**Лабораторная работа № 1**

Тема: Основные арифметические операции над целыми числами.

Интерфейс с языками высокого уровня.

1. Цель работы

Ознакомиться с правилами оформления ассемблерных процедур, в том числе вызываемых из Си-программ.

2. Краткие сведения

**Создание проекта C++**

Чтобы создать проект Visual Studio C++, выполните следующие действия.

1. Запустите Visual Studio.

2. Выберите меню File | New Project (Файл | Новый проект).

3. В дереве элементов управления диалогового окна New Project (Новый

проект) выберите строку Installed | Visual C++ | Windows Desktop

(Установленные | Visual C++ | Рабочий стол Windows).

4. Выберите Windows Console Application (Консольное приложение

Windows) в качестве типа проекта.

5. В текстовом поле Name (Имя) введите Example1.

6. В текстовом поле Location (Расположение) введите имя папки, в которой будет расположен проект. Вы также можете использовать кнопку

Browse (Обзор), чтобы выбрать папку, или оставить содержимое поля

без изменений, чтобы использовать расположение по умолчанию.

7. В текстовом поле Solution (Решение) введите TestSolution.

8. Убедитесь, что настройки диалогового окна New Project (Новый проект) такие же, как на рис. П.1 (расположение проекта может быть другим). Нажмите книпку ОК.

9. При необходимости выберите пункт меню View | Solution Explorer,

чтобы открыть окно Solution Explorer.

10. В древовидном элементе управления обозревателя решений щелкните правой кнопкой мыши текст верхнего уровня с надписью Solution

‘Example 1’ (1 Project) и выберите пункт Rename (Переименовать).

Измените имя решения на TestSolution.

11. Выберите пункт Build | Configuration Manager. В диалоговом окне

Configuration Manager выберите <Edit ...> (<Редактировать>) в разделе Active Solution Platforms (рис. П.2).

12. В диалоговом окне Edit Solution Platforms (Редактировать платформы решения) выберите x86 и нажмитеRemove (Удалить). Нажмите Close (Закрыть), чтобы закрыть диалоговое окно Edit Solutions Platforms; затем еще раз нажмите Close, чтобы закрыть диалоговое окно Configuration Manager.

**В файле CalculatorTutorial.cpp измените код таким образом, чтобы он соответствовал этому примеру:**

**C++**

**Копировать**

**// CalculatorTutorial.cpp : This file contains the 'main' function. Program execution begins and ends there.**

**//**

**#include <iostream>**

**using namespace std;**

**int main()**

**{**

**cout << "Calculator Console Application" << endl << endl;**

**cout << "Please enter the operation to perform. Format: a+b | a-b | a\*b | a/b"**

**<< endl;**

**return 0;**

**}**

**// Run program: Ctrl + F5 or Debug > Start Without Debugging menu**

**// Debug program: F5 or Debug > Start Debugging menu**

**// Tips for Getting Started:**

**// 1. Use the Solution Explorer window to add/manage files**

**// 2. Use the Team Explorer window to connect to source control**

**// 3. Use the Output window to see build output and other messages**

**// 4. Use the Error List window to view errors**

**// 5. Go to Project > Add New Item to create new code files, or Project > Add Existing Item to add existing code files to the project**

**// 6. In the future, to open this project again, go to File > Open > Project and select the .sln file**

**Пояснения к коду:**

**Операторы #include позволяют ссылаться на код, расположенный в других файлах. Иногда отображается имя файла, заключенное в угловые скобки (<>), а в других случаях — в кавычки (" "). В общем случае угловые скобки используются при ссылке на стандартную библиотеку C++, а кавычки — для других файлов.**

**Строка using namespace std; сообщает компилятору ожидать данные от стандартной библиотеки C++ для использования в этом файле. Без нее каждое ключевое слово из этой библиотеки пришлось бы предварять элементом std::, чтобы обозначить его область. Например, без этой строки каждую ссылку на cout потребовалось бы записывать в виде std::cout. Оператор using добавляется, чтобы сделать код более чистым и понятным.**

