Министерство образования Республики Беларусь

Учреждение образования

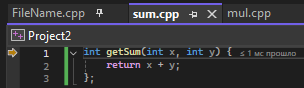
“Белорусский Государственный Технологический Университет”

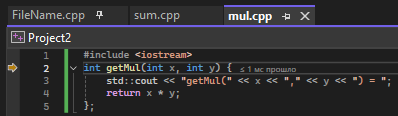
Отчёт по лабораторной работе №8

Выполнил студент 1-го курса , факультета ФИТ, специальности «Программная инженерия», 10 группы: Езерский Никита Витальевич

Минск 2024

Выполнить отладку примера из лекции 9 в пошаговом режиме с заходом в функции getSum и getMul (F11)





Задание №1

#include <iostream>

int getMul(int x, int y) {

std::cout << "getMul(" << x << "," << y << ") = ";

return x \* y;

};

int getSum(int x, int y) {

return x + y;

};

#include <iostream>

int getSum(int x, int y);

int getMul(int x, int y);

int main(int argc,char\*argv[])

{

int parm1 = 5;

for (int i = 0; i < 5; i++) {

int parm2 = parm1 + i;

int result\_getSum = getSum(parm1, parm2);

std::cout << "i = " << i;

std::cout << "getSum(" << parm1 << "," << parm2 << ") = " << result\_getSum << std::endl;

}

int result\_getMul = getMul(2, 3);

std::cout << result\_getMul << std::endl;

system("pause");

return 0;

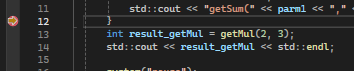
}

Пример 1

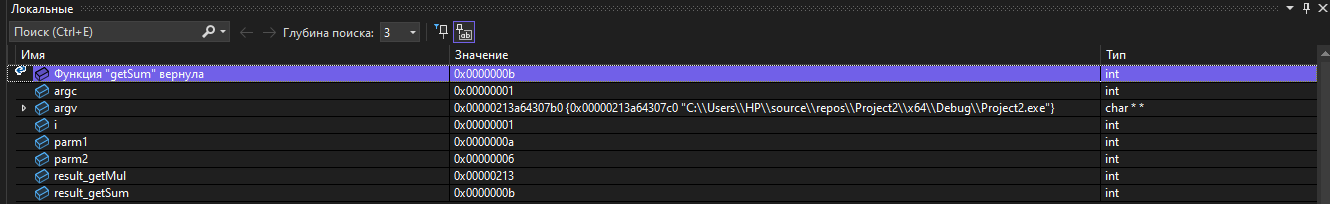
Условие:

1. Выполнить следующую последовательность действий. 1. Начать отладку. 2. Установить точку останова на 12-oй строке кода. В этом месте появится красный круг, отмечающий точку останова. Точка останова указывает, где Visual Studio приостановит выполнение кода и обеспечит возможность для выполнения необходимых действий в режиме отладки

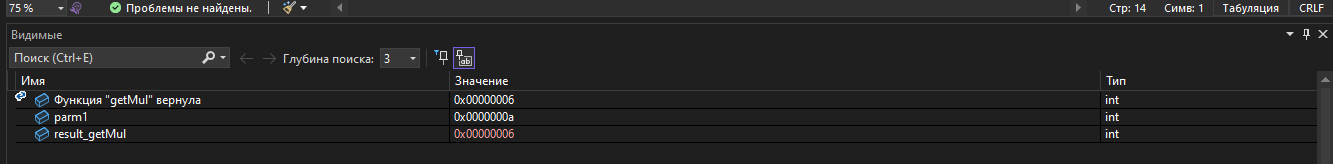
Результат:



В окне «Локальные» автоматически отображаются значения локальных переменных:



В окне «Видимые» отображаются все переменные и их текущие значения Окно «Видимые» позволяет просматривать/изменять значения переменных и выражений.

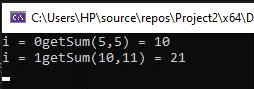
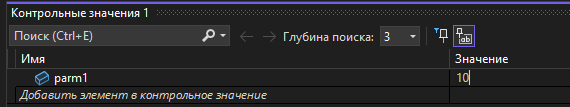


Пример 2

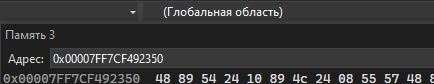
Условие:

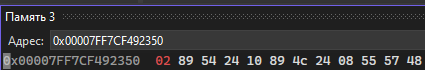
1. Начать отладку. 2. Установить точку останова на 15-й строке кода. 3. Продолжить отладку до точки останова. 4. Открыть окно «Контрольные значения». 5. Выделить имя переменной parm1 и перетащить его в окно «Контрольные значения». 6. Изменить значение переменной parm1 на 10. 7. Продолжить выполнение отладки в пошаговом режиме.

Результат:

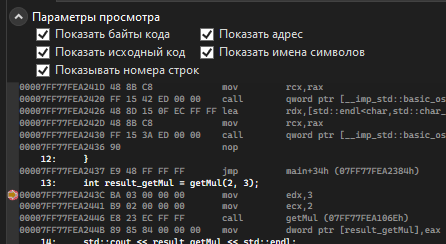
 

Окно «Памяти» позволяет просматривать содержимое ячеек памяти. Содержимое памяти может отображаться в различных форматах, которые выбираются из контекстного меню.

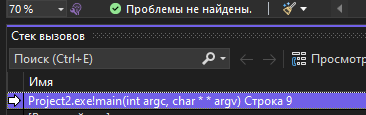


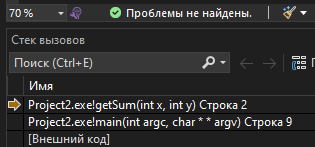


В окне «Дизассемблированный код» отображается код сборки, соответствующий инструкциям, созданным компилятором.



