Глава 2. **Адресная арифметика. «Структурные» типы данных**

МГТУ им. Н.Э. Баумана
Факультет Информатика и системы управления
Кафедра Компьютерные системы и сети
Лектор: д.т.н., проф.
Иванова Галина Сергеевна

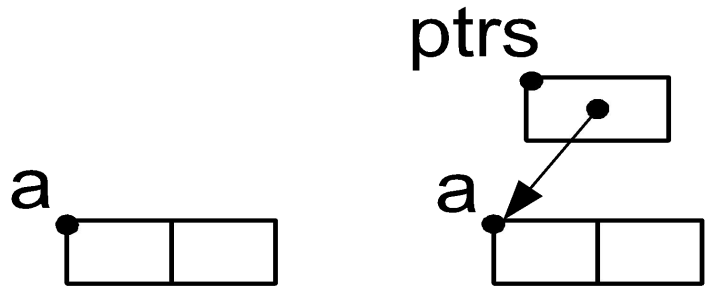
2.1 Указатели

Формат:

[<Изменяемость значения>][<Тип данных>][<Тип>]
[<Изменяемость указателя>]*<Имя>[=<Значение>];

Пример:

1) `short a,*ptrs =&a;`



2) `const short *ptrs;`

Неизменяемое значение:
можно **ptrs = &b;** нельзя ***ptrs=10;**

3) `short *const ptrs=&a;`

Неизменяемый указатель
можно ***ptrs=10;** нельзя **ptrs = &b;**

Операции над указателями

1. Присваивание

Примеры:

```
int a,*ptri,*ptrj; void *b;
1) ptri=&a;
2) ptri=nullptr; // C++11
3) ptri=ptrj;
4) b=&a;
5) ptri=b; ⇒ ptri=(int *) b;
```

Явное переопределение типа указателя (C-style)

2. Разыменование

Примеры:

```
int c, a=5,*ptri=&a;
void *b=&a;
1)  c=*ptri;
2)  *ptri=125;
3)  *b=6; ⇒
 *static_cast<int*>(b)=6;
```

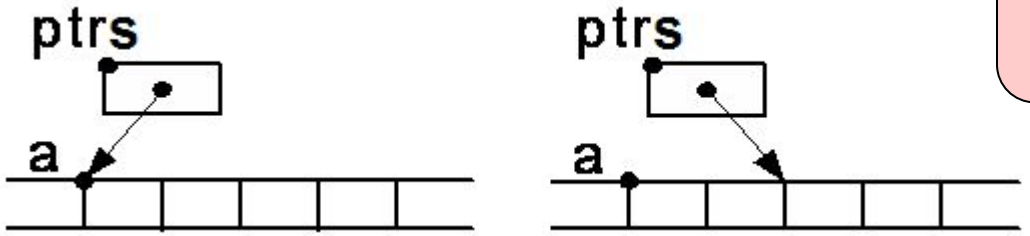
Явное переопределение типа указателя (C++-style)

Основное правило адресной арифметики

<Указатель> + n ⇔ <Адрес> + n*sizeof(<Тип данных>)

