НАСТРОЙВАНЕ НА ПРОГРАМИ В ИНТЕГРИРАНАТА СРЕДА
MS Visual C++

1. Цел на упражнението

Усвояване на технологията за работа с инструменталното средство за настройване на програми, вградено в интегрираната среда **MS Visual C++**.

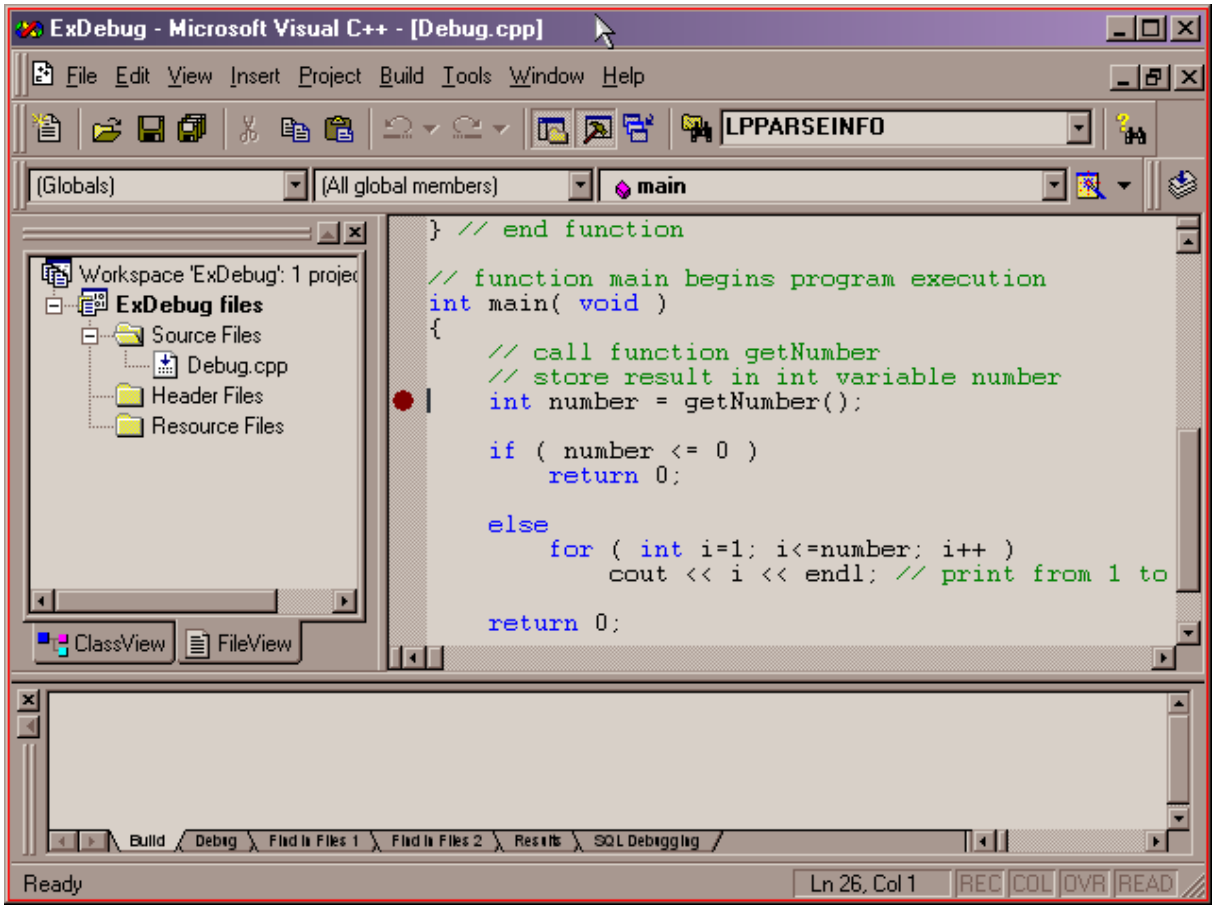
2. Основни възможности на инструменталното средство

В интегрираната среда за разработване на програми Visual C++ е вградено инструменталното средство **(DEBUGGER),** което се използва за настройване на разработените приложения. То подпомага работата на програмиста при откриване на логически грешки в кода на програмите. При този тип грешки фазите компилация и свързващо редактиране са завършили успешно, но програмата не изпълнява дефинираните функции и не се получават очакваните резултати. **Debugger**-ът позволява на програмиста да прегледа изпълнението на програмата и управлението на използваните данни стъпка по стъпка или като цялостно изпълнение. Изпълнението на програмата спира върху избрания ред от кода или върху фатална грешка по време на изпълнение (*run-time error*). Когато програмистът не може да разбере причините за некоректните резултати при изпълнение на програмата, чрез последователното (по стъпково) изпълнение на операторите от програмата той може да анализира и да идентифицира причините за грешните резултати и да коригира първичния код на програмата.