**Ключевое слово cout используется для вывода сведений в стандартный поток вывода в C++. Оператор << указывает компилятору отправить все содержимое справа от него в стандартный поток вывода.**

**Ключевое слово endl аналогично клавише ВВОД — оно завершает строку и переводит курсор на следующую строку. Для выполнения той же задачи рекомендуется помещать \n внутрь строки (заключенной в ""), так как endl всегда освобождает буфер и может негативно повлиять на производительность программы, но из-за того, что это очень небольшое приложение, в нем используется endl для улучшения удобочитаемости.**

**Все операторы C++ должны заканчиваться точкой с запятой, а все приложения C++ — содержать функцию main(). Именно ее программа выполняет при запуске. Для использования весь код должен быть доступен из main().**

**Чтобы сохранить файл, нажмите клавиши CTRL+S или выберите элемент Сохранить в верхней части интегрированной среды разработки, выглядящий как значок гибкого диска на панели инструментов в строке меню.**

**Чтобы запустить приложение, нажмите клавиши CTRL+F5 или перейдите в меню Отладка и выберите Запуск без отладки. Должно отобразится окно консоли с текстом, указанным в коде.**

**По завершении закройте окно консоли.**

**Добавление кода для выполнения математических операций**

**Пора добавить математическую логику.**

**Добавление класса Calculator**

**В меню Проект выберите команду Добавить класс. В поле ввода Имя класса введите Calculator. Нажмите кнопку ОК. В проект будут добавлены два новых файла. Чтобы сохранить сразу все измененные файлы, нажмите клавиши CTRL+SHIFT+S. Это сочетание клавиш для команды Файл>Сохранить все. Кроме того, есть кнопка Сохранить все на панели инструментов. Она имеет вид значка с двумя дискетами рядом с кнопкой Сохранить. В общем случае рекомендуется регулярно нажимать кнопку Сохранить все, чтобы не пропустить никакие файлы при сохранении.**

**Класс подобен схеме для объекта, который выполняет какие-либо действия. В этом случае мы определяем калькулятор и особенности его работы. Использованный ранее мастер Добавить класс создал файлы H и CPP, имя которых совпадает с именем класса. Вы можете просмотреть полный список файлов проекта в окне обозревателя решений, отображаемом в боковой части окна IDE. Если окно не отображается, его можно открыть из строки меню, выбрав Вид>Обозреватель решений.**

**Теперь у вас в редакторе должно быть открыто три вкладки: CalculatorTutorial.cpp, Calculator.h и Calculator.cpp. Если вы случайно закроете одну из них, то можете повторно открыть эту вкладку, дважды щелкнув ее в окне обозревателя решений.**

**На вкладке Calculator.h удалите созданные строки Calculator(); и ~Calculator();, так как здесь они вам не понадобятся. Затем добавьте следующую строку кода, чтобы файл выглядел следующим образом:**

**C++**

**Копировать**

**#pragma once**

**class Calculator**

**{**

**public:**

**double Calculate(double x, char oper, double y);**

**};**

**Пояснения к коду**

**Добавленная строка объявляет новую функцию Calculate, которую мы используем для выполнения математических операций, таких как сложение, вычитание, умножение и деление.**

**Код C++ упорядочен по файлам заголовка (H) и исходным файлам (CPP). Различные компиляторы поддерживают несколько других расширений имени файла, но это основные, о которых вам нужно знать. Функции и переменные обычно объявляются, то есть получают имя и тип, в файлах заголовка, и реализуются, то есть получают определение, в исходных файлах. Чтобы обратиться к коду, определенному в другом файле, вы можете использовать #include "filename.h", где "filename.h" — это имя файла, объявляющего переменные или функции, которые требуется использовать.**

**Две удаленные строки объявляли конструктор и деструктор для класса. Для такого простого класса компилятор создает их автоматически, а их использование выходит за рамки данного руководства.**

**Рекомендуется упорядочивать код по различным файлам в зависимости от его назначения, чтобы позднее было проще найти нужный код. В данном случае мы определяем класс Calculator отдельно от файла, содержащего функцию main(), но планируем ссылаться на класс Calculator в main().**

