Функции с переменным числом параметров

Функции с переменным числом параметров

```
int f(...);
```

- Во время компиляции компилятору не известны ни количество параметров, ни их типы.
- Во время компиляции компилятор не выполняет никаких проверок.

НО список параметров функции с переменным числом аргументов совсем пустым быть не может.

```
int f(int k, ...);
```

Функции с переменным числом параметров

Напишем функцию, вычисляющую среднее арифметическое своих аргументов.

Проблемы:

1. Как определить адрес параметров в стеке?
2. Как перебирать параметры?
3. Как закончить перебор?

Функции с переменным числом параметров

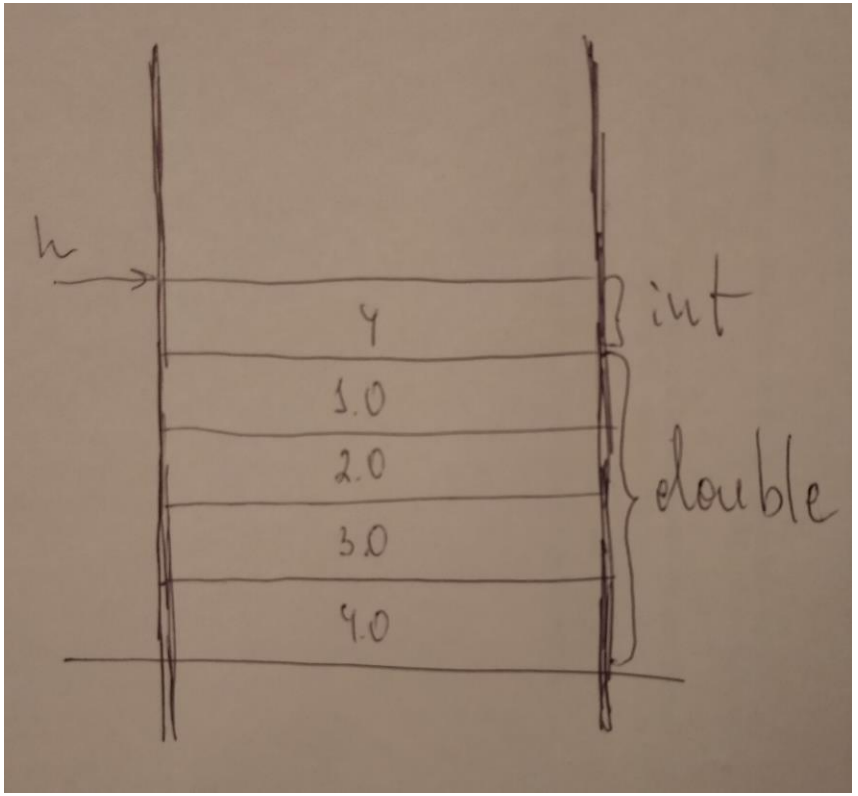
```
#include <stdio.h>

double avg(int n, ...)
{
...
}

int main(void)
{
    double a =
      avg(4, 1.0, 2.0, 3.0, 4.0);

    printf("a = %5.2f\n", a);

    return 0;
}
```



Функции с переменным числом параметров

```
#include <stdio.h>

double avg(int n, ...)
{
    int *p_i = &n;
    double *p_d =
                (double*) (p_i+1);
    double sum = 0.0;

    if (!n)
        return 0;

    for (int i = 0; i < n;
                        i++, p_d++)
        sum += *p_d;

    return sum / n;
}
```

```
int main(void)
{
    double a =
      avg(4, 1.0, 2.0, 3.0, 4.0);

    printf("a = %5.2f\n", a);

    return 0;
}
```

