**ГЛАВА 11**

**1.**

Без нулевого символа в конце это не строка, а массив символов.

char name[] = { ‘Л’, ‘у’, ‘н’, ‘a’, ‘\0’ };

**2.**

Увидимся завтра в кафе.

видимся завтра в кафе.

Увидимс

идимс

**3.**

о

но

сно

усно

кусно

Вкусно

**4.**

За всю дорогу я смог осилить лишь часть.

**5.**

а)

Хо Хо Хо! !оХ оХ оХ

б)

переменная х должна иметь тип указатиля на char.

в)

переменная х получает адрес первого символа строки «Хо Хо Хо! »

г)

выражение \*--pc означает уменьшить адрес указателя на единицу типа данных и разыменовать указатель.

а выражение --\*pc означает уменьшить значение, на которое указывает указатель на один.

д)

так как в этот момент указатель pc указывает на значение нулевого символа, то при замене \*--pc на \*pc—вывод putchar('\0') ничего не покажет, только потом сработает операция декремента.

е)

В операторе while (pc – str) выполняют роль счетчика. Так как, адрес pc больше на 9 от адреса str, то при декременте указателя pc 9 раз он будет указывать на тот же адрес, что и указатель str, а это означает, что операция pc – str в результате дает 0 и цикл прекращается.

ж)

при передаче функции пустой строки в цикл while программа не зайдет, так как, указатель \*pc будет возвращать нулевой символ.

но программа зайдет в цикл do while и попытается вывести символ из ячейка пямати, которая может быть занята другой программой и может произойти сбой.

з)

**6.**

char sign = ‘$’;

переменная sign занимает 1 байт, значение '$' занимает 1 байт и значение «$» занимает 2 байта, так как, это строка и к ней добавляется нулевой символ.

**7.**

How are ya, sweetie? How are ya, sweetie?

Beat the clock.

eat the clock.

Beat the clock. Win a toy.

Beat

chat

hat

at

t

t

at

How are ya, sweetie?

**8.**

faavrhee

\*le\*on\*sm

**9.**

char \* s\_gets(char \* st, int n)

{

char \* ret\_val;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

while (\*st != '\n' && \*st != '\0')

st++;

if (\*st == '\n')

\*st = '\0';

else

while (getchar() != '\n')

continue;

}

return ret\_val;

}

**10.**

int strlen(const char \* string)

{

int count = 0;

while (\*string++)

count++;

return count;

}

**11.**

char \* s\_gets(char \* st, int n)

{

char \* ret\_val;

char \* temp;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

temp = strchr(st, '\n');

if (temp)

\*temp = '\0';

else

while (getchar() != '\n')

continue;

}

return ret\_val;

}

**12.**

char \* find\_space(char \* string)

{

char \* space\_pt;

while (\*string != ' ' && \*string != '\0')

string++;

space\_pt = string;

if (\*string == '\0')

space\_pt = string;

return space\_pt;

}

**13.**

#include <stdio.h>

#include <string.h>

#include <ctype.h>

#define ANSWER "Grant"

#define SIZE 40

char \* s\_gets (char \* st, int n);

void toUpper(char \* string);

int main (void)

{

char try[SIZE];

char answer[] = ANSWER;

toUpper(answer);

puts("Кто похоронен в могиле Гранта?");

s\_gets(try, SIZE);

toUpper(try);

while (strcmp(try, answer) != 0)

{

puts("Неправильно! Попытайтесь еще раз.");

s\_gets(try, SIZE);

toUpper(try);

}

puts("Tenepь правильно!");

return 0;

}

void toUpper(char \* string)

{

for (int i = 0; i < strlen(string); i++)

string[i] = toupper(string[i]);

}

char \* s\_gets(char \* st, int n)

{

char \* ret\_val;

int i = 0;

ret\_val = fgets(st, n, stdin);

if (ret\_val)

{

while (st[i] != '\n' && st[i] != '\0')

i++;

if (st[i] == '\n')

st[i] = '\0';

else

while (getchar() != '\n')

continue;

}

return ret\_val;

}