**Под Calculate появится зеленая волнистая линия. Это вызвано тем, что мы еще не определили функцию Calculate в CPP-файле. Наведите курсор на это слово, щелкните появившуюся лампочку (в этом случае отвертку) и выберите Создать определение "Calculate" в Calculator.cpp.**

**Отображается всплывающее окно, сообщающее об изменении кода, внесенном в другом файле. Код был добавлен в Calculator.cpp.**

**Сейчас он возвращает просто 0.0. Давайте изменим это. Нажмите клавишу ESC, чтобы закрыть всплывающее окно.**

**Переключитесь на файл Calculator.cpp в окне редактора. Удалите разделы Calculator() и ~Calculator() (как сделали в H-файле) и добавьте в Calculate() следующий код:**

**C++**

**Копировать**

**#include "Calculator.h"**

**double Calculator::Calculate(double x, char oper, double y)**

**{**

**switch(oper)**

**{**

**case '+':**

**return x + y;**

**case '-':**

**return x - y;**

**case '\*':**

**return x \* y;**

**case '/':**

**return x / y;**

**default:**

**return 0.0;**

**}**

**}**

**Пояснения к коду**

**Функция Calculate использует число, оператор и второе число, а затем выполняет запрошенную операцию над этими числами.**

**Оператор switch проверяет, какой оператор был предоставлен, и выполняет только вариант (case), соответствующий этой операции. Вариант default: является резервным на тот случай, если пользователь вводит неприемлемый оператор, чтобы программа не прервала работу. В общем случае лучше обрабатывать некорректный ввод данных пользователем более элегантным способом, но это выходит за рамки данного руководства.**

**Ключевое слово double обозначает тип числа, который поддерживает десятичные числа. Таким образом, калькулятор может выполнять математические операции с десятичными и целыми числами. Функция Calculate должна всегда возвращать такое число из-за элемента double в самом начале кода (обозначает тип возвращаемого значения функции), поэтому возвращается 0.0 даже в случае default.**

**H-файл объявляет функцию prototype, которая заранее сообщает компилятору, какой параметр требуется и какой тип возвращаемого значения следует ожидать. CPP-файл содержит все сведения о реализации этой функции.**

**Если построить и запустить код еще раз на этом этапе, он по-прежнему завершит работу после запроса операции для выполнения. Далее вам нужно изменить функцию main для выполнения вычислений.**

**Вызов функций-членов класса Calculator**

**Теперь давайте изменим функцию main в CalculatorTutorial.cpp:**

**C++**

**Копировать**

**// CalculatorTutorial.cpp : This file contains the 'main' function. Program execution begins and ends there.**

**//**

**#include <iostream>**

**#include "Calculator.h"**

**using namespace std;**

**int main()**

**{**

**double x = 0.0;**

**double y = 0.0;**

**double result = 0.0;**

**char oper = '+';**

**cout << "Calculator Console Application" << endl << endl;**

**cout << "Please enter the operation to perform. Format: a+b | a-b | a\*b | a/b"**

**<< endl;**

**Calculator c;**

**while (true)**

**{**

**cin >> x >> oper >> y;**

**result = c.Calculate(x, oper, y);**

**cout << "Result is: " << result << endl;**

**}**

**return 0;**

**}**

**Пояснения к коду**

**Так как программы C++ всегда запускаются с функции main(), нам нужно вызвать из нее другой, для чего нужен оператор #include.**

**Объявлены начальные переменные x, y, oper и result для хранения, соответственно, первого числа, второго числа, оператора и окончательного результата. Рекомендуется всегда присваивать им некоторые начальные значения, чтобы избежать неопределенного поведения. Это мы здесь и делаем.**

**Строка Calculator c; объявляет объект "c" в качестве экземпляра класса Calculator. Сам класс является лишь схемой для описания работы калькуляторов, а объект — это конкретный калькулятор, который выполняет вычисления.**