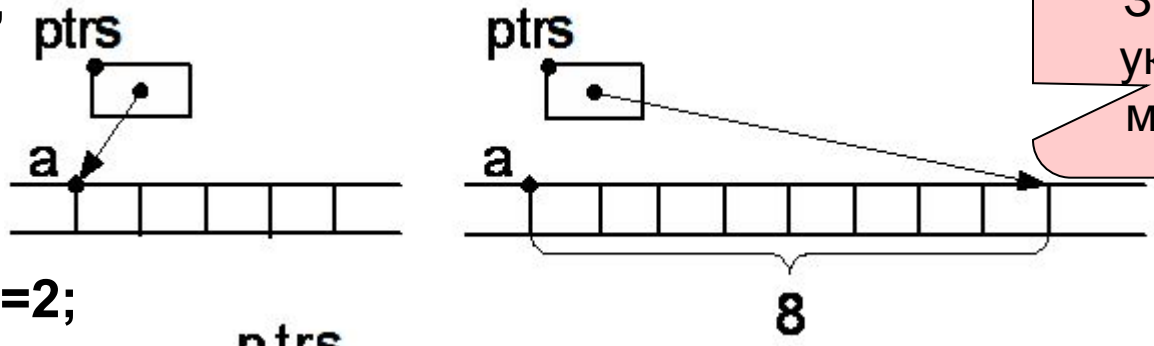
Пример: **short a, *ptrs =&a;**

1) **ptrs++;**



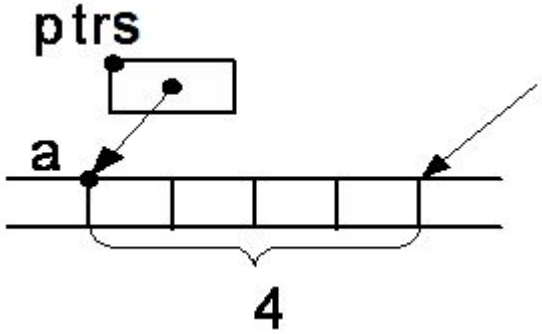
Значение указателя меняется

2) **ptrs+=4;**



Значение указателя меняется

3) ***(ptrs+2)=2;**

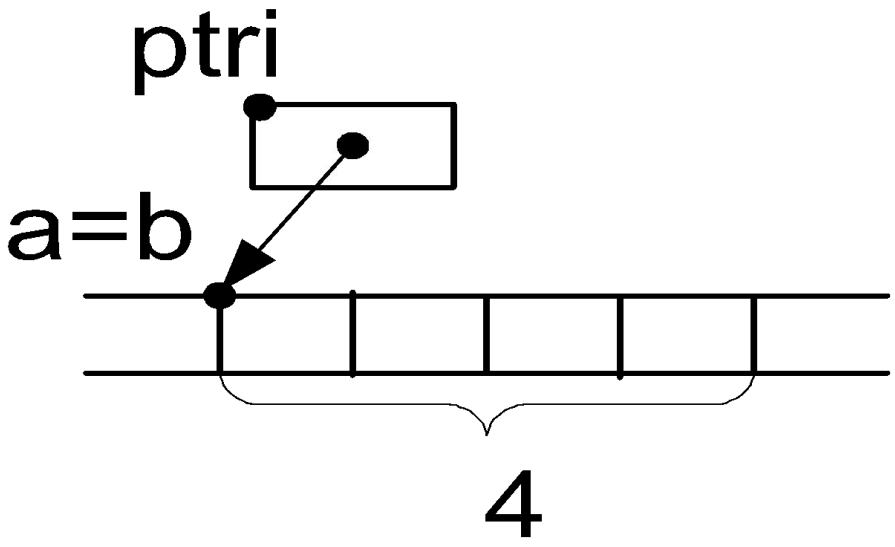


Значение указателя не меняется!!!

Ссылки

```
int a,     // переменная
    *ptri=&a, // указатель
    &b=a;     // ссылка
…
```

a=3; ⇔ *ptri=3; ⇔ b=3;



Итак, ссылка тоже физически представляет собой адрес, но в отличие от указателя при работе со ссылками не используется операция разыменования.

2.2 Управление динамической памятью (C-style)

1. Размещение в памяти одного значения

Выделение памяти

void * malloc(size_t size);

Освобождение памяти

void free(void *block);

Пример:

```
int *a;
if ((a = (int *) malloc(sizeof(int))) == NULL){
                printf("Not enough memory.");
                exit(1);
            }
*a=-244;
free(a);
```

Управление динамической памятью (C-style)

2. Размещение нескольких значений

Выделение памяти

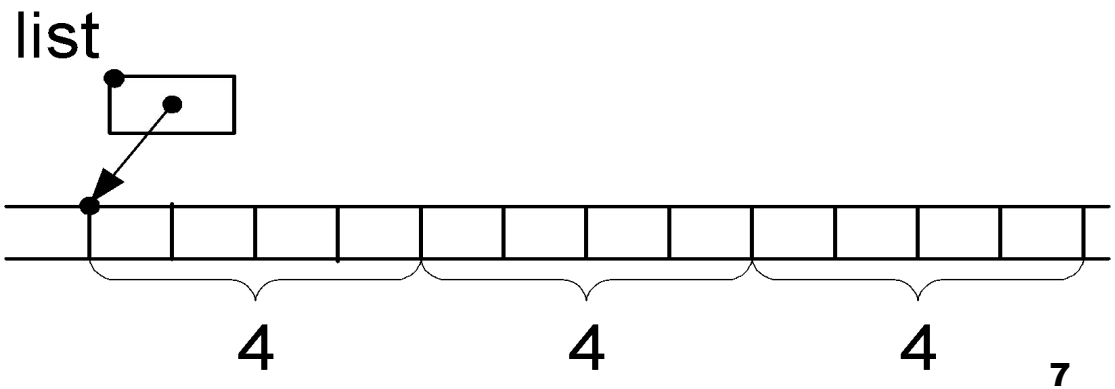
void * calloc(size_t n, size_t size);

Освобождение памяти

void free(void *block);

Пример:

```
int *list;
list = (int *) calloc(3,sizeof(int));
*list=-244;
*(list+1)=15;
*(list+2)=-45;
   …
free(list);
```



