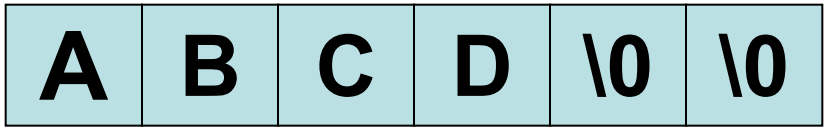
6 Строки и структуры C++

6.1 Строки C++

Строение строковых данных в С (С++)



Признак конца строки

Значение индексов элементов

В С++ символьная строка определяется как последовательность символов, которая заканчивается нуль-символом (символ «\0»).

Обращение к элементам строки осуществляется по индексу

- прямо – по номеру элемента
- косвенно – по значению переменной, которая индексирует массив

Нумерация элементов (значение индекса) начинается с 0!!!

6.1.1 Объявление строки

Объявление строки

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

Объявление указателя на строку

char *<Имя указателя>[= <Значение>];

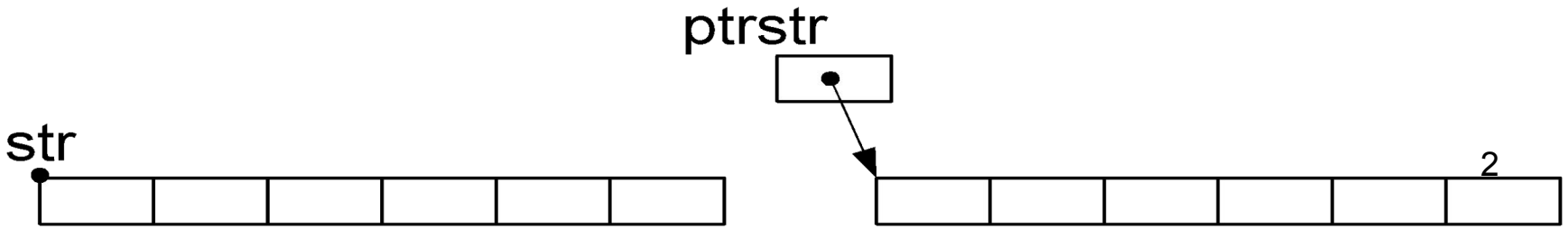
Память под строку, объявленную как одномерный массив выделяется статически на этапе компиляции.

Память под строку, объявленную как указатель на строку необходимо выделить самостоятельно на этапе выполнения из области динамической памяти. (За исключением тех, которые инициализируются при объявлении.)

Примеры:

а) **char str[6];** б) **char *ptrstr;**

ptrstr=new char[6]; **ptrstr=(char*)malloc(6);**



Объявление и инициализация строки(2)

в) **char str1[5] = {‘A’,’B’,’C’,’D’,’\0’};**

A	B	C	D	\0

Система выделит 5 байт и в них разместятся символы. Символ «\0» нужно обязательно задать!!

г) **char str1[12] = {‘A’,’B’,’C’,’D’,’\0’};**

A	B	C	D	\0	\0	\0	\0	\0	\0	\0	\0	\0

д) **char str1[ ] = “ABCD”;**

Система выделит 4 байта для размещения символов и 1 байт под символ «\0» !

е) **char *str2 = “ABCD”;**

A	B	C	D	\0

Инициализировать можно только статические и внешние строки!!

Однако, хотя строки описаны разными способами, во всех случаях имя строки является ее адресом (указателем).

Но имя строки, описанной как одномерный символьный массив, является константным (к нему нельзя применять адресную арифметику)