При използване на **debugger**-а може да се поставят една или повече *точки на прекъсване* (***breakpoints***). *Точката на прекъсване* е маркер, който е поставен на определен ред от кода на програмата и предизвиква **debugger-a** да прекъсне изпълнението на програмата при достигане до този ред. *Точките на прекъсване* помагат на програмиста да провери коректността на програмата. Точка на прекъсване може да бъде поставена по два начина:

- Чрез кликване върху реда от програмата, където трябва да бъде поставена точката и след това кликване върху бутона **<Insert/Remove Breakpoint>** от лентата **Build MiniBar**. Активирането на бутона **<Insert/Remove Breakpoint>** става чрез активиране на съответния прозорец, съдържащ първичния код на програмата.
- Чрез натискане на клавиша F9.

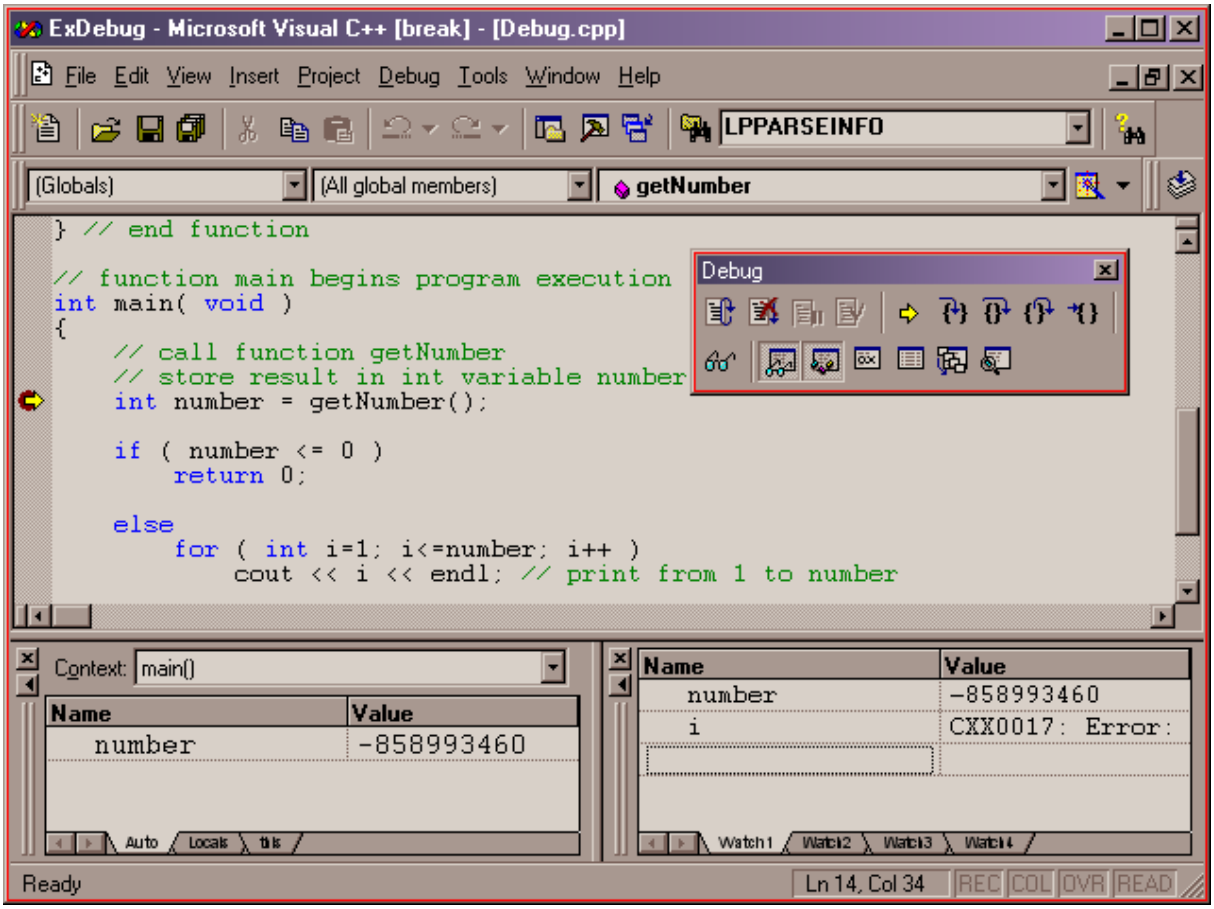
Когато точката на прекъсване е поставена, се появява червена точка в лентата вляво на реда от първичния код (фиг.1А.1). Точките на прекъсване се премахват, така както те се поставят.



