Особенности использования набора компиляторов и утилит GCC.

Цель работы.

Изучить особенности работы с набором компиляторов и утилит GNU Compiler Collection (GCC)

При использовании интегрированных средств разработки (англ. Integrated Development Environment - IDE), процесс создания исполняемого файла из исходного текста происходит после нажатия одной кнопки, при этом результат каждого отдельного этапа, как правило, скрыт от пользователя IDE. Рассмотрим данный процесс поэтапно при помощи набора компиляторов GCC, а также другие его возможности. Дальнейшее описание предполагает работу в операционной системе семейства Windows, с установленным IDE Dev-Cpp. Будет использован набор компиляторов GCC, входящий в состав IDE Dev-Cpp. Путь установки для Dev-Cpp предполагается следующий: **C:\dev-cpp.** Рабочий каталог для хранения исходных текстов - **c:\users**. При использовании других путей к рабочему каталогу и Dev-Cpp, необходимо соответствующим образом отредактировать приведённые команды. При использовании операционной системы из семейства Linux не требуется устанавливать IDE Dev-Cpp и указывать путь к программам, входящим в состав GCC. В случае использования языка программирования C++, вместо **gcc** следует использовать **g++**.

Порядок выполнения работы.

Изучение этапов компиляции при помощи набора компиляторов GCC.

Для начала создадим простой файл исходного текста программы на языке C с использованием любого текстового редактора (например Notepad) и сохраним его под именем **hworld.c** в рабочем каталоге.

```
#include <stdio.h>

int main()
{

        printf("Hello World\n");
        return 0;
}
```

После этого можно командой **C:\dev-cpp\bin\gcc -o c:\users\hworld.exe c:\users\hworld.c** создать исполняемый файл, однако в данной лабораторной рассмотрим каждый этап отдельно. В дальнейшем полный путь к файлам в командах будет опускаться, предполагается, что все исходные файлы расположены в рабочем каталоге **c:\users**. При использовании новой программы из комплекта GCC, путь к ней будет указан лишь в первый раз.

Применив команду **gcc -E -o hworld.i hworld.c** мы создадим файл **hworld.i,** содержащий исходный текст, обработанный препроцессором. В данный файл будет добавлено содержимое заголовочных файлов, будут удалены комментарии и раскрыты макросы.

Применив команду **gcc -S -o hworld.s hworld.c** мы получим файл с ассемблерным кодом, соответствующим исходному тексту.

Результатом команды **gcc -c -o hworld.o hworld.c** будет объектный файл, содержащий блоки готового к исполнению машинного кода, блоки данных, а также список определенных в файле функций и внешних переменных, но при этом в нем не заданы абсолютные адреса ссылок на функции и данные. Объектный файл не может быть запущен на исполнение непосредственно, но в дальнейшем может быть объединен с другими объектными файлами.

Важно также не менять расширения созданных файлов, так как они указывают GCC какие этапы компиляции уже были проделаны.

Оптимизация кода.

GCC позволяет провести оптимизацию кода во время компиляции. Рассмотрим этот процесс. Создадим следующий файл под именем **optim.c**:

```
#include <stdio.h>

int main()
{
        int i;
        while (i<=10)
           i++;
        printf("%d",i);
        return 0;
}
```

Теперь выполним команды

gcc -S optim.c
gcc -S -O1 -o optim1.s optim.c
gcc -S -O2 -o optim2.s optim.c
gcc -S -O3 -o optim3.s optim.c

После этого в файле **optim.s** будет ассемблерный код без оптимизаций, в файлах **optim1.s**,**optim2.s** и **optim3.s** будет код с различной степенью оптимизации.

Раздельная компиляция и статические библиотеки.

GCC поддерживает раздельную компиляцию и не требует при этом специальных указаний от пользователя. Далее будет принято, что файл **main.c** содержит исходный текст основной программы, файлы **first.c**, **second.c** - исходные тексты используемых в основной программе внешних функций, а файлы **first.h** и **second.h** - соответствующие им заголовочные файлы, подключённые в файле **main.c**. Для начала с помощью команды **gcc -c** создадим из исходных файлов объектные. Затем, применив команду **gcc -o main.exe main.o first.o second.o** можно объединить три ранее созданных объектных файла в один исполняемый файл **main.exe**.