Объявление и инициализация массивов строк

Массив указателей на строки

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

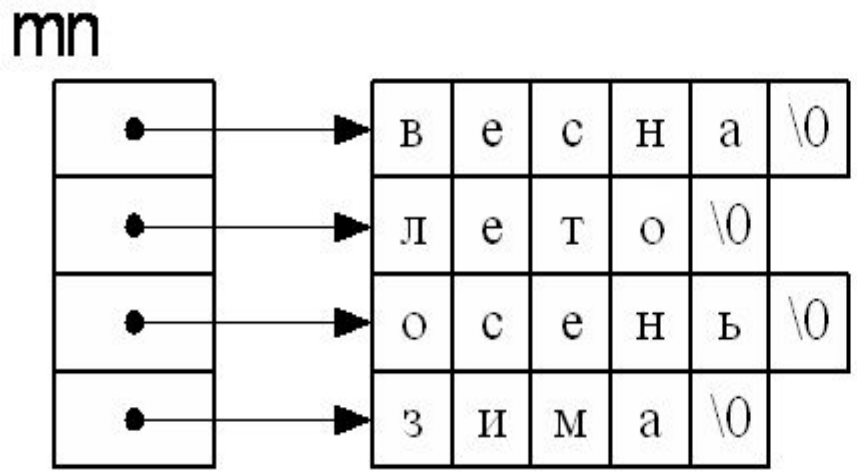
Массив строк указанной длины

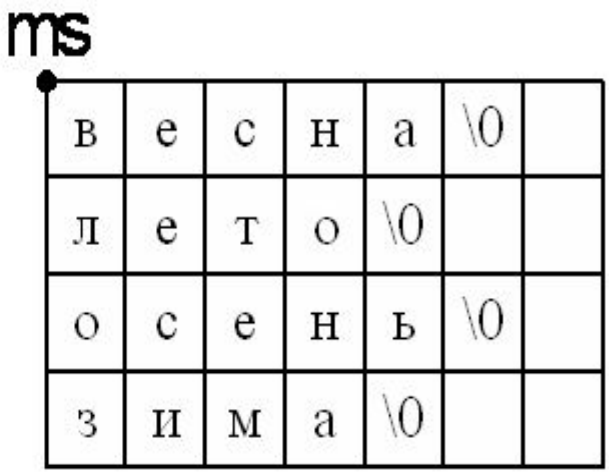
char <Имя>[<Размер 1>][Размер 2] [= <Значение>];

Примеры:

а) **char * mn[4] = {“весна”,”осень”,”зима”, “лето”};**

б) **char ms[4][7] = {“весна”,”осень”,”зима”, “лето”};**





6.1.2 Приемы обработки строк

Операции над строками

Так как имена строк являются указателями, для выполнения всех операций над строками используются специальные функции.

1. ***Ввод строк***

Процесс ввода строк выполняется в два шага: выделение памяти для сохранения строки и применение функции ввода.

Выделение памяти производится либо на этапе компиляции статически, либо на этапе выполнения – динамически.

char st1[10];

char * st2;

st2=new char [10];

Статическое выделение

Динамическое выделение

Ввод строки можно произвести с помощью двух функций – **gets** и **scanf**,

А также с помощью потоко-ориентированных средств (**cin**).

Функция gets() прототип char * gets(const * char s); :

Функция читает символы до символа новой строки (“\n”), который создается при нажатии на клавиатуре клавиши **enter.**

Функция считывает все символы до символа конца строки (исключая его), присоединяет к символам строки символ “\0” и передает строку программе

Операции над строками(2)

Если при считывании строки произошел сбой или вводилась пустая строка, то функция *gets* возвращает NULL (нулевой адрес).

Функцию можно вызывать

- как процедуру:

```
char name[10],*ptrs;
… gets(name);
   prts=name;
```

- как функцию

```
ptrs=gets(name);
```

Кроме того, функция gets может использоваться в конструкциях, подобных

while (gets(name)!=NULL){…} или if (gets(name)==NULL)

для организации циклов и проверок.

Операции над строками(3)

Функция scanf() :

Функция работает по формату %s и читает строку до первого пустого символа (пробела, знака табуляции, конца строки).

Если в формате указывается размер, например %10s, то функция читает не более 10 символов строки или до первого пустого символа.

Функция **scanf()** возвращает количество считанных символов или EOF(символ конца файла), если он встретился.

Функцию можно вызывать

- как процедуру:

```
char name[10];int count;
… scanf(“%s”,name);
```

- как функцию

```
count=scanf(“%10s”,name);
printf(“inputed %4d symbols\n”,count);
```

Кроме того, функция может использоваться в конструкциях, подобных

while (scanf(“%s”,name)!=EOF){…} или

if (scanf(“%s”,name)==NULL)

для организации циклов и проверок.

Операции над строками(4)

2. Вывод строк

Вывод строк выполняется с помощью двух функций **puts()** и **printf()** .

Функция puts() прототип : **int puts(char *s);**

Функция выводит строку, указанную в качестве аргумента. Вывод происходит до символа конца строки «\0», поэтому он обязательно должен быть.

В качестве аргумента можно указать строковую константу.

Каждая выводимая по puts строка начинается с новой строки.

В качестве результата возвращает количество выведенных символов.