Фиг. 1А.1. Поставяне точка на прекъсване

Процесът на настройване се стартира чрез командата **Go** (от менюто **Build -> Start Debug**). Появява се конзолен прозорец, тъй като ще бъде демонстриран процесът на настройване на конзолно приложение. Цялото взаимодействие (вход/изход) с програмата се изпълнява в този прозорец. Изпълнението на програмата се преустановява при настъпване на вход и на точка на прекъсване. Възможно е ръчно да се извърши превключване между интегрираната среда и конзолния прозорец, за да се осъществи въвеждането на данните. Това може да се извърши чрез **<Alt>+<Tab>** или чрез кликване върху съответния прозорец.

Жълтата стрелка вляво от оператора показва, че изпълнението е преустановено на този ред (фиг. 1А.2).



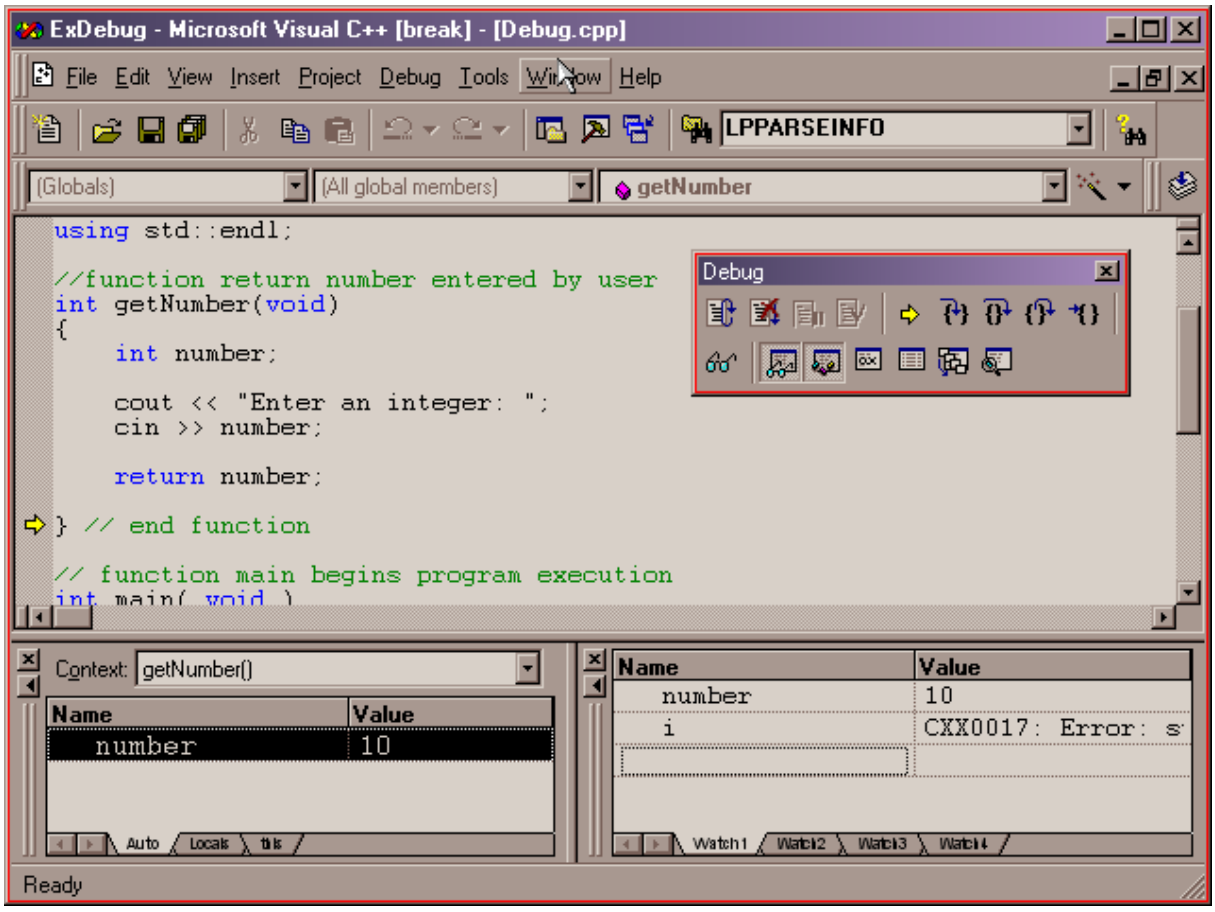
Фиг. 1А.2. Преустановено изпълнение в точка на прекъсване