Теперь рассмотрим процесс создания статичной библиотеки. Для этого потребуется заголовочный файл **hworld.h**, содержащий прототипы функций из **first.c** и **second.c**. Данный заголовочный файл будет подключён в файле **main.c**. Выполним следующую последовательность команд:

gcc -c first.c
gcc -c second.c
C:\dev-cpp\bin\ar crs libhworld.a first.o second.o

Результатом выполнения программы **ar** будет создание библиотеки **libhworld.a**, содержащей объектные файлы **first.o** и **second.o**. Для программы **ar** заданы следующие опции:

Опция **c arname** - создать архив, если архив с именем **arname** не существует, он будет создан, в противном случае файлы будут добавлены к имеющемуся архиву.

Опция **r** - задает режим обновления архива, если в архиве файл с указанным именем уже существует, он будет удален, а новый файл дописан в конец архива.

Опция **s** - добавляет (обновляет) индекс архива. Индекс архива необходим для ускорения работы с библиотекой - для того чтобы найти нужное определение, отпадает необходимость просматривать таблицы символов всех файлов архива, можно сразу перейти к файлу, содержащему искомое имя.

Для компиляции с использованием библиотеки используется следующая команда:

gcc -o main main.o -L. -lhworld

Опция **-lname** позволяет подключить библиотеку с именем **libname,** в описанном случае в качестве **name** указано **hworld**, такое же имя должен иметь подключаемый заголовочный файл.

Опция **-L <путь>** передаёт путь к библиотеке, в описанном случае **-L.** показывает что искать следует в каталоге с исходным текстом.

Задания к лабораторной работе.

Часть I

1. Написать программу 1 в соответствии с вариантом при помощи любого текстового редактора.
2. Провести поэтапную компиляцию исходного текста написанной программы, разобраться в результатах, полученных на каждом этапе компиляции.

3. Провести оптимизацию кода написанной программы с помощью набора компиляторов GCC, пояснить внесённые для оптимизации кода изменения.

Часть II

4. Написать программы 2 и 3 в соответствии с вариантом при помощи любого текстового редактора. Функции для работы с массивом вынести в отдельные файлы: в одном файле описать функции для ввода/вывода массива, в другом - для обработки массива. В обеих программах должны использоваться одни и те же функции для ввода/вывода массивов, описанные в одном из этих файлах.
5. Провести раздельную компиляцию написанных файлов.
6. Скомпилировать обе программы, используя созданные объектные файлы, обе программы должны использовать один и тот же объектный файл с функциями для ввода/вывода массива.
7. Создать статическую библиотеку для ввода/вывода массива и продемонстрировать возможности по её подключению.

Варианты к заданию.

Вариант 1

1. Дано натуральное n. Вычислить n сомножителей произведения $\frac{2}{1}\cdot\frac{2}{3}\cdot\frac{4}{3}\cdot\frac{4}{5}\cdot\frac{6}{5}\cdot\frac{6}{7}\cdot\ldots$
2. Сформировать новый массив из элементов массива M (25), встречающихся в этом массиве только один раз.
3. Определить, представляют ли собой элементы массива A (20) возрастающую последовательность.

Вариант 2

1. Определить количество натуральных трехзначных чисел, сумма цифр которых равна заданному числу N.
2. Записать элементы массива C (20) в обратном порядке $\{C_{20};C_{19};C_{18};\ldots;C_2;C_1\}$. Вспомогательный массив не использовать.
3. Вставить число 100 после второго положительного элемента массива A (15).

Вариант 3

1. Дано натуральное число N. Составить программу для сравнения цифр старшего и младшего разрядов этого числа.
2. Определить количество элементов массива M (22), больших среднего арифметического значения элементов этого массива.
3. В массиве A (45) найти локальные максимумы, определить их местоположение. Локальным максимумом назовем элемент массива, значение которого больше, чем значения двух соседних с ним элементов.

Вариант 4

1. Дано натуральное число N. Составить программу для определения количества цифр в этом числе
2. Найти 2 первых элемента в массиве C (17), значения которых не попадают в заданный с клавиатуры диапазон [A, B]. Поменять их местами.
3. В массиве A (35) найти минимум, определить его местоположение (с учетом возможного повторения).