Функцию можно вызывать

- как процедуру:

```
char name[10];int count;
… puts(name);
```

- как функцию

```
count=puts(“Example function PUTS ”);
printf(“vivod  %4d symbols\n”,count);
```

Операции над строками(5)

Функция printf():

Использует указатель на строку в качестве аргумента. Функция менее удобна, чем puts, но более гибка.

Функция выводит строку по формату %s, но автоматического перехода на новую строку не выполняет.

Для перехода необходимо указать в форматной строке символ «\n».

Вывод по формату %s позволяет выводить строки только до пробельного разделителя, поэтому используются для вывода строк без пробельных разделителей.

Преимущество функции printf() заключается в возможности объединения при выводе в одной строке нескольких строк.

Пример:

char name="Студент", MSG="GOOD" ;

printf("Examen %s %s\n",name,MSG);

6.1.3 Функции, работающие со строками

Библиотеки: **string.h, stdlib.h**

1) `size_t strlen(char *s);`
2) `char *strcat(char *dest,const char *src);`
3) `int strcmp(const char *s1,const char *s2);`
4) `char *strcpy(char *dest,const char *src);`
5) `char *strncpy(char *dest,const char *src,size_t maxlen);`
6) `char *strchr(const char *s, int c);`
7) `char *strstr(const char *s1, const char *s2);`
8) `char *strtok(char *strToken,const char *strDelimit);`
9) `int atoi(const char *s);`
10) `double atof(const char *s);`
11) `char *itoa(int value,char *s,int radix);`
12) `char *_gcvt( double value, int digits, char *buffer );`
13) `char *_ecvt(double value,int count,int *dec,int *sign);`

count - количество цифр, dec,sign – позиции точки и знака

14) `char *_fcvt(double value,int count,int *dec,int *sign);`

count - количество десятичных цифр

Примеры обработки строк

Примеры о преобразования числа в строку

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

void main( void )
{   int      decimal,    sign;   // позиция точки и знака
    char     *buffer;
    int      precision = 10;     // точность
    double   source = 3.1415926535;

    buffer =_ecvt( source, precision, &decimal, &sign );
    printf
      ("source: %2.10f buffer: '%s' decimal: %d sign: %d\n",
                              source, buffer, decimal, sign );
}
```

source: 3.1415926535 buffer: '3141592654' decimal: 1 sign: 0

Пример использования функции strtok (Ex2_03)

```
#include "stdafx.h"
#include <string.h>
#include <stdio.h>
char string[] = "A string\tof ,,tokens\nand some
                                        more tokens";
char seps[] = " ,\t\n", *token;
void main( void )
{ token = strtok( string, seps );
  while( token != NULL )
    {  printf( " %s ", token );
       token = strtok( NULL, seps );
    }
}
```

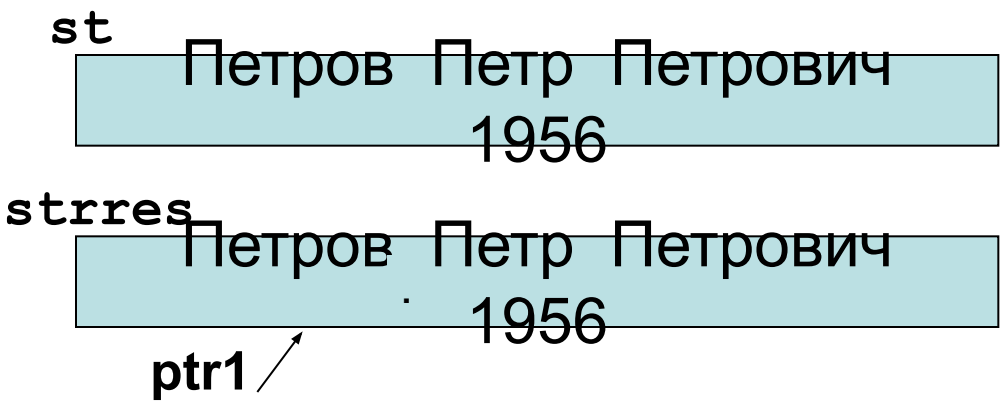
Результаты:

A string of tokens and some more tokens

Пример использования функций обработки строк

Петров Петр Петрович 1956 => Петров П.П. 50 (Ex2_04)

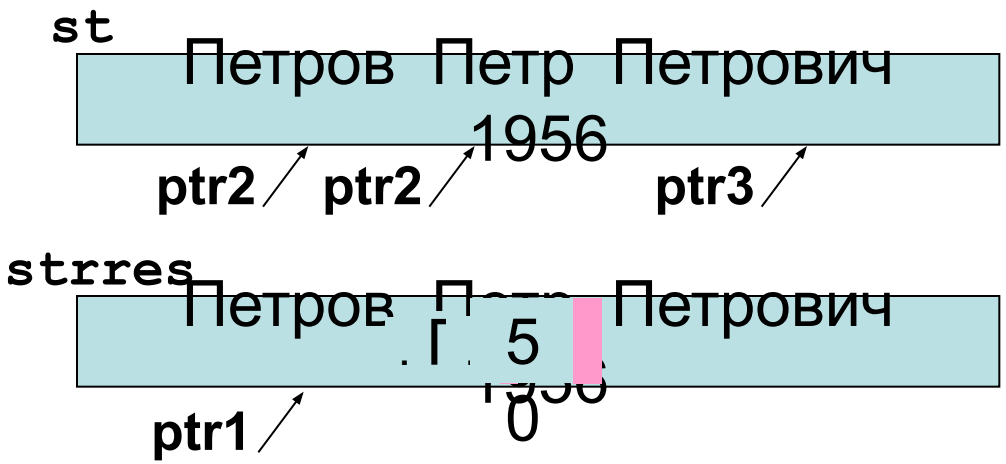
```
#include "stdafx.h"
#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include <conio.h>
```




