Вариадические функции. Параметры функции main().

Допустимость неизвестного набора параметров

В Си функции, для которых не указаны все возможные **формальные** параметры (типы и количество), являются допустимыми.

Для таких функций количество и типы параметров становятся известными только при вызове функции, когда явно задается перечень **фактических** параметров.

Такие функции (с переменным количеством параметров) называются **вариадическими**.

Каждая вариадическая функция должна иметь хотя бы один **явный параметр** (который явно указывается как формальный при описании прототипа функции).

Описание вариадических функций

Формат прототипа функции с переменным количеством параметров:

```
тип имя(явные_параметры, …);
```

тип — тип значения, возвращаемого функцией.
имя — имя функции
явные_параметры — список спецификаций параметров, количество и типы которых фиксированы и известны на момент компиляции (обязательные параметры).

Многоточие (…) указывает компилятору, что дальнейший контроль количества и типов параметров при вызове функции не требуется.

Проблема: отсутствие какого-либо признака у списка параметров (нет даже имени), поэтому не ясно, где он начинается и где заканчивается.

Описание вариадических функций

Два подхода к формированию переменного списка параметров:

1. Добавление в конец списка необязательных параметров специального параметра-ограничителя с заданным (уникальным) значением. Это значение сигнализирует об окончании списка.

В этом случае параметры функции последовательно перебираются, их значения сравниваются с заранее известным концевым признаком.

2. Передача в функцию сведений о реальном количестве необязательных параметров с помощью дополнительного обязательного параметра.

Описание вариадических функций

В любом варианте переход от одного необязательного параметра к другому выполняется с помощью указателей и адресной арифметики.

Однако: разные реализации компиляторов для разных операционных систем по-разному размещают элементы (параметры) в памяти («снизу вверх» или «сверху вниз»).

Поэтому программы, написанные с использованием этих подходов «напрямую», получаются платформо-зависимыми (см. примеры из учебника Подбельского и Фомина).

Реализация вариадических функций

В стандартной библиотеке Си имеются универсальные средства создания вариадических функций (учитывающие особенности операционных систем и компиляторов). Так обеспечивается **мобильность программ.**

Они размещаются в файле **`stdarg.h`** и включают в себя тип **`va_list`** и макросы (макроопределения) **`va_start`**, **`va_end`** и **`va_arg`**.

Вариадическая функция должна иметь хотя бы один обязательный параметр с именем.
Имя последнего обязательного параметра используется макросом **`va_start`** для инициализации переменной типа **`va_list`**.

Необязательные параметры извлекаются из стека макросом **`va_arg`**.

При завершении обработки параметров следует поместить вызов макроса **`va_end`**.

Реализация вариадических функций

`va_list` – специальный тип, обеспечивающий обработку переменного списка фактических параметров.

В теле вариадической функции обязательно должна быть объявлена локальная переменная типа **`va_list`**. Такая переменная обладает свойствами указателя.

Например
```
va_list start;
```

В процессе работы с помощью вызова **`va_start(start, param)`** этот указатель позиционируется на начало списка неявных параметров (**`param`** – последний явный (обязательный) параметр).

«На самом деле» указатель **`start`** устанавливается на адрес **`param`**, а затем перемещается на **`sizeof(param)`** байтов, что соответствует адресу первого необязательного параметра. Поэтому у вариадической функции всегда должен быть хотя бы один обязательный (явный) параметр.

Реализация вариадических функций

После этого в **start** получается адрес первого необязательного параметра. Для корректной обработки параметра нужно указать его тип.

Это делается вызовом макроса **va_arg(start, param_type)** (например, **k=va_arg(start, int);**). После выдачи значения очередного параметра макрос **va_arg()** перемещает указатель **start** на адрес следующего параметра.

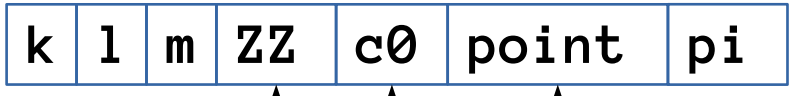
va_end(start); – обеспечивает корректный возврат из вариадической функции. Этот вызов нужно использовать только после завершения обработки всех параметров. При этом указатель **start** «сбрасывается».
Для повторной работы со списком параметров в этой же функции следует снова вызвать **va_start().**

Реализация вариадических функций

Пусть имеется
`void my_vfunc(int a, char b, float c, …);` - прототип.

Использование:

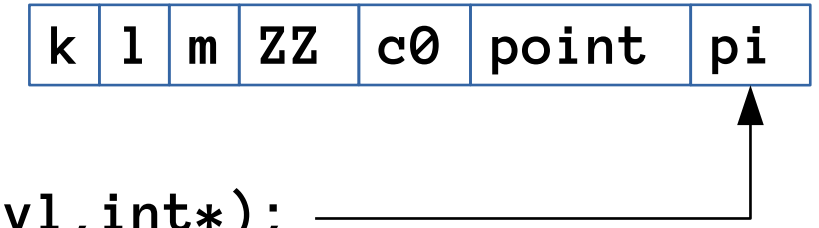
```
int YY,ZZ,p0;
char c0,c1;
double pi,phy;
int point[5], px[5];
va_list vl;
…
my_vfunc(k,l,m,ZZ,c0,point,pi);
```




```
va_start(vl,m);
YY=va_arg(vl,int);
c1=va_arg(vl,char);
```

Позиции указателя после выполнения инструкций. Указатель смещается на `sizeof(<type>)`.

