операторга айланади.

z = 4.7+3.34

Битта ифодада бир неча қиймат бериш амаллари қўлланилиши мумкин. Мисол учун:

c = y = f = 4.2+2.8;

Бундан ташқари С тилида мураккаб қиймат бериш амали мавжуд бўлиб, умумий кўриниши қуйидагичадир:

Ўзгарувчи_номи **амал** = ифода;

Бу ерда **амал** қуйидаги амаллардан бири *,/,%,+,-, &,^,|, <<,>>.

Мисол учун:

x+ = 4 ифода x = x+4 ифодага эквивалентдир;

x* = a ифода x = x*a ифодага эквивалентдир;

x/ = a+b ифода x = x/(a+b) ифодага эквивалентдир;

x>> = 4 ифода x = x>>4 ифодага эквивалентдир;

Имло белгилари амал сифатида. С тилида баъзи бир имло белгилари ҳам амал сифатида ишлатилиши мумкин. Бу белгилар оддий () ва квадрат [] қавслардир. Оддий қавслар бинар амал деб қаралиб ифодаларга ёки функцияга мурожаат қилишда фойдаланилади. Функцияга мурожаат қилиш қуйидаги шаклда амлга оширилади:

<функция номи> (<аргументлар рўйхати>). Мисол учун sin(x) ёки max(a,b).

Квадрат қавслардан массивларга мурожаат қилишда фойдаланилади. Бу мурожаат қуйидагича амалга оширилади:

<массив номи>[<индекс>]. Мисол учун a[5] ёки b[n][m].

Вергуль символини ажратувчи белги сифатида ҳам амал сифатида ҳам қараш мумкин. Вергуль билан ажратилган амаллар кетма-кетлиги бир амал деб қаралиб, чапдан ўнгга ҳисобланади ва охирги ифода қиймати натижа деб қаралади. Мисол учун:

d = 4,d+2 амали натижаси 8 га тенг.

Шартли амал. Шартли амал тернар амал дейилади ва учта операнддан иборат бўлади:

<1-ифода>?<2-ифода>:<3-ифода>

Шартли амал бажарилганда аввал 1- ифода ҳисобланади. Агар 1-ифода қиймати 0 дан фарқли бўлса 2- ифода ҳисобланади ва қиймати натижа

сифатида қабул қилинади, акс ҳолда 3-ифода ҳисобланади ва қиймати натижа сифатида қабул қилинади.

Мисол учун модульни ҳисоблаш: x<0?-x:x ёки иккита сон кичигини ҳисоблаш a<b?a:b.

Шуни айтиш лозимки шартли ифодадан ҳар қандай ифода сифатида фойдаланиш мумкин. Агар F FLOAT типга,а N – INT типга тегишли бўлса,

(N > 0) ? F : N ифода N мусбат ёки манфийлигидан қатъий назар DOUBLE типига тегишли бўлади.

Шартли ифодада биринчи ифодани қавсга олиш шарт эмас.

Амаллар устиворлиги жадвали

Ранг	Амаллар	Йўналиш
1	() [] -> :: .	Чапдан ўнгга
2	! ~ + - ++ -- & * (тип) sizeof new delete тип()	Ўнгдан чапга
3	. * ->*	Чапдан ўнгга
4	* / % (мультипликатив бинар амаллар)	Чапдан ўнгга
5	+ - (аддитив бинар амаллар)	Чапдан ўнгга
6	<< >>	Чапдан ўнгга
7	< < = > = >	Чапдан ўнгга
8	= ! =	Чапдан ўнгга
9	&	Чапдан ўнгга
10	^	Чапдан ўнгга
11	|	Чапдан ўнгга
12	&&	Чапдан ўнгга
13	||	Чапдан ўнгга
14	?:(шартли амал)	Ўнгдан чапга
15	= * = / = % = + = - = & = ^ = | = << = >> =	Ўнгдан чапга
16	, (вергуль амали)	Чапдан ўнгга

1.5. Типлар билан ишлаш

Типларни келтириш. Типларни келтириш (type casting) маълум типдаги ўзгарувчи бошқа типдаги қиймат қабул қилганда фойдаланилади. Баъзи типлар учун келтириш автоматик равишда бажарилади. Автоматик типларни келтириш ўзгарувчи типи ҳажми қийматни сақлашга етарли бўлганда бажарилади. Бу жараён кенгайтириш *(widening)* ёки юксалтириш *(promotion)* деб аталади, чунки, кичик разрядли тип катта разрядли типга кенгайтирилади. Бу ҳолда типларни автоматик келтириш хавфсиз деб аталади. Масалан int типи char типидаги қийматни сақлашга етарли, шунинг учун типларни келтириш талаб қилинмайди. Тескари жараён торайтириш *(narrowing)* деб аталади, чунки қийматни ўзгартириш талаб этилади. Бу ҳолда типларни автоматик келтириш хавфли деб аталади. Масалан ҳақиқий типни бутун типга келтирилганда каср қисм ташлаб юборилади.

Амалларда типларни автоматик келтириш. Бинар арифметик амаллар бажарилганда типларни келтириш қуйидаги қоидалар асосида амалга оширилади:

short ва char типлари int типига келтирилади;

Агар операндлардан бири long типига тегишли бўлса иккинчи операнд ҳам long типига келтирилади ва натижа ҳам long типига тегишли бўлади;

Агар операндлардан бири float типига тегишли бўлса иккинчи операнд ҳам float типига келтирилади ва натижа ҳам float типига тегишли бўлади;

Агар операндлардан бири double типига тегишли бўлса иккинчи операнд ҳам double типига келтирилади ва натижа ҳам double типига тегишли бўлади;

Агар операндлардан бири long double типига тегишли бўлса иккинчи операнд ҳам long double типига келтирилади ва натижа ҳам long double типига тегишли бўлади;

Ифодаларда типларни автоматик келтириш. Агар ифодада short ва int типидаги ўзгарувчилар ишлатилса, бутун ифода типи int га кўтарилади. Агар ифодада бирор ўзгарувчи типи— long бўлса, бутун ифода типи long типга кўтарилади. Кўзда тутилганидек ҳамма бутун константалар int типига эга деб қаралади. Ҳамма бутун константалар охирида L ёки 1 символи турган бўлса, long типига эга.

Агар ифода float типидаги операндга эга бўлса, бутун ифода float типига кўтарилади. Агар бирор операнд double типига эга бўлса, бутун ифода типи double типига кўтарилади.

Типлар билан ишловчи амаллар. Типларни ўзгартириш амали қуйидаги кўринишга эга:

(тип_номи) операнд;

Бу амал операндлар қийматини кўрсатилган типга келтириш учун ишлатилади. Операнд сифатида константа, ўзгарувчи ёки қавсларга олинган ифода келиши мумкин. Мисол учун (long)6 амали константа қийматини ўзгартирмаган ҳолда оператив хотирада эгаллаган байтлар сонини оширади. Бу мисолда константа типи ўзгармаган бўлса, (double)6 ёки (float)6 амали константа ички кўринишини ҳам ўзгартиради. Катта бутун сонлар ҳақиқий типга келтирилганда соннинг аниқлиги йўқолиши мумкин.

Масалан:

```
int x = 1.7+1.8;
int y = (int)1.7+(int)1.8;
```

Бу амаллар бажарилиши натижасида x ўзгарувчи қиймати 3 га y ўзгарувчи қиймати иккига тенг бўлади.

sizeof амали операнд сифатида кўрсатилган объектнинг байтларда хотирадаги ҳажмини ҳисоблаш учун ишлатилади. Бу амалнинг икки кўриниши мавжуд:

```
sizeof ифода
sizeof (тип)
```

Шуни таъкидлаб ўтиш лозимки sizeof функцияси препроцессор қайта ишлаш жараёнида бажарилади, шунинг учун дастур бажарилиш жараёнида вақт талаб этмайди.

Мисол учун:
sizeof 3.14 = 8
sizeof 3.14f = 4
sizeof 3.14L = 10
sizeof(char) = 1
sizeof(double) = 8.

1.6. C++ тилида дастур тузилиши

Содда дастур тузилиши. Дастур препроцессор командалари ва бир неча функциялардан иборат бўлиши мумкин. Бу функциялар орасида main номли асосий функция бўлиши шарт. Агар асосий функциядан бошқа функциялар ишлатилмаса дастур қуйидаги кўринишда тузилади:

Препроцессор_командалари
void main()
{
Дастур танаси.
}

Препроцессор директивалари компиляция жараёнидан олдин препроцессор томонидан бажарилади. Натижада дастур матни препроцессор директивалари асосида ўзгартирилади.

Препроцессор командаларидан иккитасини кўриб чиқамиз.

#include <файл_номи> Бу директива стандарт библиотекалардаги функцияларни дастурга жойлаш учун фойдаланилади.

#define <алмаштирувчи ифода> <алмашинувчи ифода>

Бу директива бажарилганда дастур матнидаги алмаштирувчи ифодалар алмашинувчи ифодаларга алмаштирилади.

Мисол тариқасида С тилида тузилган биринчи дастурни келтирамиз:

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

Бу дастур экранга **Salom, Dunyo!** жумласини чиқаради.

Алмаштирувчи define директиваси ёрдамида бу дастурни қуйидагича ёзиш мумкин:

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

Алмаштирувчи define директивасидан номланган константалар киритиш учун фойдаланиш мумкиндир.

Мисол учун:

#define ZERO 0

Агар дастурда қуйидаги матн мавжуд бўлсин:

int d = ZERO;

Препроцессор бу матнда ҳар бир ZERO константани унинг қиймати билан алмаштиради, ва натижада қуйидаги матн ҳосил бўлади.

int d = 0;

1.7. Маълумотларни киритиш ва чиқариш

Форматли чиқариш – printf. Чиқариш printf функцияси кўрсатилган параметрларни стандарт оқимга чиқариш учун ишлатилади. Стандарт оқим тушунчаси кейинги бобларда ёритилади. Хозирча стандарт оқим сифатида монитор тушунилиши етарлидир.

Функция **stdio.h** модулида жойлашган бўлиб, умумий кўриниши қуйидагичадир:

printf(control,arg1,arg2,…)

Бунда control бошқарувчи қатор деб аталиб икки турдаги символлардан иборат бўлади: оддий чиқарилувчи символлар ва навбатдаги параметрни ўзгартириб чиқарувчи спецификациялар.

Ҳар бир спецификация % символидан бошланиб ўзгартириш турини кўрсатувчи символ билан тугайди.

Ўзгартириш символлари қуйидагилардан иборат.

Бутун сонлар учун:

d– параметр ишорали ўнлик бутун сонга айлантирилади.

u - параметр ишорасиз ўнлик бутун сонга айлантирилади.

o – параметр ишорасиз ва биринчи рақами 0 бўлмаган саккизлик сонга айлантирилади.

x – параметр ишорасиз ва 0x белгисиз ўн олтилик сонга айлантирилади.

X – параметр худди x каби. Фақат ҳарф билан кўрсатилувчи рақамлар катта ҳарф яъни A,B,C,D,E,F сифатида ёзилади.

Хақиқий сонлар учун:

e – параметр float ёки double типидаги сон деб қаралади ва ишорали m.nnnnnne+-xx кўринишидаги ўнлик сонга келтирилади.

E – параметр худди e каби. Фақат мантисса белгиси катта ҳарф яъни E сифатида ёзилади.

f - параметр float ёки double типидаги сон деб қаралади ва ишорали m.nnnnnn кўринишидаги ўнлик сонга келтирилади.

g – параметр берилган сон қиймати ва аниқлиги учун энг ихчам %e ёки %f танлайди.

G – параметр худди g каби. Фақат мантисса белгиси катта ҳарф яъни E сифатида ёзилади.

Символ ва сатр учун:

c – параметр битта символ деб қаралади.

s – параметр сатр символлар нўлинчи символ учрамагунча ёки кўрсатилган сондаги символлар босилади.

Мисол:

```
#include <stdio.h>
int main()
{
int num = -27; int number = 27; float f = 123.456;
char r = 'a'; char str[4] = "abc";
printf("%d\n",    num);      /*    -27       */
printf("%u\n",    number);   /*    27        */
printf("%o\n",    number);   /*    33        */
printf("%x\n",    number);   /*    lb     */
printf( "%f\n",   f);        /*  123.456001  */
printf("%e\n",    f);        /*    1.23456e+02     */
printf("%E\n",    f);        /*    1.23456E+02     */
printf("%c\n",    r);        /*    a         */
printf("%s\n", str);         /*  abc       */
return 0;
}
```

Процент % белгиси ва ўзгартириш символи орасига қуйидаги символларни қўйиш мумкин.

Чиқарилаётган аргумент чапга текислаш лозимлигини кўрсатувчи минус белгиси.

Майдон минимал узунлигини кўрсатувчи рақамлар қатори.

Майдон узунлигини кейинги рақамлар қаторидан ажратувчи нуқта.

Бирор қатордан қанча символ ажратиб олиш лозимлигини ҳамда float ёки double типидаги сонларда нуқтадан кейин қанча каср рақамлари босиб чиқарилишини кўрсатувчи рақамлар кетма-кетлиги.

Чиқарилаётган сон long типига тегишли эканлигини кўрсатувчи узун ўнлик маркери l.

Типлар максимал ва минимал қийматлари. Турли типлар максимал ва минимал қийматлари <LIMITS.H> файлидаги константаларда сақланади.