Управление динамической памятью (C++-style)

1. Одно значение

Операция выделения памяти

<Указатель> =new<Имя типа>[(<Значение>)];

Операция освобождения памяти

delete <Указатель>;

Примеры:

а)
```
int *k;
  k = new int;
  *k = 85;
```

б)
```
int *a;
  if ((a = new int(-244)) == NULL){
    printf("Not enough memory.");
    exit(1);  }
  delete a;
```

Управление динамической памятью (C++style) (2)

2. Несколько значений

Операция выделения памяти для n значений:

<Указатель> =new<Имя типа>[<Количество>];

Операция освобождения памяти:

delete [ ] <Типизированный указатель>;

Пример:

```
int *list;
list = new int [3];
*list=-244;
*(list+1)=15;
*(list+2)=-45;
delete[ ] list;
```

2.3 Массивы

Объявление массива:

<Тип элемента> <Имя>[<Размер1>] [<Размер2>] ...[= {<Список значений >}];

Примеры:

1) **int a[4][5];**

2) **short x[3][4] ={{9,6,-56,0}, {10,3,-4,78}, {-6,8,45,7}};**

Примечания:

1) индексы массива всегда начинаются с 0;

2) многомерные массивы в памяти расположены построчно;

3) для адресации элементов массива независимо от способа описания можно использовать адресную арифметику:

```
(list+i)  ⇔ &(list[i])
*(list+i) ⇔ list[i]
```

Пример программы обработки массива (Ex2_01)

```
#include <stdio.h>
#define N 5
int main()
{
   int a[N][N], i, j, s[N];
   for (i = 0; i < N; i++)
   {
    printf("Input numbers of %2d string:\n", i);
    for (j = 0; j < N; j++) scanf("%d", &a[i][j]);
   }
   for (i = 0; i < N; i++)
    for (j = 0, s[i] = 0; j < N; j++) s[i] += a[i][j];
   for (i = 0; i < N; i++)
   {
    for (j = 0; j < N; j++) printf("%3d ", a[i][j]);
    printf("sum=%4d\n", s[i]);
   }
```

```
#define N 5 //C-style.
```
В C++ принято :
```
constexpr int N{5};
```

Пример программы обработки массива (Ex2_01)

```
#include <stdio.h>
#define N 5
int main(int argc, char* argv[])
{
  int a[N][N], i, j, s[N];
  for (i = 0; i < N; i++)
  {
   printf("Input numbers of %2d string:\n", i);
   for (j = 0; j < N; j++) scanf_s("%d", &a[i][j]);
  }
  for (i = 0; i < N; i++)
   for (j = 0, s[i] = 0; j < N; j++) s[i] += a[i][j];
  for (i = 0; i < N; i++)
  {
   for (j = 0; j < N; j++) printf("%3d ", a[i][j]);
   printf("sum=%4d\n", s[i]);
  }
```

Цикл foreach или цикл по коллекции (Ex2_07)

Цикл предложен для стандартных шаблонов коллекций, однако может использоваться в том числе и для массивов:

```
#include <iostream>
   using namespace std;
int main() {
   int arr[5] = {1, 2, 3, 4, 5};
   for(int &x : arr) {
      x*= 2;
    }
   for(auto x:arr)
     cout<<x<<' ';
   for(const auto &x:arr)
     cout<<x<<' ';
   return 0;
}
```

Размер массива должен быть известен!!!

Можно auto, тогда тип определится автоматически

Ссылка (&) - для изменения чисел в массиве

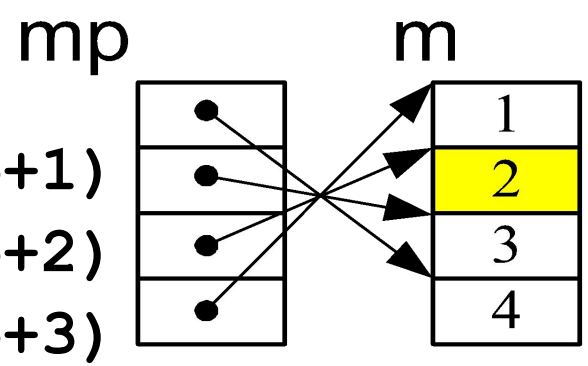
Значение – работа с копиями чисел массива

Константа – работа со значениями с запретом изменений

Многоуровневые ссылки (Ex2_0???)

