### Вопросы без ответов

Определить способ размещения последовательности переменных в общей области памяти, которая читается или заполняется функцией (формат структуры данных).

//------------------------------------------------------ 1

// Объявляется структура. в функцию передаются три символьных указателя. Далее в фукнции новому структурному указателю присваивается значение указателя p c явным приведением типа, в поле структуры name копируются данные из указателя nm, далее в байт с типом char\* сразу за границей нового структурного указателя копируется значение из указателя ad. Затем полю структуры addr присваивается то значение, что было скопировано на предыдущем шаге. Функция возвращает указатель на память сразу за окончанием значений, которые хранятся в addr.

struct man {char name[20]; int dd,mm,yy; char \*addr; };

char \*F1(char \*p, char \*nm, char \*ad)

{ man \*q =(man\*)p;

strcpy(q->name,nm);

strcpy((char\*)(q+1),ad);

q->addr = (char\*)(q+1);

for (p=(char\*)(q+1); \*p!=0; p++);

p++;

return p;}

//------------------------------------------------------ 2

// Объявляется структура. в функцию передаются три символьных указателя. Далее в фукнции новому структурному указателю присваивается значение указателя p c явным приведением типа, в поле структуры name копируются данные из указателя nm, а в поле структуры addr копируются данные из указателя ad. Функция возвращает указатель на память сразу за окончанием значений, которые хранятся в addr.

struct man1 {char name[20]; int dd,mm,yy; char addr[]; };

char \*F2(char \*p, char \*nm, char \*ad)

{ man1 \*q =(man1\*)p;

strcpy(q->name,nm);

strcpy(q->addr,ad);

for (p=q->addr; \*p!=0; p++);

p++;

return p;}

//------------------------------------------------------ 3

//Первый цикл подсчитывает количество строк в массиве указателей на символы.

В разыменованный целочисленный указатель сохраняем полученное значение.

Далее указателю на символы присваивается адрес памяти сразу за целочисленным указателем. внешний цикл пробегает по массиву символов, пока он не пустой. Внутренний цикл пробегает по строкам массива символов и сохраняет значение этих строк посимвольно в указатель на символы.

Функция возвращает указатель на память сразу за последней строкой. указатель явно приводится к типу инт\*.

int \*F3(int \*q, char \*p[])

{ int i,j;

char \*s;

for (i=0; p[i]!=NULL; i++);

\*q = i;

for (s = (char\*)(q+1), i=0; p[i]!=NULL; i++)

{

for (j=0; p[i][j]!='\0'; j++) \*s++ = p[i][j];

\*s++ = '\0';

}

return (int\*)s;}

//------------------------------------------------------- 4

//функция получает массив целочисленных значений. Нулевой элемент массива хранит число, которое используется для проверки условия цикла. далее в указатель на дабл сохраняется адрес 1-го элемента массива приведенного к типу дабл. На каждом шаге цикла к сумме прибавляется текущее значение по указателю. Функция возвращает сумму.

double F4(int \*p)

{ double \*q,s; int m;

for (q=(double\*)(p+1), m=\*p, s=0.; m>=0; m--) s+= \*q++;

return s;}

//------------------------------------------------------- 5

//Внешний цикл подсчитывает количество строк в массиве указателей на символы. Внутренний цикл копирует посимвольно все строки из массива символов в указатель на символы. После копирования в указатель сохраняется дополнительный символ конца строки и функция возвращает адрес получившегося указателя.

char \*F5(char \*s, char \*p[])

{ int i,j;

for (i=0; p[i]!=NULL; i++)

{

for (j=0; p[i][j]!='\0'; j++) \*s++ = p[i][j];

\*s++ = '\0';

}

\*s = '\0';

return s;}

//------------------------------------------------------- 6

//интовому полю объединения присваиваем указатель, переданный в функцию. далее, пока значение по указателю не равно 0, в случае если значение в поле pi равно 1,2,3 прибавляем к сумме значение, находящееся следом за pi, преобразованное к определенному типу, в зависимости от case и двигаем указатель на +1 того типа, который прибавляли.

Функция возвращает сумму типа дабл.

union x {int \*pi; long \*pl; double \*pd;};

double F6(int \*p)

{ union x ptr;

double dd=0;

for (ptr.pi=p; \*ptr.pi !=0; )

switch (\*ptr.pi++)

{

case 1: dd += \*ptr.pi++; break;

case 2: dd += \*ptr.pl++; break;

case 3: dd += \*ptr.pd++; break;

}

return dd;}

//------------------------------------------------------- 7

//сначала в указатель сохраняется количество символов в строке с явным приведением типа, а затем копируется вся строка без символа конца строки.

Функция возвращает указатель на следующую ячейку памяти после последнего символа.

unsigned char \*F7(unsigned char \*s, char \*p)

{ int n;

for (n=0; p[n] != '\0'; n++);

\*((int\*)s)++ = n;

for (; \*p != '\0'; \*s++ = \*p++);

s++;

return s;}

//------------------------------------------------------- 8

//сначала в указатель копируется число (скорее всего обозначающее, сколько элементов массива дабл надо обработать). затем в цикле копируется значение дабл в указатель с явным приведением типа. Возвращается указатель на целое, указывающий на ячейку памяти, находящуюся сразу после последнего скопированного дабл числа.

int \*F8(int \*p, int n, double v[])

{ \*p++ = n;

for (int i=0; i<n; i++) \*((double\*)p)++ = v[i];

return p;}

//------------------------------------------------------- 9

//Если целочисленное число больше 0, то к сумме дабл прибавляем значение по текущему указателю и сдвигаем указатель на 1. Если число меньше ноля, то сначала двигаем указатель на 1, а потом к сумме дабл прибавляем значение указателя приведенного тоже к типу дабл. После этого двигаем указатель на одну величину дабл. Функция возвращает сумму значений.

double F9(int \*p)

{ double s=0;

while(\*p!=0)

{

if (\*p>0) s+=\*p++;

else

{ p++; s += \*((double\*)p)++; }

}

return s; }

//------------------------------------------------------ 10

//сначала сдвигаем указатель q на длину переданной строки. а затем, в зависимости от того, какое значение лежит в указателе p, преобразуем указатель q в один из трех типов и берём значение из него и добавляем к сумме, после этого сдвигаем указатель и идем на следующий шаг цикла.

Функция возвращает сумму значений.

double F10(char \*p)

{ double s;

char \*q;

for (q=p; \*q!=0; q++);

for (q++; \*p!=0; p++)

switch(\*p)

{

case 'd': s+=\*((int\*)q)++; break;

case 'f': s+=\*((double\*)q)++; break;

case 'l': s+=\*((long\*)q)++; break;

}

return s; }

//-------------------------------------------------------11

//сдвигаем указатель q на длину строки без символа конца строки. Сдвигаем q еще на 1. Далее указателю v присваиваем указатель q приведенный к типу инт.

Далее в цикле проходим по строке p и если значение по текущему указателю является цифрой, то в сумму s добавляем значение, которое лежит по индексу указателя v равному цифре из указателя p.

Функция возвращает сумму.

int F11(char \*p)

{ int s=0, \*v;

char \*q;

for (q=p; \*q!=0; q++);

q++; v=(int\*)q;

for(;\*p!=0;p++)

if (\*p>='0' && \*p<='9') s+=v[\*p-'0'];

return s;

}