Трябва да се има предвид, че в интегрираната среда менюто **Build** се заменя с менюто **Debug** и в заглавната лента на прозореца се изписва **[break]**, което показва, че средата е в режим на настройване. Цикли, които се изпълняват много пъти, могат да бъдат изпълнени като едно цяло (без да се спира всеки път в цикъла) чрез разполагане точката на прекъсване след цикъла избиране на командата **Go** от менюто **Debug**.

Долната част на интегрираната среда се разделя на два прозореца: левият прозорец е прозорец на променливите (**Variables** window), а десният – е прозорец за наблюдение (**Watch** window).

Прозорецът **Variables** съдържа списък на променливите в програмата. Трябва да се има пред вид, че различни променливи могат да бъдат визуализирани в различни моменти от време, чрез кликване върху етикетите **Auto**, **Locals** или **this**. Етикетът **Auto** визуализира името и стойността на променливите или обектите, използвани в предходните и в текущия оператор. Етикетът **Locals** визуализира името и текущата стойност на всички локални променливи или обекти, намиращи се в съответната област на видимост. Етикетът **this** визуализира данните за обектите, към които принадлежи изпълняващата се в момента функция. Стойностите на променливите, визуализирани в прозореца **Variables**, могат да бъдат модифицирани от потребителя в процеса на тестване, чрез кликване върху полето **Value** и въвеждане на новата стойност. Всяка модифицирана стойност се оцветява в червено, за да се покаже, че тя е била променена по време на **debug** сесията от програмиста.

Много често някои променливи се наблюдават от програмиста по време на процеса на настройване. Прозорецът **Watch** позволява на програмиста да наблюдава променливите, така както се променят техните стойности. Промените се визуализират в прозореца **Watch**. Имената на променливите могат да бъдат директно написани в прозореца или да бъдат изтеглени с мишката от прозореца **Variables**, или от първичния код и след това пуснати в прозореца **Watch**. Една променлива може да бъде изтрита от прозореца **Watch** чрез избиране на нейното име и след това натискане на клавиша <**Delete**>. Четирите етикета отдолу в прозореца **Watch** се използват от програмиста при групови променливи. Подобно на прозореца **Variables**, стойностите на променливите от прозореца **Watch** също могат да бъдат променяни, чрез промяна на полето **Value**. Променените стойности се оцветяват в червено. Текущата стойност на променливата по време на процеса на настройване също може да бъде визуализирана чрез задържане курсора на мишката върху променливата в прозореца с първичния код (фиг. 1А.3).



