# ՆԵՐԱԾՈՒԹՅՈՒՆ

Ժամանակակից ՀՏ-ի զարգացման ներկա մակարդակում մի շարք պատճառներով պրոցեսորների տակտային հաճախականությունն ավելացնել այլևս հնարավոր չէ: Դրա փոխարեն պրոցեսորների արտադրողականությունն սկսել են ավելացնել ոչ թե տակտային հաճախականության ավելացման շնորհիվ, այլ պրոցեսորի մեկ բյուրեղի վրա մի քանի հաշվողական միջուկների տեղադրման միջոցով՝ չփոխելով, նույնիսկ նվազեցնելով պրոցեսորի տակտային հաճախականությունը: Այդ պատճառով էլ ծրագրային միջոցների աշխատանքի արագագործության ավելացման համար հարկ է նորովի մոտենալ ծրագրային կոդի կազմակերպմանը՝ հարմարեցնելով կոդը բազմամիջուկային համակարգերին: Ներկայումս բազմամիջուկային պրոցեսորները սովորական երևույթ են դարձել: Դրա հետ մեկտեղ մշտապես աճում է ծրագրերի արտադրողականության բարձրացման պահանջը: Այդ պահանջի բավարարման համար էլ մշակվել է զուգահեռ ծրագրավորման տեխնոլոգիան։

Հաշվի առնելով զուգահեռ ծրագրավորման կարևորությունը և կիրառման արդիականությունը, կուրսային աշխատանքում դիտարկվել է զուգահեռ ծրագրավորման տեխնոլոգիայի առանձնահատկությունները։ Այս տեխնոլոգիայի կիրառմամբ մշակվել է ծրագիր ժամանակակից ամենահայտնի ծրագրավորման լեզուներից մեկով՝ C#-ով։ Մշակված ծրագիրը փորձում է միանալ տասնյակ հասցեների միաժամանակ, ամեն հասցեին համապատասխանեցնելով մեկ հոսք։ Դա զգալիորեն կրճատում է աշխատանքի կատարման ժամանակը և բարձրացնում է աշխատանքի արտադրողականությունը։

# ԶՈՒԳԱՀԵՌ ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԻՄՆԱԿԱՆ ՀԱՍԿԱՑՈՒԹՅՈՒՆՆԵՐԸ

Զուգահեռ ծրագրավորումը մի տեխնոլոգիա է, որն օգտագործում է բազմապրոցեսորային համակարգիչների կամ բազմամիջուկային պրոցեսորների առավելությունները: Այն առաջարկում է նույն ծրագրի տարբեր մասերի միաժամանակյա կատարման համար օգտագործել մի քանի հաշվողական սարքավորումներ: Զուգահեռ ծրագրավորումը ավելի լայն հասկացման՝ բազմահոսաքայնության (*multithreading*) բաղկացուցիչ մասն է կազմում:

Զուգահեռ հաշվարկներ ասելով կարելի է հասկանալ խնդիրների լուծման այնպիսի գործընթաց, որի դեպքում ժամանակի միևնույն պահին միաժամանակ կատարվում են մի քանի հաշվարկային գործողություններ: Զուգահեռ հաշվարկները քոմփյութերային հաշվարկների կատարման այնպիսի կազմակերպում է, որտեղ ծրագրերը մշակվում են որպես զուգահեռ աշխատող և միմյանց հետ փոխգործակցող հաշվարկային գործընթացների հավաքակազմ: Զուգահեռ հաշվարկները կարելի է իրականացնել բազմապրոցեսորային համակարգերի վրա՝ օգտագործելով մի քանի գործողությունների միաժամանակ կատարելու հնարավորությունը:

Զուգահեռ ծրագրավորման նպատակն է կրճատել խնդրի լուծման համար անհրաժեշտ մեքենայական ժամանակը: Գործնականում բավականին խնդիրներ կան, որոնք լուծում են պահանջում իրական ժամանակում, կամ նրանց լուծման համար կատարվում են հսկայածավալ հաշվարկներ: Զուգահեռ հաշվարկնրի կատարման խնդիրն է մշակել զուգահեռ ալգորիթմ, որը թույլ կտա առավելագույն արդյունավետությամբ օգտագործել բազմապրոցեսորային հաշվողական տեխնիկայի հնարավորությունները: Այդպիսի ալգորիթմ ստանալու համար անհրաժեշտ է զուգահեռացնել առկա հաջորդական ալգորիթմը կամ մշակել նոր զուգահեռ ալգորիթմ:

Զուգահեռ ծրագրավորման կիրառման պատճառներից մեկն այն է, որ մի շարք պատճառներով պրոցեսորների տակտային հաճախականությունն ավելացնել այլևս հնարավոր չէ: Դրա փոխարեն պրոցեսորների արտադրողականությունն սկսել են ավելացնել ոչ թե տակտային հաճախականության ավելացման շնորհիվ, այլ պրոցեսորի մեկ բյուրեղի վրա մի քանի հաշվողական միջուկների տեղադրման միջոցով՝ չփոխելով, նույնիսկ նվազեցնելով պրոցեսորի տակտային հաճախականությունը: Այդ պատճառով էլ ծրագրային միջոցների աշխատանքի արագագործության ավելացման համար հարկ է նորովի մոտենալ ծրագրային կոդի կազմակերպմանը՝ հարմարեցնելով կոդը բազմամիջուկային համակարգերին:

# ԶՈՒԳԱՀԵՌ ԾՐԱԳՐԱՎՈՐՄԱՆ ՏԵԽՆՈԼՈԳԻԱՆԵՐԸ

**Բազմահոսքային ծրագրավորում:** Ընդհանուր հիշողությամբ համակարգերում զուգահեռ ծարգրերի ստեղծման ընդհանուր միջոցը հոսքերի կիրառումն է: Զուգահեռացումը հեշտությամբ ստացվում է տարբեր հոսքային ֆունկցիաներ գրելու միջոցով, իսկ տվյալների զուգահեռացումն ապահովվում է ընդհանուր վիրտուալ հասցեային տարածության շնորհիվ: Ծրագրավորողը հոսքերի հետ կարող է աշխատել ինչպես օպերացիոն համակարգի API-ի միջոցով, այնպես էլ հոսքերի սեփական գրադարանների միջոցով:

Հոսքերի վրա հիմնված զուգահեռ ծրագրի մշակումը, հատկապես տվյալների զուգահեռացման դեպքում, ենթադրում է սինքրոնացման խնդրի լուծում, որի նպատակն է հաշվարկների կատարման ճշգրիտ արդյունքների ստացումը, օրինակ՝ չպետք է թույլ տալ մի հոսի կողմից տվյալների կարդացում այն պահին, երբ մի այլ հոսք այդ տվյալները գրանցում է հիշողության մեջ:

Օպերացիոն համակարգերը տրամադրում են միջոցներ բազմահոսքային ծրագրավորման ընթացքում առաջացող բոլոր խնդիրների լուծման համար, սակայն դրանց գրագետ օգտագործումը շատ ջանքեր է պահանջում:

**OpenMP տեխնոլոգիան**: Այս տեխնոլոգիան առաջարկում է ընհանուր հիշողությամբ համակարգերի համար զուգահեռ ծրագրերի մշակման ընթացքում առաջացող խնդիրների լուծումը դնել կոմպիլյատորի վրա, քանի որ այդ խնդիրները շատ դեպքերում իրար նման են: Այս տեխնոլոգիան առաջարկում է դիրեկտիվների հատուկ հավաքակազմ, ֆունկցիաների գրադարաններ և փոփոխականներ C, C++, Fortran լեզուների համար: OpenMP տեխնոլոգիան կարելի է համարել POSIX կամ Windows Threads հոսքերի գրադարանների բարձր մակարդակի իրականացում:

OpenMP տեխնոլոգիայի առավելություններն են.

