Программирование на языке С

Модуль 5.
ПРЕПРОЦЕССОР

- Препроцессорные директивы: `#include`, `#define`, `#undef`, `#if-#else-#endif`
- Макроопределения с параметрами
- Правила оформления деклараций

Назначение препроцессора. Стадии препроцессорной обработки

- **Препроцессор** — компонент среды разработки, выполняющий предварительную обработку исходных текстов программы до передачи их компилятору.
- **Стадии препроцессорной обработки** [Под04]
  - унификация системно-зависимых обозначений (индикаторов конца строки и т.д.);
  - объединение строк, разделенных символами '\' и ↲ (перевод строки);
  - распознавание директив и лексем препроцессора (см. ниже), замена каждого комментария одним обобщенным пробельным символом;
  - выполнение директив препроцессора и макроподстановок `#define`;
  - замена escape-последовательностей числовыми кодами символов;
  - конкатенация строковых литералов;
  - перевод лексем препроцессора: символьных и строковых литералов, имен включаемых файлов, идентификаторов, знаков операций и пунктуации, препроцессорных чисел и всех

- **Обобщенный формат директив препроцессора**

  ␣#␣<*имя директивы*> <*лексемы препроцессора*>␣

  где ␣ суть необязательный обобщенный пробельный символ

- **Директивы включения исходного текста**
  - из файла, находящегося в стандартных системных каталогах (1)

    `#include` <*имя файла*>*, **
  - из файла, находящегося в текущем каталоге или стандартных системных каталогах (2)

    `#include "`*имя файла*`"`

- из файла, предусмотренного случаями (1) или (2) (см. выше)

  `#include` *<идентификатор макроопределения>*[***]

  - [*] — здесь парные угловые скобки <> являются элементом грамматики, а не метасимволом описания;
  - [**] — стандартными заголовочными файлами языка Си являются: assert.h — функции диагностического характера, ctype.h — функции обработки символьных данных, errno.h — функции проверки ошибок, float.h — функции обработки вещественных данных, limits.h — предельные (препроцессорные) значения целочисленных данных, locate.h — функции поддержки национальной операционной среды, math.h — библиотека математических функций, setjump.h — поддержка нелокальных переходов вычислительного процесса, signal.h — функции обработки исключительных ситуаций, stdarg.h — поддержка функций с переменным числом параметров, stddef.h — дополнительные определения, stdio.h — функции стандартного ввода-вывода, stdlib.h — функции общего назначения, string.h — функции обработки символьных строк, time.h — функции для работы с датой и временем;
  - [***] — здесь парные угловые скобки <> являются метасимволом описания, а *идентификатор макроопределения* после конечного количества подстановок должен быть преобразован к виду (1) или (2) (см. выше);

- **Директивы определения (отмены макроопределения) макросов и препроцессорных идентификаторов** (не выполняются в комментариях, строковых и символьных литералах)

Определяют идентификатор (имя макро) и последовательность символов, которая будет подставляться вместо этого идентификатора каждый раз, когда он встретится в исходном файле. Этот процесс еще называют макрозаменой или макроподстановкой.

1 сп. **#define <***имя_макро***>**

2 сп. **#define <***имя_макро***> <***строка подстановки***>**

3 сп. **#define <***имя_макро***>(<***список формальных параметров***>) <***строка подстановки***>**

Примечание: Такие макроопределения (3 сп.) называют также макрофункциями

- «Антипод» к директиве **#define** это директива

  #undef <*имя_макро***>**

  Она отменяет сделанное ранее макроопределение.

Практика

- Использовать препроцессор для создания удобных/коротких имён
- Написать макрофункцию для обмена значений у 2-х переменных

- Написать макрофункцию для возведения числа в куб

- **Директивы условной компиляции**

#if *<целочисленное константное выражение>*
#ifdef *<идентификатор>* -- эквивалентно
```
#if defined
```
#ifndef *<идентификатор>* -- эквивалентно
```
#if !defined
```
#else
#elif *<целочисленное константное выражение>*

#endif

#ifdef *имя_макро*

последовательность строк исходного кода

#else

последовательность строк исходного кода

#endif

Примечание: ifdef – сокращение от if defined

#ifndef *имя_макро*

последовательность строк исходного кода

#else

последовательность строк исходного кода

#endif

Примечание: ifndef – сокращение от if not defined

#if *константное_выражение*

последовательность строк исходного кода

#elif *константное_выражение*

последовательность строк исходного кода

#else

последовательность строк исходного кода

#endif

Примечание: elif – сокращение от else if

Защита файлов от повторного включения директивой include

Во включаемом файле необходимо поместить следующую конструкцию:

```
#ifndef <имя_макро>
#define <имя_макро>
    <текст заголовочного файла>
#endif
```

Примечание: <имя_макро> - формальное имя, которое, как правило, содержит имя (часть имени) включаемого файла. Например, при подключении файла с именем rus.h, имя макро может быть:

A) _INC_RUS

или

B) _RUS_H

- **Вспомогательные директивы**
  - смена номера следующей строки файла с исходным текстом программы

    `#line` *<десятичная целочисленная константа>*
  - формирование сообщения об ошибке времени компиляции

    `#error` *<последовательность лексем>*
  - выполнение действий, предусмотренных средой разработки

    `#pragma` *<последовательность лексем>*
  - пустая директива

    `#`

Препроцессорные операции

- **Унарная операция в директивах `#if` и `#elif`**

  `defined [(]<`*операнд*`>[)]` — выражение принимает значение 1L, если операнд определен как препроцессорный идентификатор, в противном случае — 0L

  `!defined [(]<`*операнд*`>[)]` — выражение принимает значение 1L, если операнд не определен как препроцессорный идентификатор, в противном случае — 0L

- **Операции в *строке замещения* директивы `#define`**

  `#<`*операнд*`>` — унарная операция с префиксной формой записи, требующая заключить текст, замещающий *операнд* при выполнении макроподстановки `#define`, в двойные кавычки;

  `##` — бинарная операция с инфиксной формой записи, реализующая конкатенацию препроцессорных лексем, которые она связывает

- **Стандартные встроенные макроимена препроцессора**
  - `__LINE__` — номер текущей обрабатываемой строки исходного текста;
  - `__FILE__` — имя компилируемого файла (чувствительно к директиве `#include`);
  - `__DATE__` — дата начала препроцессорной обработки исходного текста в формате "*Mon dd yyyy*";
  - `__TIME__` — время начала препроцессорной обработки исходного текста в формате "*hh:mm:ss*«
  - `__func__` — имя функции.

Список литературы

- [КР92] Керниган Б., Ритчи Д. Язык программирования Си / Пер. с англ. — М.: Финансы и статистика, 1992. — 272 с.
- [КР06] Керниган Б., Ритчи Д. Язык программирования С / Пер. с англ. — М.: Вильямс, 2006. — 304 с.
- [Под04] Подбельский В.В., Фомин С.С. Программирование на языке Си. – 2-е доп. изд. – М., Финансы и статистика, 2004. – 600 с.