```
int main(int argc, char* argv[])
{  char st[80],stres[80],strab[80],
        *ptr1,*ptr2,*ptr3;
   int old;
   while ((puts("Input string or Ctrl_Z:"),
                    gets(st))!=NULL)
   {  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=2011-atoi(ptr3+1);
    strcat(stres,itoa(old,strab,10));
    puts(stres);  }
  getch();  return 0;
}
```

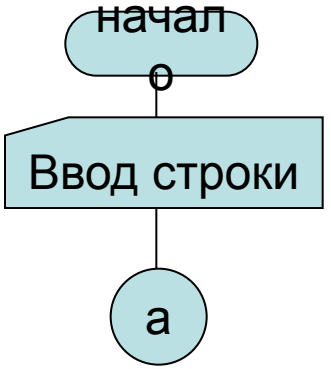


Пример определения количества слов в строке

Дана строка. Определить количество слов и их длину.

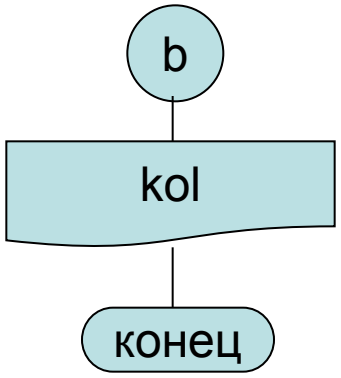
```
// Ex6_4.cpp
#include "stdafx.h"
#include <stdio.h>
#include <string.h>
int main(int argc, char* argv[])
{char s[80];
unsigned int i,kols,dls;
puts("input  words and space");
gets(s);
```

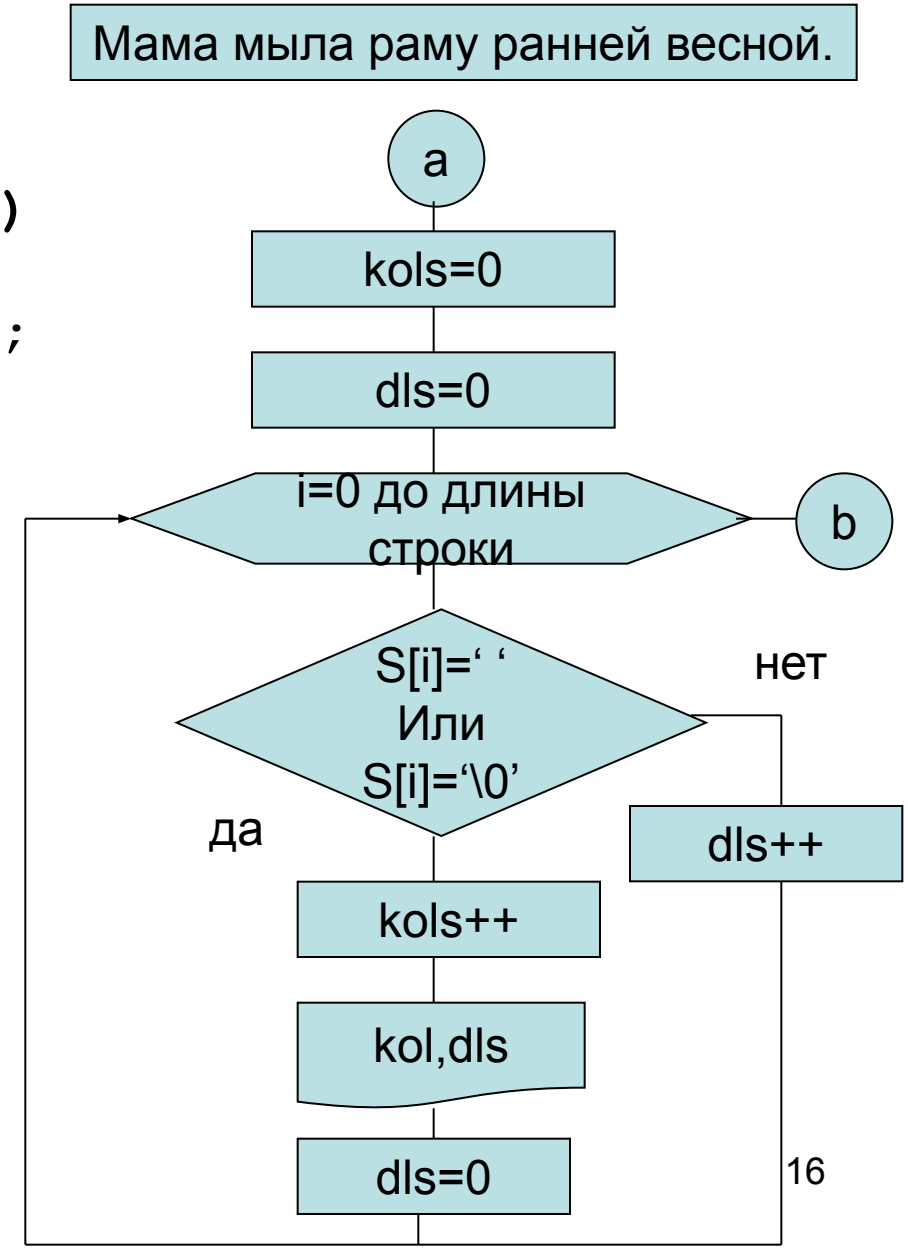
Мама мыла раму ранней весной



Пример определения количества слов в строке

