Zadania:

1. Funkcja przeliczająca F na C:

```
float FtoC(float f){
     return (f – 32.0) * 5.0 / 9.0;
}
```

2. Utworzyć dodatkową funkcję, która przelicza F na K, na bazie poprzedniej F to C (z poprzednich zajęć), wg wzoru:
   Kelvin = (Fahrenheit + 459.67) * 5.0 / 9.0
   Niech się wyświetla w 3 kolumnach tekst:
   F = (wartość) C = (wyliczona wartość) K = (wyliczona wartość)
   rozdzielone tabulacją od 0 do 200F. (+0,5 oceny).
3. Należy zmienić nazwy funkcji, które Państwo nadaliście dla przeliczania F na C na **Fahr_na_Cels** oraz F na K na **Fahr_na_Kelw**.
4. Należy samodzielnie utworzyć pozostałe funkcje:

Cels_na_Fahr	F = C * 9.0/5.0 + 32.0
Cels_na_Kelw	K = C + 273.15
Kelw_na_Cels	C = K – 273.15
Kelw_na_Fahr	F = K * 9.0/5.0 - 459.67

5. **Debuggowanie czyli poszukiwanie błędów w napisanym kodzie:**
   a. Debuggowanie jest procesem wyszukiwania oraz usuwania błędów znajdujących się w pisanym przez nas kodzie programu. Program wykorzystywany do tego celu nazywany jest debuggerem (ang. debugger – w wolnym tłumaczeniu odpluskwiacz). Istnieje szereg debuggerów w postaci komercyjnych rozwiązań zintegrowanych z pakietami Visual Studio, Visual C++ itd. Nie jesteśmy oczywiście zmuszeni do korzystania z powyższych środowisk. Uwagę swoją można bowiem skierować na narzędzia oparte na licencji publicznej GNU. Przykładem deguggera GNU jest program gdb. Zwykle nie musimy korzystać bezpośrednio z gdb gdyż narzędzia te są częścią zintegrowanego środowiska programistyczne (IDE), jak CodeBlocks czy Visual Studio.
   b. Aby skorzystać z przedstawionych metod wyszukiwania błędów konieczne jest, aby każdy testowany przez nas program uruchamiać w trybie „Debug” (nie wybierać opcji „Release”). W środowisku CodeBlocks ustawienie tej opcji można kontrolować w polu „Build target” na pasku „Compiler”. Podobna sytuacja ma miejsce w przypadku środowiska Visual Studio.
   c. **Uwaga, żeby skorzystać z opcji debuggowania trzeba utworzyć projekt w Code::Blocks!**
      - Utworzyć nowy projekt: File->New->Project

- W dalszej kolejności wybrać typ tworzonego projektu: Console application
- W wyświetlonym oknie dialogowym wybrać język w jakim będziemy programowali: C++
- Wpisać nazwę projektu oraz jego lokalizację: np. Project title: Test
- Naciśnięcie przycisku „Finish" kończy procedurę tworzenia projektu. W okienku „Menadzera" w zakładce „Projects" można przejrzeć drzewo przedstawiające strukturę plików w naszym projekcie. W chwili obecnej posiadamy jeden plik *.cpp o nazwie main.cpp zawierający wygenerowany przez środowisko prosty kod programu, przedstawiony poniżej (**należy go nadpisać własnym!**):

```
#include <iostream> // dodanie pliku nagłówkowego
using namespace std; // korzystamy z przestrzeni nazw „std"
int main() // główna funkcja programu
{
   cout << "Hello world!" << endl; // wyświetlenie na standardowym
  wyjściu Hello World !
   return 0;
}
```

d. Zadanie, napisz krótki program, np.:
   - Obliczający zadaną potęgę n liczby 2, tzn. 2^n, np. korzystając z podanego kodu:

```
int n=4;
int wynik=1;
   for(int i=0;i<n;i++){
      wynik=wynik*2;
   }
   printf("Wynik=%d\n", wynik);
```