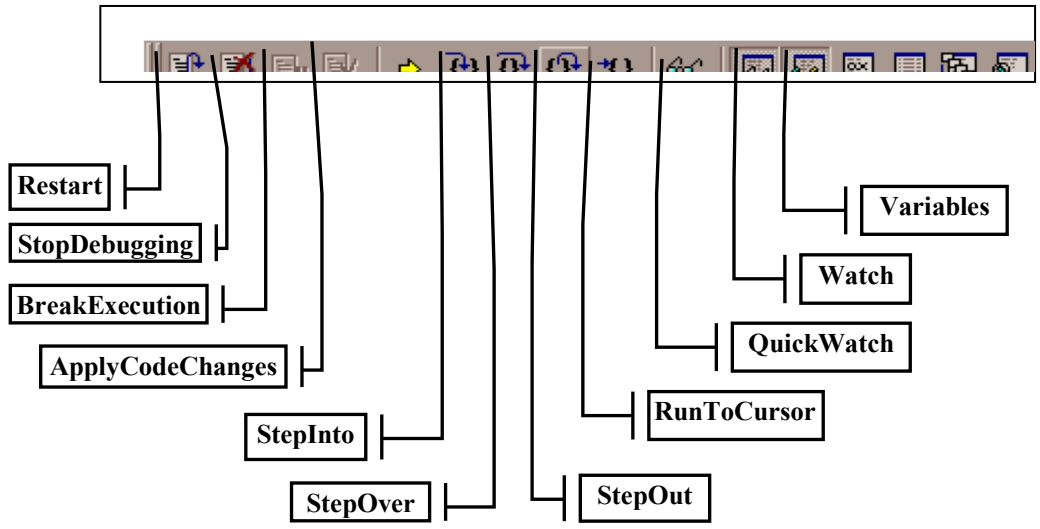
Фиг. 1А.3. Прозорците Variables и Watch

Лентата с инструменти **Debug** съдържа бутони, които управляват процеса на настройване. Тези бутони изпълняват същите действия, както тези от менюто **Debug**. Лентата с инструменти **Debug** може да бъде визуализирана чрез позициониране указателя на мишката върху празна област от главното меню, кликване върху десния бутон и избиране на опцията **Debug** от падащото меню (фиг. 1А.4).

Бутонът **Restart** повторно стартира приложението, спирайки в началото на програмата, като позволява на програмиста да постави точките на прекъсване

преди стартиране изпълнението на кода. Бутонът **StopDebugging** завършва сесията на настройване, като дава възможност на програмиста да редактира и повторно да компилира програмата преди следващото тестване.

Бутонът **BreakExecution** преустановява изпълнението на програмата в текущата позиция. Бутонът **ApplyCodeChanges** позволява на програмиста да модифицира първичния код в режим на настройване. Бутонът **ShowNextStatement** установява курсора на следващия оператор от същия ред, където сочи жълтата стрелка. Този бутон е полезен за повторно позициониране върху същия ред, където сочи жълтата стрелка, при визуализиране на първичния код в процеса на настройване.



Фиг. 1А.4. Лента с инструменти Debug

При всяко кликване върху бутона **StepInto** се изпълнява един оператор от програмата, включително и кода във функциите. Това се отнася за функции, които могат да бъдат изпълнявани по стъпки. За функции от библиотеката на C++ може да се изиска специфициране местоположението на библиотеката.

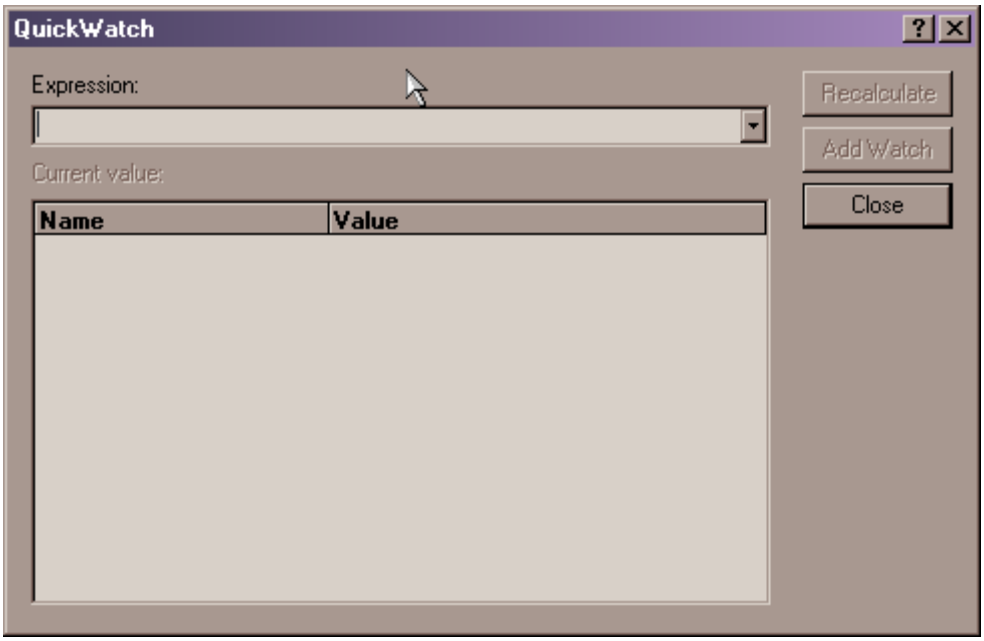
Бутонът **StepOver** изпълнява следващия изпълним ред от кода на програмата и придвижва напред жълтата стрелка към следващия изпълним ред в програмата. Ако редът от кода съдържа извикване на функция, тя се изпълнява изцяло като една стъпка. Това позволява на програмиста да изпълнява програмата ред по ред и да проверява изпълнението й без да се съобразява с детайлите на всяка една функция, която се извиква. Това е особено полезно за операторите cin и cout.

