Краткие сведения о препроцессоре Си

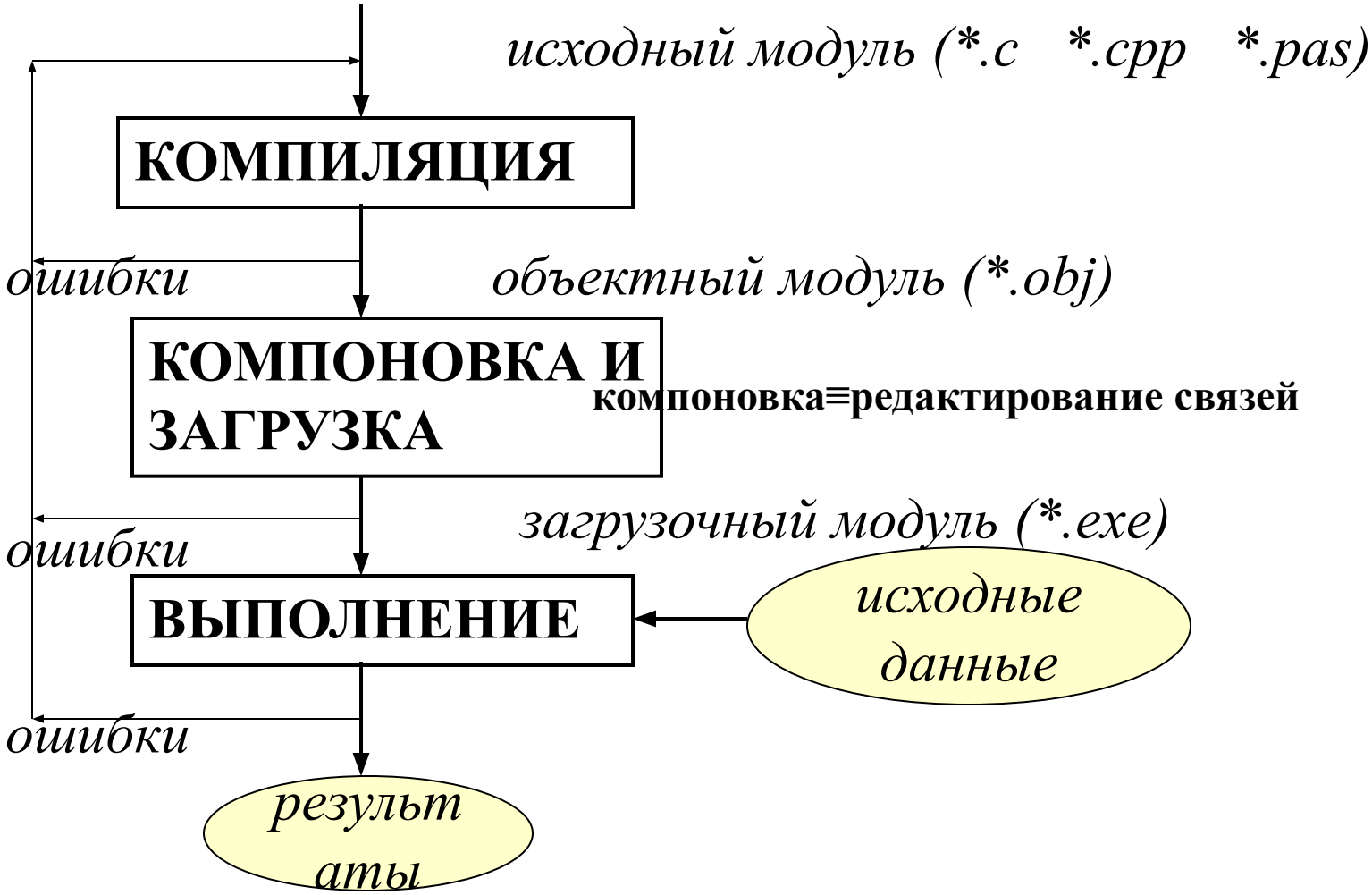
Лекция №3

Часть 3

Что такое препроцессор?

- Исходный текст Си-программы, кроме инструкций алгоритмического языка Си, содержит также директивы препроцессора (которые не являются инструкциями алгоритмического языка Си).
- Препроцессор – это программа, которая обрабатывает текст Си-программы и свои директивы, в результате чего формируется программа на алгоритмическом языке Си, не содержащая директив препроцессора и в дальнейшем подлежащая компиляции.
- Как правило, код на выходе препроцессора существенно превышает код на входе.