```
kols=0;dls=0;
for(i=0;i<=strlen(s);i++)
{
 if ((s[i]==' ')||(s[i]=='\0'))
 {kols=kols+1;
  printf("Slovo  %3d) -",kols);
  printf("dlina%6d\n",dls);
  dls=0;}
  else dls++;
}
 printf("V stroke %5d",kols);
 printf)"  slov\n");
   return 0;
}
```

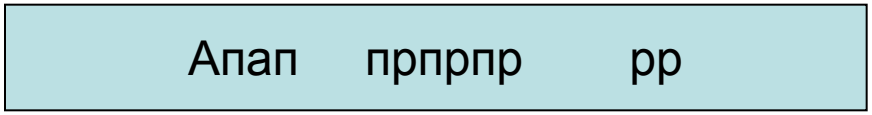


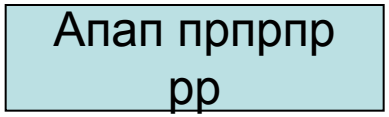


Пример программы вычеркивания лишних пробелов

Лишними называют более одного пробела между словами, а также в начале и конце строки.

```
// Ex6_5.cpp
#include "stdafx.h"
#include <stdio.h>
#include <string.h>
int main(int argc, char*
  argv[])
{char st[80],*ptr;
unsigned int i,kol,dlsl;
puts("input string ");
gets(st);
puts("isxodnaya stroka");
puts(st);
```