Бутонът **StepOut** позволява на програмиста да напусне текущата функция и да върне управлението обратно на реда, който е извикал функцията. Ако чрез бутона **StepIn** вече е започнато изпълнението на някоя функция, която няма нужда от тестване, кликвайки върху **StepOut**, управлението се връща в точката на извикване на функцията.

Чрез кликване с мишката върху ред от кода, който се намира няколко реда, след текущия изпълнен ред, след което се кликне върху бутона **RunToCursor**, изпълнението на програмата ще продължи до реда, където курсорът е позициониран. Тази техника е полезна за изпълнение на цикли или функции. Цикли, които имат голям брой итерации могат да бъдат изпълнявани като едно

цяло чрез позициониране на курсора след цикъла в прозореца с първичния код и след това се кликне върху бутона **RunToCursor**. Ако при настройване на програмата се е попаднало на библиотечна функция чрез бутона **StepOut** изпълнението ще се върне в потребителския код.

Бутонът **QuickWatch** визуализира диалога **QuickWacth** (фиг. 1А.5), който се използва при наблюдаване стойностите на изрази и променливи. Диалогът **QuickWacth** представя моментното състояние на стойностите на една или повече променливи в съответна точка от изпълнението на програмата.



Фиг. 1А.5. Диалогът QuickWatch

За да се наблюдава една променлива е необходимо да се въведе нейното име или израз в полето **Expression** и да се натисне клавишът **Enter**. Както в прозорците **Variables** и **Watch**, стойностите могат да бъдат редактирани в полето **Value**, но променените стойности не се оцветяват в червено. За да се наблюдават повече стойности, чрез бутона **AddWatch** се добавят променливи в прозореца **Watch**. Когато диалогът **QuickWatch** се затвори чрез бутона **Close**, променливите в диалога не се запазват. При следващо визуализиране на диалога **QuickWatch** полетата **Name** и **Value** са празни. Прозорецът **QuickWatch** може също да бъде използван за настройване на изрази от аритметични изчисления, чрез написване на израза в полето **Expression**.

Бутонът **Watch** визуализира прозореца **Watch**. Бутонът **Variables** визуализира прозореца **Variables**.

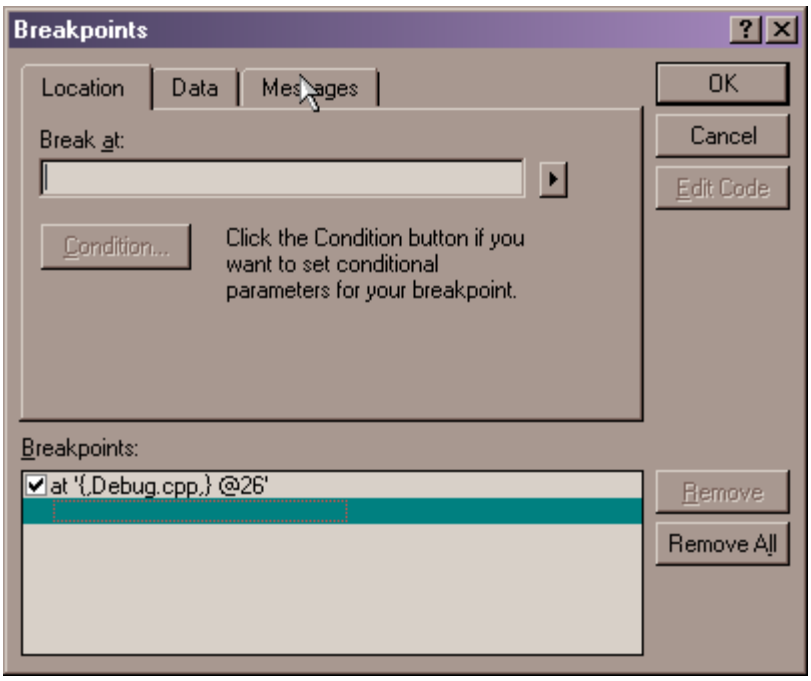
Бутонът **Memory** визуализира прозореца **Memory**, а бутонът **Registers** визуализира прозореца **Registers**.