* Օգտագործման պարզությունը. Ծրագրավորողը նոր զուգահեռ ծրագիր չի մշակում, այլ հաջորդական ծրագրի տեքստի մեջ ավելացնում է անհրաժեշտ դիրետիվներ կամ ֆունկցիաներ, որոնք կոմպիլյատորին ցույց են տալիս տվյալների և հաշվարկների զուգահեռացման միջոցները: OpenMP-ի հիմնական գաղափարն է զուգահեռացնել ցիկլերի կատարումը, որոնց վրա ընկած է հաշվարկների կատարման հիմնական ծանրությունը:
* Ճկունությունը. OpenMP-ն ծրագրավորողին զուգահեռ ծրագրի վարքը կառավարելու բավականին մեծ հնարավորություններ է ընձեռում:
* Կրկնակի օգտագործումը. OpenMP-ծրագիրը շատ դեպքերում կարելի է օգտագործել որպես սովորական հաջորդական ծրագիր, եթե անհրաժեշտ է այ կատարել մեկպրոցեսորային համակարգերում: Այդ դեպքում OpenMP-ի իրականացումից հրաժարվելու համար բավական է այն մեկ անգամ թողարկել հաջորդական կոմպիլյատորում: OpenMP-ի դիրեկտիվներն այս դեպքում կանտեսվեն:

Համապատասխան կոմպիլյատորներում OpenMP ստանդարտի միջոցով գրված ծրագրերը կարող են նույնքան արդյունավետ լինել, որքան բազմահոսքային ծրագրերը, սակայն OpenMP տեխնոլոգիան աշխատում է միայն SMP (սիմետրիկ բազմապրոցեսորային) համակարգերում:

**MPI տեխնոլոգիա** (**Message Passing Interface**  - հաղորդագրությունների փոխանակման ինտերֆեյս): Բաշխված հիշողությամբ համակարգերում զուգահեռ ծրագրավորումը օպերացիոն համակարգերի կողմից դեռևս բավականաչափ չի աջակցվում: MPI ստանդարտն առաջարկում է մեխանիզմ, որը հնարավորություն է տալիս զուգահեռաբար մեկ խնդիր կատարող պրոցեսների միջև հաղորդագրություններ փոխանակել: Այս տեխնոլոգիան նույնպես իրականացվում է C, C++, Fortran լեզուներում:

MPI ստանդարտն իրացումներ ունի գործնականորեն բոլոր սուպերկոմպյուտերային պլատֆորմների վրա, ինչպես նաև UNIX\Linux և Windows-ով աշխատանքային կայանների վրա: Ներկայումս MPI-ը առավել լայնորեն կիրառվող և դինամիկ զարգացող ինտերֆեյսն է:

MPI տեխնոլոգիայի առավելություններից են.

* Տեղափոխելիությունը: Այս տեխնոլոգիան թույլ է տալիս զգալիորեն նվազեցնել զուգահեռ ծրագրերի տեղափոխելիությունը տարբեր քոմփյութերային համակարգերի միջև: C կամ Fortran լեզվով և MPI տեխնոլոգիայի կիրառմամբ գրված զուգահեռ ծրագիրը կաշխատի գործնականորեն բոլոր համակարգիչներում, որտեղ ներդված է MPI ստանդարտի իրականացումը:
* Արդյունավետության բարձրացումը: MPI ստանդարտն օգնում է բարձրացնել զուգահեռ հաշվարկների կատարման արդյունավետությունը, քանի որ ներկայումս բոլոր տիպերի հաշվողական համակարգերի համար կա ստանդարտի իրականացումը, որը առավելագույնս հաշվի է առնում օգտագործվող ապարատային միջոցների հնարավորությունները:

MPI ծրագրերի կառուցման հիմնական մոտեցումը հաշվարկների և տվյալների հստակ բաշխումն է պրոցեսների միջև, ինչպես նաև հաղրդագրությունների փոխանակումն է տվյալների փոխանցման համար, ինչի պատճառով MPI ծրագիրը էականորեն տարբերվում է հաջորդական ծրագրից, իսկ որոշ դեպքերում նույնիս չի կարող իրականացվել մեկպրոցեսորային համակարգերում: Եվ իհարկե, MPI ծրագրի ստեղծումն ու շտկումն ավելի շատ ջանքեր է պահանջում, քան նույն խնդրի հաջորդական ծրագիրը:

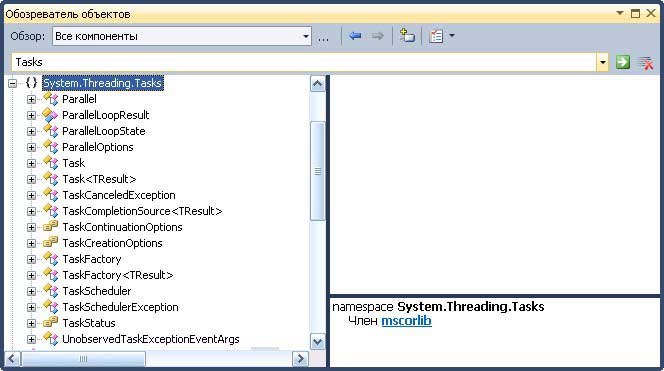
# Զուգահեռ ծրագրավորումը .NET Framework 4-ում

Նախկինում ալգորիթմների զուգահեռացումը պահանջում էր հոսքերի ղեկավարում և փոխադարձ արգելափակում ցածր մակարդակի վրա: *Visual Studio* 2010 և .*NET* Framework 4-ը բարելավել են զուգահեռ ծրագրավորման աջակցումը` շնորհիվ նոր տիպերի դասերի գրադարանների (*TPL*) և դիագնոստիկայի նոր միջոցների (*Concurrency* Visualizer): Այս միջոցները պարզեցնում են զուգահեռ ալգորիթմների մշակումը, ծրագրավորողին հնարավորություն են տալիս արդյունավետ, մասշտաբավորված զուգահեռ կոդ գրել բնական, արտահայտիչ միջոցներով, առանց հոսքերի հետ անմիջական աշխատանքի:

Խնդիրների զուգահեռացման գրադարանը (TPL) բարելավում է զուգահեռ ծրագրավորումը երկու միջոցներով. Առաջին հերթին այն պարզեցնում է շատ հոսքերի ստեղծումն ու կիրառումը, և երկրորդ՝ այն թույլ է տալիս ավտոմատ օգտագործել մի քանի պրոցեսորներ: Այլ կերպ ասած` TPL-ը բացահայտում է ծրագրերի մասշտաբավորման նոր հնարավորություններ՝ մատչելի պրոցեսորների արդյունավետ օգտագործման նպատակով: Այս երկու առանձնահատկությունների շնորհիվ TPL գրադարանն ավելի շատ է կիրառվում բազմահոսքային խնդիրների լուծման համար: .*NET* Framework 4-ում զուգահեռ ծրագրավորման ևս մեկ միջոց է ինտեգրացված հարցումների զուգահեռ լեզուն՝ PLINQ: Այն հնարավորություն է տալիս ստեղծել այնպիսի հարցումներ, որոնց մշակման համար ավտոմատ կերպով օգտագործվում են մի քանի պրոցեսորներ, ինչպես նաև հարկ եղած դեպքում՝ զուգահեռացման սկզբունքը:

Այսպիսի կարևոր նորամուծությունների գլխավոր պատճառը ժամանակակից ծրագրավորման մեջ զուգահեռացման կարևորությունն է: Ներկայումս բազմամիջուկային պրոցեսորները սովորական երևույթ են դարձել: Դրա հետ մեկտեղ մշտապես աճում է ծրագրերի արտադրողականության բարձրացման պահանջը: Այս ամենն, իր հերթին, առաջացրեց այնպիսի տեխնոլոգիայի մշակման անհրաժեշտություն, որը հնարավորություն կտա շահավետ օգտագործել մի քանի պրոցեսների կիրառման հնարավորությունը՝ ծրագրային ապահովման արտադրողականության բարձրացման համար:

Բանն այն է, որ նախկինում դա անելն այնքան էլ հեշտ չէր: Հենց դրա համար էլ մշակվել են TPL-ը և PLINQ-ը, հնարավորություն տալով ավելի անվտանգ և հեշտությամբ օգտագործել համակարգային ռեսուրսները: TPL գրադարանը սահմանված է System.Threading.Tasks անունների տարածությունում: Բայց դրա հետ աշխատելու համար պահանջվում է ներառել նաև System.Threading տարածությունը, քանի որ այն աջակցում է բազմահոսքային մշակման սինքրոնացմանը և այլ միջոցներին: Նկարում ցույց է տրված *NET* 4․0-ի անունների նոր տարածությունները։



# Ընդհանուր տեղեկություններ C# ծրագրավորման լեզվից

**C#**-ը (արտասանվում է *«սի շարփ»*) [օբյեկտ կողմոնորոշված](https://hy.wikipedia.org/wiki/%D5%95%D5%A2%D5%B5%D5%A5%D5%AF%D5%BF_%D5%AF%D5%B8%D5%B2%D5%B4%D5%B6%D5%B8%D6%80%D5%B8%D5%B7%D5%BE%D5%A1%D5%AE_%D5%AE%D6%80%D5%A1%D5%A3%D6%80%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4) [ծրագրավորման լեզու](https://hy.wikipedia.org/wiki/%D4%BE%D6%80%D5%A1%D5%A3%D6%80%D5%A1%D5%BE%D5%B8%D6%80%D5%B4%D5%A1%D5%B6_%D5%AC%D5%A5%D5%A6%D5%B8%D6%82) է։ Ստեղծվել է [Microsoft](https://hy.wikipedia.org/wiki/Microsoft) ընկերությունում [1998](https://hy.wikipedia.org/wiki/1998) - [2001](https://hy.wikipedia.org/wiki/2001) թվականներին՝ [Անդերս Հեյլսբերգի](https://hy.wikipedia.org/w/index.php?title=%D4%B1%D5%B6%D5%A4%D5%A5%D6%80%D5%BD_%D5%80%D5%A5%D5%B5%D5%AC%D5%BD%D5%A2%D5%A5%D6%80%D5%A3&action=edit&redlink=1" \o "Անդերս Հեյլսբերգ (դեռ գրված չէ)) ղեկավարած ինժիներական խմբում՝ [Microsoft .NET Framework](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) հարթակի ծրագրերի մշակման նպատակով և ավելի ուշ համապատասխանեցվել է[ECMA](https://hy.wikipedia.org/w/index.php?title=ECMA&action=edit&redlink=1)-334 և [ISO](https://hy.wikipedia.org/wiki/ISO)/[IEC](https://hy.wikipedia.org/w/index.php?title=IEC&action=edit&redlink=1) 23270 չափորոշիչներին։

C#-ը ծրագրավորման [C](https://hy.wikipedia.org/wiki/C_(%D5%AE%D6%80%D5%A1%D5%A3%D6%80%D5%A1%D5%BE%D5%B8%D6%80%D5%B4%D5%A1%D5%B6_%D5%AC%D5%A5%D5%A6%D5%B8%D6%82)) ընտանիքի լեզու է, որի շարահյուսությունն ավելի մոտ է [C++](https://hy.wikipedia.org/wiki/C%2B%2B) և [Java](https://hy.wikipedia.org/wiki/Java" \o "Java) լեզուներին։ Լեզուն ունի [ստատիկ տիպավորում](https://hy.wikipedia.org/wiki/%D5%8D%D5%BF%D5%A1%D5%BF%D5%AB%D5%AF_%D5%BF%D5%AB%D5%BA%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4), աջակցում է[պոլիմորֆիզմ](https://hy.wikipedia.org/wiki/%D5%8A%D5%B8%D5%AC%D5%AB%D5%B4%D5%B8%D6%80%D6%86%D5%AB%D5%A6%D5%B4_(%D5%AE%D6%80%D5%A1%D5%A3%D6%80%D5%A1%D5%BE%D5%B8%D6%80%D5%B8%D6%82%D5%B4)), [օպերատորի գերբեռնում](https://hy.wikipedia.org/w/index.php?title=%D5%95%D5%BA%D5%A5%D6%80%D5%A1%D5%BF%D5%B8%D6%80%D5%AB_%D5%A3%D5%A5%D6%80%D5%A2%D5%A5%D5%BC%D5%B6%D5%B8%D6%82%D5%B4&action=edit&redlink=1) (այդ թվում բացահայտ և թաքնված տիպի օպերատորներ), իրադարձություններ, հատկություններ, տեսակների և մեթոդների ընդհանրացում, իտերացիա, անանուն ֆունկցիաներ, [LINQ](https://hy.wikipedia.org/w/index.php?title=Language_Integrated_Query&action=edit&redlink=1), բացառում, [XML](https://hy.wikipedia.org/wiki/XML) ֆորմատի մեկնաբանություն։