Стек вызовов (call stack) – это список всех активных функций, которые вызывались, до текущей точки выполнения исходного кода.





#include <windows.h>

#include <iostream>

using namespace std;

int main() {

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

char upper\_case, lower\_case, number;

int n;

cout << "Введите число от 1 до 4:";

cin >> n;

switch (n)

{

case 1: {

cout << "Прописное написание:";

cin >> upper\_case;

cout << "Строчном написание: ";

cin >> lower\_case;

if (upper\_case >= 0x41 && upper\_case <= 0x5A || lower\_case >= 0x61 && lower\_case <= 0x7A)

cout << "Разница значений составляет:"<< (int)upper\_case - (int)lower\_case << endl;

else

{

cout << "Ошибка" << endl;

}

break;

}

case 2: {

cout << "Прописное написание: ";

cin >> upper\_case;

cout << "Строчном написание: ";

cin >> lower\_case;

if ((unsigned char)upper\_case >= 0xC0 && (unsigned char)upper\_case <= 0xDF || (unsigned char)lower\_case >= 0xE0 && (unsigned char)lower\_case <= 0xFF)

cout << "Разница значений составляет:" << (int)upper\_case - (int)lower\_case << endl;

else

{

cout << "Ошибка" << endl;

}

break;

}

case 3: {

cout << "number: ";

cin >> number;

if (number < 0x30 || number > 0x39)

{

cout << "Ошибка" << endl;

}

else

{

cout << "Вывод в консоль кода символа : "<< (int)number << endl;

}

break;

}

case 4: {

cout << "Выход из программы" << endl;

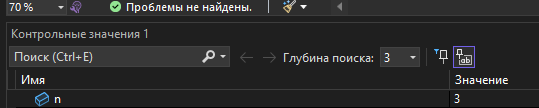
break;

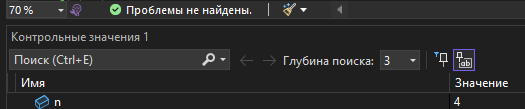
}

}

}

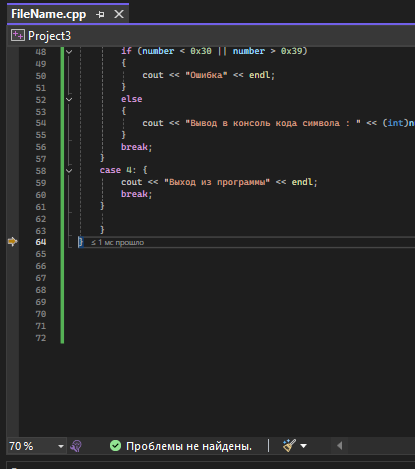
**b. Проверить и изменить значения переменных всеми возможными способами (используя окна отладчика, подсказки).**



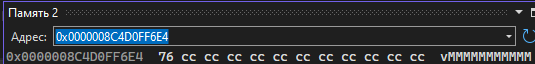


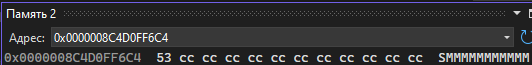


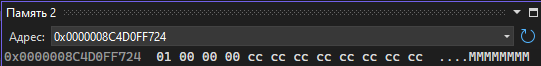
**a. Выполнить отладку программы в пошаговом режиме.**

****

**c. Просмотреть значения переменных в памяти.**







1. Назначение отладчика – устранение ошибок в исходном коде программы.
2.  Пошаговое выполнение программы (Step Over, Step Into, Step Out).

 Установка и управление точками останова.

 Просмотр и изменение значений переменных.

 Анализ стека вызовов.

 Отображение памяти, регистров и дизассемблированного кода.

1. Точка останова (breakpoint) – это точка, в которой процесс выполнения программы приостанавливается и отладчик получает управление.
2.  Щелчок левой кнопкой мыши на полосе слева от строки кода в редакторе.

 Использование контекстного меню с опцией «Установить/удалить точку останова».

 Нажатие клавиши F9 при выборе строки кода.

### 5. Какие способы запуска отладчика вы знаете?

* Запуск отладки с использованием клавиши **F5** (Start Debugging).
* Запуск без отладки с использованием **Ctrl + F5**.
* Запуск отладки через меню **Debug -> Start Debugging**.

### Назовите основные возможности управления режимами отладки

* **Step Over (F10)** — выполнение текущей строки кода без захода в вызванные функции.
* **Step Into (F11)** — пошаговое выполнение с заходом в вызванные функции.
* **Step Out (Shift + F11)** — выход из текущей функции к месту её вызова.
* **Continue (F5)** — продолжение выполнения до следующей точки останова.
* **Break (Ctrl + Alt + Break)** — приостановка выполнения программы
  1. Для контроля значений

### Для каких целей используется окно памяти?

### ***Окно «Памяти» позволяет просматривать содержимое ячеек памяти.*** Содержимое памяти может отображаться в различных форматах, которые выбираются из контекстного меню

### 10. Для каких целей используется окно регистров?

**Окно регистров отображает текущее состояние регистров процессора.** Оно полезно для отладки программ на уровне машинного кода или при анализе производительности и оптимизации.

### 11. Для каких целей используется окно дизассемблированного кода?

**Окно дизассемблированного кода позволяет просматривать скомпилированный ассемблерный код программы**. Это полезно для анализа выполнения программы на уровне машинного кода, особенно если нет доступа к исходному коду.

### 12. Что отражается в окне дизассемблированного кода?

В окне «Дизассемблированный код» отображается код сборки, соответствующий инструкциям, созданным компилятором.