(list+i) ⇔ &(list[i])
*(list+i) ⇔ list[i]

```
int m[]={1,2,3,4};
int *mp[]={m+3,m+2,m+1,m};
```

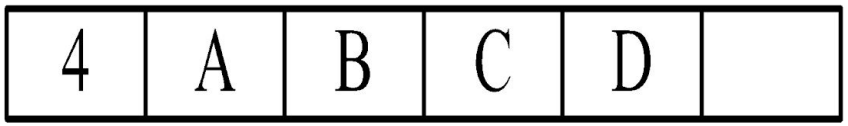



```
mp[0],*mp
mp[1],*(mp+1)
mp[2],*(mp+2)
mp[3],*(mp+3)
```

```
m[1],*(m+1)
```
или
```
mp[0][-2],
*(mp[0]-2),
*(*mp-2),
mp[1][-1],
*(mp[1]-1),
*(*(mp+1)-1)
```

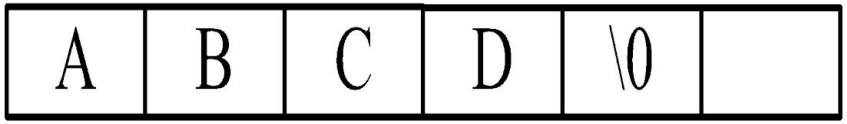
2.4 Строки

Строка Паскаля



Длина

Строка С (С++)



Признак конца

Строка в С и С++– последовательность символов, завершающаяся нулем.

Примечание. Цикл **`for`** по коллекции для строк лучше не использовать, поскольку он не видит завершающего нуля…

Объявление строки

Объявление строки:

`char` <Имя>[<Объем памяти>] [= <Значение>];

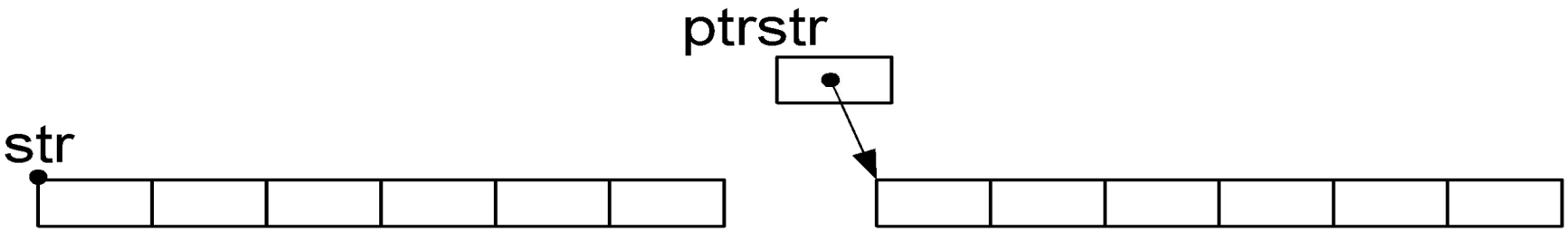
Объявление указателя на строку:

`char` *<Имя указателя>[= <Значение>];

Примеры:

а) `char str[6];`

б)
```
char *ptrstr;
ptrstr = new char[6];
delete [] ptrstr;
```



в) `char str1[5] = {'A','B','C','D','\0'};`

г) `char str2[5] = "ABCD";`

д) `char str3[] = "ABCD";`

е) `const char *str4 = "ABCD"; // важно!`

Объявление и инициализация массивов строк

Массив указателей на строки

char * <Имя>[<Размер 1>] [= <Значение>];

