8. Подпрограммы

8.1. Функции и процедуры

В программе на Си **должна быть обязательно функция main** (в VC++ при использовании непустого проекта _tmain). Она обычно явно не вызывается, но ее можно и вызвать (и передать ей параметры):

```
#include <stdio.h>
int i=10;
int main(void)
  {
   print f(“%d”, --i);
   if (i) return ( main( ) ) ;
   return (0);
  }
```

действия (вывод i на экран) производятся на спуске
(на возврата return что-то возвращает, но мы это не используем

рекурсия (функция вызывает саму себя)

при рекурсивных вызовах

На экран будет выведено: 9876543210

по return(0);

В Паскале есть процедуры и функции, а **в Си - только функции**. Однако в Си можно объявить функцию с возвращаемым значением типа void, что позволит ей не возвращать никакого значения (не нужен будет и оператор return), кроме того, можно просто игнорировать значение, выдаваемое функцией, и **вызывать функцию как процедуру**.

1

В Си имеется возможность явно **указывать порядок передачи параметров**. Это выполняется с помощью модификатора типа функции cdecl (_cdecl), _pascal, _stdcall. С этими модификаторами определение функции, например, может иметь следующий вид:

void _cdecl f(void){…}.

1) _stdcall задает, что: а) аргументы передаются в стек справа налево; б) добавляется символ подчеркивания к имени (в начале имени); в) стек чистит (выталкивает из него через pop свои собственные параметры) вызываемая (called) функция; д) регистр букв учитывается (не меняется); г) параметры передаются по значению. Используется для вызова Win32-функций;

2) _cdecl задает, что: а) учитывается регистр букв; б) добавляется символ подчеркивания к имени (в начале имени); в) параметры копируются ("пушируются") в стек в обратном порядке (начиная с последнего); г) стек чистит вызывающая (calling) функция. _cdecl выбирается по умолчанию.

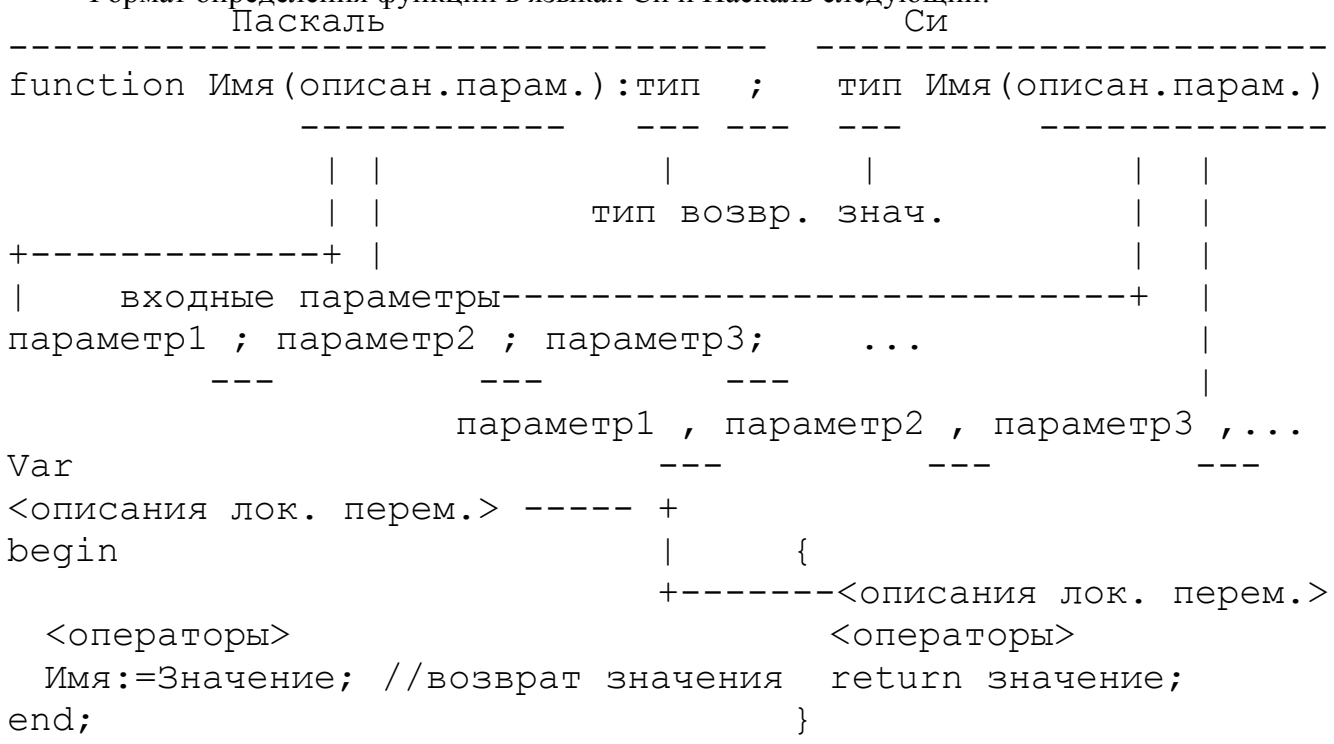
3) _pascal (больше не поддерживается так же как и _fortran) задает, что: а) все символы будут приведены к одному регистру (в итоге не учитывается регистр букв); б) не добавляется символ подчеркивания к имени (в начале имени); в) параметры копируются ("пушируются") в стек в прямом порядке (начиная с первого слева направо); г) стек чистит вызываемая функция.