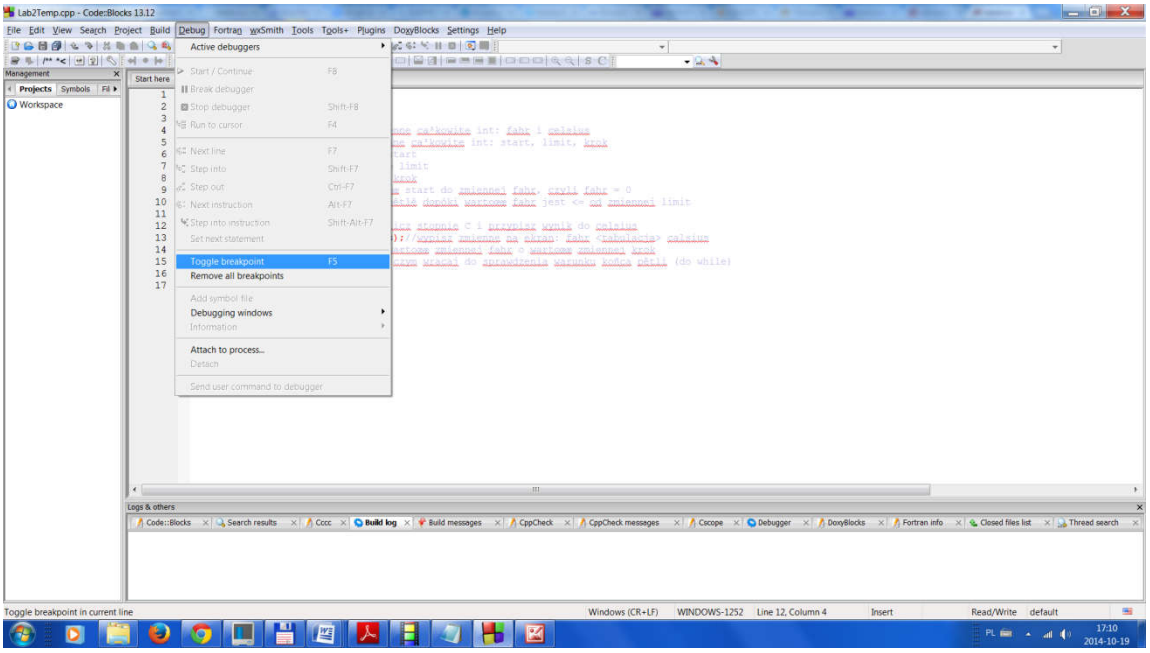
   - Obliczający pierwiastki wybranego równania kwadratowego. W tym celu skorzystaj z następujących uwag:
     - Funkcja sqrt(N) oblicza pierwiastek z liczby N; funkcja ta wymaga jednak dołączenia biblioteki <math.h>
   - Skompiluj program w trybie „Debug" (wykonywanie programu będzie prowadzone pod kontrolą debugger'a) i uruchom w celu sprawdzenia jego poprawności.

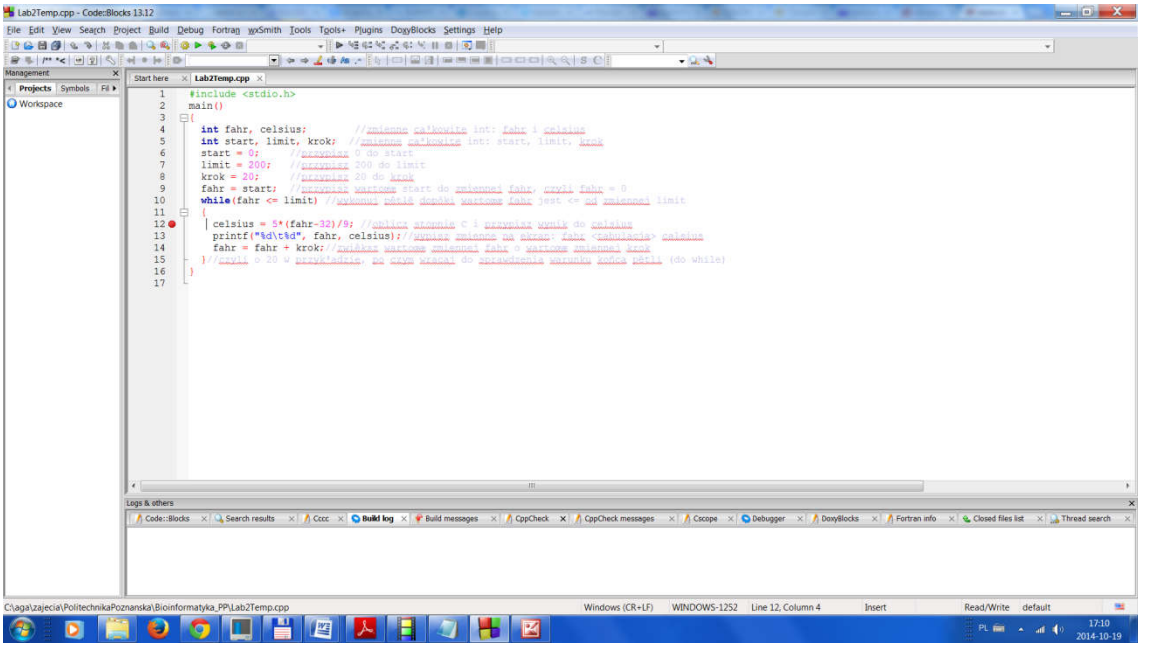
e. Kiedy dysponujemy już gotowym i działającym kodem rozwiązującym postawione zadanie, możemy przejść do ustawienia pierwszych punktów wstrzymania. Punktem wstrzymania (ang. breakpoint) jest pewne miejsce w programie w którym debugger automatycznie zatrzyma jego działanie. Nie jesteśmy oczywiście ograniczeni do pojedynczego punktu wstrzymania a ich