Қуйидаги дастурда char типидаги битлар сони ва char типи максимал ва минимал қиймати экранга чиқарилади

```
#include<stdio.h>
#include<limits.h>
int main()
{
printf("CHAR_BIT = %d\n",CHAR_BIT);
printf("CHAR_MIN = %d CHAR_MAX = %d\n",CHAR_MIN,CHAR_MAX);
return 0;
```

```
}
```

Қуйидаги дастурда short, int, long типлари максимал ва минимал қиймати экранга чиқарилади

```
#include<stdio.h>
#include<limits.h>
int main()
{
printf("SHRT_MIN = %d SHRT_MAX = %d\n",SHRT_MIN,SHRT_MAX);
printf("INT_MIN = %d INT_MAX = %d\n",INT_MIN,INT_MAX);
printf("LONG_MIN = %ld LONG_MAX = %ld\n",LONG_MIN,LONG_MAX);
return 0;
}
```

Қуйидаги дастурда unsigned char, unsigned short, unsigned int, unsigned long типлари максимал ва минимал қиймати экранга чиқарилади

```
#include<stdio.h>
#include<limits.h>
int main()
{
printf("UCHAR_MAX = %u\n",UCHAR_MAX);
printf("USHRT_MAX = %u\n",USHRT_MAX);
printf("UINT_MAX = %u\n",INT_MAX);
printf("ULONG_MAX = %ul\n",ULONG_MAX);
return 0;
}
```

Форматли киритиш Scanf. Scanf функцияси stdio.h модулида жойлашган бўлиб, умумий кўриниши қуйидагичадир:

Scanf(control, arg1, arg2,…)

Функция стандарт оқимдан символларни ўқиб бошқарувчи қатор асосида форматлаб мос параметрларга ёзиб қўяди. Параметр кўрсаткич бўлиши лозим.

Бошқарувчи қатор қуйидаги ўзгартириш спецификацияларидан иборат:

Бўшлиқ, табуляция, кейинги қаторга ўтиш символлари;

Оддий символлар (% дан ташқари) киритиш оқимидаги навбатдаги символлар билан мос келиши лозим;

% символидан бошланувчи спецификация символлари;

% символидан бошланувчи қиймат беришни таъқиқловчи * символи;

% символидан бошланувчи майдон максимал узунлигини кўрсатувчи сон;

қуйидаги спецификация символларини ишлатиш мумкин:

d – ишорали ўнли бутун сон кутилмоқда.

o – ишорали саккизлик бутун сон кутилмоқда.

x –ишорали ўн олтилик бутун сон кутилмоқда.

h - ишорасиз ўнлик сон кутилмоқда.

c – битта символ кутилмоқда.

s – сатр кутилмоқда.

f - float типидаги сон кутилмоқда. Киритилаётган соннинг бутун рақамлари ва нуқтадан сўнг каср рақамлари сони ва E ёки e белгисидан сўнг мантисса рақамлари сони кўрсатилиши мумкин.

```
#include <stdio.h>
int main(void)
{
 unsigned width, precision;
 int number = 256;
 double weight = 242.5;
 printf("What field width?\n");
 scanf("%d", &width);
 printf("The number is :%*d:\n", width, number);
 printf("Now enter a width and a precision:\n");
 scanf("%d %d", &width, &precision);
 printf("Weight = %*.*f\n", width, precision, weight);
 printf("Done!\n");
 return 0;
}
```

Локал ва глобал ўзгарувчилар. С тилида ўзгарувчи таърифи албатта блок бошида жойлашиши лозим.

Ўзгарувчи мавжудлик соҳаси деб шу ўзгарувчига ажратилган хотира мавжуд бўлган дастур қисмига айтилади. Ўзгарувчи кўриниш соҳаси деб ўзгарувчи қийматини олиш мумкин бўлган дастур қисмига айтилади. Бирор блокда таърифланган ўзгарувчи локал ўзгарувчи дейилади. Ҳар қандай блокдан ташқарида таърифланган ўзгарувчи глобал ўзгарувчи дейилади.

Локал ўзгарувчи мавжудлик ва кўриниш соҳаси таърифдан то шу таъриф жойлашган блок охиригачадир.

Ташқи блокдаги ўзгарувчи номи шу блокда жойлашган ёки шу блокдаги ички блокда ўзгарувчи номи билан бир хил бўлмаслиги керак.

Глобал ўзгарувчи мавжудлик соҳаси таърифдан то дастур охиригачадир.

Агар ички блокдаги ўзгарувчи номи глобал ўзгарувчи номи билан бир хил бўлса локал ўзгарувчи кўриниш соҳасида глобал ўзгарувчи кўринмай қолади.

Мисол:

```
#include<stdio.h>
int i = 5;
int k = 6;
int main()
{
int i = 9;
printf("%d\n",i);
printf("%d\n",k);
```

```
return 0;
}
```
Натижа:
```
9
6
```

Назорат саволлари

1. Бутун сонли ва ҳақиқий типларни қандай фарқи бор?
2. Ишорасиз unsigned типининг хоссаларини кўрсатинг.
3. Ишорасиз unsigned short int ва long int типларининг ўзаро фарқи нимада?
4. Хизматчи сўзлар ва уларнинг вазифаси нима?
5. Ўзгарувчи нима?
6. C++ да қандай типлар мавжуд?
7. Ўзгармас нима?
8. Сановчи константа нима?
9. Нуль кўрсаткич нима?
10. C++ да неча хил амал бор?
11. Разрядли амаллар нима?
12. Мантиқий амалларни санаб беринг.
13. #define нима?
14. printf ва cout орасидаги фарқни санаб ўтинг.
15. Локал ва глобал ўзгарувчилар ўртасидаги фарқ нима?
16. Сонли ва символли ўзгармаслар ўртасидаги фарқ нима?
17. Биринчи қайси функция бажарилади?
18. Символли киритиш функциялари.
19. Шартли амал умумий кўриниши.
20. Типларни келтириш қоидалари.
21. Бош main() функциясининг ўзига хос хусусияти нимадан иборат?
22. Изоҳлар бир неча қаторда ёзилиши мумкинми?

Адабиётлар рўйхати

1. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013, 488 b.
2. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. –СПб: ООО «ДиаСофтЮП», 2003.-560с.
3. Культин Н. Б. C/C++ в задачах и примерах. — СПб.: БХВ-Петербург, 2005. -288 с.
4. acm.tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.

2-мавзу. Операторлар ва функциялар

Режа:

1. Операторлар турлари.
2. Танлаш операторлари.
3. Цикл операторлари.
4. Ўтиш операторлари.
5. Фойдаланувчи функциялари
6. Рекурсия

Таянч иборалар: оператор, танлаш, цикл, ўтиш, рекурция, функция, инкремент, декремент.

2.1. Операторлар турлари

Операторлар ва блоклар. Ҳар қандай дастур функциялар кетма-кетлигидан иборат бўлади. Функциялар сарлавҳа ва функция танасидан иборат бўлади. Функция сарлавҳасига void main() ифода мисол бўла олади. Функция танаси объектлар таърифлари ва операторлардан иборат бўлади.

Ҳар қандай оператор нуқта-вергуль белгиси билан тугаши лозим. Қуйидаги ифодалар x = 0, ёки i++ операторга айланади агар улардан сўнг нуқтали вергуль келса

```
x = 0;
i++;
```

Операторлар бажарилувчи ва бажарилмайдиган операторларга ажратилади. Бажарилмайдиган оператор бу изоҳ операторидир.

Изоҳ оператори /* белгиси билан бошланиб */ белгиси билан тугайди. Бу икки символ орасида ихтиёрий жумла ёзиш мумкин. Компилятор бу жумлани текшириб ўтирмайди. Изоҳ операторидан дастурни тушунарли қилиш мақсадида изоҳлар киритиш учун фойдаланилади.

Бажарилувчи операторлар ўз навбатида маълумотларни ўзгартирувчи ва бошқарувчи операторларга ажратилади.

Бошқарувчи операторлар дастурни бошқарувчи конструкциялар деб аталади. Бу операторларга қуйидагилар киради:

Танлаш операторлари;

Цикл операторлари;

Ўтиш операторлари;

Маълумотларни ўзгартирувчи операторларга қиймат бериш операторлари ва нуқта вергуль билан туговчи ифодалар киради. Мисол учун:

```
i++;
x* = i;
i = x-4*i;
```

Қўшма операторлар. Бир неча операторлар { ва } фигурали қавслар ёрдамида қўшма операторларга ёки блокларга бирлаштирилиши мумкин. Блок ёки қўшма оператор синтаксис жиҳатдан битта операторга эквивалентдир. Блокнинг қўшма операторлардан фарқи шундаки блокда объектлар таърифлари мавжуд бўлиши мумкин.

Қуйидаги дастур қисми қўшма оператор:

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

Бу фрагмент бўлса блок:

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

2.2. Танлаш операторлари

Шартли оператор. Шартли оператор икки кўринишда ишлатилиши мумкин:

if (ифода**)**
1- оператор
else
2- оператор
ёки
if (ифода**)**
1-оператор

Шартли оператор бажарилганда аввал ифода ҳисобланади; агар қиймат рост яъни нольдан фарқли бўлса 1- оператор бажарилади. Агар қиймат ёлғон яъни ноль бўлса ва else ишлатилса 2-оператор бажарилади. Оператор else қисми ҳар доим энг яқин if га мос қўйилади.

```
if( n>0)
 if(a>b)
  Z = a;
 else
  Z = b;
```

Агар else қисмни юқори if га мос қўйиш лозим бўлса, фигурали қавслар ишлатиш лозим.

```
if( n>0) {
```

```
 if(a>b)
  z = a;
}
else
  z = b;
```

Мисол тариқасида учта берилган соннинг энг каттасини аниқлаш дастури:

```
#include<stdio.h>
int main()
{
float a,b,c,max;
scanf("%f",&a);
scanf("%f",&b);
scanf("%f",&c);
if (a>b)
if (a>c) max = a; else max = c;
else
if (b>c) max = b; else max = c;
printf("\n max = %f", max);
return 0;
}
```

Кейинги мисолда киритилган балл ва максимал балл асосида баҳо аниқланади:

```
#include<stdio.h>
int main()
{
int ball,max_ball,baho;
printf( "\n ball = ");
scanf("%d",&ball);
printf("\n max_ball = ");
scanf("%d",&max_ball);
float d = (float)ball/max_ball;
if (d>0.85) baho = 5; else
{
if (d>0.71) baho = 4; else
{
if (d>0.55) baho = 3; else baho = 2;
}
}
printf("\n baho = %d",baho);
return 0;
}
```

Калит бўйича танлаш оператори. Калит бўйича танлаш switch оператори умумий кўриниши қуйидагича:

```
switch(<ифода>) {
case <1-қиймат>:<1-оператор>
    …
break;
    …
default: <оператор>
    …
case: <n-оператор>;
 }
```

Олдин қавс ичидаги бутун ифода ҳисобланади ва унинг қиймати ҳамма вариантлар билан солиштирилади. Бирор вариантга қиймат мос келса шу вариантда кўрсатилган оператор бажарилади. Агар бирор вариант мос келмаса default орқали кўрсатилган оператор бажарилади. Узиш break оператори ишлатилмаса шартга мос келган вариантдан ташқари кейинги вариантдаги операторлар ҳам автоматик бажарилади. Қуйидаги default, break ва белгиланган вариантлар ихтиёрий тартибда келиши мумкин. Умуман default ёки break операторларини ишлатиш шарт эмас. Белгиланган операторлар бўш бўлиши ҳам мумкин.

Мисол тариқасида баҳони сон миқдорига қараб аниқлаш дастурини кўрамиз.

```
#include <stdio.h>
int main()
{
int baho;
scanf("%d", &baho);
switch(baho)
{
case 2:printf("\n yomon");break;
case 3:printf("\n o'rta");break;
case 4:printf("\n yahshi");break;
case 5:printf("\n alo");break;
default: printf("\n noto'g'ri kiritilgan");
};
return 0;
}
```

Кейинги мисолда киритилган символ унли ҳарф эканлиги аниқланади:

```
#include <stdio.h>
int main()
{
char c;
scanf("%c", &c);
switch(c)
```

```
{
case 'a':
case 'u':
case 'o':
case 'i':
printf("\n Simvol unli");break;
default: printf("\n Simvol unli emas");
};
return 0;
}
```

2.3. Цикл операторлари

Олдинги шартли while оператори. Олдинги шартли while оператори қуйидаги умумий кўринишга эгадир:

```
while(ифода)
Оператор
```

Бу оператор бажарилганда аввал ифода ҳисобланади. Агар унинг қиймати 0 дан фарқли бўлса оператор бажарилади ва ифода қайта ҳисобланади. То ифода қиймати 0 бўлмагунча цикл қайтарилади.

Агар дастурда while (1); сатр қўйилса бу дастур ҳеч қачон тугамайди.

Мисол. Берилган n гача сонлар йиғиндиси.

```
#include <stdio.h>
void main()
{
long n,i = 1,s = 0;
scanf("%d",&n);
while (i< = n )
 s+ = i++;
printf("\n s = %d",s);
}
```

Бу дастурда s+ = i++ ифода s = s+i; i = i+1 ифодаларга эквивалентдир.

Қуйидаги дастур то нуқта босилмагунча киритилган символлар ва қаторлар сони ҳисобланади:

```
#include <stdio.h>
int main()
{
int nc = 0,nl = 0;
char c;
while ((c = getchar())! = '.' )
{
++nc;
```

```
if (c = = '\n') ++nl;
};
printf("satrlar = %d simvollar = %d \n",nl,nc);
return 0;
}
```

Кейинги шартли do-while операторы. Кейинги шартли do-while оператори умумий кўриниши қуйидагича:

```
do
Оператор
while(ифода)
```

Цикл операторининг бу кўринишида аввал оператор бажарилади сўнгра ифода ҳисобланади. Агар унинг қиймати 0 дан фарқли бўлса оператор яна бажарилади ва ҳоказо. То ифода қиймати 0 бўлмагунча цикл қайтарилади.

Мисол. Берилган n гача сонлар йиғиндиси.

```
#include <stdio.h>
int main()
{
long n,i = 1,s = 0;
scanf("%d",&n);
do
s+ = i++;
while (i< = n);
printf("\n s = %d",s);
return 0;
}
```

Бу дастурнинг камчилиги шундан иборатки агар n қиймати 0 га тенг ёки манфий бўлса ҳам, цикл танаси бир марта бажарилади ва s қиймати бирга тенг бўлади.