Бутонът **CallStack** визуализира прозорец, съдържащ разпределението на стека в резултат на извикването на функциите в програмата. Този стек представлява списък на функциите, които са били извикани, до текущия ред на програмата.

Бутонът **Disassembly** визуализира прозореца **Disassembly**.

Когато един проект се затвори и след това повторно се отвори, всички поставени точки на прекъсване по време на предишната сесия на настройване

все още съществуват. Информация за точките на прекъсване може да се извлече чрез избиране от менюто Edit на елемента **Breakpoints…**, в резултат на което се визуализира диалогът **BreakPoint** (фиг. 1А.6).



Фиг. 1А.6. Диалогът BreakPoint

Този диалог в визуализира всички установени точки на прекъсване към текущия момент. До всяка точка на прекъсване се появява checkbox. Ако точката е активна, checkbox-ът е маркиран, в противен случай checkbox-ът е празен. Ако точката не е активна, това няма да предизвика спиране на изпълнението на програмата при нейното достигане, но тя може да бъде активирана по-късно. Допълнителни точки на прекъсване могат да бъдат добавени, чрез въвеждане на желания номер на ред в полето **Break at**. За да се въведе желаният номер на ред в полето **Break at**, той трябва да се предхожда от точка. Програмната среда позволява активирането на точка на прекъсване при настъпването на определени условия. След като програмистът е въвел номера на реда в полето **Break at**, той може да дефинира условията чрез бутона **Condition…**, чрез който се визуализира диалога **BreakpointCondition**. Зададените условия се потвърждават чрез бутона **OK**.

За приключване на процеса на настройване се използва бутонът **StopDebugging** от лентата с инструменти **Debug**.

3. Настройване на конкретно приложение

Някои от елементите на процеса на настройване ще бъдат демонстрирани чрез едно опростено приложение. Представената програма изчаква въвеждането на число от потребителя и брои от 1 до това число.

1. Решението на тази задача е представено във файла `H:\PROBLEMS\Debug.CPP` (фиг. 1А.7).

Да се създаде проект с име `ExDebug`.

Файлът `Debug.CPP` да се добави към създадения проект (виж “Добавяне на .cpp файл към съществуващ проект”)

Проектът да се компилира и изпълни.

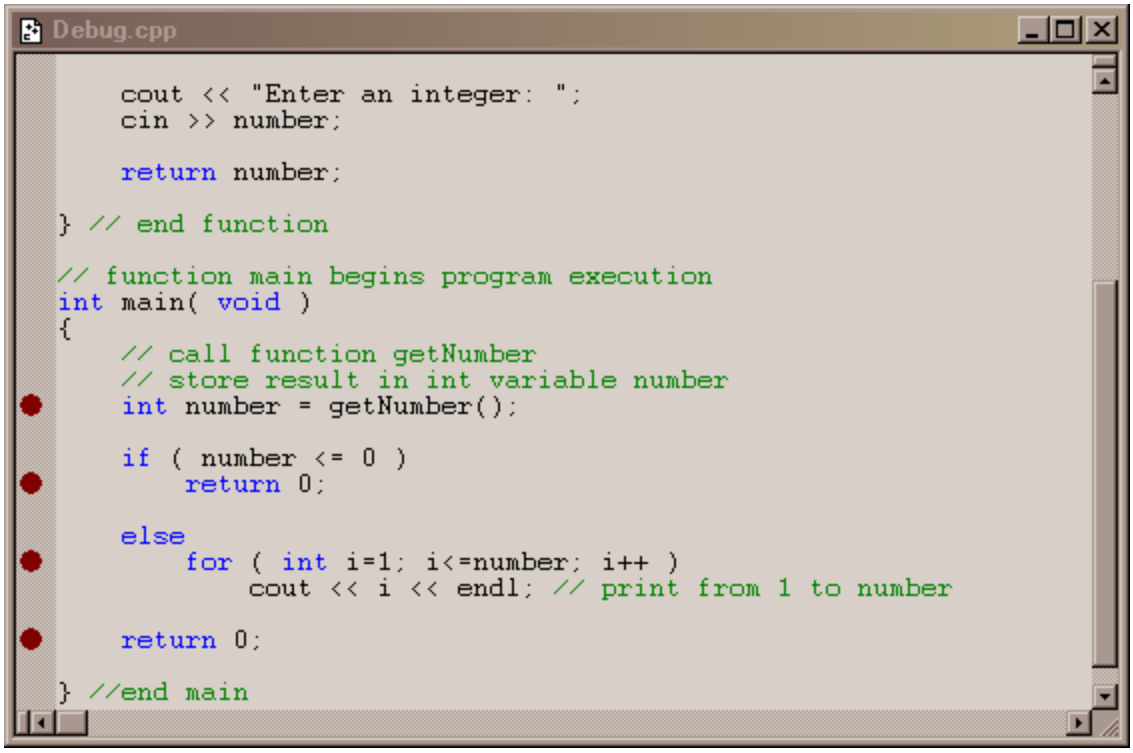
Да се анализират получените резултати.

```
1:    // Debug.cpp
2:
3:    #include <iostream>
4:
5:    using std::cout;
6:    using std::cin;
7:    using std::endl;
8:
9:    //function return number entered by user
10:   int getNumber(void)
11:   {
12:         int number;
13:
14:         cout << "Enter an integer: ";
15:         cin >> number;
16:
17:         return number;
18:
19:   } // end function
20:
21:   // function main begins program execution
22:   int main( void )
23:   {
24:         // call function getNumber
25:         // store result in int variable number
26:         int number = getNumber();
27:
28:         if ( number <= 0 )
29:               return 0;
30:
31:         else
32:               for ( int i=1; i<=number; i++ )
33:                     cout << i << endl; // print
34:
35:         return 0;
36:
37:   } //end main
```