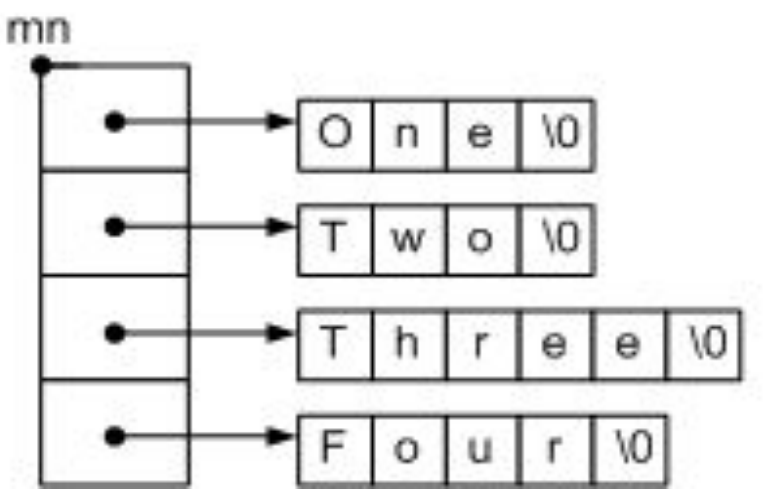
Массив строк указанной длины

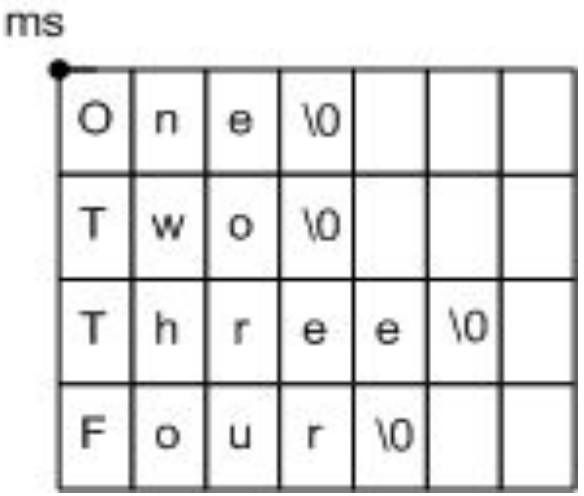
char <Имя>[<Размер 1>][Размер 2] [= <Значение>];

Примеры:

а) `const char * mn[4] = {"One","Two","Three","Four"};`

б) `char ms[4][7] ={"One","Two","Three","Four"};`





Функции, работающие со строками

Библиотеки: **string.h**, **stdlib.h**

1) определение длины строки: **size_t strlen(char *s);**
2) конкатенация строк:
   char *strcat(char *dest,const char *src);
3) сравнение строк:
   int strcmp(const char *s1,const char *s2);
4) копирование строки **src** в **dest**:
   char *strcpy(char *dest,const char *src);
5) копирование фрагмента **dest** строки **src**:
   char *strncpy(char *dest,const char *src,size_t maxlen);
6) поиск символа c в строке s:
   char *strchr(const char *s, int c);
7) поиск подстроки s2 в строке s1:
   char *strstr(const char *s1, const char *s2);
8) поиск токенов в строке:
   char *strtok_s(char *strToken,
   const char *strDelimit, char **nexttoken);

Функции, работающие со строками (2)

9) преобразование строки в целое число:

```
int atoi(const char *s);
```

10) преобразование строки в вещественное число:

```
double atof(const char *s);
```

11) преобразование числа в строку:

```
char *itoa(int value,char *s,int radix);
```

12) преобразование числа в строку:

```
char *_gcvt( double value, int digits,
                        char *buffer );
```

13) преобразование числа в строку:

```
char *_ecvt_s(buf, bufsize, double value,
                      int count, int *dec, int *sign);
```

buf – адрес буфера, bufsize – размер буфера,

count - количество преобразуемых цифр,

dec, sign – позиции точки и знак

Пример преобразования числа в строку (Ex2_02)

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

int main()
{
   char buf[12];      // буфер
   int decimal,sign; // позиция десятичной точки и знак
   int count=10;      // количество преобразуемых разрядов
   int err;           // код ошибки

   err = _ecvt_s(buf,12,3.1415926535,count,&decimal,&sign);
   printf("Converted value to string: %s\n", buf);
   printf("Decimal= %d, Sign= %d.", decimal, sign);

   return 0;
}
```

```
Converted value to string: 3141592654
Decimal= 1, Sign= 0.
```

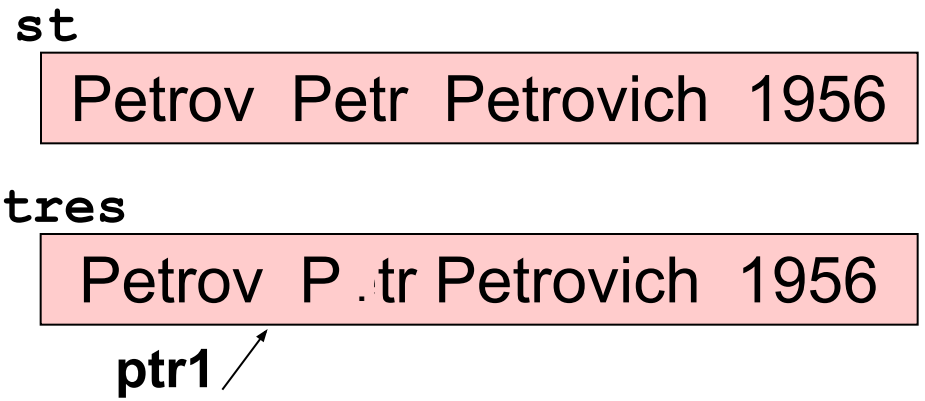
Пример использования функции strtok_s (Ex2_03)

```
#include <string.h>
#include <stdio.h>
int main()

char str[] = "A string\tof ,,tokens\nand some
                                        more tokens";
char seps[] = " ,\t\n", *token, *nexttoken;

{
   token = strtok_s(str, seps, &nexttoken);
   while (token != nullptr)
   {
    printf("%s ", token);
    token =
   }
   return 0;
}
```