Параметрли for операторы. Параметрли for оператори умумий кўриниши қуйидагича:

```
for( 1-ифода;2- ифода; 3-ифода)
Оператор
```

Бу оператор қуйидаги операторга мосдир.

```
1-ифода;
while(2-ифода) {
оператор
3-ифода
}
```

Мисол. Берилган n гача сонлар йиғиндиси.

```
#include <stdio.h>
int main()
{
int n;
scanf("%d",&n);
int s = 0;
for(int i = 1;i< = n; i++) s+ = i;
printf("\n%d",s);
return 0;
}
```

Циклда бир нечта счётчикни қўлланилиши. Параметрли for циклининг синтаксиси унда бир нечта ўзгарувчи - счётчикни қўлланилишига, циклни давом этишини мураккаб шартларини текширишга ва цикл счётчиклари устида кетма-кет бир нечта операцияни бажарилишига имкон беради.

Агарда бир нечта счётчикка қиймат ўзлаштирилса ёки улар ўртасида бир нечта операция бажарилса, бу ифодалар вергул билан ажратилган ҳолда кетма – кет ёзилади.

for циклида бир нечта счётчикни қўлланилиши

```
#include <stdio.h>
#include<conio.h>
int main()
{
int i,j;
for (i = 0, j = 0; i<3; i++, j++)
printf("i:%d  j:%d\n",i,j);
getch();
return 0;
}
```

Натижа:

```
i: 0          j: 0
i: 1          j: 1
i: 2      j: 2
```

2.4. Ўтиш операторлари

Узиш break оператори. Баъзи ҳолларда цикл бажарилишини ихтиёрий жойда тўхтатишга тўғри келади. Бу вазифани break оператори бажаришга имкон беради. Бу оператор дархол цикл бажарилишини тўхтатади ва бошқарувни циклдан кейинги операторларга узатади.

Мисол:

```
#include <stdio.h>
int main()
{
int n;
while(1)
{
scanf("%d",&n);
if(n == 1||n == 0) break;
}
printf("Sikl tugadi");
return 0;
}
```

Бу мисолда while(1) оператори ёрдамида чексиз цикл ҳосил қилинади. Агар 1 ёки 0 сони киритилса цикл тўхтатилади.

Қайтариш continue оператори. Цикл бажарилишига таъсир ўтказишга имкон берадиган яна бир оператор continue операторидир. Бу оператор цикл қадамини бажарилишини тўхтатиб for ва while да кўрсатилган шартли текширишга ўтказади.

Мисол:

```
#include <stdio.h>
int main()
{
int n;
for(;;)
{
scanf("%d",&n);
if(n == 1||n == 0) continue;
break;
}
printf("Sikl tugadi");
return 0;
}
```

Бу мисолда for(;;) оператори ёрдамида чексиз цикл ҳосил қилинади. Агар 1 ёки 0 сонлардан фарқли сон киритилса цикл тўхтатилади.

Ўтиш оператори goto. Ўтиш операторининг кўриниши:

goto <идентификатор>. Бу оператор идентификатор билан белгиланган операторга ўтиш кераклигини кўрсатади.

Мисол учун goto A1;…;A1:y = 5;

Структурали дастурлашда goto операторидан фойдаланмаслик маслаҳат берилади. Лекин баъзи ҳолларда ўтиш операторидан фойдаланиш дастурлашни осонлаштиради.

Мисол учун бир неча циклдан бирдан чиқиш керак бўлиб қолганда, тўғридан-тўғри break операторини қўллаб бўлмайди, чунки у фақат энг ички циклдан чиқишга имкон беради.

```
#include <stdio.h>
int main()
{
int n = 16,s = 0;
int i,j;
for(i = 1;i<5;i++)
for(j = 1;j<5;j++)
{
if(i*j>n) goto A;
s++;
}
A:printf("Sikl tugadi s = %d",s);
return 0;
}
```

2.5. Фойдаланувчи функциялари

Функцияларни таърифлаш ва уларга мурожаат қилиш. Функция таърифида функция номи, типи ва формал параметрлар рўйхати кўрсатилади. Формал параметрлар номларидан ташқари типлари ҳам кўрсатилиши шарт. Формал параметрлар рўйхати функция сигнатураси деб ҳам аталади.

Функция таърифи умумий кўриниши қуйидагичадир:

Функция типи функция номи(формал_параметрлар_таърифи)

Формал параметрларга таъриф берилганда уларнинг бошланғич қийматлари ҳам кўрсатилиши мумкин.

Функция қайтарувчи ифода қиймати функция танасида **return** <ифода> ; оператори орқали кўрсатилади.

Мисол:

```
float min(float, float b)
{
if (a<b) return a;
return b;
}
```

Функцияга мурожаат қилиш қуйидагича амалга оширилади:

Функция номи (хақиқий параметрлар рўйхати)

Хақиқий параметр ифода ҳам бўлиши мумкин. Ҳақиқий параметрлар қиймати ҳисобланиб мос формал параметрлар ўрнида ишлатилади.

Мисол учун юқоридаги функцияга қуйидагича мурожаат қилиш мумкин:

int x = 5,y = 6,z; z = min(x,y) ёки **int z = min(5,6)** ёки **int x = 5; int z = min(x,6)**

Функцияга мурожаат қилинганда ҳақиқий параметрлар типлари формал параметрлар типларига мос келмаслиги мумкин. Бу ҳолда автоматик равишда типларни келтириш бажарилади.

Функция қиймат қайтармаса типи **void** деб кўрсатилади.

Мисол учун:

```
void print()
{
printf("\n Salom!");
};
```

Бу функцияга print() шаклида мурожаат қилиш экранга Salom! ёзилишига олиб келади.

Қиймат қайтармайдиган функция танасида return оператори ишлатилиши мумкин. Бу оператор функциядан чиқишни билдиради. Масалан:

```
void print()
{
printf("\n Salom!");
return;
printf("\n Dunyo!");
}
```

Бу функцияга print() шаклида мурожаат қилиш экранга Salom! ёзилишига олиб келади, лекин **Dunyo!** сўзи ёзилмай қолади.

Қиймат қайтармайдиган функция формал параметрларга эга бўлиши мумкин.

Масалан:

```
#include <stdio.h>
void print_baho(int baho)
{
switch(baho)
{
case 2:printf("\n yomon");break;
case 3:printf("\n o'rta");break;
case 4:printf("\n yaxshi");break;
case 5:printf("\n alo");break;
default:printf("\n noto'ri kiritilgan");
};
};
int main()
{
int a;
scanf("%d",&a);
print_baho(5);
```

```
return 0;
};
```

Функция прототипи. Агар программада функция таърифи мурожаатдан кейин берилса, ёки функция бошқа файлда жойлашган бўлса, мурожаатдан олдин шу функциянинг прототипи жойлашган бўлиши керак. Прототип функция номи ва формал параметрлар типларидан иборат бўлади. Формал параметрлар номларини бериш шарт эмас.

Мисол учун y = min(a,b)+2*max(c,d) ифодани ҳисоблашни кўрамиз:

```
#include <stdio.h>
int max(int a,int b)
{
if (a<b) return b;else return a;
}
int main()
{
int a,b,c,d,y;
int min(int,int);
scanf("%d%d%d%d",&a,&b,&c,&d);
y = min(a,b)+2*max(c,d);
printf("\n %d",y);
return 0;
};
int min(int a,int b)
{
if (a<b) return a;
else return b;
}
```

Функцияга параметрлар узатиш. Функцияга параметрлар қиймат бўйича узатилади ва қуйидаги босқичлардан иборат бўлади:

1. Функция бажаришга тайёрланганда формал параметрлар учун хотирадан жой ажратилади, яъни формал параметрлар функцияларнинг ички параметрларига айлантирилади. Агар параметр типи float бўлса double типидаги объектлар ҳосил бўлади, char ва shortint бўлса int типидаги объектлар яратилади.
2. Хақиқий параметрлар сифатида ишлатилган ифодалар қийматлари ҳисобланади.
3. Хақиқий параметрлар ифодалар қийматлари формал параметрлар учун ажратилган хотира қисмларига ёзилади. Бу жараёнда float типи double типига, char ва short int типлари int типига келтирилади.
4. Функция танаси ички объектлар – параметрлар ёрдамида бажарилади ва қиймат чақирилган жойга қайтарилади.
5. Хақиқий параметрлар қийматларига функция ҳеч қандай таъсир ўтказмайди.

6. Функциядан чиқишда формал параметрлар учун ажратилган хотира қисмлари бўшатилади.

C++ тилида чақирилган функция чақирувчи функциядаги ўзгарувчи қийматини ўзгартира олмайди. У фақат ўзининг вақтинчалик нусхасини ўзгартириши мумкин холос.

Қиймат бўйича чақириш қулайлик туғдиради. Чунки функцияларда камроқ ўзгарувчиларни ишлатишга имкон беради. Мисол учун шу хусусиятни акс эттирувчи POWER функцияси вариантини келтирамиз:

```
int power(int x, int n)
{
int p;
for (p = 1; n > 0; --n)
p = p * x;
return (p);
}
```

Аргумент n вақтинчалик ўзгарувчи сифатида ишлатилади. Ундан то қиймати 0 бўлмагунча бир айрилади. Параметр n функция ичида ўзгариши функцияга мурожаат қилинган бошланғич қийматига таъсир қилмайди.

2.6. Рекурсия

Рекурсив функциялар. Рекурсив функция деб ўзига ўзи мурожаат қилувчи функцияга айтилади. Мисол учун факториални ҳисоблаш функциясини келтирамиз:

```
long fact(int k)
{
if (k<0) return 0;
if (k == 0) return 1;
return k*fact(k-1);
}
```

Манфий аргумент учун функция 0 қиймат қайтаради. Параметр 0 га тенг бўлса функция 1 қиймат қайтаради. Акс ҳолда параметр қиймати бирга камайтирилган ҳолда функциянинг ўзи чақирилади ва узатилган параметрга кўпайтирилади. Функциянинг ўз ўзини чақириш формал параметр қиймати 0 га тенг бўлганда тўхтатилади.

Кейинги мисолимизда ихтиёрий ҳақиқий соннинг бутун даражасини ҳисоблаш рекурсив функциясини келтирамиз.

```
double expo(double a, int n)
{
if (n == 0) return 1;
if (a == 0.0) return 0;
if (n>0) return a*expo(a,n-1);
```

```
if(n<0) return expo(a,n+1)/a;
}
```

Мисол учун функцияга expo(2.0,3) шаклда мурожаат қилинганда рекурсив равишда функциянинг иккинчи параметри камайган ҳолда мурожаатлар ҳосил бўлади:

expo(2.0,3),expo(2.0,2),expo(2.0,1),expo(2.0,0). Бу мурожаатларда қуйидаги кўпайтма ҳисобланади: 2.0*2.0*2.0*1 ва керакли натижа ҳосил қилинади.

Шуни кўрсатиб ўтиш керакки бу функциямизда ноаниқлик мавжуддир яъни 0.0 га тенг соннинг 0 чи даражаси 0 га тенг бўлади. Математик нуқтаи назардан бўлса бу ҳолда ноаниқлик келиб чиқади. Юқоридаги содда мисолларда рекурсиясиз итератив функциялардан фойдаланиш мақсадга мувофиқдир.

Масалан даражани ҳисоблаш функцияни қуйидагича тузиш мумкин:

```
#include <stdio.h>
double expo(double a, int n)
{
if (n == 0) return 1;
if (a == 0.0) return 0;
int k = (n>0)?n:-n;
double s = 1.0;
for(int i = 0;i<k;i++) s* = a;
if (n>0) return s; else return 1/s;
}
void main()
{
printf("%f",expo(2,-2));
};
```

Натижа

0.250000

Рекурсияга мисол сифатида сонни сатр шаклида чиқариш масаласини кўриб чиқамиз. Сон рақамлари тескари тартибда ҳосил бўлади. Биринчи усулда рақамларни массивда сақлаб сўнгра тескари тартибда чиқаришдир.

Рекурсив усулда функция ҳар бир чақириқда бош рақамлардан нусха олиш учун ўз ўзига мурожаат қилади, сўнгра охирги рақамни босиб чиқаради.

```
#include <stdio.h>
void printd(int n){
int i;
if (n < 0) {
printf("-");
n = -n;
}
```

```
if ((i = n/10)! = 0)
printd(i);
printf("%d",n % 10);
}
void main()
{
printd(123);
};
```

Бу мисолда printd (123) чақирикда биринчи функция printd n = 123 қийматга эга. У 12 қийматни иккинчи printd га узатади, бошқариш ўзига қайтганда 3 ни чиқаради.

Назорат саволлари

1. Оператор нима?
2. Тармоқланувчи операторга мисол келтиринг.
3. Танлаш операторга мисол келтиринг.
4. Цикл while операторининг умумий кўриниши.
5. Цикл do while оператори while операторидан қандай фарқ қилади?
6. Цикл for операторининг умумий кўриниши.
7. Сирпаниш деб нимага айтилади?
8. Қайтариш continue операторидан нима учун фойдаланилади?
9. Узиш break операторидан нима учун фойдаланилади?
10. Ўтиш goto оператори бошқаришни қаерга узатади?
11. Функция нима?
12. Функция прототипини эълон қилиш ва функцияни аниқлаш ўртасида қандай фарқ бор?
13. Агарда функция ҳеч қандай қиймат қайтармаса уни қандай эълон қилиш керак?
14. Функцияда локал ва глобал ўзгарувчилар ўртасидаги фарқ нима?
15. Рекурсия нима?

Адабиётлар рўйхати

1. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013, 488 b.
2. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. –СПб: ООО «ДиаСофтЮП», 2003.-560с.
3. Культин Н. Б. C/C++ в задачах и примерах. — СПб.: БХВ-Петербург, 2005. -288 с.
4. acm.tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.

3-мавзу. Массивлар ва сатрлар

Режа:

1. Бир ўлчовли массивлар.
2. Кўп ўлчовли массивлар.
3. Белгили ахборот ва сатрлар.
4. Сўзлар массивлари.
5. Излаш ва тартиблаш.

