FT_PRINTF

MAKROLAR:

#define yapısı gibi, argüman alan define yapıları diyebiliriz.

VARIATIC FUNCTIONS:

***variatic functions**/varying number of arguments: printf'de fonksiyonun içine birden fazla değişken girebiliriz ve kullanıcının ne kadar girecegini bilemeyiz. varying demek tamamıyla kullanıcıya bağlı sayıda girdi alacagım olayı. örnek: printf("**%d%d**…")

→sınırsız parametre alan fonksiyonlarda … kullanırız (int topla(int miktar, …)

<stdarg.h> : va_start, va_arg, va_copy, va_end fonksiyonlarını içeren kütüphane

A function may be called with a ***varying number of arguments of varying types. The include file <stdarg.h> declares **a type (va_list)** and defines …skipping…

→ **void va_start(va_list arg, last):**

va list tipindeki argümanı oluşturur

arg: değişkenleri içeren argüman listesi

last: son parametrenin adı / hangi parametreden sonra sınırsız fonksiyonlar başlıyorsa onu yazıyoruz. örn: aşagıdaki fonksiyon için miktar değişkeni

return value: does not return anything.

→**type va_arg(va_list arg, type);**

arg: sonraki argümanın değerini alır

type: argümanların tipi ne ise o girilir.

→ void va_end(va_list ap);

va list'i kapatır. dosya işlemlerindeki gibi başlattıgımız va_listi bitirmemiz gerekir aksi halde hata çıkarabilir.

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

int topla(int miktar, ...)
{
  va_list ag;
  int i,toplam;

  va_start(ag, miktar);
  toplam = 0;
  for(i=0; i<miktar; i++)
    toplam += va_arg(ag, int);

  va_end(ag);
  return toplam;
}

int main()
{
  printf("%d\n", topla(2, 5, 5)); //2 argüman giricem onları topla diyor // 10
  printf("%d\n", topla(3, 5, 5)); // beklenenden az argüman verirsem //10
  printf("%d\n", topla(1, 5, 5)); // beklenenden çok argüman verirsem //5
}
```