|  |  |
| --- | --- |
| **0** | Значение добавляет нули (вместо пробелов).   * Если флаги “0”, и ” –“, то флаг 0 игнорируется. * Если преобразование указано точно (d, i, o, u, x и X), то флаг 0 игнорируется |
| **.** | Точность сопровождается необязательной строкой десятичных цифр,   * вместо строки десятичных цифр можно указать '\*', чтобы определить значение точности по следующему аргументу, которые должны иметь тип int. |
| **-** | Выравнивает результат преобразования по левой границе поля.   * Отменяет флаг 0, если оба флага были установлены. |
| **Digit/\*** | Ширина поля  Необязательная строка из десятичных цифр (с первой цифрой, отличной от нуля) определяет минимальную ширину поля.   * Если преобразованное значение имеет меньшее количество знаков, чем ширина поля, то оно слева дополняется пробелами (или справа, если указан флаг левого преобразования). * Вместо строки десятичных цифр можно указать '\*' |

|  |  |  |
| --- | --- | --- |
| **c** | unsigned char | |
| **s** | const char \* | |
| **p** | void \* | |
| **d**  **i** | Int  если результат можно показать с помощью меньшего количества цифр, то слева добавляются незначащие нули. По умолчанию значение точности равно единице. | |
| **u** | unsigned int беззнаковое десятичное | если результат можно показать с помощью меньшего количества цифр, то слева добавляются незначащие нули. По умолчанию значение точности равно единице. |
| **x** | unsigned int беззнаковое шестнадцатеричное |
| **X** | unsigned int беззнаковое шестнадцатеричное |
| **%** | Выводит символ `%' без преобразующих параметров. | |

if (c == 'c')

count **= ft\_treat\_char**(va\_arg(aps, int), flags);

else if (c == 's')

count **= ft\_treat\_string**(va\_arg(aps, char \*), flags);

else if (c == 'p')

count = **ft\_treat\_pointer**(va\_arg(aps, unsigned long long), flags);

else if (c == 'd' || c == 'i')

count = **ft\_treat\_int**(va\_arg(aps, int), flags);

else if (c == 'u')

count += **ft\_treat\_uint**((unsigned int)va\_arg(aps, unsigned int),

flags);

else if (c == 'x')

count += **ft\_treat\_hexa**(va\_arg(aps, unsigned int), 1, flags);

else if (c == 'X')

count += **ft\_treat\_hexa**(va\_arg(aps, unsigned int), 0, flags);

else if (c == '%')

count += **ft\_treat\_percent**(flags);

ft\_putstrprec

ft\_treat\_width

flags.dot – точность - precision

flags.minus - минус;

flags.star – кол-во звездочек - ;

flags.type = 0;

flags.width = 0;

flags.zero = 0;

#include "include/ft\_printf.h"

str

int ft\_printf(const char \*apFormat, ...);

int ft\_treat\_save(const char \*save, va\_list aps);

t\_flags ft\_init\_flags();

int ft\_flag\_parse(const char \*save, int i, t\_flags \*flags, va\_list aps);

int ft\_is\_in\_type\_list(int c);

int ft\_is\_in\_flags\_list(int c);

int ft\_flag\_precision(const char \*save, int start, t\_flags \*flags, va\_list aps);

t\_flags ft\_flag\_minus(t\_flags flags);

t\_flags ft\_flag\_width(va\_list aps, t\_flags flags);

t\_flags ft\_flag\_digit(char c, t\_flags flags);

int ft\_treatment(int c, t\_flags flags, va\_list aps);

int ft\_treat\_char(char c, t\_flags flags);

int ft\_treat\_width(int width, int minus, int has\_zero);

int ft\_treat\_string(char \*str, t\_flags flags);

int ft\_put\_part\_str(char \*str, t\_flags flags);

int ft\_putstrprec(char \*str, int prec);

int ft\_treat\_pointer(unsigned long ull, t\_flags flags);

char \*ft\_ull\_base(unsigned ull, int base);

char \*treat\_base(unsigned ull\_save, int base, char \*rtn, int count);

char \*ft\_str\_tolower(char \*str);

int ft\_in\_put\_part\_pointer(char \*pointer, t\_flags flags);

int ft\_treat\_int(int i, t\_flags flags);

int ft\_put\_part\_int(char \*d\_i, int save\_i, t\_flags flags);

int ft\_in\_put\_part\_int(char \*d\_i, int save\_i, t\_flags flags);

int ft\_treat\_uint(unsigned int unsi, t\_flags flags);

char \*ft\_u\_itoa(unsigned int n);

int ft\_estim(long n);

char \*ft\_gen(char \*rtn, long nbr, int len, int isneg);

int ft\_put\_part\_uint(char \*unsi\_int, t\_flags flags);

int ft\_in\_put\_part\_uint(char \*unsi\_int, t\_flags flags);

int ft\_treat\_hexa(unsigned ui, int lower, t\_flags flags);

int ft\_put\_part\_hexa(char \*hexa, t\_flags flags);

int ft\_in\_put\_part\_hexa(char \*hexa, t\_flags flags);

int ft\_treat\_percent(t\_flags flags);

int ft\_putchar(int c);

char \*ft\_itoa(int i);