Пример:

```
int f(...)...;          //без модификатора (по умолчанию будет _cdecl)
int pascal f (...) ...; //с модификатором
```

модификатор всегда располагается слева от того, что модифицируется.

Формат определения функции в языках Си и Паскаль следующий:



Другие важные различия видны из следующего **примера**: пусть имеются 2 глобальные переменные и надо написать функцию, которая, получая их через заголовок, складывает их и делит на некоторую локальную переменную.

Паскаль	Си

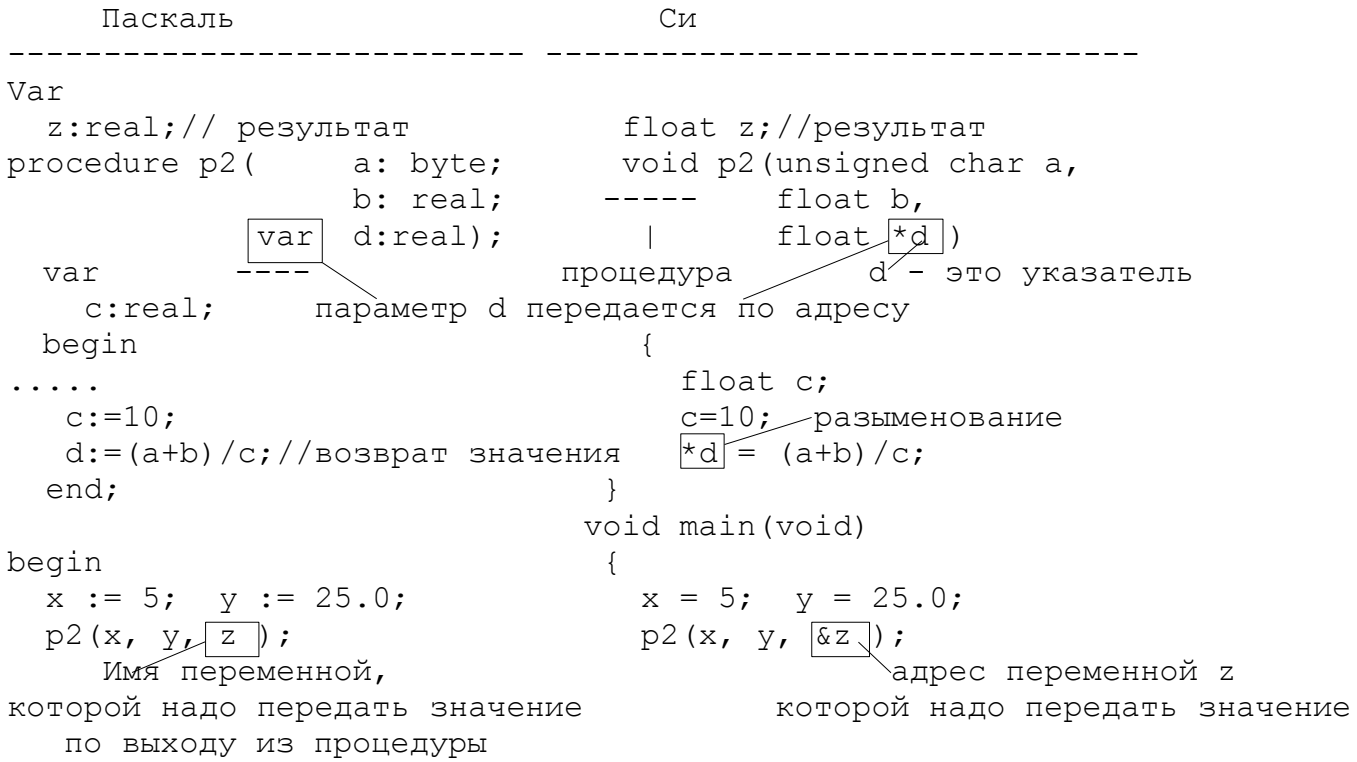
var
X: byte;
Y, Z : real;
function f2(a: byte; b: real):real;
var c: real;
begin
c := 10;
f2:= (a+b)/c;
end;
 	
begin
X := 5; Y := 25.0;
Z := f2(x, y);
end;
	
	unsigned char x;
	float y, z;
	float f2(unsigned char a, float b)
	