Реализация вариадических функций




```
px=va_arg(vl,int*);
phy=va_arg(vl,double);

p0=px[0];
```

Позиции указателя после выполнения инструкций. Указатель смещается на **`sizeof(<type>)`**.

Реализация вариадических функций

Пример: (lect-019-01.c):
Функция, которая вычисляет сумму своих необязательных параметров (целых чисел).
Обязательный параметр — какое-то целое число.
Признаком окончания списка параметров является значение 0.

```
long summa(int k,...)
{
    va_list vl;
    long total,s;

    total=0;
    va_start(vl,k);
    while((s=va_arg(vl,int))!=0) total = total+s;
    va_end(vl);
    return total;
}
```

Д/З: написать функцию, вычисляющую произведение вещественных параметров, если задано количество фактических параметров

Реализация вариадических функций

Следующий пример: (lect-019-02.c):
Функция принимает на вход произвольное количество пар «указатель на строку — целое число» пока очередной указатель не равен `NULL` и печатает каждую строку соответствующее число раз.

```
void print_times(char *str,...)
{
    va_list vl;
    char *p;
    int n,i;

    va_start(vl,str);
    for(p=str;p;p=va_arg(vl,char*))
    {
        n=va_arg(vl,int);
        for(i=0;i<n;i++) printf("%s ",p);
        printf("\n");
    }
    va_end(vl);
}
```

Реализация вариадических функций

Следующий пример: (lect-019-03.c):
Функция выводит значения `int` и `float` в соответствии с форматной строкой (модель `printf()`).

```
void miniprint(char *format,...)
{
    va_list ap;
    char *p;
    int ii;
    double dd;

    va_start(ap,format);
    for(p=format;*p;p++)
    {
        if((*p)=='%') {/* тело ветки «Да» */}
        else printf("%c",*p);
    }
    va_end(ap);
}
```

Реализация вариадических функций

```
/* тело ветки «Да» */

switch(*(++p))
{
  case 'd':
    ii=va_arg(ap,int);
    printf("%d",ii);
    break;
  case 'f':
    dd=va_arg(ap,double);
    printf("%f",dd);
    break;
  default:
    printf("%c",*p);
}

/* Конец тела ветки «Да» */
```

Вариадические функции в стандартной библиотеке

```
scanf(format_string, param_list);
printf(format_string, param_list);

fscanf(FILE*, format_string, param_list);
fprintf(FILE*, format_string, param_list);

sscanf(string, format_string, param_list);
sprintf(string, format_string, param_list);
snprintf(string, MAXLEN, format_string, param_list);

vscanf(format_string, param_va_list);
vprintf(format_string, param_va_list);

vfscanf(FILE*, format_string, param_va_list);
vfprintf(FILE*, format_string, param_va_list);

vsscanf(string, format_string, param_va_list);
vsprintf(string, format_string, param_va_list);
vsnprintf(string, MAXLEN, format_string, param_va_list);
```

Вариадические функции в стандартной библиотеке

Пример динамического форматного вывода:
(из базы знаний IBM, lect-019-04.c)

```
#include <stdarg.h>
#include <stdio.h>

void vout(char *fmt, ...)
{
    va_list arg_ptr;

    va_start(arg_ptr, fmt);
    vprintf(fmt, arg_ptr);
    va_end(arg_ptr);
}

int main()
{
    char fmt1[]=" %s\t %d\n %s\t %d\n %s\t %d\n";

    vout(fmt1,"Mon",1,"Wed",3,"Fri",5);
    return 0;
}
```

Параметры функции `main()`

Программа, написанная на Си и откомпилированная, может запускаться не только из среды разработки, но и из командной строки (диалоговый режим).

При этом программе может передаваться несколько параметров (например, **`format F: FS:/NTFS /Q`**
или **`ls –al /home/user`**)

Для передачи исходных данных программе из операционной системы (при вызове программы) используются параметры функции **`main()`**.

Синтаксис:

```
int main(int argc, char **argv)
{
…
}
```

Параметры функции `main()`

`argv` (argument vector) — массив строк (указателей на массивы символов), разделенных пробелами. Последний (дополнительный) элемент массива — указатель на NULL.
Первый элемент массива (**`argv[0]`**) - [полное] имя исполняемого файла (программы).
Значение **`argc`** (argument count) считается автоматически (пока не нажата <ENTER>), указывать не надо.

Пример lect-019-05.c — для программы с рекурсивным формированием и выводом списка можем передать начальное значение n через командную строку и попросить вывести имя программы.

Вызов (Linux)
```
$ ./lect-019-05 5
```
Вывод:
```
…
Name of program: ./lect-19-05
Number of arguments: 2
…
```