Фиг. 1А.7. Първичен код на програмата Debug.cpp

2. В прозореца, който съдържа първичния код да се добави точка на прекъсване на **ред 26** чрез кликване върху реда от програмата и след това натискане на клавиша ***F9***. Червеното кръгче вляво от реда индицира, че на този ред е поставена точка на прекъсване.

3. Да се повтори стъпка 2, за да се поставят точки на прекъсване на **редове 29, 32 и 35**. Прозорецът ще изглежда, така както е показан на фиг. 1А.8.



Фиг. 1А.8. Поставени точки на прекъсване в програмата Debug.cpp

4. От менюто **Build** се избира подменюто **StartDebug**, откъдето чрез елемента **Go** се стартира процесът на настройване. Тъй като ще се настройва конзолно приложение, се визуализира конзолен прозорец. Изпълнението на програмата се преустановява при необходимост от въвеждане на данни и при достигане на точка на прекъсване (**ред 26**). Жълтата стрелка вляво от оператора

```
int number = getNumber();
```

показва, че изпълнението е преустановено на този ред.

5. За да се добави *възможност за следене* (***watch***) на определена променлива, нейното име, в случая **number** трябва да се напише в етикета **Watch1**, който се намира в долния край на интегрираната среда. В момента стойността на **number** е нейният адрес от паметта, тъй като все още не е присвоена стойност.

6. Кликнете отново върху елемента **Go**. В този момент програмата се изпълнява и се визуализира конзолният прозорец. Въведете 10 от клавиатурата и натиснете клавиша **Enter**. Програмата за кратко се изпълнява и след това се преустановява. Добавете друг watch за променливата **i**. Този watch може да бъде добавен само ако се намирате в областта на действие на променливата **i**, т.е. във for цикъла. Ако изпълнението на програмата не е в областта на действие на

променливата **i**, ще се появи съобщение за грешка за променливата **i** в прозорчето Watch1. В момента там се визуализира информация за променливата **number** и за променливата **i**. На фигура 1А.9 са представени установените наблюдения (watchs) за променливите **number** и **i**.

Фиг. 1А.9. Установени наблюдения за променливите number и i

7. Чрез кликване върху бутона **StepOver** ще се изпълняват отделни стъпки от цикъла for. Кликвайки върху бутона **StepOut** или **Go** целият цикъл ще се изпълни като една стъпка.

8. След като цикълът *for* е бил завършен, изпълнението на програмата се преустановява на **ред 35**. Основният прозорец показва, че debugger-ът е завършил и преброените стойности от 1 до 10 се визуализират в конзолния прозорец. Освен това, точка на прекъсване беше поставена на **ред 29**. Програмата не прекъсна своето изпълнение върху този ред, тъй като той никога не беше изпълнен. Изпълнението на **ред 29** от програмата зависи от числото, което е въведено от потребителя. Необходимо е повторно стартиране на debugger-а, чрез кликване върху елемента **Restart** от менюто **Debug** и въвеждане на неположително число от конзолния прозорец.

9. Завършването на процеса на настройване може да бъде осъществено чрез кликване върху бутона **StopDebugging** от лентата с инструменти **Debug**.