Практическое занятие 5. Препроцессор.

ВАЖНО! Препроцессор языка Си предоставляет возможность компиляции с условиями. Это позволяет иметь несколько вариаций одного и того же кода. Обычно такой подход используется для учета особенностей компилятора, платформы и пр.

Задание 1. Условная компиляция.

Цель – познакомиться с возможностями условной компиляции.

1. В домашнем каталоге создаем и переходим в папку *TASK_05_01*, в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_05_01 <Enter>
$cd TASK_05_01 <Enter>
$
```

2. Создадим файл "**prep1.c**" и добавляем в него следующие строки:

```
// В заголовочном фале <stdio.h> описана функция printf
#include <stdio.h>

int main(int argc, char *argv[]){
#ifdef MY_DEF
  // Если MY_DEF определена, то препроцессор выбирает этот код
  printf("1. Hello world! MY_DEF is DEFINED.\n");
#else
  printf("2. Hello world! MY_DEF is UNDEFINED.\n");
#endif
}
```

3. Собираем программу:

```
$gcc prep1.c -o prep1.x <Enter>
$
```

В результате появится выполняемый файл с именем "**prep1.x**".

ЗАМЕЧАНИЕ Ключ "**-o**" позволяет задать имя выходного исполняемого файла. В случае если эта опция не задана, то формируется файл с именем "**a.out**".

4. Запускаем программу и анализируем результат вывода на экран. Учитывая, что "**MY_DEF**" не определена, на экран будет выведена строка

```
./prep1.x  <Enter>
2. Hello world! MY_DEF is UNDEFINED.
$
```

5. Инициализируем константу препроцессора через аргумент компилятора "**-D MY_DEF**", собираем программу:

```
$gcc prep1.c -DMY_DEF -o prep1.x <Enter>
$
```

ЗАМЕЧАНИЕ Ключ компилятора "**-D**" позволяет задавать (инициализировать или передавать программе) любые константы препроцессора. Обратите внимание, на синтаксис: имя константы идет сразу после ключа "**-D**", то есть без пробела.

6. Запускаем новую версию программы. Так как константа "**MY_DEF**" теперь определена, то на экране появится первая строка из условия:

```
./prep1.x  <Enter>
1. Hello world! MY_DEF is DEFINED.
$
```

7. Рассмотри способ задания константы препроцессора непосредственно в коде при помощи директивы "**#define**". Для этого в текст программы добавляем инициализацию "**MY_DEF**":

```
#include <stdio.h>

// Определяем константу MY_DEF
#define MY_DEF

int main(int argc, char *argv[]){
#ifdef MY_DEF
  // Если MY_DEF определена, то препроцессор выбирает этот код
  printf("Hello world! MY_DEF is DEFINED.\n");
#else
  printf("Hello world! MY_DEF is UNDEFINED.\n");
#endif

  return 0;
}
```

8. Самостоятельно собрать и запустить программу. Сравнить и проанализировать результат выполнения модифицированной и немодифицированной версий программы.

Задание 2. Константные макроопределения.

ВАЖНО! Как правило, все постоянные в коде программ задают с использованием константных макроопределений.

Цель – научиться использовать константные макроопределения.

1. В домашнем каталоге создаем и переходим в папку "**TASK_05_02**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
```

```
$mkdir TASK_05_02 <Enter>
$cd TASK_05_02 <Enter>
$
```

2. Создаем файл "**prep2.c**" и добавляем в него следующие строки:

```
#include <stdio.h>

// Определяем константу HELLO_STR
#define HELLO_STR "Hello world!\n"

int main(int argc, char *argv[]){
  printf(HELLO_STR);

  return 0;
}
```

3. Самостоятельно собрать/запустить программу и проанализировать результат вывода на экран.
4. Самостоятельно модифицировать программу, таким образом, чтобы константа "**HELLO_STR**" задавалась ключом при компиляции.

Задание 3. Препроцессор. Макросы (макроопределения).

ВАЖНО! Константы и макросы препроцессора используются для определения небольших фрагментов кода. Макросы позволяют упрощать программу: вместо фразы можно указать её идентификатор.

1. В домашнем каталоге создаем и переходим в папку "**TASK_05_03**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_05_03 <Enter>
$cd TASK_05_03 <Enter>
$
```

2. Создаем файл "**prep3.c**" и добавляем в него следующие строки, в том числе макрос "**HYPOTENUSE**" (вычисление гипотенузы):

```
#include <stdio.h>
// math.h  - заголовочный файл стандартной библиотеки простых
// математических операций. В этом заголовочном фале задекларирована
// функция sqrtf()
#include <math.h>

// Макрос HYPOTENUSE с двумя аргументами для вычисления гипотенузы
// прямоугольного треугольника
#define HYPOTENUSE(X, Y) sqrtf(X*X + Y*Y)

int main(int argc, char *argv[]){
```

```
  float x = 3.0;
  float y = 4.0;