**Оператор while (true) является циклом. Код внутри этого цикла продолжает выполняться снова и снова, пока выполняется условие в (). Так как условие просто указано как true, оно всегда выполняется, поэтому цикл выполняется бесконечно. Чтобы закрыть программу, пользователю нужно вручную закрыть окно консоли. В противном случае программа будет постоянно ожидать новые входные данные.**

**Ключевое слово cin используется для получения входных данных от пользователя. Этот входной поток способен обработать строку текста, введенного в окне консоли, и поместить ее внутрь каждой из перечисленных переменных в указанном порядке, при условии, что введенные пользователем данные соответствуют требуемой спецификации. Вы можете изменить эту строку, чтобы принять другие типы входных данных, например, больше двух чисел, однако для этого также потребуется изменить функцию Calculate().**

**Выражение c.Calculate(x, oper, y); вызывает определенную ранее функцию Calculate и предоставляет введенные входные значения. Затем она возвращает число, сохраняемое в result.**

**Наконец, в консоль выводится result, благодаря чему пользователь видит результат вычисления.**

**Повторная сборка и проверка кода**

**Пришло время снова протестировать программу еще раз, чтобы убедиться в ее правильной работе.**

**Нажмите клавиши CTRL+F5, чтобы перестроить и запустить приложение.**

**Введите 5 + 5 и нажмите клавишу ВВОД. Убедитесь, что выводится результат 10.**

**Пример 1. Calculator.cpp**

**#include "****Calculator.h"**

**double Calculator::Calculate(double x, char oper, double y)**

**{**

**switch (oper)**

**{**

**case '+':**

**return x + y;**

**case '-':**

**return x - y;**

**case '\*':**

**return x \* y;**

**case '/':**

**return x / y;**

**default:**

**return 0.0;**

**}**

**}**

**Описание Calculator.h**

#pragma once

class Calculator

{

public:

double Calculate(double x, char oper, double y);

};

// CalculatorTutorial.cpp : Этот файл содержит функцию "main". Здесь начинается и заканчивается выполнение программы.

//

#include <iostream>

#include "Calculator.h"

using namespace std;

int main()

{

double x = 0.0;

double y = 0.0;

double result = 0.0;

char oper = '+';

cout << "Calculator Console Application" << endl << endl;

cout << "Please enter the operation to perform. Format: a+b | a-b | a\*b | a/b"

<< endl;

Calculator c;

while (true)

{

cin >> x >> oper >> y;

result = c.Calculate(x, oper, y);

cout << "Result is: " << result << endl;

}

return 0;

}

// Запуск программы: CTRL+F5 или меню "Отладка" > "Запуск без отладки"

// Отладка программы: F5 или меню "Отладка" > "Запустить отладку"

// Советы по началу работы

// 1. В окне обозревателя решений можно добавлять файлы и управлять ими.

// 2. В окне Team Explorer можно подключиться к системе управления версиями.

// 3. В окне "Выходные данные" можно просматривать выходные данные сборки и другие сообщения.

// 4. В окне "Список ошибок" можно просматривать ошибки.

// 5. Последовательно выберите пункты меню "Проект" > "Добавить новый элемент", чтобы создать файлы кода, или "Проект" > "Добавить существующий элемент", чтобы добавить в проект существующие файлы кода.

// 6. Чтобы снова открыть этот проект позже, выберите пункты меню "Файл" > "Открыть" > "Проект" и выберите SLN-файл.

Отладка приложения

Так как пользователь может ввести в окне консоли что угодно, нужно убедиться, что калькулятор обрабатывает некоторые входные данные правильно. Вместо запуска программы давайте выполним ее отладку, чтобы можно было проверить ее выполнение шаг за шагом.

Запуск приложения в отладчике

Установите точку останова на строке result = c.Calculate(x, oper, y);, сразу после запроса входных данных у пользователя. Чтобы задать точку останова, щелкните рядом со строкой серую вертикальную панель, идущую вдоль левой части окна редактора. Появится красная точка.

Снимок экрана: Visual Studio с красной точкой слева от строки кода, представляющей точку останова.