położenie zależy wyłącznie od nas samych i wiedzy na temat możliwych błędów jakie mogą się pojawić w danym miejscu kodu źródłowego.

CodeBlocks:

- Ustaw kursor w linijce, w której ma znaleźć się punkt wstrzymania
- Z menu wybierz: Debug -> Toggle breakpoint (F5)
- Uruchom program: Debug -> Start (F8)





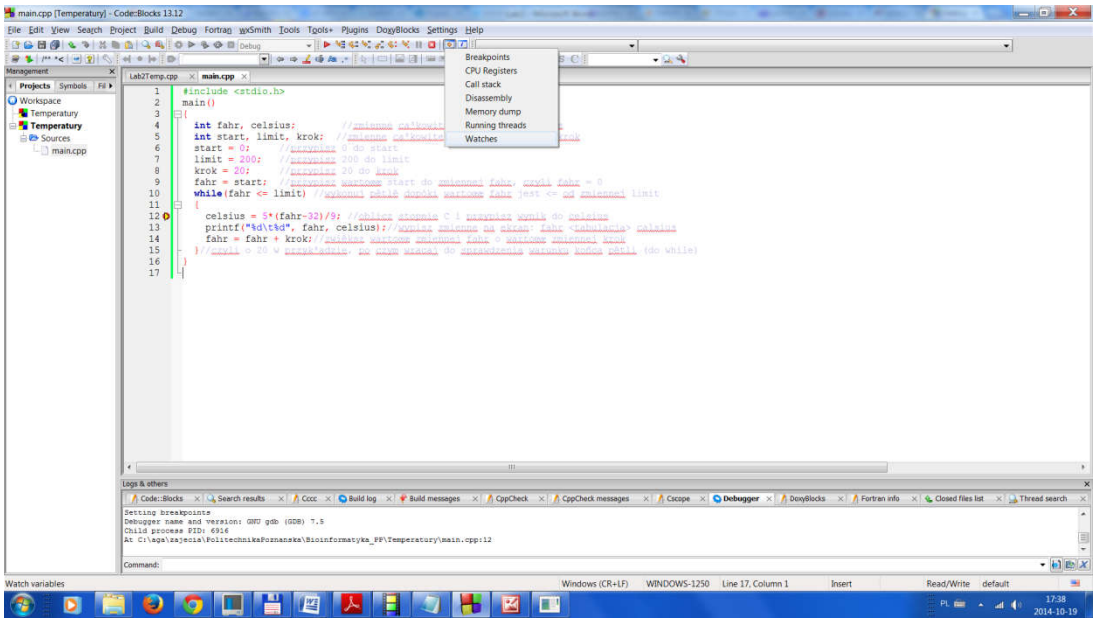
f. Zadania:

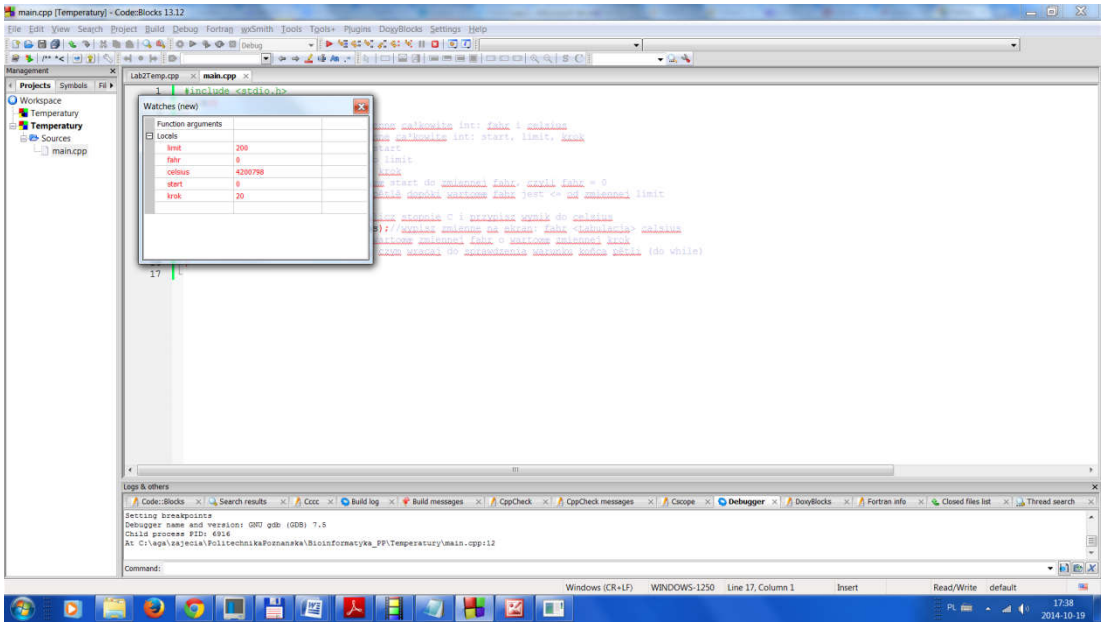
- Ustaw kilka punktów wstrzymania w różnych linijkach kodu.
- Sprawdź działanie poleceń dostępnych w menu Debug tj.
    - Next Line (F7),

Przejście do następnej linii kodu źródłowego.
    - Next instruction (ALT+F7),
      Przejście do następnej linii kodu maszynowego.
    - Step into (SHIFT+F7),
      Wykonanie następnej instrukcji z wejściem do funkcji.
    - Step out (SHIFT+CTRL+F7),
      Wyjście z funkcji.
    - Run to cursor (F4)
      Przejście do pozycji wskazywanej przez kursor.

g. Opcje debuggowania można znaleźć na pasku „Debugger" pod ikonką „Debugging windows":
    - Okno: Call stack: okno stosu wywołań (ang. Call stack) zawiera listę wywołań funkcji. Lista ta przedstawiana jest w postaci hierarchicznej od funkcji wywołanej najwcześniej aż do wywołanej w bieżącej chwili. Kliknięcie prawym przyciskiem na nazwę funkcji oraz wybranie z pop-up menu polecenia „switch to this Frome" powoduje przeskoczenie do wskazanej funkcji.
    - Okno: CPU Registers: okno to przedstawia aktualną zawartość rejestrów procesora. Możliwa jest bieżąca obserwacja zmian wartości zapisanych w rejestrach. Bezpośrednia manipulacja wartościami rejestrów procesora możliwa jest z poziomu języka maszynowego tj. assemblera.
    - Okno: Disassembly: okno to (zwane oknem disassemblera) zawiera widok kodu maszynowego zapisanego w języku assemblera i wygenerowanego przez kompilator.
    - Okno: Memory dump: polskim odpowiednikiem dla tej nazwy jest okno zrzutu pamięci. Okno to pozwala na bezpośrednie podglądanie zawartości pamięci. Możliwość podglądania zawartości pamięci ograniczona jest do obszaru należącego do przestrzeni adresowej naszego procesu. Nie możemy zatem podejrzeć obszaru pamięci wykraczającego poza ten przydzielony (przez system operacyjny) dla aplikacji którą właśnie debuggujemy.
    - Okno: Running threads: w oknie tym znajdują się informacja o wątkach pracujących w bieżącej chwili. Wszystkie programy pisane na tych zajęciach będą programami jednowątkowymi.
    - **Okno: Watches (zdecydowanie nas interesuje)**: okno to umożliwia podejrzenie bieżącej wartości zmiennych jak również śledzenie zmian ich wartości. Kliknięcie prawym przyciskiem w oknie „Watches" powoduje pojawienie się menu podręcznego zawierającego szereg przydatnych funkcji tj. „Add watch" (śledzenie zmian wybranej

zmiennej) czy też „Change value” (pozwala zmienić bieżącą wartość przechowywaną w zmiennej).





h. **Zadania:**

- Korzystając z przedstawionych możliwości debuggowania prześledź napisany dotychczas program przeliczania temperatur.
- Zapoznaj się z funkcjami dostępnymi w menu: Debug -> Information