Լեզվի անվանումը

[](https://hy.wikipedia.org/wiki/%D5%8A%D5%A1%D5%BF%D5%AF%D5%A5%D6%80:Music-sharp.png)

«Սի շարփ» անվանումը ([անգլ.](https://hy.wikipedia.org/wiki/%D4%B1%D5%B6%D5%A3%D5%AC%D5%A5%D6%80%D5%A5%D5%B6)՝ sharp - դիեզ) գալիս է երաժշտական [նոտագրությունից](https://hy.wikipedia.org/wiki/%D5%86%D5%B8%D5%BF%D5%A1%D5%A3%D6%80%D5%B8%D6%82%D5%A9%D5%B5%D5%B8%D6%82%D5%B6" \o "Նոտագրություն), որտեղ դիեզը երաժշտական նշան է, որը ցույց է տալիս ձայնի կես աստիճան բարձրացում,  ինչպես օրինակ [«C++»](https://hy.wikipedia.org/wiki/C%2B%2B)-ի անվանման մեջ «++»-ը նշանակում է փոփոխականի արժեքի ավելացում մեկ միավորով (ինկրեմենտ)։ Անվանումը հանդիսանում է նաև շղթայի շարունակությունը՝ C → C++ → C++++(C#), քանի որ «#» նշանը կարելի է ստանալ 4 հատ «+»-ով։

Լեզվի անվանումն ընդունված չէ թարգմանել, ուստի խորհուրդ է տրվում օգտագործել [անգլերեն](https://hy.wikipedia.org/wiki/%D4%B1%D5%B6%D5%A3%D5%AC%D5%A5%D6%80%D5%A5%D5%B6) C Sharp (Սի շարփ) անվանումը։

Տարբերակները

C# լեզվի մշակման արդյունքում թողարկվել են նրա մի քանի տարբերակներ.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Տարբերակ** | **Լեզվի չափանիշների համակարգ** | | | **Ամսաթիվ** | [**.NET Framework**](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) | [**Visual Studio**](https://hy.wikipedia.org/w/index.php?title=Visual_Studio&action=edit&redlink=1) |
| [**ECMA**](https://hy.wikipedia.org/w/index.php?title=Ecma_International&action=edit&redlink=1) | [**ISO/IEC**](https://hy.wikipedia.org/w/index.php?title=ISO/IEC&action=edit&redlink=1) | **Microsoft** |
| C# 1.0 | [Դեկտեմբեր 2002](http://www.ecma-international.org/publications/files/ECMA-ST-WITHDRAWN/ECMA-334,%202nd%20edition,%20December%202002.pdf) | [Ապրիլ 2003](http://download.www.techstreet.com/cgi-bin/pdf/free/378672/ISO+IEC+23270-2003.pdf) | [Հունվար 2002](http://download.microsoft.com/download/a/9/e/a9e229b9-fee5-4c3e-8476-917dee385062/CSharp%20Language%20Specification%20v1.0.doc) | Հունվար 2002 | [.NET Framework 1.0](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) | [Visual Studio .NET 2002](https://hy.wikipedia.org/w/index.php?title=Microsoft_Visual_Studio&action=edit&redlink=1) |
| C# 1.2 | [Հոկտեմբեր 2003](http://download.microsoft.com/download/5/e/5/5e58be0a-b02b-41ac-a4a3-7a22286214ff/csharp%20language%20specification%20v1.2.doc) | Ապրիլ 2003 | [.NET Framework 1.1](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) | [Visual Studio .NET 2003](https://hy.wikipedia.org/w/index.php?title=Microsoft_Visual_Studio&action=edit&redlink=1) |
| C# 2.0 | [Հունիս 2006](http://www.ecma-international.org/publications/files/ECMA-ST/Ecma-334.pdf) | [Սեպտեմբեր 2006](http://standards.iso.org/ittf/PubliclyAvailableStandards/c042926_ISO_IEC_23270_2006(E).zip) | [Սեպտեմբեր 2005](http://download.microsoft.com/download/9/8/f/98fdf0c7-2bbd-40d3-9fd1-5a4159fa8044/csharp%202.0%20specification_sept_2005.doc)[[4]](https://hy.wikipedia.org/wiki/C_Sharp#cite_note-note-4) | Նոյեմբեր 2005 | [.NET Framework 2.0](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) | [Visual Studio 2005](https://hy.wikipedia.org/w/index.php?title=Microsoft_Visual_Studio&action=edit&redlink=1) |
| C# 3.0 | բացակայում է[[5]](https://hy.wikipedia.org/wiki/C_Sharp#cite_note-autogenerated1-5) | | [Օգոստոս 2007](http://download.microsoft.com/download/3/8/8/388e7205-bc10-4226-b2a8-75351c669b09/CSharp%20Language%20Specification.doc) | Նոյեմբեր 2007 | [.NET Framework 3.5](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) | [Visual Studio 2008](https://hy.wikipedia.org/w/index.php?title=Microsoft_Visual_Studio&action=edit&redlink=1) |
| C# 4.0 | [Ապրիլ 2010](http://www.microsoft.com/downloads/details.aspx?displaylang=en&FamilyID=dfbf523c-f98c-4804-afbd-459e846b268e) | Ապրիլ 2010 | [.NET Framework 4](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) | [Visual Studio 2010](https://hy.wikipedia.org/w/index.php?title=Microsoft_Visual_Studio&action=edit&redlink=1) |
| C# 5.0 |  | Օգոստոս 2012 | [.NET Framework 4.5](https://hy.wikipedia.org/w/index.php?title=.NET_Framework&action=edit&redlink=1) | [Visual Studio 2012](https://hy.wikipedia.org/w/index.php?title=Microsoft_Visual_Studio&action=edit&redlink=1) |