```
A string of tokens and some more tokens
```

Пример использования функций обработки строк

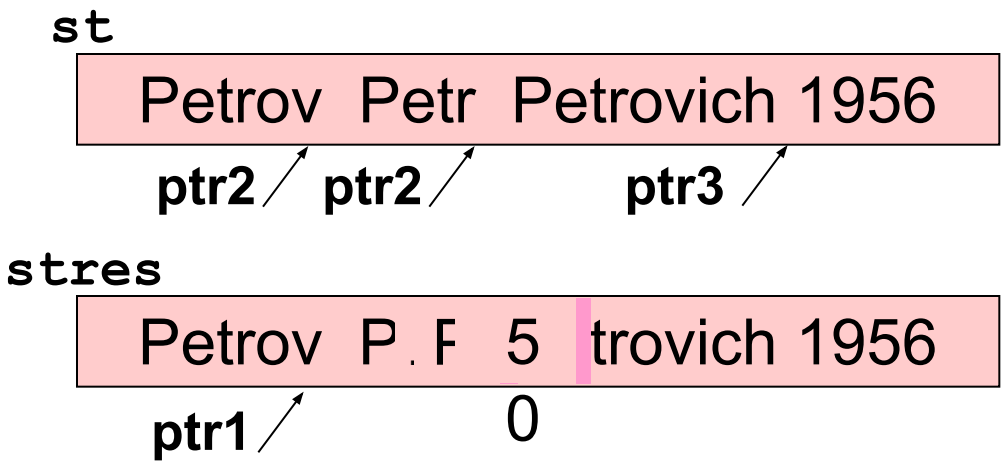
Petrov Petr Petrovich 1956 => Petrov P.P. 50 (**Ex2_04**)




```
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

int main()
{  char st[80],stres[80],strab[80],
        *ptr1,*ptr2,*ptr3;
   int old;
   while ((puts("Enter string or end:"),
           strcmp(gets(st),"end")!=0) {
     strcpy(stres,st);
     ptr1=strchr(stres,' ');
     *(ptr1+2)='.';
```

Пример использования функций обработки строк (2)

```
    ptr2=strchr(st,' ');
    ptr2=strchr(ptr2+1,' ');
    strncpy(ptr1+3,ptr2+1,1);
    strncpy(ptr1+4,". \0",3);
    ptr3=strchr(ptr2+1,' ');
    old=2006-atoi(ptr3+1);
    strcat(stres,itoa(old,strab,10));
    puts(stres);  }
    return 0;
}
```



2.5 Структуры

1. Объявление (C-style)

`struct` [<Имя структуры>] {<Описание полей>}
[<Список переменных [и значений>]];

Примеры:

а)
```
struct student { char name[22];  char family[22];  int old;};
  struct student stud1={"Petr","Petrov",19},stud[10], *ptrstud;
```

б)
```
struct {char name[22];char family[22];int old;}
                                    stud1, stud[10], *ptrstud;
```

2. Объявление (C++-style)

typedef struct {<Описание полей>} <Имя структуры>;

<Имя структуры> <Список переменных [и значений]>;

Пример:

```
typedef struct
            {char name[22];char family[22];int old;} student;
struct student stud1={"Petr","Petrov",19}, stud[10],*ptrstud;
```

Имя переменной типа «структура» не является ее адресом !

Обращение к полям структуры

<Имя переменной>.<Имя поля>

<Имя массива>[<Индекс>].<Имя поля>

(*<Имя указателя>).<Имя поля> или

<Имя указателя> -> <Имя поля>

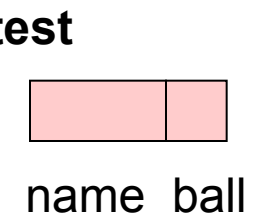
Примеры:

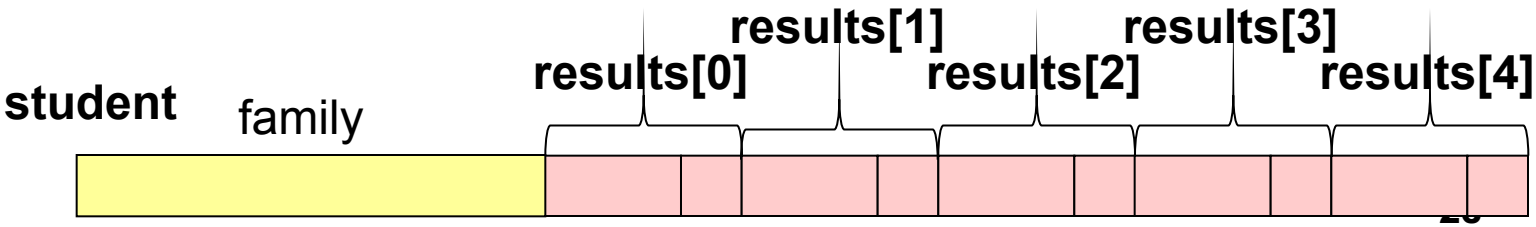
```
stud1.name
stud[i].name
(*ptrstud).name    ⇔ ptrstud -> name
```

Пример использования структуры (Ex2_05)

Программа определения среднего балла каждого студента и группы в целом