Таянч иборалар: массив, сатр, ўлчам, белги, излаш, саралаш, вақт лимити.

3.1. Бир ўлчовли массивлар

Массив тушунчаси. Массив бу бир типли номерланган маълумотлар жамланмасидир. Массив индексли ўзгарувчи тушунчасига мос келади. Массив таърифланганда типи, номи ва индекслар чегараси кўрсатилади. Масалан, type туридаги length та элементдан иборат a номли массив шундай эълон қилинади:

```
type a[length];
```

Бу махсус a[0], a[1], ..., a[length -1] номларга эга бўлган type туридаги ўзгарувчиларнинг эълон қилинишига тўғри келади.

Массивнинг ҳар бир элементи ўз рақамига - индексга эга. Массивнинг x-нчи элементига мурожаат индекслаш операцияси ёрдамида амалга оширилади:

```
int x = ...;      //butun sonli indeks
TYPE value = a[x];   //x-nchi elementni o‘qish
a[x] = value;     //x- elementga  yozish
```

Индекс сифатида бутун тур қийматини қайтарадиган ҳар қандай ифода қўлланиши мумкин: char, short, int, long. C да массив элементларининг индекслари 0 дан бошланади (1 дан эмас), length элементдан иборат бўлган массивнинг охирги элементининг индекси эса - бу length -1 (length эмас) га тенг. Массивнинг int z[3] шаклдаги таърифи, int типига тегишли z[0], z[1], z[2] элементлардан иборат массивни аниқлайди.

Массив чегарасидан ташқарига чиқиш (яъни мавжуд бўлмаган элементни ўқиш/ёзишга уриниш) дастур бажарилишида кутилмаган натижаларга олиб келиши мумкин. Шуни таъкидлаб ўтиш лозимки, бу энг кўп тарқалган хатолардан биридир.

Агар массив инициализация қилинганда элементлар чегараси кўрсатилган бўлса, рўйхатдаги элементлар сони бу чегарадан кам бўлиши мумкин, лекин ортиқ бўлиши мумкин эмас.

Мисол учун int a[5] = {2,-2}. Бу ҳолда a[0] ва a[1] қийматлари

аниқланган бўлиб, мос ҳолда 2 ва –2 га тенг. Агар массив узунлигига қараганда камроқ элемент берилган бўлса, қолган элементлар 0 ҳисобланади:

```
int a10[10] = {1, 2, 3, 4};  //va 6 ta nol
```

Агар номланган массивнинг тавсифида унинг ўлчамлари кўрсатилмаган бўлса, компилятор томонидан массив чегараси автоматик аниқланади:

```
int a3[] = {1, 2, 3};
```

Массивда мусбат элементлар сони ва суммасини ҳисоблаш (1-листинг).

1-листинг.

```
#include<stdio.h>
int main()
{
int s = 0,k = 0;
int x[] = {-1,2,5,-4,8,9};
for(int i = 0; i<6; i++)
{
if (x[i]< = 0) continue;
k++;
s+ = x[i];
};
printf("%d\n",k);
printf("%d\n",s);
return 0;
};
```

Массивнинг энг катта, энг кичик элементи ва ўрта қийматини аниқлаш (2-листинг):

2-листинг

```
#include<stdio.h>
int main()
{
int i,j,n;
float min,max,s = 0;
float a,b,d,x[100];
while(1)
{
printf("\n n = ");
scanf("%d",&n);
if ( n>0 && n< = 100 ) break;
printf("\n Hato 0<n<101 bo'lishi kerak");
}
printf("\n elementlar qiymatlarini kiriting:\n");
for (i = 0;i<n;i++)
{
printf("x[%d] = ",i);
scanf("%f",&x[i]);
```

```
}
max = x[0];
min = x[0];
for(i = 0;i<n;i++)
{
s+ = x[i];
if (max<x[i]) max = x[i];
if (min>x[i]) min = x[i];
};
s/ = n;
printf("\n max = %f",max);
printf("\n min = %f",min);
printf("\n o'rta qiymat = %f",s);
return 0;
};
```

Массивларни навларга ажратиш. Навларга ажратиш - бу берилган кўплаб объектларни бирон-бир белгиланган тартибда қайтадан гуруҳлаш жараёни.

Массивларнинг навларга ажратилиши тез бажарилишига кўра фарқланади. Навларга ажратишнинг n*n та қиёслашни талаб қилган оддий усули ва n*log(n) та қиёслашни талаб қилган тез усули мавжуд. Оддий усуллар навларга ажратиш тамойилларини тушунтиришда қулай ҳисобланади, чунки содда ва калта алгоритмларга эга. Мураккаблаштирилган усуллар камроқ сонли операцияларни талаб қилади, бироқ операцияларнинг ўзи мураккаброқ, шунинг учун унча катта бўлмаган массивлар учун оддий усуллар кўпроқ самара беради.

Оддий усуллар учта асосий категорияга бўлинади:

- оддий киритиш усули билан навларга ажратиш;
- оддий ажратиш усули билан навларга ажратиш;
- оддий алмаштириш усули билан навларга ажратиш.

Оддий киритиш усули билан навларга ажратиш

Массив элементлари аввалдан тайёр берилган ва дастлабки кетма-кетликларга бўлинади. i = 2 дан бошлаб, ҳар бир қадамда дастлабки кетма-кетликдан i-нчи элемент чиқариб олинади ҳамда тайёр кетма-кетликнинг керакли ўрнига киритиб қўйилади. Кейин i биттага кўпаяди ва ҳ.к.

Керакли жойни излаш жараёнида, кўпроқ ўнгдан битта позициядан танлаб олинган элементни узатиш амалга оширилади, яъни танлаб олинган элемент, j: = i-1 дан бошлаб, навларга ажратиб бўлинган қисмнинг навбатдаги элементи билан қиёсланади. Агар танлаб олинган элемент a[i] дан катта бўлса, уни навларга ажратиш қисмига қўшадилар, акс ҳолда a[j] битта позицияга сурилади, танлаб олинган элементни эса навларга ажратилган кетма-кетликнинг навбатдаги элементи билан қиёслайдилар. Тўғри келадиган жойни қидириш жараёни иккита турлича шарт билан тугалланади:

- агар a[j]>a[i] элементи топилган бўлса;
- агар тайёр кетма-кетликнинг чап учига борган бўлса.

```
int i, j, x;
for(i = 1; i < n; i++)
{
x = [i]; // kiritib qo‘yishimiz lozim bo‘lgan elementni esda saqlab qolamiz
j = i - 1;
while(x<a[j]&&j> = 0)//to‘g‘ri keladigan joyni qidirish
}
a[j+1] = a[j]; //o‘ngga surish
j--;
}
a[j+1] = x; //elementni kiritish
}
```

Оддий танлаш усули билан навларга ажратиш

Массивнинг минимал элементи танланади ҳамда массивнинг биринчи элементи билан жой алмаштирилади. Кейин жараён қолган элементлар билан такрорланади ва ҳ.к.

```
int i, min, n_min, j;
for(i = 0; i < n-1; i++)
{
min = a[i]; n_min = i; //minimal qiymatni qidirish
for(j = i + 1; j < n; j++)
if(a[j] < min){min = a[j]; n + min = j;}
a[n_min] = a[i]; //almashtirish
a[i] = min;
}
```

Оддий алмаштириш усули билан навларга ажратиш

Элементлар жуфтлари охиргисидан бошлаб қиёсланади ва ўрин алмашинади. Натижада массивнинг энг кичик элементи унинг энг чапки элементига айланади. Жараён массивнинг қолган элементлари билан давом эттирилади.

```
for(int i = 1; i < n; i++)
for(int j = n - 1; j > = i; j—
if(a[j] < a[j-1])
{int r = a[j]; a[j] = a[j-1]; a[j - 1] = r;}
}
```

Бир ўлчамли массивларни функция параметрлари сифатида узатиш. Массивдан функция параметри сифатида фойдаланганда, функциянинг биринчи элементига кўрсаткич узатилади, яъни массив ҳамма вақт адрес бўйича узатилади. Бунда массивдаги элементларнинг миқдори

ҳақидаги ахборот йўқотилади, шунинг учун массивнинг ўлчамлари ҳақидаги маълумотни алоҳида параметр сифатида узатиш керак.

Мисол:

Массив барча элементлари чиқарилсин (3-листинг):

3-листинг.

```
#include <stdio.h>
#include <stdlib.h>
int form(int a[100])
{
int n;
printf("\nEnter n:");
scanf("%d",&n);
for(int i = 0;i < n; i++)
a[i] = rand()%100;
return n;
}
void print(int a[100], int n)
{
for(int i = 0;i < n; i++)
printf("%d ", a[i]);
printf("\n");
}
int main()
{
int a[100];
int n;
n = form(a);
print(a,n);
return 0;
}
```

Функцияга массив бошланиши учун кўрсаткич узатилгани туфайли (адрес бўйича узатиш), функция танасининг операторлари ҳисобига массив ўзгариши мумкин.

Функцияларда бир ўлчовли сонли массивлар аргумент сифатида ишлатилганда уларнинг чегарасини кўрсатиш шарт эмас.

Мисол. Массивдан барча жуфт элементлар чиқарилсин (4-листинг).

4-листинг.

```
#include <stdio.h>
#include <stdlib.h>
void form(int a[], int n)
{
for(int i = 0; i < n; i++)
a[i] = rand()%100;
}
void print(int a[],int n)
```

```
{
for(int i = 0; i < n; i++)
printf("%d ", a[i]);
printf("\n");
}
int Dell(int a[],int n)
{
int j = 0, i, b[100];
for(i = 0; i < n; i++)
if(a[i]%2! = 0)
{
b[j] = a[i]; j++;
}
for(i = 0; i < j; i++) a[i] = b[i];
return j;
}
int main()
{
int a[100];
int n;
printf("\nEnter n:");
scanf("%d",&n);
form(a, n);
print(a, n);
n = Dell(a, n);
print(a, n);
system("pause");
return 0;
}
```

Функцияларда бир ўлчовли сонли массивлар аргумент сифатида ишлатилганда уларнинг чегарасини кўрсатиш шарт эмас. Мисол тариқасида n ўлчовли векторлар билан боғлиқ функцияларни таърифлашни кўриб чиқамиз.

Векторнинг узунлигини аниқлаш функцияси:

```
float mod_vec(int n, float x[])
{
float a = 0;
for (int i = 0; i < n; i++)
 a+ = x[i] * x[i];
return sqrt(double(a));
}
```

Функцияларда бир ўлчовли массивлар қайтарилувчи қийматлар сифатида ҳам келиши мумкин. Мисол учун икки вектор суммасини ҳисобловчи функция процедурани кўрамиз:

```
void sum_vec(int n, float a, float b, float c)
{
for(int i = 0;i < n; i++,c[i] = a[i] + b[i]);
}
```

Бу функцияга қуйидагича мурожаат қилиш мумкин:

```
float a[] = {1, -1.5, -2};
b[] = {-5.2, 1.3, -4},c[3];
sum_vec(3, a, b, c);
```

Полином. Полином қийматини ҳисоблаш функцияси poly деб номланади.

Прототипи:

```
double poly(double x, int degree, double coeffs[]);
```

Алгебраик полином коэффициентлари coeffs[0], coeffs[1], ..., coeffs[degree] массив элементларида берилади.

Мисол:

```
#include <stdio.h>
#include <math.h>
/* polynomial: x**3 - 2x**2 + 5x - 1 */
int main(void)
{
double array[] = { -1.0, 5.0, -2.0, 1.0};
double result;
result = poly(2.0, 3, array);
printf("The polynomial: x**3 - 2.0x**2 + 5x - 1 at 2.0 is %lf\n", result);
return 0;
}
```

3.2. Кўп ўлчовли массивлар

Кўп ўлчовли массивлар таърифи. Икки ўлчовли массивлар математикада матрица ёки жадвал тушунчасига мос келади. Жадвалларнинг инициализация қилиш қоидаси, икки ўлчовли массивнинг элементлари массивлардан иборат бўлган бир ўлчовли массив таърифига асослангандир.

Мисол учун икки қатор ва уч устундан иборат бўлган ҳақиқий типга тегишли d массив бошланғич қийматлари қуйидагича кўрсатилиши мумкин:

```
float d[2][3] = {(1, -2.5, 10),(-5.3, 2, 14)};
```

Бу ёзув қуйидаги қиймат бериш операторларига мосдир:

```
d[0][0] = 1; d[0][1] = -2.5; d[0][2] = 10;
d[1][0] = -5.3; d[1][1] = 2; d[1][2] = 14;
```

Бу қийматларни битта рўйхат билан ҳосил қилиш мумкин:

```
float d[2][3] = {1, -2.5, 10, -5.3, 2, 14};
```

Инициализация ёрдамида бошланғич қийматлар аниқланганда массивнинг ҳамма элементларига қиймат бериш шарт эмас.

Мисол учун: int x[3][3] = {(1, -2, 3), (1, 2), (-4)}.

Бу ёзув қуйидаги қиймат бериш операторларига мосдир:

x[0][0] = 1; x[0][1] = -2; x[0][2] = 3;

x[1][0] = -1; x[1][1] = 2; x[2][0] = -4;

Инициализация ёрдамида бошланғич қийматлар аниқланганда массивнинг биринчи индекси чегараси кўрсатилиши шарт эмас, лекин қолган индекслар чегаралари кўрсатилиши шарт.

Мисол учун:

double x[][2] = {(1.1,1.5),(-1.6,2.5),(3,-4)}

Бу мисолда автоматик равишда қаторлар сони учга тенг деб олинади.

Қуйидаги кўрадиган мисолимизда жадвал киритилиб ҳар бир қаторнинг максимал элементи аниқланади (5-листинг).