	{
	float c; c = 10;
	return ((a +b)/c) ;
	}
	void main (void)
	{
	x = 5; y = 25.0;
	z = f2(x, y);
	}

Рассмотрим, как сделать то же самое с помощью **процедуры**. При этом надо помнить, что в Паскале параметры передаются или по адресу, или по значению. В Си можно передавать **только по значению**. Поэтому для передачи по адресу поступают следующим образом:

1) в заголовке процедуры соответствующую переменную (которую надо передать по адресу) описывают как указатель на тип передаваемого значения;

2) в теле процедуры все действия с этой переменной выполняются через разыменование указателя;

3) при вызове процедуры в качестве соответствующего фактического параметра используется адрес той переменной, которой должно быть передано значение по выходе из процедуры.



Замечание: в C++ можно передавать параметры не по адресу, а **по ссылке**:

```
void  p2(unsigned char  a, float  b, float  &d )
 {
  float c;
  c = 10;
  d = (a/b) + c;
 }
void  main(void)
 {
  p2(x, y, z );
 }
```

ссылка может указывать только на переменную, а **не на любое место в памяти** (в отличие от обычного указателя)

здесь не надо указывать на то, что d – это указатель (и его надо разыменовывать)

здесь не надо указывать, что z - это адрес

В языке Си **недопустимы вложенные функции**. В программах на Паскале их использование, например, позволяет сократить область видимости подпрограмм и их объектов. В языке Си для изменения области видимости используются другие средства (классы памяти extern, static).

8.2. Прототипы функций

Паскаль всегда делает проверку на соответствие количества и типов параметров, определенных в функции, количеству и типам параметров, используемых при вызове этой функции. Допустим, что определена функция с целочисленными формальными параметрами:

function Max(I,J : Integer) : Integer;

а вызвать ее попытаемся с действительными фактическими параметрами

A:=Max(12, 34.56);

В случае Паскаля получим при трансляции ошибку, сообщающую, что присутствует несоответствие типов, так как значение 34.56 не является целым.

В случае Си это не так: по умолчанию компилятор Си не производит проверку ошибок при вызове функции. Кроме этого может возникнуть ошибка (не замеченная

компилятором), связанная с неправильным числом фактических параметров при вызове. Избежать этого позволяет использование **прототипов функций**.

Прототип функции **должен предшествовать фактическому вызову функции**. Прототип функции имеет формат:

<тип> имя(<тип><имя>,<тип><имя> и т.д.).

Это **похоже на опережающее описание функций в Паскале**, но с некоторыми отличиями:

1) описание каждого параметра разделяет запятая (а не точка с запятой), т.е. нельзя писать

   int f(float x ; float y);

   а надо писать

   int f (float x , float y);

2) нельзя задавать список нескольких <имен> для одного <типа>, т.е. нельзя писать

   int f(float x, y); y будет иметь тип int

   а надо писать

   int f(float x, float y).

Замечание: если описание функции (предположительно размещенное ниже прототипа) не будет соответствовать прототипу, компилятор Си выдаст ошибку.

Наличие прототипов в программе заставляет компилятор выполнять следующие действия:

- **контроль соответствия** числа и типов фактических и формальных параметров;

-**преобразование (если возможно) типов** фактических параметров в типы соответствующих формальных параметров.

Прототипы стандартных функций хранятся в виде специальных **заголовочных файлов** (h-файлов, от англ. header - заголовок), которые надо подключать к программе с помощью команды (оператора) препроцессора

#include <имя h-файла>.

В прототипах необязательно указывать имена формальных параметров. Можно указывать только их типы. В данном случае прототип процедуры p2 будет иметь вид:

void p2(unsigned char, float, float *).

Библиографический список

1. Полубенцева М.И. С/С++. Процедурное программирование. СПб.: БХВ-Петербург, 2008. 448 с.

2. Павловская Т.А. Паскаль. Программирование на языке высокого уровня. СПб.: Питер, 2010. 464 с.

3. Павловская Т.А. С/С++. Программирование на языке высокого уровня. СПб.: Питер, 2012. 461 с.

4. Культин Н.Б. Основы программирования в Turbo C++. СПб.: БХВ-Петербург, 2012. 464 с.

5. Пахомов Б.И. С/С++ и MS Visual C++ 2010 для начинающих. СПб.: БХВ-Петербург, 2011. 736 с.