```
#include <stdio.h>
#include <string.h>
typedef struct {
      char name[10];
       int ball;
     } test;
typedef struct {
       char family[22];
       test results[5];
     }student;
```





Пример использования структуры (2)

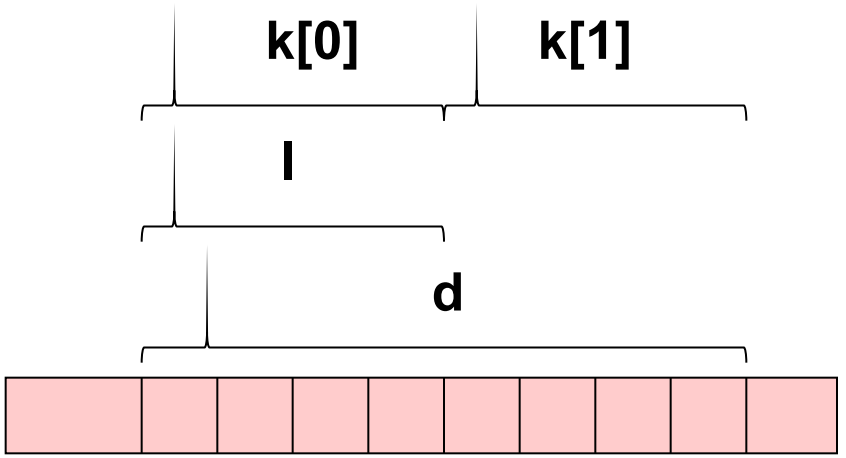
```
int main()
{student stud[10]; int i,n=0; float avarstud,avarage=0;
 while (puts("Input names, subjects and marks or end"),
        scanf("\n%s",stud[n].family),
        strcmp(stud[n].family,"end")!=0) {
     for (avarstud=0,i=0; i<3; i++)
       { scanf("\n%s %d",stud[n].results[i].name,
                        &stud[n].results[i].ball);
         avarstud+=stud[n].results[i].ball;}
     printf("Average:%s=%5.2f\n",
                                stud[n].family,avarstud/3);
     avarage+=avarstud;
     n++; }
 printf("Group average mark=%5.2f\n",avarage/n/3);
 return 0;
}
```

2.6 Объединения

union <Имя объединения> {
<Список элементов объединения>}
[<Список переменных [и значений]>];

Пример:

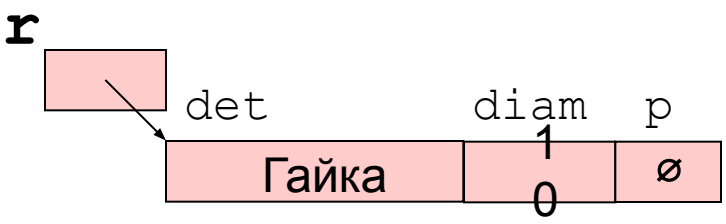
```
union mem
{
  double d;
  long l;
  int k[2];
};
```

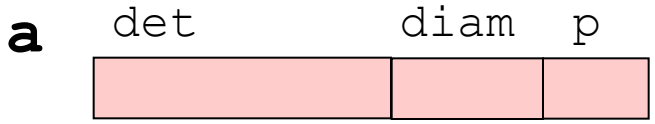


2.7 Динамические структуры данных (Ex2_06)

Пример. Стек записей.

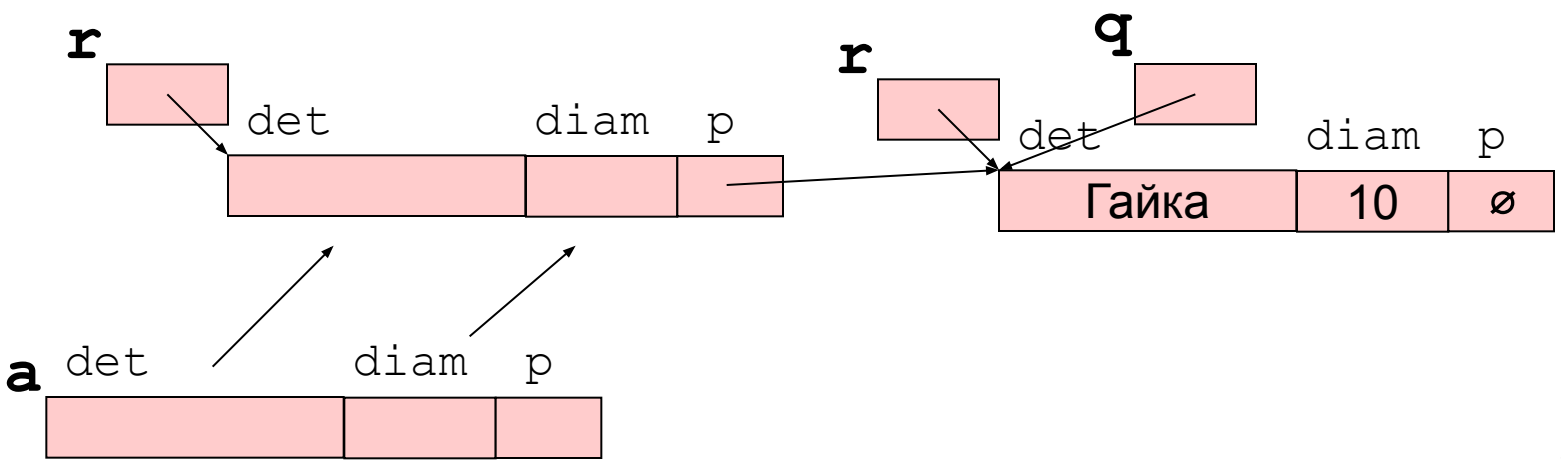
```
#include <stdio.h>
#include <string.h>
struct zap  {char det[10];  float diam; zap *p; };