 // Вычисляем значение гипотенузы при помощи макроса
  float hyp = HYPOTENUSE(x, y);

  // Выводим на экран вычисленное значение
  // Синтаксис printf. Вместо %f будет подставлен аргумент hyp
  // типа float
  printf("hypotenuse = %f\n", hyp);

  return 0;
}
```

3. Собираем программу:

```
$gcc prep3.c -o prep3.x -lm <Enter>
$
```

В результате появится выполняемый файл с именем "**prep3.x**".

ЗАМЕЧАНИЕ А) **"-lm"** в конце команды для компиляции означает прилинковать библиотеку "**libm**" простых математических функций. Именно в этой библиотеке реализована функция вычисления квадратного корня "**sqrtf**".

В) В данном контексте "**l**" - это сокращение от "**lib**".

4. Запускаем программу и анализируем результат вывода на экран.

Задание 4. Включение заголовочных файлов.

ВАЖНО! Как правило, декларирование всех функций и все макросы содержатся в заголовочных файлах, имеющих расширение "***.h**". Далее эти заголовочные файлы используют, включая их в коде директивой "#include". Именно так включались заголовочные файлы стандартной библиотеки "**stdio.h**" и "**math.h**" в примерах выше.

1. В домашнем каталоге создаем и переходим в папку "**TASK_05_04**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_05_04 <Enter>
$cd TASK_05_04 <Enter>
$
```

2. Создаем файл "**prep4.c**" и добавляем в него следующие строки:

```
#include <stdio.h>
// Включаем собственный заголовочный файл (на жаргоне - хидер) prep4.h
// в котором определим макрос HYPOTENUSE
#include <math.h>
```

```
#include "prep4.h"

int main(int argc, char *argv[]){
  float x = 3.0;
  float y = 4.0;

  float hyp = HYPOTENUSE(x, y);

  printf("hypotenuse = %f\n", hyp);

  return 0;
}
```

3. Создаем заголовочный файл "**prep4.h**" и добавляем в него определение макроса "**HYPOTENUSE**":

```
#define HYPOTENUSE(X, Y) sqrtf(X*X + Y*Y)
```

4. Самостоятельно собрать/запустить программу и проанализировать результат вывода на экран.

Задание 5. Защита от повторного включения файлов.

ВАЖНО! Очень часто, когда в код программы директивой "**#include**" включается большое число файлов, возможно их повторное включение. Это может привести к рекурсии и, как следствие, к ошибке при компиляции. Чтобы этого не происходило, используют простую, но эффективную защиту, которую мы рассмотрим в этом задании.

1. В домашнем каталоге создаем и переходим в папку "**TASK_05_05**", в которой будем выполнять все операции задания:

```
$cd ~ <Enter>
$mkdir TASK_05_05 <Enter>
$cd TASK_05_05 <Enter>
$
```

2. На основе кода из предыдущего задания создаем файлы "**prep5.c**" и "**prep5.h**". Файл "**prep5.c**":

```
#include <stdio.h>
#include <math.h>
#include "prep5.h"

int main(int argc, char *argv[]){
  float x = 3.0;
  float y = 4.0;

  float hyp = HYPOTENUSE(x, y);
```

```
  printf("hypotenuse = %f\n", hyp);

  return 0;
}
```

Файл "**prep5.h**":

```
#define HYPOTENUSE(X, Y) sqrtf(X*X + Y*Y)
```

3. Собираем программу, убеждаемся в том, что программа компилируется и корректно работает.
4. Далее, в файл "**prep5.h"** добавляем строку включения самого себя:

```
#define HYPOTENUSE(X, Y) sqrtf(X*X + Y*Y)
#include "prep5.h"
```

Очевидно, что это приведет к рекурсивному включению файла "**prep5.h"**, что, конечно, является ошибочной ситуацией.

5. Собираем программу и убеждаемся в наличие ошибки компиляции, связанной с многократным включением заголовочного файла "**prep5.h"**.
6. Решаем проблему повторного включения файлов при помощи условной компиляции. В заголовочный файл "**prep5.h**" добавляем проверку на инициализацию константы препроцессора "**_PREP5_H_**". Если эта константа не определена, то мы попадаем внутрь условия и, в том числе, определим константу "**_PREP5_H_**". При повторном включении заголовочного файла константа "**_PREP5_H_**" будет уже определена, условие не будет выполнено, а повторное использование заголовочного файла будет исключено.

```
#ifndef _PREP5_H_
#define _PREP5_H_

#define HYPOTENUSE(X, Y) sqrtf(X*X + Y*Y)
#include "prep5.h"

#endif
```

7. Собираем программу, убеждаемся в том, что программа компилируется и корректно работает.

ЗАМЕЧАНИЕ Мы будем использовать защиту повторного включения файлов во всех созданных заголовочных файлах. Константу препроцессора, используемую в конструкции защиты будем называть по имени файла, заглавными буквами, вместо символа "**.**" будем

писать символ "_", а также этот символ будем добавлять в начале и в конце имени константы.