Функции с переменным числом параметров

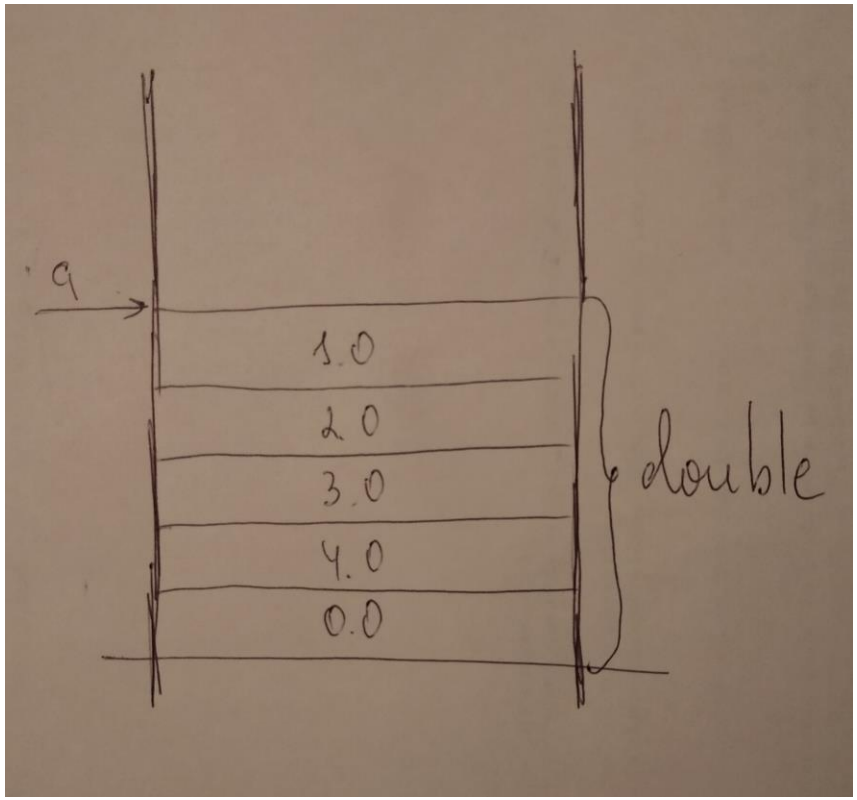
```
#include <stdio.h>

double avg(double a, ...)
{
...
}

int main(void)
{
    double a =
         avg(1.0, 2.0, 3.0,
                        4.0, 0.0);

    printf("a = %5.2f\n", a);

    return 0;
}
```



Функции с переменным числом параметров

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

#define EPS    1.0e-7

double avg(double a, ...)
{
    int n = 0;
    double *p_d = &a;
    double sum = 0.0;

    while (fabs(*p_d) > EPS)
    {
        sum += *p_d;
        n++;

        p_d++;
    }
    if (!n)
        return 0;

    return sum / n;
}

int main(void)
{
    double a =
        avg(1.0, 2.0, 3.0,
                    4.0, 0.0);

    printf("a = %5.2f\n", a);

    return 0;
}
```

Функции с переменным числом параметров

```
#include <stdio.h>

void print_ch(int n, ...)
{
    int *p_i = &n;
    char *p_c = (char*) (p_i+1);

    for (int i = 0; i < n; i++, p_c++)
        printf("%c %d\n", *p_c, (int) *p_c);
}

int main(void)
{
    print_ch(5, 'a', 'b', 'c', 'd', 'e');

    return 0;
}
```

Стандартный способ работы с параметрами функций с переменным числом параметров

stdarg.h

- va_list
- void va_start(va_list argptr, last_param)
- type va_arg(va_list argptr, type)
- void va_end(va_list argptr)

Функции с переменным числом параметров

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

double avg(int n, ...)
{
    va_list vl;
    double sum = 0, num;

    if (!n)
        return 0.0;

    va_start(vl, n);

    for (int i = 0; i < n; i++)
    {
        num = va_arg(vl, double);

        printf("%f\n", num);

        sum += num;
    }

    va_end(vl);

    return sum /n;
}

int main(void)
{
    double a =
      avg(4, 1.0, 2.0, 3.0, 4.0);

    printf("a = %5.2f\n", a);

    return 0;
}
```

Функции с переменным числом параметров

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

#define EPS      1.0e-7

double avg(double a, ...)
{
    va_list vl;
    int n = 0;
    double num, sum = 0.0;

    va_start(vl, a);
    num = a;

    while (fabs(num) > EPS)
    {
        sum += num;
        n++;
        num = va_arg(vl, double);
    }

    va_end(vl);
    if(!n)
        return 0;

    return sum / n;
}

int main(void)
{
    double a =
         avg(1.0, 2.0, 3.0,
                        4.0, 0.0);

    printf("a = %5.2f\n", a);

    return 0;
}
```

Функции с переменным числом параметров: журналирование

```
// log.c
#include <stdio.h>
#include <stdarg.h>

static FILE* flog;

int log_init(const char
                        *name)
{
    flog = fopen(name, "w");
    if(!flog)
        return 1;

    return 0;
}

void log_message(const char
                  *format, ...)
{
    va_list args;
    va_start(args, format);
    vfprintf(flog, format, args);
    va_end(args);
}

void log_close(void)
{
    fclose(flog);
}
```

```
// log.h

#ifndef __LOG__H__

#define __LOG__H__

#include <stdio.h>

int log_init(const char
                    *name);

void log_message(const char
                   *format, ...);

void log_close(void);

#endif  //  __LOG__H__
```