int main()
{     zap a,*r,*q,*f;
      r=new zap;
      r->p=nullptr;
      puts("Input strings");
      scanf("%s %f\n",r->det,&r->diam);
```





Динамические структуры данных (2)

```
while((scanf("\n%s",a.det)),strcmp(a.det,"end")!=0)
        { scanf("%f",&a.diam);
           q=r;
           r=new zap;
           strcpy(r->det,a.det);
           r->diam=a.diam;
           r->p=q;
        }
```

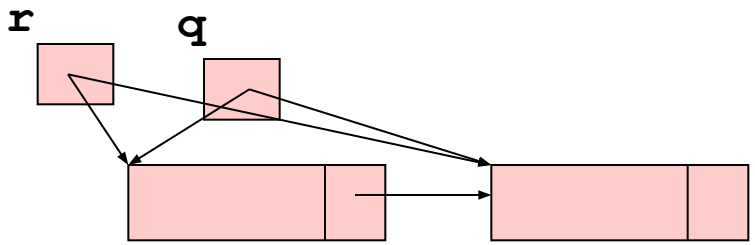


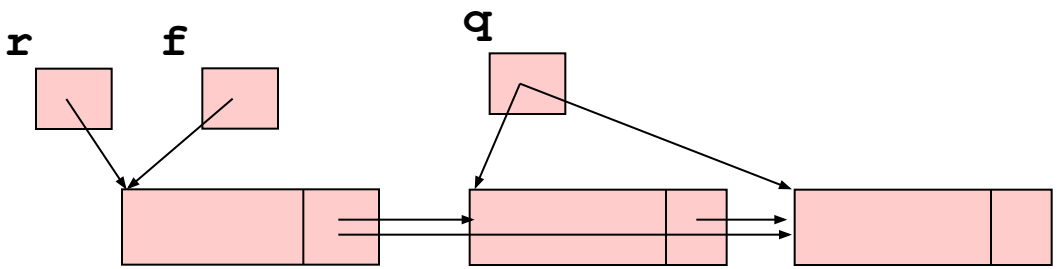
Динамические структуры данных (3)

```
q=r;
puts("List");
if(q==nullptr) puts("No information");
else
    do {
          printf("%s %5.1f\n",q->det,q->diam);
          q=q->p;
       }
    while (q!=nullptr);
```

Динамические структуры данных (4)

```
q=r;
do
 { if(q->diam<1)
    { if(q==r) {
       r=r->p;
       delete q;
       q=r;}
      else {
    q=q->p;
    delete f->p;
    f->p=q;}
    }
   else {
    f=q;
    q=q->p;}
 }
while(q!=nullptr);
```





Динамические структуры данных (5)

```
    q=r;
    puts("Result");
    if(q==nullptr) puts("No information");
    else
      do {
           printf("%s %5.1f\n",q->det,q->diam);
           q=q->p;
         }
      while (q!=nullptr);
    return 0;
}
```