5-листинг.

```
#include <stdio.h>
#include <stdlib.h>
int main(){
int a[4][3];
int max;
int i,j;
for(i = 0;i < 4; i++)
{
for(j = 0; j < 3; j++)
{
printf("a[%d][%d] = ",i,j);
scanf("%d",&a[i][j]);
}
printf("\n");
};
for(i = 0; i < 4; i++)
{
max = a[i][0];
for(j = 0; j < 3; j++)
if (max<a[i][j]) max = a[i][j];
printf("%d ",max);
}
system("pause");
return 0;
}
```

Функцияга кўп ўлчамли массивларни узатиш. Кўп ўлчамли массивларни функцияга узатишда барча ўлчамлар параметрлар сифатида узатилиши керак. С тилида кўп ўлчамли массивлар аниқланиши бўйича мавжуд эмас. Агар биз бир нечта индексга эга бўлган массивни тавсифласак (масалан, int mas [3][4]), бу дегани, биз бир ўлчамли mas массивини тавсифладик, бир ўлчамли int [4] массивлар эса унинг элементларидир.

Мисол: Квадрат матрицани узатиш (транспортировка қилиш).

Агар void transp(int a[][], int n){.....} функциясининг сарлавҳасини аниқласак, бу ҳолда биз функцияга номаълум ўлчамдаги массивни узатишни хоҳлаган бўлиб қоламиз. Аниқланишига кўра массив бир ўлчамли бўлиши керак, ҳамда унинг элементлари бир хил узунликда бўлиши керак. Массивни узатишда унинг элементларининг ўлчамлари ҳақида ҳам бирон нарса дейилмаган, шунинг учун компилятор хато чиқариб беради.

Бу муаммонинг энг содда ечими функцияни қуйидагича аниқлашдир:

void transp(int a[][4],int n){.....}, бу ҳолда ҳар бир сатр ўлчами 4 бўлади, массив кўрсаткичларининг ўлчами эса ҳисоблаб чиқарилади (6-листинг).

6-листинг.

```
#include <stdio.h>
#include <stdlib.h>
const int N = 4;//global ўзгарувчи
void input_array(int a[][N],int n)
{
int i,j;
for(i = 0;i<n;i++)
{
for(j = 0;j<n;j++)
{
printf("a[%d][%d] = ",i,j);
scanf("%d",&a[i][j]);
}
printf("\n");
}
}
void print_array(int a[][N],int n)
{
int i,j;
for(i = 0;i<n;i++)
{
for(j = 0;j<n;j++)
{
printf("a[%d][%d] = ",i,j);
printf("%d ",a[i][j]);
}
printf("\n");
}
}
void transp(int a[][N], int n)
{
int r;
for(int i = 0; i < n; i++)
for(int j = 0; j < n; j++)
if(i < j)
```

```
{
r = a[i][j]; a[i][j] = a[j][i]; a[j][i] = r;
}
}
int main()
{
int mas[N][N];
int n;
printf("\n n = ");
scanf("%d",&n);
input_array(mas,n);
transp(mas, n);
print_array(mas, n);
system("pause");
return 0;
}
```

Мисол тариқасида уч ўлчовли квадрат матрицани уч ўлчовли векторга кўпайтириш функциясини кўриб чиқамиз:

```
void umn_vec( float a[3][3],float b[3], float c[3])
{
for(int i = 0; i<3; i++)
{
c[i] = 0;
for(int j = 0; j<3; j++)
c[i]+ = a[i][j]*b[j];
};
}
```

3.3. Белгили ахборот ва сатрлар

Сатрлар. С да белгили маълумотлар учун char тури қабул қилинган. Белгили ахборотни тақдим этишда белгилар, символли ўзгарувчилар ва матнли константалар қабул қилинган.

Мисоллар:

```
const char c = 'c';
char a,b;
```

С даги сатр - бу нул-белги - \0 (нул-терминатор)- билан тугалланувчи белгилар массиви. Нул-терминаторнинг ҳолатига қараб сатрнинг амалдаги узунлиги аниқланади. Бундай массивдаги элементлар сони, сатр тасвирига қараганда, биттага кўп.

Символли массивлар қуйидагича инициализация қилинади:

char capital[] = "TASHKENT"; Бу ҳолда автоматик равишда массив элементлари сони аниқланади ва массив охирига сатр кўчириш '\0' символи қўшилади.

Юқоридаги инициализацияни қуйидагича амалга ошириш мумкин:

char capital[] = {'T','A','S','H','K','E','N','T','\0'};

Бу ҳолда сўз охирида '\0' символи аниқ кўрсатилиши шарт.

Қиймат бериш оператори ёрдамида сатрга қиймат бериш мумкин эмас. Сатрни массивга ёки киритиш пайтида ёки номлантириш ёрдамида жойлаштириш мумкин (7-листинг).

7-листинг.

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
char s1[10] = "string1";
int k = sizeof(s1);
printf("\n%s %d",s1,k);
char s2[] = "string2";
k = sizeof(s2);
printf("\n%s %d",s2,k);
char s3[] = {'s','t','r','i','n','g','3','\0'};
k = sizeof(s3);
printf("\n%s %d",s3,k);
char *s4 = "string4";//satr ko'rsatkichi, uni o'zgartirib bo'lmaydi
k = sizeof(s4);
printf("\n%s %d",s4,k);
system("pause");
return 0;
}
```

Натижа:

string1 10
string2 8
string3 8
string4 4

Кейинги мисолда киритилган сўздан берилган ҳарфни олиб ташлаш дастури берилган (8-листинг).

8-листинг.

```
#include <stdio.h>
int main()
{
char s[100];
scanf("%s",&s);
int i, j;
```

```
for ( i = j = 0; s[i] ! = '\0'; i++)
if ( s[i] ! = 'c' )
s[j++] = s[i];
s[j] = '\0';
printf("%s",s);
return 0;
}
```

Хар сафар 'c' дан фарқли символ учраганда, у j позицияга ёзилади ва фақат шундан сўнг j нинг қиймати 1 га ошади. Бу қуйидаги ёзувга эквивалент:

```
if ( s[i] ! = c )
s[j] = s[i];
j++;
```

Функциялар ва сатрлар. Функцияларда сатрлар ишлатилганда уларнинг чегарасини кўрсатиш шарт эмас. Сатрларнинг узунлигини ҳисоблаш len функциясини қуйидагича таърифлаш мумкин:

```
int len(char c[])
{ int m = 0;
for(m = 0;c[m]! = ’0’; m++);
return m;
};
```

Шу функциядан фойдаланилган дастурни келтирамиз:

```
#include <stdio.h>
int len(char c[])
{
int m = 0;
while(c[m++]);
return m-1;
};
int main()
{
char e[] = "Pro Tempore!";
printf("\n%d", len(e));
return 0;
}
```

Бу функциянинг стандарт варианти strlen деб аталади ва бу функциядан фойдаланиш учун string.h сарлавҳа файлидан фойдаланиш лозим.

Сатрдан нусха олиш функцияси strcpy ни С тилида қуйидагича таърифлаш мумкин:

```
#include <stdio.h>
void strlen(char s1[], char s2[])
```

```
{
int i = 0;
while(s2[i]! = '\0') s1[i++] = s2[i];
s1[i] = s2[i];
}

int main()
{
char s1[] = "aaa";
char s2[] = "ddd";
strcpy(s1,s2);
printf("%s",s1);
return 0;
}
```

Натижа:

ddd

Берилган сатрни тескарига айлантирувчи функция:

```
reverse(char s[])
{
int c, i, j;
for(i = 0, j = strlen(s) - 1; i < j; i++, j--)
c = s[i];
s[i] = s[j];
s[j] = c;
}
```

Кейинги мисолимизда T қаторни S қатор охирига уловчи STRCAT(S,T) функциясини кўриб чиқамиз:

```
strcat(char s[], t[])
 {
int i, j;
i = j = 0;
while (s[i] ! = '\0')
i++;
while((s[i++] = t[j++]) ! = '\0')
}
```

3.4. Сўзлар массивлари

Сўзлар массивини киритиш. С тилида сўзлар массивлари икки ўлчовли символли массивлар сифатида таърифланади. Мисол учун:

char name[4][5].

Бу таъриф ёрдамида ҳар бири 5 та ҳарфдан иборат бўлган 4 та сўзли массив киритилади. Сўзлар массивлари қуйидагича инициализация қилиниши мумкин:

char Name[3][8] = { "Анвар","Миркомил","Юсуф"}.

Бу таърифда ҳар бир сўз учун хотирадан 8 байт жой ажратилади ва ҳар бир сўз охирига ' \0' белгиси куйилади.

Сўзлар массивлари инициализация қилинганда сўзлар сони кўрсатилмаслиги мумкин. Бу ҳолда сўзлар сони автоматик аниқланади:

char comp[][9] = { "компьютер","принтер","картридж"}.

Қуйидаги дастурда берилган ҳарф билан бошланувчи сўзлар рўйхати босиб чиқарилади:

```
#include <stdio.h>
int main()
{
char a[10][10];
char c = 'a';
int i;
for (i = 0;i<3;i++) scanf("%s",&a[i]);
for (i = 0;i<3;i++)
if (a[i][0] == c) printf("\n%s",a[i]);
return 0;
}
```

Қуйидаги дастурда фан номи, талабалар рўйхати ва уларнинг баҳолари киритилади. Дастур бажарилганда икки олган талабалар рўйхати босиб чиқарилади:

```
#include <stdio.h>
int main()
{
char a[10][10];
char s[10];
int k[10];
scanf("%s",&s);
for (int i = 0;i<3;i++)
{
scanf("%s",&a[i]);
scanf("%d",&k[i]);
};
for (int i = 0;i<3;i++)
if (k[i] == 2) printf("%s\n",a[i]);
return 0;
}
```

Функциялар ва сўзлар массивлари. Сатрли массивлар функция аргументи сифатида ишлатилганда сатрларнинг умумий узунлиги аниқ кўрсатилиши шарт.

Мисол тариқасида ихтиёрий сондаги сатрлар массивини алфавит бўйича тартиблаш функциясидан фойдаланилган дастурни кўриб чиқамиз:

```
#include <stdio.h>
#define m 10
void sort(int n, char a[][m])
{
char c;
int i,j,l;
for (i = 0;i<n;i++)
for (j = i+1;j<m;j++)
if (a[i][0]<a[j][0] )
for(l = 0;l<m;l++)
{
c = a[i][l];
a[i][l] = a[j][l];
a[j][l] = c;
};
};
int main()
{
char aa[][m] = {"Alimov","Dadashev","Boboev"};
sort(3,aa);
for(int i = 0; i<3;i++) printf("%s\n",aa[i]);
return 0;
}
```

3.5. Излаш ва тартиблаш

Иккига бўлиб излаш. Қуйидаги дастурда тартибланган массивда иккига бўлиб калит сонни излаш алгоритми асосида тузилган функциядан фойдаланиш келтирилган:

```
#include<stdio.h>
#include<conio.h>
int bsearch(int a[],int key,int n)
{
int m1,m2,m;
m1 = 0;m2 = n-1;
while(m1< = m2)
{
m = (m2+m1)/2;
if (a[m] == key) return m;
if (a[m]>key) m2 = --m;
if (a[m]<key) m1 = ++m;
}
return -1;
```

```
};
int main(int argc, char* argv[])
{
int m;
int a[] = {5,6,9,11};
m = bsearch(a,7,4);
printf("%d",m);
getch();
return 0;
}
```

Кейинги мисолда шу функция сатрлар учун варианти келтирилган:

```
#include<stdio.h>
#include<string.h>
#include<conio.h>
#define size 5
int strbsearch(char a[][size],char key[],int n)
{
int m1,m2,m,pr;
m1 = 0;m2 = n-1;
while(m1< = m2)
{
m = (m2+m1)/2;
pr = strcmp(a[m],key);
if (pr == 0) return m;
if (pr == -1) m2 = --m;
if (pr == 1) m1 = ++m;
}
return -1;
};
int main(int argc, char* argv[])
{
int m;
char a[][size] = {"aaa","aab","aac","aad"};
m = strbsearch(a,"aab",4);
printf("%d",m);
getch();
return 0;
}
```

Тезкор тартиблаш. Қуйидаги дастурда тезкор тартиблаш алгоритмига асосланган функциядан фойдаланилган. Алгоритм моҳияти шундан иборатки, аввал етакчи элемент танланади. Функцияда етакчи элемент сифатида бошланғич элемент танланади. Шундан сўнг массив икки қисмга ажратилади. Етакчи элементдан кичик элементлар паст қисмга, катта

элементлар юқори қисмга тўпланади. Шундан сўнг рекурсия асосида алгоритм иккала қисмга алоҳида қўлланади.