Теперь при отладке программа всегда приостанавливает выполнение на этой строке. Мы уже имеем примерное представление о том, что в простых случаях программа работает. Так как нам не нужно приостанавливать выполнение при каждом запуске, создадим условную точку останова.

Щелкните правой кнопкой мыши красную точку, обозначающую точку останова, и выберите команду Условия. В поле ввода для условия введите (y == 0) && (oper == '/'). По завершении операции нажмите кнопку Закрыть. Условия сохранятся автоматически.

Снимок экрана: всплывающее окно

Теперь выполнение приостанавливается в точке останова, в частности при попытке деления на 0.

Чтобы отладить программу, нажмите клавишу F5 или выберите Локальный отладчик Windows — кнопку панели инструментов со значком зеленой стрелки. Если ввести в консольном приложении нечто вроде "5 - 0", программа работает как обычно и продолжает выполнение. Но если ввести "10 / 0", она приостанавливает выполнение в точке останова. Вы можете ввести любое число пробелов между оператором и числами; функция cin достаточно интеллектуальна для того, чтобы правильно проанализировать входные данные.

Снимок экрана: выполнение программы приостановлено в условной точке останова в Visual Studio.

Полезные окна в отладчике

Каждый раз при отладке кода вы можете заметить, что появляется несколько новых окон. Они могут оказаться полезными в процессе отладки. Давайте рассмотрим окно Видимые. Окно Видимые содержит текущие значения переменных, используемых по меньшей мере за три строки перед текущей строкой. Чтобы просмотреть все переменные из этой функции, переключитесь в окно Локальные. Вы можете изменять значения этих переменных во время отладки, чтобы оценить их влияние на программу. В данном случае мы оставим их без изменений.

Снимок экрана: окно

Кроме того, вы можете просто навести указатель на переменные в самом коде, чтобы просмотреть их текущие значения на участке, где приостановлено выполнение. Сначала щелкните окно редактора, чтобы сделать его активным.

Снимок экрана: подсказка со значением переменной oper, выраженным символом

Продолжение отладки

Желтая линия слева показывает текущую точку выполнения. Текущая строка вызывает Calculate, поэтому нажмите клавишу F11, чтобы выполнить Шаг с заходом для этой функции. Вы переходите в тело функции Calculate. Обдуманно применяйте функцию Шаг с заходом, чтобы не терять время из-за чрезмерного ее использования. С помощью функции анализируется любой код, используемый в текущей строке, включая стандартные функции библиотеки.

Теперь, когда точка выполнения находится в начале функции Calculate, нажмите клавишу F10 для перехода на следующую строку в рамках выполнения программы. При нажатии клавиши F10 активируется функция Шаг с обходом. Шаг с обходом можно использовать для перемещения по строкам, не вдаваясь в подробности того, что происходит в каждой части строки. В общем случае следует использовать Шаг с обходом, а не Шаг с заходом, если только вы не хотите подробно изучить код, вызываемый из другого расположения (как в случае с переходом к телу функции Calculate).

Продолжайте использовать клавишу F10, чтобы выполнять Шаг с обходом для каждой строки, пока не вернетесь к функции main() в другом файле, и остановитесь на строке cout.

Похоже, программа работает правильно — принимает первое число и делит его на второе. В строке cout наведите указатель мыши на переменную result или взгляните на result в окне Видимые. Соответствующее значение отображается как inf. Это неправильное поведение, и нужно его скорректировать. В строке cout просто выводится любое значение, хранящееся в result. Поэтому при продвижении еще на один шаг с помощью клавиши F10 в окне консоли отображается следующее:

Снимок экрана: консоль отладки Visual Studio с результатом деления на ноль, равным inf.

Причина такого результата в том, что деление на ноль является неопределенным, поэтому у программы нет числового ответа на запрошенную операцию.

Исправление ошибки деления на ноль

Давайте обеспечим корректную обработку деления на ноль, чтобы пользователь мог осознать проблему.