Этапы прохождения через ЭВМ программы на любом алгоритмическом языке, требующем компиляции



Этапы прохождения через ЭВМ Си-программы

Директивы препроцессора позволяют:

- Сделать исходный код (**.c *.cpp*) короче.
- Сделать исходный код более наглядным, удобочитаемым.
- Получать разные Си-программы (**.i*) в зависимости от выполнения некоторых условий.

Директивы препроцессора:

- Директива макроопределения #define
- Директива #include включения файлов
- Директивы условной компиляции

Директивы препроцессора начинаются со знака # в первой позиции строки.

Директива макроопределения #define

- Эта директива обеспечивает замену сокращений (так называемых *макро*) на полный текст.
- Макро также называют *макросом* или *макроопределением*.
- Общий вид директивы:

  #define *Имя_Макро Последовательность_Символов*
- Каждое вхождение *Имя_Макро* заменяется препроцессором на *Последовательность_Символов*.
- Процесс замены называется *расширением макро*.
- Расширению не подлежат группы символов, входящие в состав комментариев, строковых литералов и символьных констант, инструкций языка.

Простейшие примеры #define

#define N 100 *Чем отличается от const N=100?*

#define PRIGL "Введите исходные данные "

Наиболее распространенное применение #define - это обозначение часто встречающихся констант, числовых и строковых.

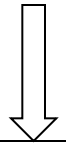
Принято для обозначения макро использовать большие буквы, так же как малые для имен переменных программы (но это не обязательно).

Если последовательность символов не помещается на одной строке, то для переноса используется символ \:

```
#define PRIMER "Наш текст не помещается\
на одной строке"
```

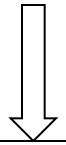
Отличие простейшего макро от константы программы

#define N 100



После препроцессорной обработки в Си-программе имени N не будет. Препроцессор все N заменит на 100.

const N=100



N – обозначение содержимого ячейки памяти, защищенной от записи, т. е. константа программы.

Директива #undef

- Директиву #define можно отменить директивой **#undef**:

  #undef *ИмяМакро*

- После директивы #undef *ИмяМакро* становится неопределенным, препроцессор прекращает расширение этого макроса.

Макро с параметрами

#define *Имя_Макро(Список_Параметров) Тело_Макро*

Каждый раз при расширении параметры заменяются параметрами макро в программе.

Чем макро с параметрами отличается от подпрограммы-функции?

Пример:

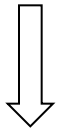
```
#define square(x) ((x)*(x)) /*макро с формальным параметром*/
...   int i,j; ...
j=square(i);  /*расширяется в j=((i)*(i)) */
```

Аргумент макро необходимо заключать в круглые скобки, чтобы вместо формального параметра можно было подставлять выражение. Пример "неприятности" при отсутствии скобок:

```
#define square(x) x*x /*макро с форм. параметром без скобок*/
...      int i,j,l;...
j=square(i+l);/*расширяется в j=i+l*i+l –вычисляется другое выражение!!!*/
```

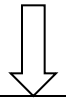
Отличие макро с параметрами от подпрограммы-функции

Макро с параметрами



Расширение макро приводит к вставке *в исходную программу* тела макро

Подпрограмма-функция



Обращение к подпрограмме генерирует *в объектном модуле* команду безусловного перехода с возвратом и передачу параметров

1. Применение макро увеличивает код программы, но экономит время выполнения (за счет отсутствия пересылки параметров).
2. Обычно не рекомендуется использовать макро вместо подпрограмм, так как при применении макро отсутствует всякий контроль правильности подстановки параметров.

Предопределенные макро

Директива #include включения файлов

Директива **#include** позволяет включать в исходную программу любые текстовые файлы.

Формы директивы include

1. **#include <СпецификацияФайла>** //поиск файла в стандартных //директориях
2. **#include "СпецификацияФайла"**

// поиск файла по маршруту, заданному //спецификацией

3. **#include ИмяМакро** //существует макроопределение, заменяющее *Имя_Макро* на спецификацию файла в угловых скобках или двойных кавычках.

Директивы условной компиляции

Эти директивы позволяют производить выборочную компиляцию программы.

Самый простой вид такой директивы:

#if ЛогическоеВыражение
    TRUE-секция
#else
    FALSE-секция
#endif

Пример:

В зависимости от того, определен ли макрос NAME, регулируется значение макроса BUF.

```
#ifdef NAME
#define BUF 135
#else
#define BUF 80
#endif
```