```
#include<stdio.h>
#include<conio.h>
int a[] = {5,4};
void sqsort(int k1, int k2)
{
if(k1 < k2){
int i, j, k;
i = k1; j = k2;
while(i < j)
{
if (a[k1] > a[i]) {i++; continue;}
if (a[k1] < a[j]) {j--; continue;}
k = a[j]; a[j] = a[i]; a[i] = k;
}
k = a[k1]; a[k1] = a[i]; a[i] = k;
sqsort(k1, i);
sqsort(i+1, k2);
};
}
int main(int argc, char* argv[])
{
int i;
sqsort(0, 1);
for(i = 0; i < 2; i++) printf("%d ", a[i]);
getch();
return 0;
}
```

Назорат саволлари

1. Массив нима?
2. Бир, икки ва уч ўлчовли массивларни геометрик маъноси нима?
3. Сатр символли массивдан қандай фарқ қилади?
4. Бир ўлчовли массивларни инициализация қилиш усулларини кўрсатинг.
5. Кўп ўлчовли массив таърифи хусусиятларини келтиринг.
6. Кўп ўлчовли масивлар инициализацияси хусусиятлари.
7. Сатрларни инициализация қилиш усулларини кўрсатинг.
8. Смиволлар массиви қандай киритилади?
9. Символли ва сонли массивлар ўртасидаги фарқ нима?
10. Қандай қилиб бир ўлчовли массивлар формал параметрлар сифатида ишлатилиши мумкин?
11. Қандай қилиб кўп ўлчовли массивлар формал параметрлар сифатида ишлатилиши мумкин?

12. Сатр таърифлаш усуллари.
13. Сатрлар функция параметри сифатида.
14. Массивларни саралаш дастурини тузинг.
15. Массивларда қидириш дастурини тузинг.

Адабиётлар рўйхати

1. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013, 488 b.

2. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. –СПб: ООО «ДиаСофтЮП», 2003.-560с.

3. Культин Н. Б. C/C++ в задачах и примерах. — СПб.: БХВ-Петербург, 2005. -288 с.

4. acm.tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.

4-мавзу. Структуралар ва файллар

Режа:

1. Структура таърифи.
2. Структуралар ва массивлар
3. Структура хоссалари
4. Бирлашмалар
5. Файллар
6. Файлга кетма-кет мурожаат қилиш

Таянч иборалар: структура, бирлашма, файл, оқим, ўқиш, ёзиш.

4.1. Структура таърифи

Ўзгарувчиларнинг қўшимча турлари. Структура бу турли типдаги маълумотлар бирлаштирилган типдир. Структура ҳар хил типдаги элементлар-компоненталардан иборат бўлади.

Бир хил типдаги элементлардан ташкил топувчи массивлардан фарқли ўлароқ, структураларнинг ташкил этувчилари ҳар хил бўлиши мумкин ва улар ҳар хил номланиши керак. Масалан: омбордаги товарларнинг рўйхатини тузиш учун яратилган структура компоненталари билан қуйидагичадир:

Товар номи (char[20])

Улгуржи нархлар (long)

Сотиш нархлари процент ҳисобида (float)

Товар партиясининг ҳажми (int)

Товар кириб келган вақти (char [9])

“Омбордаги товарлар” структурасидаги элементларни санаб ўтиш давомида дастурчи томонидан шу элементларнинг типи қўшилган. Шунга асосланиб компоненталар ҳар қандай типга эга бўлиши мумкин.

Структуралар қуйидагича таърифланиши мумкин:

struct струтурали_тип_номи
{Элементлар_таърифлари}

Масалан:
```
struct Date
{
int year;
char month, day;
};
```

Юқорида кўриб ўтилган мисолни қуйидагича ёзиш мумкин.
```
struct goods {
char name[20]; / *номланиши*/
long price; /*улгуржи нархлар */
float percent; /*нархлар % */
int vol;          /* товар партияси */
char date [9]; /* товар кириб келган вақти*/
};
```

Дастурда тузилма туридаги ўзгарувчи қуйидаги шаклда киритилади:
struct Тузилма_номи идентификаторларнинг_рўйхати;

Масалан:
```
struct Date s1, s2;
```
Мисол учун:
```
struct complex
{
double real;
double imag;
}
```
Бу мисолда комплекс сонни тасвирловчи структурали тип complex киритилган бўлиб, комплекс сон ҳақиқий қисмини тасвирловчи real ва мавҳум қисмини тасвирловчи imag компоненталаридан иборатдир.

Конкрет структуралар бу ҳолда қуйидагича тасвирланади:

struct complex sigma,alfa;

Қуйидаги мисолда каср сонни тасвирловчи numerator–суръат ва denominator-махраж компоненталаридан иборат структура таърифи келтирилган.
```
struct fraction;
{
int numerator;
int denominator;
}
```
Бу ҳолда конкрет структуралар қуйидагича тасвирланиши мумкин:

struct fraction beta;

Структуралар таърифланганда конкрет структуралар рўйхатини киритиш мумкин:

struct струтурали_тип_номи
{Элементлар_таърифлари}
Конкрет_структуралар_рўйхати.
Мисол:

```
struct student
{
char name[15];
char surname[20];
int year;
} student_1, student_2, student_3;
```

Бу ҳолда student структурали тип билан бирга учта конкрет структура киритилади. Бу структуралар студент исми (name[15]), фамилияси (surname[20]), туғилган йилидан (year) иборат.

Структурали тип таърифланганда тип номи кўрсатилмай, конкрет структуралар рўйхати кўрсатилиши мумкин:

struct
{Элементлар_таърифлари}
Конкрет_структуралар_рўйхати.

Қуйидаги таъриф ёрдамида учта конкрет структура киритилади, лекин структурали тип киритилмайди.

```
struct
{
char processor [10];
int frequency;
int memory;
int disk;
} IBM_486, IBM_386, Compaq;
```

Структураларга мурожаат. Конкрет структуралар таърифланганда массивлар каби инициализация килиниши мумкин. Масалан

struct complex sigma = {1.3; 12.6};
struct goods coats = {"пиджак", 40000, 7.5, 220, "12.01.97");
Бир хил типдаги структураларга қиймат бериш амалини қўллаш мумкин:
struct complex alfa; alfa = sigma;
Лекин структуралар учун солиштириш амаллари аниқланмаган.
Структуралар элементларига қуйидагича мурожаат қилиш мумкин:

Структура номи.элемент_номи.

'Нуқта амали' структура элементига мурожаат қилиш амали дейилади. Бу амал қавс амаллари билан бирга энг юқори устиворликка эга.

Мисол:

struct complex alfa = {1.2, -4.5}, betta = {5.6, -7.8), sigma;

sigma.real = alfa.real + betta.real;

sigma.imag = alfa.imag + betta.imag;

Конкрет структуралар элементлари дастурда алоҳида киритилиши ва чиқарилиши зарур. Қуйидаги мисолда хизматчи структураси киритилади:

```
#include <stdio.h>
struct employee
{
char name [64];
long employee_id;
float salary;
char phone[10];
int office_number;
} worker;
void show_emploee(employee worker)
{
printf("\nIsmi:  %s", worker.name);
printf("\nTelefon: %s",worker.phone);
printf("\nNomer: %ld",worker.employee_id);
printf("\nOylik: %f",worker.salary);
printf("\nOfis: %d",worker.office_number);
};
int main()
{
worker.employee_id = 12345;
worker.salary = 25000.00;
worker.office_number = 102;
printf("\n ismi:");
scanf("%s",&worker.name);
printf("\n telefon:");
scanf("%s",&worker.phone);
show_emploee(worker);
return 0;
}
```

Структуравий тип киритиш. Структуравий типни киритишда яна бир имкониятни ёрдамчи сўз typedef яратади. Структуравий тип киритилган ва қайта номланган хол учун таъриф қуйидаги кўринишга эга бўлади:

typedef struct{элементлар тарифи}

структура типи

Масалан:

```
typedef struct {
double real;
double imag;
}
complex;
```

Келтирилган қоида структуравий тип ва унга белгилаш киритади. Бу белгилаш орқали структура типидаги ўзгарувчиларни худди шундай оддий номланган структуравий тип каби киритиш мумкин.

Мисол: **complex sigma, alfa;**

Структуравий типга дастурчи typedef ёрдамида ном беради шу билан бирга у иккинчи номга ҳам struct ёрдамчи сўз орқали эга бўла олади. Мисол тариқасида рационал касрнинг структуравий типини аниқлашни кўриб чиқамиз.

```
typedef struct rational_fraction
{
int numerator;  /* Сурат */
int denominator; /* Махраж*/
} fraction;
```

бу ерда fraction – структуравий типнинг белгиланиши typedef- ёрдамида киритилмокда.

Структуравий типларни стандарт кўринишда киритиш учун rational_fraction ишлатилади.

4.2. Структуралар ва массивлар

Массивлар структуралар элементлари сифатида. Массивларни структуралар элементи сифатида ишлатилиши ҳеч қандай қийинчилик туғдирмайди. Биз юқорида символли массивлардан фойдаланишни кўрдик.

Қуйидаги мисолда фазода берилган нуқтавий жисмни тасвирловчи компоненталари жисм массаси ва координаталаридан иборат структура киритилган бўлиб, нуқтанинг координаталар марказигача бўлган масофаси ҳисобланган.

```
#include <stdio.h>
#include <math.h>
struct
 {
double mass;
float coord[3];
} point = {12.3,{1.0,2.0,-3.0}};
int main()
{
```

```
int i;
float s = 0.0;
for (i = 0;i<3; i++)
s+ = point.coord[i]*point.coord[i];
printf("\n masofa = %f",sqrt(s));
return 0;
}
```

Бу мисолда point структураси номсиз структурали тип орқали аниқланган бўлиб, қийматлари инициализация ёрдамида аниқланади.

Структуралар массивлари. Структуралар массивлари оддий массивлар каби тасвирланади. Юқорида киритилган структурали типлар асосида қуйидаги структуралар массивларини киритиш мумкин:

```
struct goods list[100];
complex set [80];
```

Бу таърифларда list ва set структуралар номлари эмас, элементлари структуралардан иборат массивлар номларидир. Конкрет структуралар номлари бўлиб set[0], set[1] ва ҳоказолар хизмат қилади. Конкрет структуралар элементларига қуйидагича мурожаат қилинади: set[0].real – set массиви биринчи элементининг real номли компонентасига мурожаат.

Қуйидаги мисолда нуқтавий жисмларни тасвирловчи структуралар массиви киритилади ва бу нуқталар системаси учун оғирлик маркази координаталари (x_c, y_c, z_c) ҳисобланади. Бу координаталар қуйидаги формулалар асосида ҳисобланади:

$m = \sum m_i; x_c = (\sum x_i m_i)/m; y_c = (\sum y_i m_i)/m; z_c = (\sum z_i m_i)/m;$

```
#include <stdio.h>
struct particle
{
double mass;
double coord [3];
};
int main()
{
struct particle mass_point[] = { 20.0, {2.0, 4.0, 6.0}, 40.0, {6.0, -2.0, 6.0},
10.0, {1.0, 3.0, 2.0}};
int N;
struct particle center = { 0.0, {0.0, 0.0, 0.0}};
int I;
N = sizeof(mass_point)/sizeof(mass_point[0]);
for (I = 0; I < N; I++)
{
```

```
center.mass+ = mass_point[I].mass;
center.coord[0]+ = mass_point[I].coord[0]* mass_point[I].mass;
center.coord[1]+ = mass_point[I].coord[1]* mass_point[I].mass;
center.coord[2]+ = mass_point[I].coord[2]* mass_point[I].mass;
}
printf("\n Massa markazi koordinatalari:");
for (I = 0; I < 3; I++)
{
center.coord[I]/ = center.mass;
printf("\n Koordinata %d %f", (I+1), center.coord[I]);
}
return 0;
}
```

4.3. Структура хоссалари

Структуралар ва функциялар. Структуралар функциялар аргументлари сифатида ёки функция қайтарувчи қиймат сифатида келиши мумкин. Бундан ташқари функция аргументи сифатида структура типидаги массив келиши мумкин.

Мисол учун комплекс сон модулини ҳисоблаш дастурини келтирамиз:

```
Double modul(complex a)
{return sqrt(a.real*a.real+a.imag*a.imag}
```

Икки комплекс сон йиғиндисини ҳисоблаш функцияси:

```
complex add(complex a, complex b)
{ complex c;
c.real = a.real+b.real;
c.imag = a.imag+b.imag;
return c;
}
```

Мисол:

```
#include<stdio.h>
#include<conio.h>
typedef struct
{
char name[20];
int year;
} person;
person old_person(person a[], int n)
{
int i;
person s = a[0];
```

```
for(i = 1; i < n; i++)
if(a[i].year > s.year) s = a[i];
return s;
}
void print_person(person s)
{
printf("name %s\n", s.name);
printf("year %d", s.year);
}
int main()
{
person a[] = {{"smit", 34}, {"bobbi", 45}, {"pit", 56}};
person s = old_person(a, 3);
print_person(s);
getch();
return 0;
}
```

Функцияда битта структура типидаги ўзгарувчи қийматини ўзгартириш мумкин эмас, лекин массив элементлари қийматини ўзгартириш мумкин:

```
#include<stdio.h>
#include<conio.h>
struct goods {
char name[20];
long price;
float percent;
};
void change_percent(struct goods a[], int n, float percent)
{
int i;
for(i = 1;i <n; i++) a[i].percent = percent;
}
void print_goods(struct goods s)
{
printf("%s %ld %f\n", s.name, s.price, s.percent);
}
void all_print(struct goods a[], int n)
{
int i;
for(i = 0;i < n; i++) print_goods(a[i]);
};
int main()
{
struct goods a[] = {{"smit", 34, 0.5},{"bobbi", 45, 0.7},{"pit", 56, 0.8}};
all_print(a, 3);
change_percent(a, 3, 0.5);
```

```
all_print(a, 3);
getch();
return 0;
}
```

Структуралар учун хотирадан жой ажратиш. Структура учун ажратилган жой ҳажмини қуйидаги амаллар ёрдамида аниқлаш мумкин:

sizeof (структурали_тип_номи);
sizeof (структура_номи);
sizeof структура_номи.

Охирги ҳолда структура номи ифода деб қаралади. Ифоданинг типи аниқланиб, ҳажми ҳисобланади.

Мисол учун:
sizeof (struct goods)
sizeof (tea)
sizeof coat

Мураккаб типлар яъни массивлар ва структурали типлар учун хотирага талаб уларнинг таърифига боғлиқдир. Масалан, double array[10] таъриф хотирадан 10*sizeof байт жой ажратилишига олиб келади.

```
struct mixture
{
int ii;
long ll;
char cc[8];
};
```

Бу таъриф ҳар бир struct mixture типидаги объект хотирада sizeof(int)+sizeof(long)+8*sizeof(char) байт жой эгаллашини кўрсатади. Объект аниқ ҳажмини қуйидаги амал ҳисоблайди:

sizeof(struct mixture)

Хотирани текислаш. Структурали тип киритилиши бу тип учун хотирадан жой ажратилишига олиб келмайди. Ҳар бир конкрет структура (объект) таърифланганда, шу объект учун элементлар типларига қараб хотирадан жой ажратилади. Хотирадан жой зич ажратилганда структура учун ажратилган жой ҳажми ҳар бир элемент учун зарур бўлган хотира ҳажмлари йиғиндисига тенг бўлади. Шу билан бирга хотирадан жой зич ажратилмаслиги ҳам мумкин, яъни элементлар орасида бўш жойлар ҳам қолиши мумкин. Бу бўш жой кейинги элементни хотира қисмларининг қабул қилинган чегаралари бўйича текислаш учун қолдирилади. Мисол учун бутун типдаги элементлар жуфт адреслардан бошланса, бу элементларга мурожаат тезроқ амалга оширилади.