Внесите следующие изменения в CalculatorTutorial.cpp. (Прерывать выполнение программы при редактировании не требуется благодаря функции отладчика Изменить и продолжить):

C++

Копировать

// CalculatorTutorial.cpp : This file contains the 'main' function. Program execution begins and ends there.

//

#include <iostream>

#include "Calculator.h"

using namespace std;

int main()

{

double x = 0.0;

double y = 0.0;

double result = 0.0;

char oper = '+';

cout << "Calculator Console Application" << endl << endl;

cout << "Please enter the operation to perform. Format: a+b | a-b | a\*b | a/b" << endl;

Calculator c;

while (true)

{

cin >> x >> oper >> y;

if (oper == '/' && y == 0)

{

cout << "Division by 0 exception" << endl;

continue;

}

else

{

result = c.Calculate(x, oper, y);

}

cout << "Result is: " << result << endl;

}

return 0;

}

Теперь нажмите клавишу F5 один раз. При этом выполнение программы продолжается вплоть до приостановки для запроса данных, вводимых пользователем. Снова введите 10 / 0. Теперь выводится более информативное сообщение. У пользователя запрашиваются дополнительные входные данные, и программа продолжает выполнение в обычном режиме.

Снимок экрана: консоль отладки Visual Studio с окончательным результатом после изменений.

В окне консоли отображаются две строки: "10 / 0 Результат: inf" и "10 / 0 Исключение: деление на 0".

Примечание

При редактировании кода в режиме отладки существует риск того, что код устареет. Это происходит, когда отладчик продолжает выполнять старый код, не учитывающий внесенные вами изменения. В подобной ситуации отладчик выводит диалоговое окно с соответствующим сообщением. В некоторых случаях может потребоваться нажать клавишу F5, чтобы обновить выполняемый код. В частности, если вы вносите изменения внутри функции, когда точка выполнения находится внутри этой же функции, вам потребуется выполнить шаг с выходом из функции, а затем вернуться в нее, чтобы получить обновленный код. Если по какой-либо причине это не работает и вы видите сообщение об ошибке, можно остановить отладку, щелкнув красный квадрат на панели инструментов под меню в верхней части интегрированной среды разработки, а затем перезапустить отладку, нажав клавишу F5 или выбрав зеленую стрелку "воспроизведения" рядом с кнопкой остановки на панели инструментов.

Пояснения по сочетаниям клавиш для выполнения и отладки

Клавиша F5 (или команда Отладка>Запустить отладку) запускает сеанс отладки, если он еще не активен, и выполняет программу до тех пор, пока не будет встречена точка останова либо программе не потребуются вводимые пользователем данные. Если вводимые пользователем данные не требуются и точка останова отсутствует, программа завершает работу, а окно консоли автоматически закрывается после завершения выполнения программы. Если вы выполняете программу уровня "Hello World", используйте сочетание клавиш CTRL+F5 или задайте точку останова перед нажатием клавиши F5, чтобы окно оставалось открытым.

Сочетание клавиш CTRL+F5 (или команда Отладка>Запуск без отладки) запускает приложение без перехода в режим отладки. Это немного быстрее отладки, а окно консоли остается открытым после завершения выполнения программы.

Клавиша F10 (соответствующая возможности Шаг с обходом) позволяет выполнять построчную итерацию по коду, а также визуализировать выполнение кода и значения переменных на каждом шаге выполнения.

Клавиша F11 (соответствующая возможности Шаг с заходом) работает аналогично возможности Шаг с обходом, за исключением того, что заходит в любые функции, вызываемые в строке выполнения. Например, если в выполняемой строке вызывается функция, при нажатии клавиши F11 указатель перемещается в тело функции, чтобы вы могли проследить выполнение кода функции, прежде чем вернуться на исходную строку. При нажатии клавиши F10 выполняется шаг с обходом по вызову функции и простой переход на следующую строку. Вызов функции все равно выполняется, но программа не приостанавливается, чтобы показать вам, что именно она делает.

Закрытие приложения

Если оно все еще выполняется, закройте окно консоли для приложения калькулятора.