# Զուգահեռ ծրագրավորման տեխնոլոգիայի կիրառմամբ խնդրի լուծում C# ծրագրավորման լեզվով

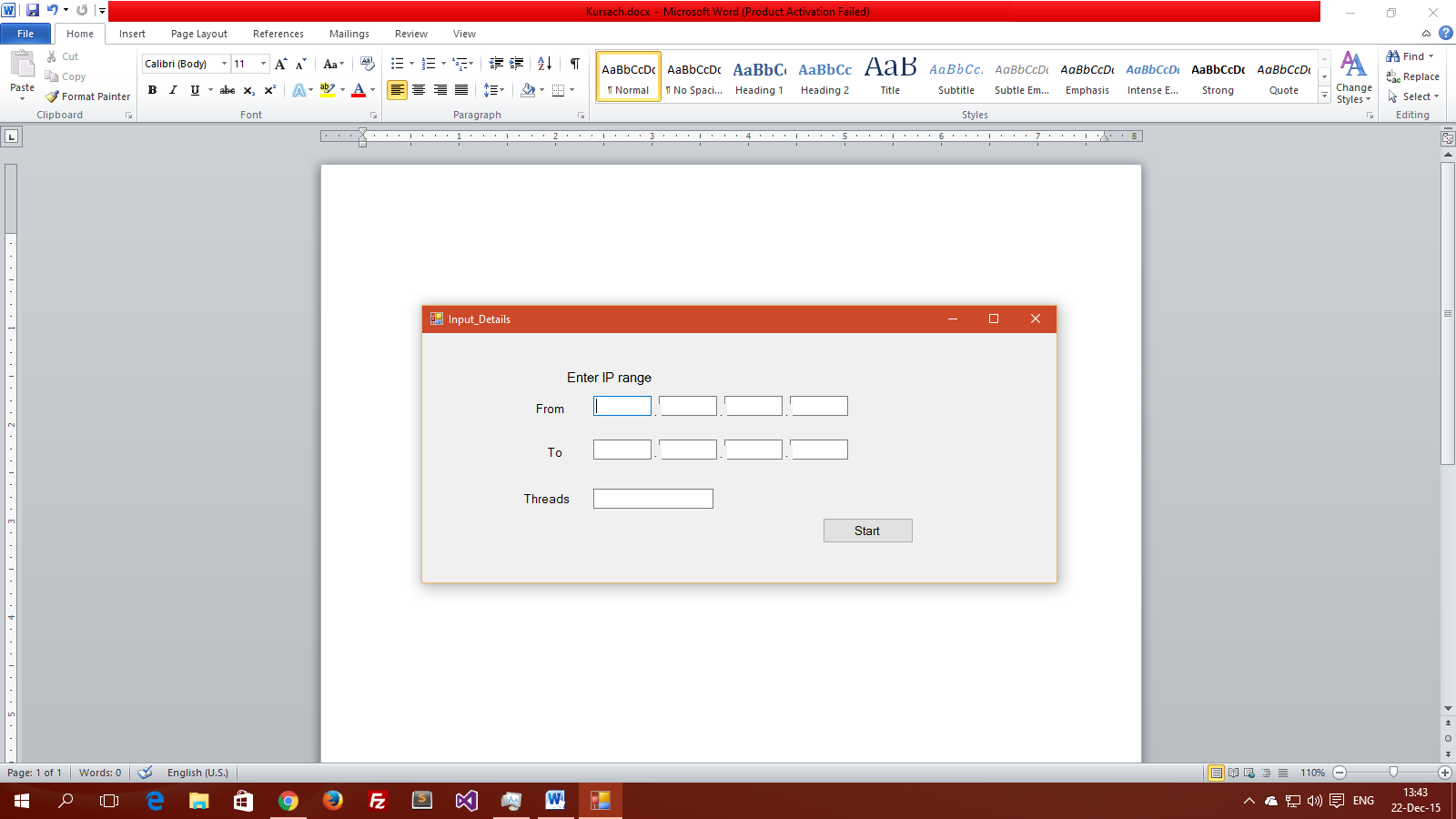
Ծրագիրը նախատեսված է նշված IP-ների միջակայքում գտնվող բոլոր հնարավոր հասցեների տարբերակներով համակարգիչներ գտնելու համար։ Նշեմ, որ այն գտնում է այն համակարգիչները, որոնք ունեն իրենց վրա տեղադրված ու աշխատող WEB սերվեր։ Ծրագիրը գրվել է C# ծրագրավորման լեզվով։ Օգտագործվել են այս ծրագրավորման լեզվի ստանդարտ գրադարանները և Windows օպերացիոն համակարգի WindowsForms գրադարանը։

Ծրագիրն աշխատացնելու համար բացեք այս փաստաթղթին կից սկավառակի մեջ Parallels.exe ֆայլը (Նկ․ 1).



Նկ․ 1

Ծրագիրը բացելուց կհայտնվի այսպիսի պատուհան (Նկ․ 2)