```
#include <stdio.h>
```

```
struct Foo
{
char c;
int i;
char s;
};
int main()
{
printf("Foo hajmi = %d",sizeof(Foo));
return 0;
}
```

Натижа:

Foo hajmi = 12

Конкрет структураларни жойлашувига баъзи компиляторларда #pragma препроцессор директиваси ёрдамида таъсир ўтказиш мумкин. Бу директива қуйидаги шаклда:

pragma pack(n)

Бу ерда n нинг қиймати 1, 2 ёки 4 га тенг бўлиши мумкин.

pack(1) – элементларни байт чегаралари бўйича текислаш;

pack(2) – элементларни сўзлар чегаралари бўйича текислаш;

pack(4) – элементларни иккиланган сўзлар чегаралари бўйича тексилаш.

Масалан:

```
#include <stdio.h>
#pragma pack(1)
struct Foo
{
char c;
int i;
char s;
};
int main()
{
printf("Foo hajmi = %d",sizeof(Foo));
return 0;
}
```

Натижа:

Foo hajmi = 6

4.4. Бирлашмалар

Структураларга яқин тушунча бу бирлашма тушунчасидир. Бирлашмалар union хизматчи сўзи ёрдамида киритилади. Мисол учун:

```
union
{
long h;
```

```
int i,j;
char c[4]
}UNI;
```

Бирлашмаларнинг асосий хусусияти шундаки, унинг ҳамма элементлари бир хил бошланғич адресга эга бўлади.

Бирлашмаларнинг асосий афзалликларидан бири хотира бирор қисми қийматини ҳар хил типдаги қиймат шаклида қараш мумкиндир.

Мисол учун қуйидагича бирлашма

```
union
{
float f;
unsigned long k;
char h[4];
}fl;
```

Хотирага fl.f = 2.718 ҳақиқий сон юборсак унинг ички кўриниши кодини fl.l ёрдамида кўришимиз ёки алоҳида байтлардаги қийматларни fl.h[0]; fl.h[1] ва ҳоказо ёрдамида кўришимиз мумкин.

Қуйидаги дастур ёрдамида бирлашма хусусиятини текшириш мумкин:

```
#include <stdio.h>
enum paytype{CARD, CHECK};
struct{
paytype ptype;
union{
char card[4];
long check;
};
}info;
int main()
{
info.ptype = CHECK;
info.check = 77;
switch (info.ptype)
{
case CARD:printf("\nKarta bilan to'lash:%s", info.card); break;
case CHECK:printf("\nChek bilan to'lash:%ld", info.check); break;
}
return 0;
}
```

Натижа

```
Chek bilan to'lash:77
```

Бирлашмалар имкониятларини кўрсатиш учун bioskey() функциясидан фойдаланишни кўриб чиқамиз. Бу функция bios.h сарлавҳали файлда жойлашган бўлиб, қуйидаги прототипга эга:

int bioskey(int);

MS DOS операцион тизимида ихтиёрий клавишанинг босилиши клавиатура буферига маълумот ёзилишига олиб келади.

Агар функцияга bioskey(0) шаклда мурожаат қилинса ва буфер бўш бўлса бирор клавишага босилиши кутилади, агар буфер бўш бўлмаса функция буфердан икки байтли кодни ўқиб бутун сон сифатида қайтаради. Функцияга bioskey(0) шаклда мурожаат қилинса ва буфер бўш бўлса бирор клавиша босилиши кутилади, агар буфер бўш бўлмаса функция буфердаги навбатдаги кодни қайтаради. Функцияга bioskey(1) шаклда мурожаат қилиш буфер бўш ёки бўшмаслигини аниқлашга имкон беради. Агар буфер бўш бўлмаса функция буфердаги навбатдаги кодни қайтаради, лекин бу код буфердан ўчирилмайди.

Қуйидаги дастур буферга келиб тушувчи кодларни ўқиб мониторга чиқаришга имкон беради:

```
#include <stdio.h>
#include <bios.h>
int main()
{
union
{
char hh[2];
int ii;
} cc;
unsigned char scn,asc;
printf("\n\n Ctrl+Z билан чикиш.");
printf("\n Клавишани босиб, кодини олинг. \n ");
printf("\n SCAN || ASCII");
printf("\n (10) (16) (10) (16)");
do
{
printf("\n");
cc.ii = bioskey(0);
asc = cc.hh[0];
scn = cc.hh[1];
printf("  %4d %3xH || %4d %3xH |", scn, scn, asc, asc);
}
while(asc! = 26 || scn! = 44);
return 0;
}
```

Бу дастурда cc номли бирлашма киритилган бўлиб, cc.ii элементига bioskey(0) функцияси натижаси ёзилади. Сўнгра натижанинг алоҳида байтлари скэн ва ASCII кодлар сифатида мониторга чиқарилади.

Цикл то 26 ASCII код ва 44 скэн код пайдо бўлмагунча (ctrl+Z клавишлари босилмагунча) давом этади.

Разрядли майдонлар. Разрядли майдонлар структуралар ва бирлашмалар майдонларининг хусусий холидир. Разрядли майдон таърифланганда унинг узунлиги битларда кўрсатилади (бутун мусбат константа).

Мисол:

```
#include <stdio.h>
struct
{
int a:8;
int b:6;
} xx = {64, 64};
int main()
{
printf("\n%d", xx.a);
printf("\n%d", xx.b);
return 0;
}
```

Натижа

```
64
0
```

Разрядли майдонлар ихтиёрий бутун типга тегишли бўлиши мумкин. Разрядли майдонлар адресини олиш мумкин эмас. Хотирада разрядли майдонларни жойлаштириш компилятор ва аппаратурага боғлиқ.

Разрядли майдонлар ёрдамида разрядли массивлар ҳосил қилиш мумкин. Юқорида кўрилган сон ҳамма битларини чиқариш дастурини қуйидагича ёзиш мумкин:

```
include<stdio.h>
#include<conio.h>
struct bit
{
unsigned int i:1;
};
unsigned printbits(int c, struct bit pp[])
{
unsigned int i;
unsigned k = 8*sizeof(int);
for(i = 0; i < k; i++)
{
```

```
pp[i].i = c&1;
printf(" %d", c&1);
c>> = 1;
}
return k;
}
int main()
{
unsigned int k, i;
struct bit pp[100];
k = printbits(-5, pp);
printf("\n");
for(i = k-1; i > 0; i--)
printf("%d ", pp[i].i);
printf("%d ", pp[0].i);
getch();
return 0;
}
```

4.5. Файллар

C++ тилининг асосий хусусиятларидан бири олдиндан режалаштирилган файллар структураси йўқлигидир. Ҳамма файллар, байтлар кетма-кетлиги деб кўрилади. UNIX операцион системасида ҳар бир қурилмага «Махсус файл» мос келади, шунинг учун С библиотекасидаги функциялар файллар билан ҳам, қурилмалар билан ҳам маълумот алмашиниши учун фойдаланилади. С тили библиотекасида киритиш–чиқариш, қуйи даражадаги киритиш, чиқариш ва портлар учун киритиш–чиқариш, оқимли даража тизим хусусиятларига боғлиқ бўлиши учун бу ерда қаралмайди.

Оқимли киритиш ва чиқариш. Оқимли чиқариш ва киритишда маълумотлар билан алмашиш байтма-байт амалга оширилади. Лекин ташқи хотира қурилмалари билан алмашиш олдидан белгиланган маълумотлар блоки орқали амалга оширилади. Одатда бу блокнинг минимал ҳажми 512 ёки 1024 байтга тенг бўлади. Дискдан яъни файлдан ўқишда маълумотлар операцион тизим буферига ёзилади, сўнгра байтма-байт ёки маълум порциялар билан фойдаланувчи дастурига узатилади. Дискка яъни файлга ёзишда буферга йиғилади, сўнгра дискка бир мурожаат қилинганда ягона блок сифатида узатилади. Буферлар оператив хотира қисмлари сифатида яратилади, шунинг учун маълумот алмашиши дискка тўғридан-тўғри мурожаат қилишига кўра тезроқ амалга ошади. Шундай қилиб оқим бу буферлаш воситалари ва файлдир.

Оқим билан ишлашда қуйидаги вазифаларни бажариш мумкин.

- Оқимларни очиш ва ёпиш;

- Символ, қатор, сатр, форматланган маълумот ихтиёрий узунликдаги маълумотларни киритиш ёки чиқариш ва файл охирига етганлик шартини таҳлил қилиш;
- Буферлаш ва буфер ҳажмини бошқариш;
- Кўрсаткич оқимдаги ўрнини аниқлаш ёки янги ўрингa кўчириш.

Бу вазифаларни бажарувчи функциялардан фойдаланиш учун дастурга stdio.h – файлини улаш лозим.

Дастур бажарилиши бошланганда автоматик равишда қуйидаги оқимлар очилади:

- Стандарт киритиш оқими stdin;
- Стандарт чиқариш оқими stdout;
- Хатолар ҳақида маълумотлар стандарт оқими stderr;

Оқимларни очиш ва ёпиш. Оқим очилиши учун, олдиндан киритилган FILE типидаги структура билан боғлаш лозимдир. FILE структураси таърифи stdio.h библиотекасида жойлашган.

Бу структурада буферга кўрсаткич, ўқилаётган позицияга кўрсаткич ва бошқа маълумотлар сақланади.

Оқим очилганда дастурга оқимга кўрсаткич, яъни FILE структурали типдаги объектга кўрсаткич қайтарилади. Бу кўрсаткич қуйидагича эълон қилиниши лозим.

FILE * <кўрсаткич номи>

Мисол учун FILE * fp

Оқим очиш функцияси қуйидаги кўринишга эга;

<оқимга кўрсаткич номи> = fopen(<файл-номи>,<очиш режими>)

Мисол учун:fp = fopen(“t.txt”, “r”)

Оқим билан боғлиқ файлни қуйидаги режимларда очиш мумкин:

“w”- Янги файл ўқиш учун очилади. Агар файл мавжуд бўлмаса янгидан яратилади.

“r” - Мавжуд файл фақат ўқиш учун очилади.

“a” - Файл давом эттириш учун очилади.

“w+” - Файл ёзиш ва кейинги тахрирлаш учун очилади. Файл ихтиёрий жойидан ўқиш ёки ёзиш мумкин.

“r+”- файл ихтиёрий жойидан ўқиш ёки ёзиш мумкин, лекин файл охирига қўшиш мумкин эмас.

“a+” - Файл ихтиёрий жойидан ўқиш ва ёзиш учун очилади. Қуйидаги “w+” режимдан фарқли файл охирига маълумот қўшиш мумкин.

Матнли режимда оқимдан ўқилган қуйидаги символлар CR(қиймати 13) “кареткани қайтариш” ва LF(қиймати 10)- “янги қатор бошига ўтиш” битта символга “\n” (қиймати LF яъни 10га тенг) символга алмаштиради.

Агар файл матнли эмас ихтиёрий маълумотни сақласа бинар режимда очилади. Бунинг учун режимлар белгиларига b ҳарфи қўшилади, масалан “wb” ёки “r+b”. Баъзи компиляторларда матнли режим t ҳарфи ёрдамида кўрсатилади масалан ”rt”.

Оқим очилганда қуйидаги хатолар келиб чиқиши мумкин: кўрсатилган файл мавжуд эмас (ўқиш режимида); диск тўла ёки ёзишдан ҳимояланган ва ҳоказо. Яна шуни айтиш керакки fopen() функцияси бажарилганда динамик хотира ишлатилади. Агар хотирада жой қолмаган бўлса “not enough memory ” - хатоси келиб чиқади.

Кўрсатилган холларда кўрсаткич NULL қийматга эга бўлади.

Бу хатолар ҳақидаги маълумотларни экранга чиқариш учун perror () функцияси ишлатилади. Бу функция stdio.h библиотекасида сақланувчи прототипи қуйидаги кўринишга эга.:

void perror(const char * s);

Дискда очилган файлларни беркитиш учун қуйидаги функциядан фойдаланилади.

int fclose (<оқимга-кўрсаткич номи>).

4.6. Файлга кетма-кет мурожаат қилиш

Файллар билан ишлашнинг битли режими. Файл билан битли алмашиш режими getc( ) ва putc( ) функциялари ёрдамида ташкил этилади. Бу функцияларга қуйидаги шаклда мурожаат этилади:

```
c = getc(fp);
putc(c,fp);
```

Бу ерда fp-кўрсаткич

c-int типидаги ўзгарувчи

Мисол тариқасида клавиатурадан символ киритиб файлга ёзишни кўрамиз. Матн охирини ‘#’ белгиси кўрсатади. Файл номи фойдаланувчидан сўралади. Агар <enter> клавишаси босилса файлга CR ва LF (қийматлари 13 ва 10) константалар ёзилади. Кейинчалик файлдан символларни ўқишда бу константалар сатрларни ажратишга имкон беради.