Вариант 5

1. Составить программу поиска двузначных чисел таких, что если к сумме цифр этого числа прибавить квадрат этой суммы, то получится это число.
2. Удалить из массива M (25) все элементы, значения которых в этом массиве повторяются, оставив по одному.
3. Вычислить сумму отрицательных элементов массива D (19), кратных четырем.

Вариант 6

1. Найти сумму целых положительных чисел из промежутка от A до B, кратных 4. Значения A и B вводятся с клавиатуры.
2. Поменять местами максимальный отрицательный и первый положительный элементы массива B (18).
3. Вставить число 0 в середину массива M (20), предварительно сдвинув вправо значения элементов массива, начиная с 11-го.

Вариант 7

1. Для натурального числа N получить все его натуральные делители.
2. Вычислить сумму элементов массива M (15), значения которых лежат в введенном с клавиатуры диапазоне [X, Y].
3. Удалить из массива A (20) первый отрицательный элемент.

Вариант 8

1. Сумма цифр трехзначного числа кратна 7, само число также делится на 7. Найти все такие числа.
2. Поменять местами максимальный и последний отрицательный

элементы массива A (40).

3. Сформировать новый массив из элементов заданного целочисленного массива M (50), кратных 7 или содержащих в записи числа цифру 7.

Вариант 9

1. Дано натуральное N. Составить программу для поиска первой цифры этого числа.
2. Удалить из массива M (26) первый положительный элемент.
3. Заменить все четные элементы массива A (20) на их квадраты, а нечетные удвоить.

Вариант 10

1. Среди четырехзначных чисел выбрать те, у которых все 4 цифры различны.
2. Удалить из массива B (50) все элементы, кратные 3 или 5.
3. Определить, есть ли в массиве Q (10) заданное число X, и если нет, то найти ближайшее к нему.

Вариант 11

1. Поменять местами цифры старшего и младшего разрядов данного натурального числа (например, из числа 3879 получится 9873).
2. Найти сумму элементов массива A (45), находящихся между максимальным и минимальным значениями.
3. Сформировать новый массив из положительных нечетных элементов заданного массива P (20).

Вариант 12

1. Ввести натуральные числа A и B. Определить все числа, кратные

A и B, меньшие A*B.

2. Определить местоположение элементов массива A (30), не встречающихся в массиве B (15).
3. Найти сумму четных элементов массива M (15), имеющих четные индексы.

Вариант 13

1. Определить, являются ли натуральные числа A и B взаимно простыми. Взаимно простые числа не имеют общих делителей, кроме единицы.
2. Определить, есть ли в массиве M (15) пары соседних одинаковых элементов.
3. Найти наибольший отрицательный элемент массива A (23) и удалить его.

Вариант 14

1. Дано натуральное число N. Вычислить $S=1+2^2+3^3+\ldots+N^N$. Формулу возведения в степень не использовать.
2. Поменять местами первый и последний отрицательные элементы массива B (18).
3. Определить, является ли массив M (20) перестановкой последовательности натуральных чисел от 1 до 20, т. е. проверить, все ли числа из этого диапазона входят в указанный массив.

Вариант 15

1. Исходное данное - натуральное число K, выражающее площадь. Написать программу для нахождения всех таких

прямоугольников, площадь которых равна K и стороны выражены натуральными числами.

2. Дан массив целых чисел B (30). Определить, сколько из них делится на 7 без остатка.
3. Вычислить $Z = \frac{S_n + S_o}{S_n - S_o}$, где S_n и S_o – суммы положительных и отрицательных элементов массива A (70).

Контрольные вопросы.

1) Что такое GCC и что входит в его состав?
2) Что такое компиляция?
3) Этапы компиляции.
4) Параметры программы gcc.
5) Что происходит при оптимизации кода?
6) Что такое раздельная компиляция?
7) Как создать и использовать статичные библиотеки?

Использована статья “О GCC, компиляции и библиотеках“, расположенная по адресу http://pyviy.blogspot.ru/2010/12/gcc.html

Задачи взяты из методического пособия “Основы программирования на языке Си”, составители: Арипова Ольга Владимировна, Бузюкина Ольга Александровна