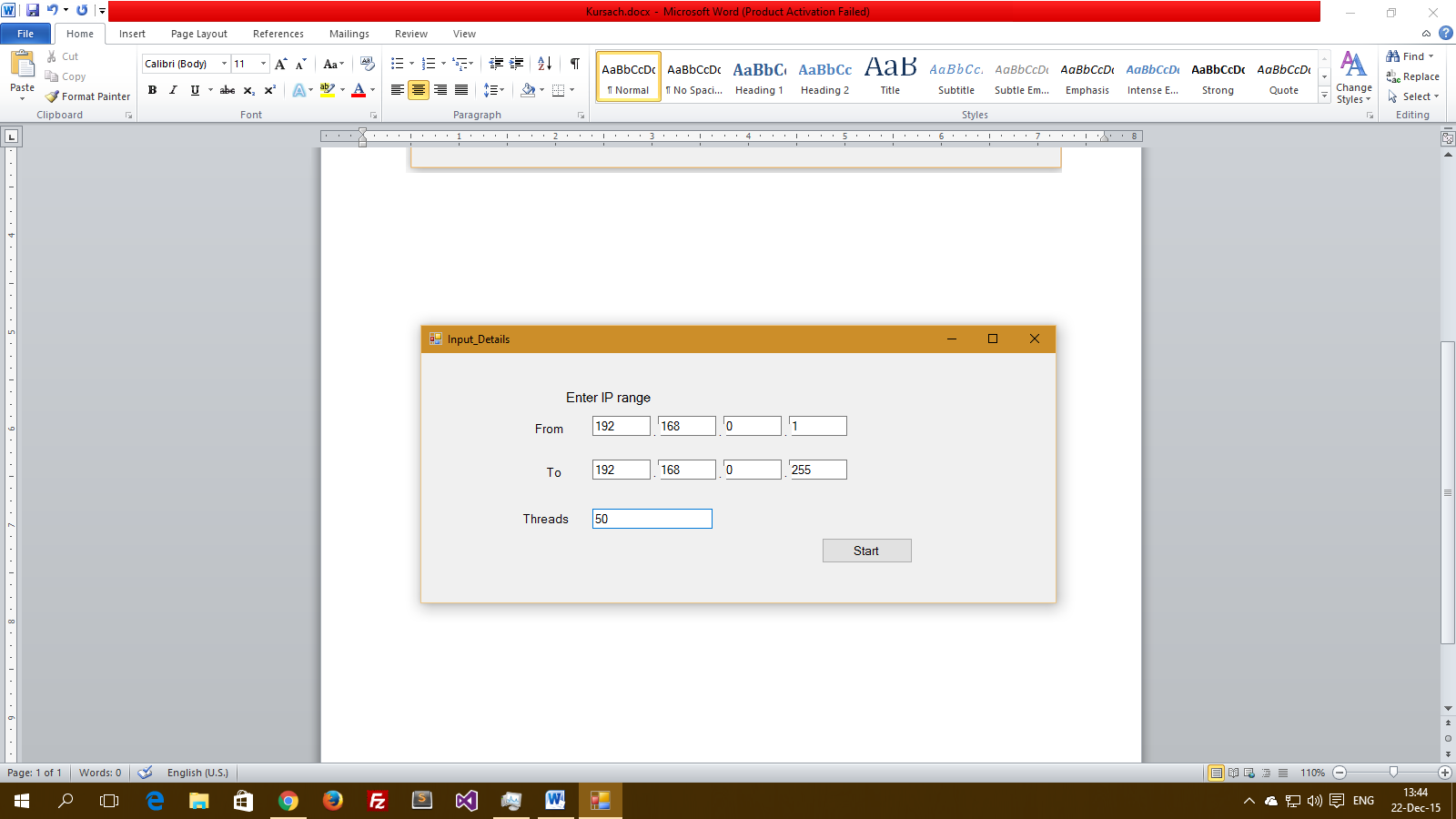
Նկ․ 2

From – Այս տողում պետք է գրվի IP դիապազոնի առաջին IP հասցեն

To – Այս տողում պետք է գրվի IP դիապազոնի վերջին IP հասցեն

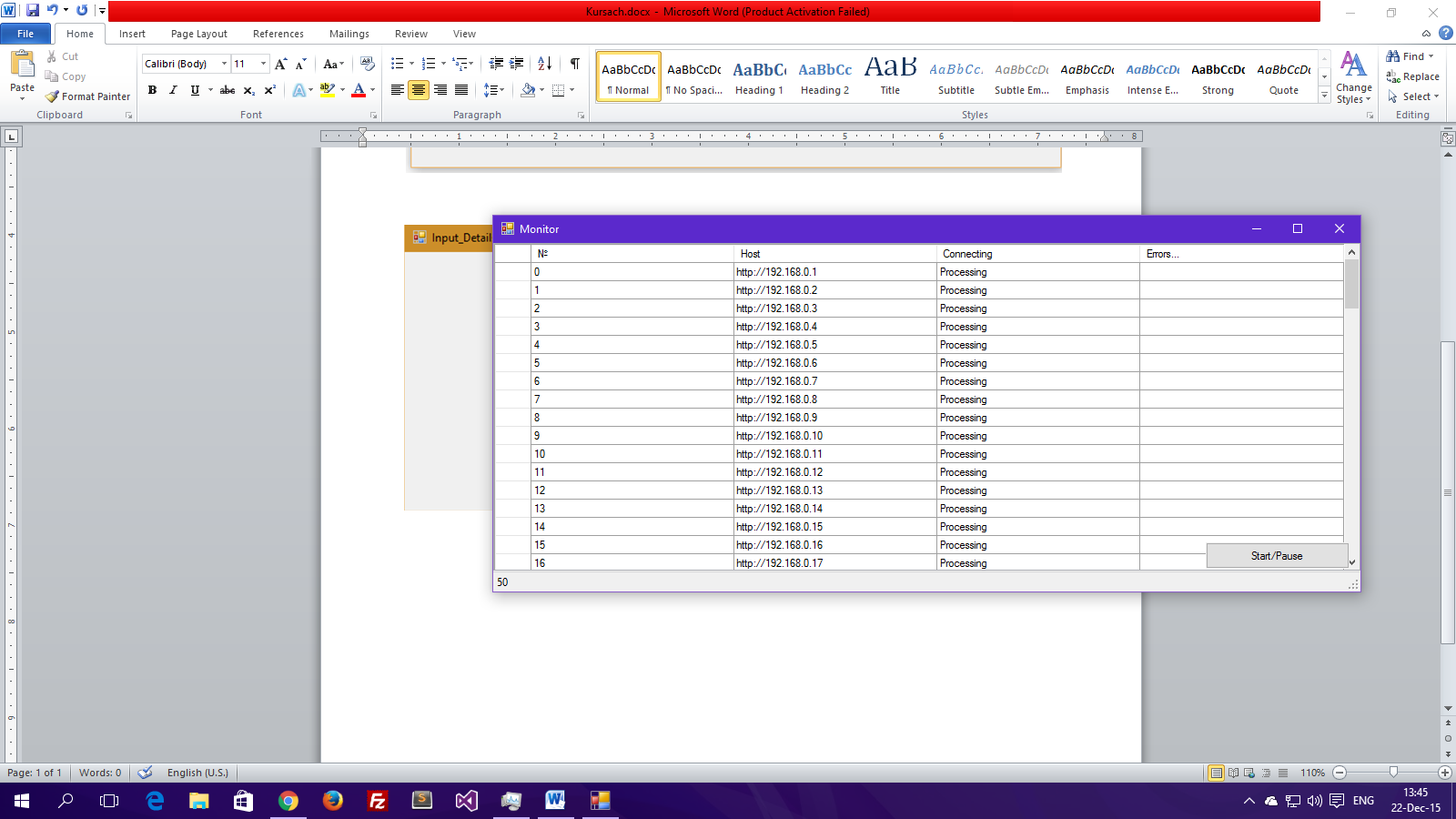
Threads – Այս տողը նախատեսված է ծրագրի հոսքերի\* քանակը տալու համար

\* հոսքը նախատեսված է միաժամանակ մի քանի հասցեներին միանալու փորձ կատարելու համար



