Mihez:	Mi:	Megjegyzés:
alap típusok	int string bool	egész szöveg logikai
operátorok	= == < > (+,-,*,/)	értékadás egyenlőség vizsgálat kisebb nagyobb triviális
cin cout endl string ifstream ofstream	#include <iostream> using namespace std;	iostream sosem árt ha benn van mindenhez amit a CodeBlocks zöldel jelöl nem kötelező, ha nincs a használt parancsok elé std::irandó pl: std::cin

ciklusok	for(futóindex=kezdőérték;feltétel;futóindex_növelés)
	{ //ciklusmag }
	
	while(feltétel) { //ciklusmag }
	
	do { //ciklusmag } while(feltétel);
		Addig futnak amíg a feltétel igaz
		Használjuk: ha fontos hogy éppen hányadiknál járunk a futásokban
		
		Használjuk: ha nem tudjuk meddig kell fusson, tipikusan várunk valami történést
		
		Használjuk: ha egyszer mindenképpen szeretnénk hogy lefusson a ciklusmag, tipikusan előreolvasás

elágazás	if(feltétel){ //lefut ha igaz } else { //lefut ha hamis }	
random szám generálása	#include <cstdlib> #include <ctime>	srand(time(0)); kell a main elején utána rand()%(intervallum_hossza)+eltolás pl:[-50,50] rand()%101-50
beolvasás	cin >> valtozó; getline(honnan,hova);	változó: · ha szöveg típusú akkor az első szóközig olvas bele, ha nincs benne akkor a végéig · ha szám egy számot olvas be honnan: cin vagy ifstream neve hova: szöveg típusú változó
fájlkezelés	#include <fstream>	olvasás: ifstream v_neve(„fájlneve”); írás: ofstream v_neve(„fájlneve”); — szavanként olvasás: v_neve >> string_tipusu_valtozo; sor beolvasása: getline(v_neve, string_tipusu_valtozo);

n.hatványra emelés
négyzetgyökvonás
	#include <cmath>
	pow(kitevő,alap);
	sqrt(szam);
		négyzetre emelést érdemesebb önmagával megszorozva számolni ;) alap*alap;

szövegkezelés string szoveg1 = „kre a tiv”; string szoveg2 = „ vagyok”;	szoveg1.length(); szoveg1+szoveg2; szoveg2[szam]; (szam = 3)	Művelet: tárolt szöveg hosszának lekérdezése összefűzés számadik indexű karaktere a tárolt szövegnek — Eredmeny: 9 kre a tiv vagyok y
int és string konvertálás stringstream ss; int szam = 0; string sz = "25";	#include <sstream>	számból szöveget: ss << szam; sz = ss.str(); — szövegből számot: (csak ha számot tartalmaz) ss << szoveg; ss >> szam;

függvényírás	visszatérés_típusa fv_neve(bemeneti_paraméterek){
	//amit akarsz
	return visszatérés_tipusú_változó;
	}
		visszatérés_típusa: int,string,bool,void
		bemeneti_pm:
		nem kötelező, típus és változónév páros
		//amit akarsz:
		vigyázz más függvények változóit nem látod
		return:
		létezik visszatérési érték nélküli fv (void) ekkor nem kell
			Függvényhívás:
			fv_neve(bemeneti1,bemeneti2,...);
			fontos a bemeneti változók
			sorrendje,száma és típusa

tömb	tárolt_tipus tomb_neve [merete]	0 -> (meret-1) indexelhető	

vector
(méretét változtató tömb)
	#include <vector>
	vector <tárolt_típus> vektor_neve;
	vektor_neve.size();
	vektor_neve.push_back(érték);
	vektor_neve.pop_back();
	vektor_neve.erase(vektor_neve.begin()+n);
	vektor_neve.erase(vektor_neve.begin()+a,
	vektor_neve.begin()+b);
		Leírás:
		üres vektor létrehozása
		visszaadja a vektor méretét
		első üres eleme lesz az 'érték' (tárolt_típusú)
		törli az utolsó elemet
		törli az n. indexű elemet
		[a -> b) indexek között töröl minden elemet
		ebben a is benne van
			0 -> (size()-1) indexelhető
			
			a begin() a kezdeti helyét adja meg fontos: INDEXEDIK