```
#include <stdio.h>
void main()
{
FILE *fp;
char c;
const char CR = '\015';
const char LF = '\012';
char fname[20];
puts("fayl nomini kiriting:\n");
gets(fname);
if((fp = fopen(fname, "w")) == NULL)
{
perror(fname);
return 1;
}
while ((c = getchar())! = '#')
{
if (c == '\n')
```

```
{ putc(CR,fp);
putc(LF,fp);
}
else putc (c,fp);
}
fclose(fp);
}
```

Кейинги программа файлдан символларни ўқиб экранга чиқаради:

```
#include <stdio.h>
#include<conio.h>
int main()
{
FILE *fp;
char c;
char fname[20];
puts("fayl nomini kiriting:\n");
gets(fname);
if((fp = fopen(fname, "r")) = = NULL)
{
perror(fname);
return 1;
}
while ((c = getc(fp))! = EOF)
putchar(c);
fclose (fp);
getch();
}
```

Сатрлар ёрдамида файллар билан боғланиш. Матнли файллар билан ишлаш учун fget ва fputs функцияларидан фойдаланилади. Бу функциялар прототиплари stdio.h файлида қуйидаги кўринишга эга:

int fputs (const char *s, FILE *stream);

char *fgets (char * s, int n, FILE * stream);

fputs функцияси ‘\0’ символи билан чегараланган сатрни stream кўрсаткичи орқали аниқланган файлга ёзади. ‘\0’ символи файлга ёзилмайди.

fgets() функцияси stream кўрсаткичи орқали аниқланган файлдан (n-1) символни ўқийди ва S кўрсатган сатрга ёзиб қўяди. Функция n-1 символни ўқиб бўлса ёки 1-чи қатор символи ‘\n’ни учратса ишини тўхтатади. Ҳар бир сатр охирига қўшимча ‘\0’ белгиси қўшилади. Хато бўлганда ёки файл охирига етганда агар файлдан бирорта символ ўқилмаган бўлса NULL қиймат қайтарилади.

Мисол тариқасида файл номи фойдаланувчидан сўраб яратувчи ва бу файлга иккита киритилган сўзни ёзиб қўювчи программани кўриб чиқамиз:

```
#include <stdio.h>
#include <conio.h>
void main()
{
FILE *fp;
char s[256];
char fname[20];
int n;
puts("fayl nomini kiriting:\n");
gets(fname);
if((fp = fopen(fname, "w")) == NULL)
{
perror(fname);
getch();
return;
}
for(n = 1; n<3;n++) {
gets(s);
fputs(s,fp);
fputs("\n",fp);
}
fclose(fp);
getch();
}
```

Кейинги мисолда номи киритилган файлдан мониторга ўқишни кўрамиз:

```
#include <stdio.h>
#include <conio.h>
void main()
{
FILE *fp;
char s[256];
char fname[20];
int n;
puts("fayl nomini kiriting:\n");
gets(fname);
if((fp = fopen(fname, "r")) == NULL)
{
perror(fname);
getch();
return;
}
for(n = 1; n<3;n++) {
fgets(s,256,fp);
puts(s);
```

```
}
fclose(fp);
getch();
}
```

Қуйидаги дастурда бир файлдаги матнни иккинчи файлга ёзишни кўриб чиқамиз.

Программа матни:

```
#include <stdio.h>
#include <conio.h>
void main()
{
FILE *f1,*f2;
char s[256];
char fname1[20];
char fname2[20];
puts("fayl nomini kiriting:\n");
gets(fname1);
if((f1 = fopen(fname1, "r")) == NULL)
{
perror(fname1);
getch();
return;
}
puts("fayl nomini kiriting:\n");
gets(fname2);
if((f1 = fopen(fname2, "w")) == NULL)
{
perror(fname2);
getch();
return;
}
while (fgets(s,256,f1)! = NULL)
fputs(s,f2);
fclose(f1);
fclose(f2);
getch();
}
```

Файллар билан форматли алмашинув. Кўп ҳолларда маълумотни тўғридан-тўғри мониторга чиқаришда қулай шаклда файлда сақлаш зарур бўлади. Бу ҳолда файлга форматли киритиш ва чиқариш функцияларидан фойдаланиш мумкин. Бу функциялар қуйидаги прототипларга эга:

int fprintf(оқимга кўрсаткич, форматлаш-қатори, ўзгарувчилар рўйхати);

int fscanf (оқимга кўрсаткич, форматлаш-қатори, ўзгарувчилар рўйхати);

Мисол тариқасида файл номи фойдаланувчидан сўраб яратувчи ва бу файлга 1 дан 100 гача бўлган сонларнинг символли тасвирини ёзиб қўювчи программани кўриб чиқамиз:

```
#include <stdio.h>
void main()
{
FILE *fp;
int n;
char fname[20];
puts("fayl nomini kiriting:\n");
gets(fname);
if((fp = fopen(fname, "w")) == NULL)
{
perror(fname);
return;
}
for(n = 1; n<11;n++)
fprintf(fp,"%d ",n);
fclose(fp);
}
```

Кейинги мисолда номи киритилган файлдан мониторга ўқишни кўрамиз:

```
#include <stdio.h>
#include <conio.h>
void main()
{
FILE *fp;
int n;
char fname[20];
puts("fayl nomini kiriting:\n");
gets(fname);
if((fp = fopen(fname, "r")) == NULL)
{
perror(fname);
getch();
return;
}
for(n = 1; n<11;n++) {
fscanf(fp,"%d ",&n);
printf("%d ",n);
}
fclose(fp);
getch();
}
```

Стандарт оқимга чиқариш. Кейинги дастурда сатрларни стандарт киритиш оқими яъни клавиатурадан киритиш ва стандарт чиқариш оқимига, мониторга чиқариш кўрсатилган:

```
#include<stdio.h>
#define MAXLINE 20
int main(void)
{
char line[MAXLINE];
while (fgets(line, MAXLINE, stdin) ! = NULL &&
line[0] ! = '\n')
fputs(line, stdout);
return 0;
}
```

Назорат саволлари

1. Структура нима?
2. Структура умумий кўринишини кўрсатинг.
3. Структуралар қандай инициализация қилинади?
4. Структура элементларига қандай мурожаат қилинади?
5. Структурага кўрсаткич таърифи.
6. Структура ҳажми қандай ўлчанади?
7. Структуралар таркибидаги массивлар элементларига қандай мурожаат қилинади?
8. Структура элементларига қандай қилинади?
9. Структуралар массиви қандай инициализация қилинади?
10. Бирлашмалар хоссаларини кўрсатинг.
11. Разрядли майдон деб қандай майдонларга айтилади?
12. Структуралар орасида муносабатлар.
13. Бирлашма нима?
14. Бирлашма ва структура орасидаги фарқ нима?
15. Файл нима?
16. Файлларга қандай мурожаат қилинади?
17. Файлларга маълумот қиритиш ва чиқариш?
18. Файлларга маълуомт ёзиш режимлари.
19. Функцияга кўрсаткич таърифи умумий кўриниши.
20. Функцияга кўрсаткич массивлари.

Адабиётлар рўйхати

1. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013, 488 b.

2. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. –СПб: ООО «ДиаСофтЮП», 2003.-560с.

3. Культин Н. Б. C/C++ в задачах и примерах. — СПб.: БХВ-Петербург, 2005. -288 с.

4. acm.tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.

Мустақил ишлаш учун мисоллар

1. Математик амаллардан фойдаланишни кўрсатувчи дастур тузинг.
2. Мантиқий амаллардан фойдаланишни кўрсатувчи дастур тузинг.
3. Нисбат амаллардан фойдаланишни кўрсатувчи дастур тузинг.
4. Муносабат амаллардан фойдаланишни кўрсатувчи дастур тузинг.
5. Сон абсолют қийматини шартли амал ёрдамида ҳисобловчи дастур тузинг.
6. Берилган eps аниқликда умумий хади 1/n! бўлган кетма-кетлик йиғиндисини ҳисобловчи дастур тузинг.
7. Умумий хади x/n! бўлган кетма-кетлик n та хади йиғиндисини ҳисобловчи дастур тузинг.
8. Киритилган n та сон қатъий ўсувчи эканлигини текширувчи дастур яратинг.
9. Киритилган n символдан нечтаси ўнли ҳарф эканлигини switch оператори ёрдамида ҳисобловчи дастур тузинг.
10. Рекурсия ёрдамида Паскаль учбурчагини ҳисобловчи функция тузинг. Бу функция ёрдамида учбурчакни экранга чиқарувчи функция тузиб дастурда фойдаланинг.
11. Берилган массив қатъий камаювчи эканлигини текширувчи функция тузинг ва дастурда фойдаланинг.
12. Матрицани векторга кўпайтириш функциясини тузинг ва дастурда фойдаланинг.
13. Берилган сатр телефон рақами эканлигини аниқловчи функция тузинг ва дастурда фойдаланинг.
14. Берилган сатр ўзгарувчи эканлиги текширувчи функция тузинг ва дастурда фойдаланинг.
15. Берилган жумладан энг қисқа сўз ажратиб олувчи функция тузинг ва дастурда фойдаланинг.
16. Абитуриент (исми, туғилган йили, йиққан бали, аттестат ўрта бали) структурасини яратинг. Структура типидаги массив яратинг.
17. Кўрсатилган рақамли элементни олиб ташланг ва кўрсатилган фамилияли элементдан сўнг элемент қўшинг.
18. Ходим (исми, лавозими, туғилган йили, ойлиги) структурасини яратинг. Структура типидаги массив яратинг.
19. Кўрсатилган фамилияли элементни олиб ташланг ва кўрсатилган рақамли элементдан сўнг элемент қўшинг.
20. Мамлакат (номи, пойтахти, ахоли сони, эгаллаган майдони) структурасини яратинг. Структура типидаги массив яратинг. Кўрсатилган аҳоли сонидан кичик бўлган элементни олиб ташланг ва кўрсатилган номга эга бўлган элементдан сўнг элемент қўшинг.
21. Давлат (номи, давлат тили, пул бирлиги, валюта курси) структурасини яратинг. Структура типидаги массив яратинг. Кўрсатилган номга эга бўлган элементни ўчиринг ва файл охирига иккита элемент қўшинг.

22. Инсон (исми, яшаш манзили, телефон рақами, ёши) структурасини яратинг. Структура типидаги массив яратинг. Кўрсатилган ёшга эга бўлган элементни ўчиринг ва берилган телефон рақамидаги элементдан олдин элемент қўшинг.

23. Берилган сатр ўзгарувчи эканлигини аниқловчи функция тузинг ва дастурда фойдаланинг. Функция танасида фақат кўрсаткичлар устида амаллардан фойдаланинг.

24. Матрицани векторга кўпайтириш функциясини яратиб дастурда фойдаланинг. Матрица кўрсаткичлар массиви сифатида киритилсин.

25. Динамик равишда ўқувчилар фамилиялари ва баҳолари массивларини ҳосил қилувчи функциялар яратинг. Ҳамма аълочилар фамилияларини чиқарувчи функция тузиб дастурда фойдаланинг.

26. Учбурчак динамик массив ёрдамида Паскаль учбурчагини ҳисобловчи функция тузинг. Бу функциядан ташқари учбурчакни экранга чиқарувчи функция тузиб дастурда фойдаланинг.

27. Дихотомия усули ёрдамида f(x) = 0 тенгламани ечиш учун функция тузинг. Функцияга кўрсаткич параметр сифатида узатилсин.

Фойдаланилган адабиётлар рўйхати

Асосий адабиётлар

1. Nazirov Sh.A., Qobulov R.V., Bobojanov M.R., Raxmanov Q.S. C va C++ tili. “Voris-nashriyot” MCHJ, Toshkent 2013, 488 b.
2. Қосимов С.С. Ахборот технологиялари. Техника олий ўрта юртлари бакалавриат босқичи талабалари учун ўқув қўлланма сифатида тавсия этилган. Тошкент, “Алоқачи” нашриёти, 2006 й.
3. Aripov M., Begalov B., Begimqulov U., Mamarajabov M. Axborot texnologiyalari. Toshkent: Noshir, 2009. -368 b.
4. G’ulomov S.S., Begalov B.A. Informatika va axborot texnologiyalari. Toshkent, Fan, 2010,-686c.

Қўшимча адабиётлар

1. Меньшиков Ф. В. Олимпиадные задачи по программированию. - СПб.: Питер, 2006. - 315 с.
2. Кнут Д. Искусство программирования. Том 1-4., СПб. Вильямс 2007.
3. Холзнер С. Visual C++ 6. Учебный курс. — СПб.: Питер, 2007. - 570 с.
4. Смайли Джон. Учимся программировать на C++ вместе с Джоном Смайли. –СПб: ООО «ДиаСофтЮП», 2003.-560с.
5. Культин Н. Б. C/C++ в задачах и примерах. — СПб.: БХВ-Петербург, 2005. -288 с.
6. Подбельский В. В., Фомин С. С. Программирование на языке Си: Учеб. пособие. - 2-е доп. изд. - М.: Финансы и статистика, 2004. - 600 с
7. Долинский М. С. Решение сложных и олимпиадных задач по программированию: Учебное пособие. — СПб.: Питер, 2006. — 366 с.

8. Павловская Т.С., Щупак Ю.С. С/С++. Структурное программирование. Практикум. -СПб.: Питер, 2007-240с

9. Павловская Т.С., Щупак Ю.С. С++. Объектно- ориентированное программирование. Практикум.-СПб.: Питер, 2005-265с.

10. Романов Б.А. Практикум по программированию на С++: Учебное пособие. СПб.: BXB-Петербург, Новосибирск: Из-во НГТУ, 2006.- 432с.

Электрон дарсликлар, ўқув қўлланмалар ва Интернет ресурслар

1. www.ziyonet.uz - Ўзбекистан Республикаси ахборот-таълим портали.
2. Martijn Koster "Robots in the Web: threat or treat?". http://info.webcrawler.com/mak/projects/robots/threat-or-treat.html
3. acm.tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.
4. neerc.ifmo.ru – Дастурлаш бўйича жаҳон чемпионатининг Шимолий Шарқий Европа Минтақасини расмий сайти.
5. icpc.baylor.edu - Дастурлаш бўйича жаҳон чемпионатининг расмий сайти.
6. acm.timus.ru – дастурий ечим тўғрилигини автоматик тестловчи тизим.
7. algo.urgench-tuit.uz - дастурий ечим тўғрилигини автоматик тестловчи тизим.
8. codeforces.com - дастурий ечим тўғрилигини автоматик тестловчи тизим.