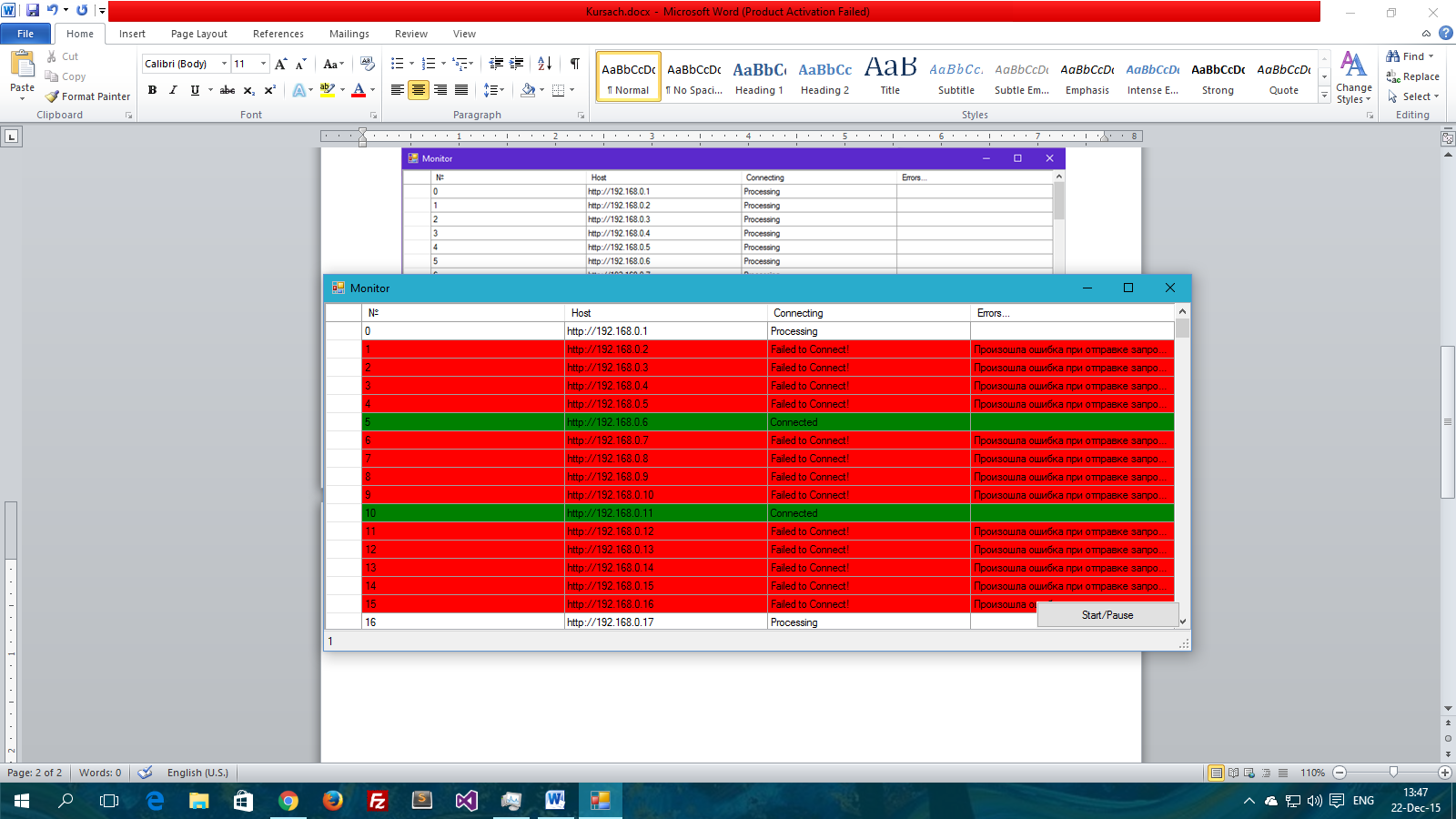
Նկ․ 3

Տողերը մուտք անելուց (Բոլոր տողերը պարտադիր են) հետո սղմեք Start կոճակը (Նկ․ 3)։ Օրինակի համար ես մուտքագրել եմ իմ համակարգչի լոկալ ցանցի հասցեները։ Դրանից հետո կբացվի այսպիսի մի պատուհան (Նկ․ 4)։



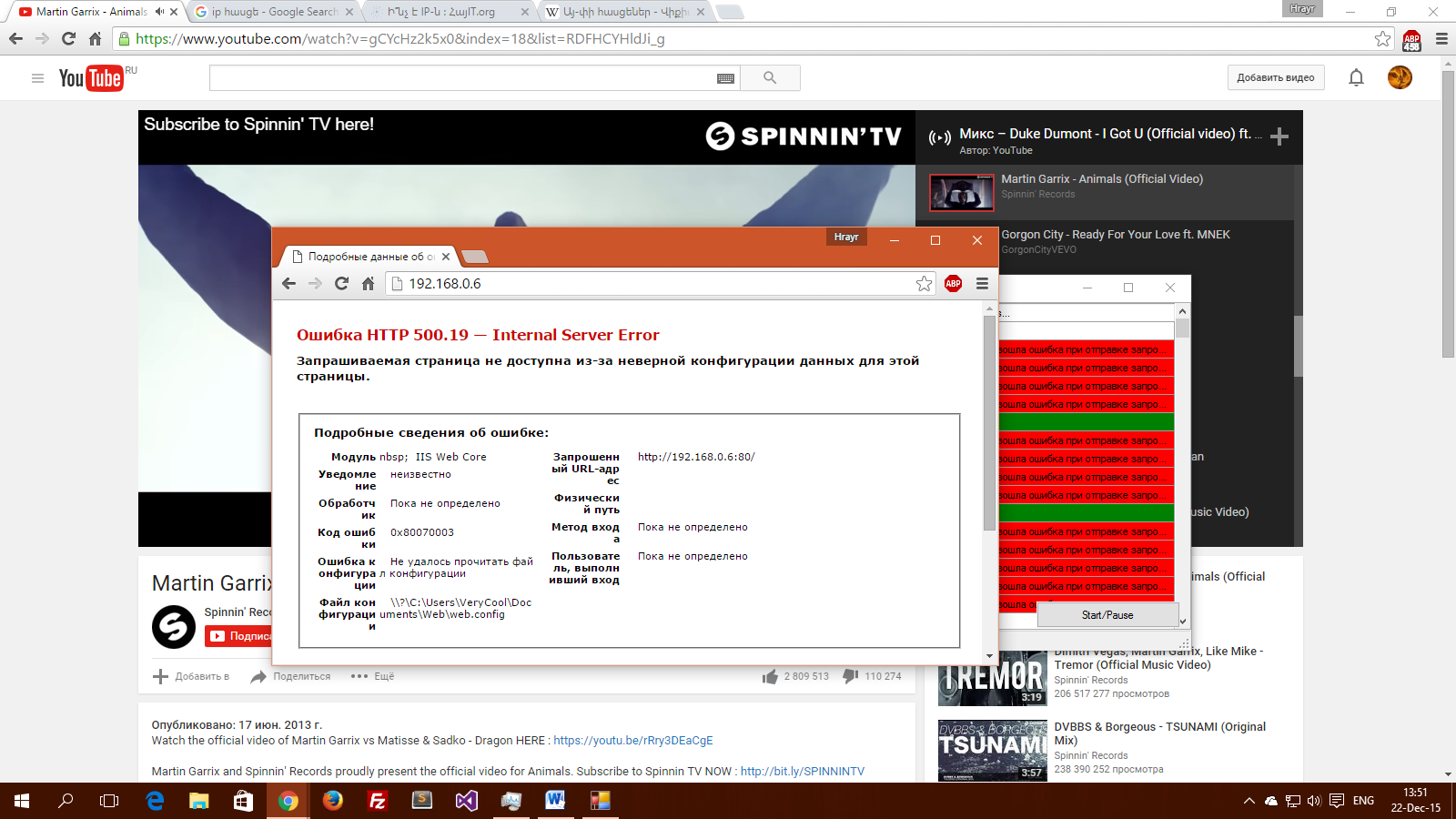
Նկ․ 4

Այժմ մեր ծրագիրը սկսել է աշխատել։ Ժամանակի ընթացքում նա գեներացնում է հոսքերի քանակով հասցեներ և փորձում է կապ հաստատել նրանց հետ (Նկ․ 5)։



Նկ․ 5

Ինչպես տեսնում եք նկարում տողերի մի մասը կարմրել են, մի մասը կանաչել։ Կանաչ տողերում գրված են հաջողությամբ կապ հաստատված հասցեները։ Օրինակի համար ես կփորձեմ իմ բրաուզերի միջոցով միանալ հաջողված հասցեներից մեկին (<http://192.168.0.6> – Նկ․ 6)։



Նկ․ 6

Խնդրի հիմնական աշխատող կոդը բերված է ներքև։ Ծրագրի ամբողջ կոդը դուք կարող եք գտնել այս փաստաթղթին կից սկավառակի մեջ։

# ԾՐԱԳՐԻ ԿՈԴ

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Net.Http;

using System.Net;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

namespace Parallels

{

public partial class Monitor : Form

{

public static int thr\_quantity = 0;

public static bool th\_pause = false;

public static bool running = true;

public Monitor()

{

InitializeComponent();

Task.Factory.StartNew(delegate

{

while (true) { this.view\_quantity(thr\_quantity); Thread.Sleep(100); }

});

}

public void start\_load(int[] ipfrom,int[] ipto,int thrq) {

int[] ip\_range = new int[4];

button1.Anchor = (AnchorStyles.Bottom | AnchorStyles.Right);

if (ipfrom == null)

{

ip\_range[0] = 0;

ip\_range[1] = 0;

ip\_range[2] = 0;

ip\_range[3] = 0;

}

else

ip\_range = ipfrom;

int n=thrq;

int k = 0;

String host;

Thread[] t = new Thread[n + 1];

Task.Factory.StartNew(delegate {

while (running)

{

Thread.Sleep(500);

BeginInvoke((MethodInvoker)delegate

{

if (thr\_quantity < n && !th\_pause)

{

for (int i = 0; i < n && running; i++)

{

if (t[i] != null && t[i].ThreadState == ThreadState.Aborted)

{

t[i] = null;

}

if (t[i] == null || t[i].ThreadState == ThreadState.Stopped || t[i].ThreadState == ThreadState.Unstarted)

{

host = "http://" + ip\_range[0] + "." + ip\_range[1] + "." + ip\_range[2] + "." + ip\_range[3];

dataGridView1.Rows.Add(k.ToString(), host, "Processing", "");

t[i] = new Thread(() => HTTP\_GET(host, k));

t[i].Start();

for (int p=3;p>0;p--)

{

if (ip\_range[p] > ipto[p])

{

ip\_range[p] = 0;

ip\_range[p - 1]++;

}

if (ip\_range[0]>ipto[0])

running = false;

}

ip\_range[3]++;

k = k + 1;

thr\_quantity = thr\_quantity + 1;

}

else if (t[i].ThreadState != ThreadState.Stopped || t[i].ThreadState != ThreadState.Unstarted)

{

t[i].Join();

}

}

}

});

}

});

}

public void view\_quantity(int d) {

BeginInvoke((MethodInvoker)delegate

{

thrqnt.Text = d.ToString();

});

}

public void success(int i)

{

BeginInvoke((MethodInvoker)delegate

{

for (int l = 0; l <= 3; l++)

dataGridView1.Rows[i].Cells[l].Style.BackColor = Color.Green;

dataGridView1.Rows[i].Cells[2].Value = "Connected";

});

}

public void failed(int i, string message)

{

BeginInvoke((MethodInvoker)delegate

{

for(int l=0;l<=3;l++)

dataGridView1.Rows[i].Cells[l].Style.BackColor = Color.Red;

dataGridView1.Rows[i].Cells[2].Value = "Failed to Connect!";

dataGridView1.Rows[i].Cells[3].Value = ""+message;

});

}

public async void HTTP\_GET(string host, int count)

{

var TARGETURL = host;

try

{

HttpClientHandler handler = new HttpClientHandler();

HttpClient client = new HttpClient(handler);

var byteArray = Encoding.ASCII.GetBytes("admin:admin");

client.DefaultRequestHeaders.Authorization = new System.Net.Http.Headers.AuthenticationHeaderValue("Basic", Convert.ToBase64String(byteArray));

HttpResponseMessage response = null;

HttpContent content = null;

response = await client.GetAsync(TARGETURL);

content = response.Content;

string result = await content.ReadAsStringAsync();

if (result != null && result.Length > 0)

{

success(count);

}

}

catch (HttpRequestException exc)

{

failed(count, exc.Message);

}

catch (TaskCanceledException exc)

{

failed(count, exc.Message);

}

thr\_quantity--;

}

private void button1\_Click(object sender, EventArgs e)

{

th\_pause = th\_pause ? false : true;

}

}

}

# ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ

Կուրսային աշխատանքում դիտարկվել է զուգահեռ ծրագրավորման տեխնոլոգիայի կիրառությունը։ Խնդրի ալգորիթմը մշակվել է հաշվի առնելով զուգահեռացման սկզբունքները և հոսքերի կիրառությունը։ Զուգահեռ ծրագրավորումը արմատապես նոր մոտեցում է և հնարավորություն է տալիս առավելագույնս օգտագործել ժամանակակից բազմամիջուկային պրոցեսորների տեխնիկական հնարավորությունները։

Կուրսային աշխատանքում մշակված ծրագիրը առավել արդյունավետ է աշխատում մեծ միջակայքերի հետ, քանի որ զուգահեռացման շնորհիվ այն կրկնակի անգամ կրճատում է ծրագրի աշխատանքի ժամանակը։

# ԳՐԱԿԱՆՈՒԹՅՈՒՆ

1․ Шилдт Герберт. Полный справочник по C#. : Пер. с англ. — М. : Издательский дом “Вильямс”, 2004. — 752 с. : ил. — Парал. тит. англ.

2. Сысоев А.В., Мееров И.Б., Свистунов А.Н., Курылев А.Л., Сенин А.В., Шишков А.В., Корняков К.В., Сиднев А.А. « Параллельное программирование в системах с общей памятью. Инструментальная поддержка». Учебно-методические материалы по программе повышения квалификации « Технологии высокопроизводительных вычислений для обеспечения учебного процесса и научных исследований». Нижний Новгород, 2007, 110 с.

Оглавление

[ՆԵՐԱԾՈՒԹՅՈՒՆ 1](#_Toc438579789)

[ԶՈՒԳԱՀԵՌ ԾՐԱԳՐԱՎՈՐՄԱՆ ՀԻՄՆԱԿԱՆ ՀԱՍԿԱՑՈՒԹՅՈՒՆՆԵՐԸ 2](#_Toc438579790)

[ԶՈՒԳԱՀԵՌ ԾՐԱԳՐԱՎՈՐՄԱՆ ՏԵԽՆՈԼՈԳԻԱՆԵՐԸ 2](#_Toc438579791)

[Զուգահեռ ծրագրավորումը .NET Framework 4-ում 4](#_Toc438579792)

[Ընդհանուր տեղեկություններ C# ծրագրավորման լեզվից 6](#_Toc438579793)

[Տարբերակները 7](#_Toc438579796)

[Զուգահեռ ծրագրավորման տեխնոլոգիայի կիրառմամբ խնդրի լուծում C# ծրագրավորման լեզվով 8](#_Toc438579797)

[ԾՐԱԳՐԻ ԿՈԴ 11](#_Toc438579798)

[ԵԶՐԱԿԱՑՈՒԹՅՈՒՆ 13](#_Toc438579799)

[ԳՐԱԿԱՆՈՒԹՅՈՒՆ 13](#_Toc438579800)