```
ptr=st;
while((ptr=strstr(ptr,
"  "))!=NULL)
  strcpy(ptr,ptr+1);
ptr=st;
if (st[0]==' ')
  strcpy(ptr,ptr+1);
if (st[strlen(st)-1]==' ')
 st[strlen(st)-1]='\0';
puts("Result string 2");
puts(st);
   return 0;
}
```

6.2 Структуры

Иногда, при составлении программ необходимо объединить в единое целое разнородную, но логически связанную информацию.

Например, нам необходимо хранить данные библиотечной карточки, содержащей следующую информацию о книге:

- фамилия и инициалы автор;
- название книги;
- место издания;
- издательство; -;
- год издания;
- количество страниц.

Объединить такую разнородную информацию удобно с помощью структуры.

Структура – это объединенное в единое целое множество поименованных элементов разных типов.

Структуры(2)

1. Объявление (Си)

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

Примеры:

а) **struct student { char name[22]; char family[22]; int old; };**
struct student stud1={“Петр”, “Петров”, 19}, stud[10], *ptrstud;

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

2. Объявление (C++)

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={“Петр”, “Петров”, 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

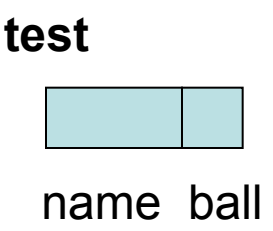
stud[i].name

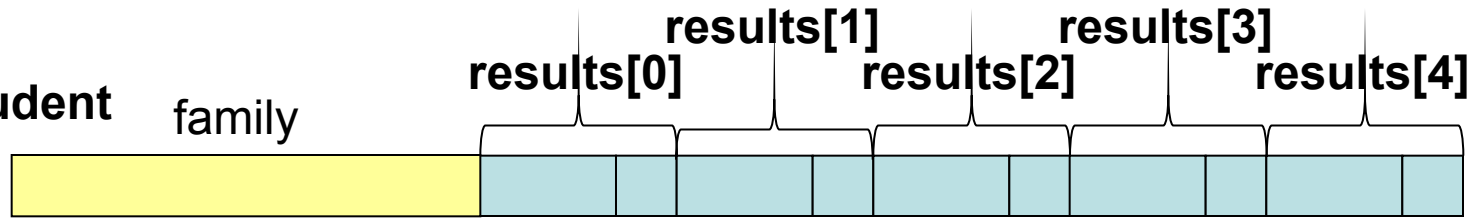
(*ptrstud).name ⇔ ptrstud -> name

Пример использования структуры (Ex6_6)

Программа определения среднего балла каждого студента и группы в целом (Пример 6.6)

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





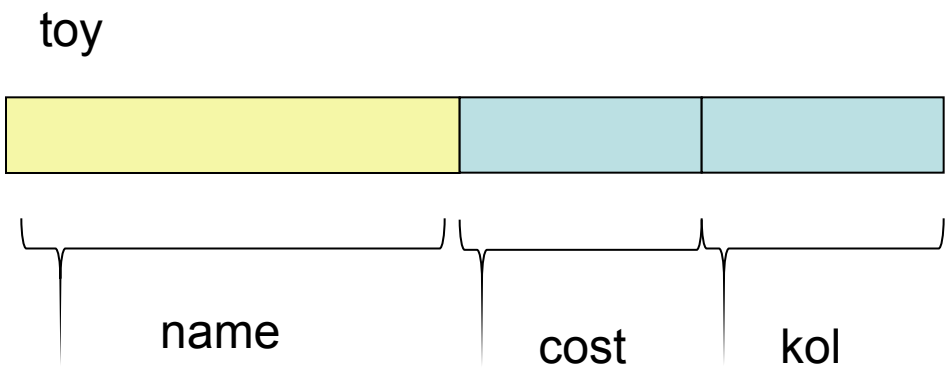
Пример использования структуры (2)

```
int main(int argc, char* argv[])
{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);
 getch();
 return 0;}
```

Пример использования структуры (3)

Написать программу формирования массива данных об игрушках, содержащих их название, количество и стоимость, и определения товара с наибольшей стоимости.

```
// Ex6_7.cpp
#include "stdafx.h"
#include <stdio.h>
#include <string.h>
typedef struct
 {char name[15];
float cost; int kol;}toy;
int main()
{toy mas[10];
int i,n=-1,k;
char st[15];
while((puts("input toy name or end"),
scanf("\n%s",st),strcmp(st,"end")!=0)&&(n<9))
{n++;
strcpy(mas[n].name,st);
printf("input cost kol  ");
scanf("%f %d",&mas[n].cost,&mas[n].kol);
}
```



Пример использования структуры (4)

```
float sumcost=0,maxcost=0, num;
st[0]='\0';
puts("=====    massiv of toys   ========");
puts(" N         name      cost        kol ");
puts("=================================");
for(i=0;i<=n;i++)
{
printf("%3d   %10s”,i+1,mas[i].name);
printf(“ %6.2f  %5d\n",mas[i].cost,mas[i].kol);
sumcost=mas[i].cost*mas[i].kol;
if (sumcost>maxcost) {
maxcost=sumcost;
strcpy(st,mas[i].name);
}
}
printf("tovar %10s Have maxcost= %8.3f\n",st,maxcost);
return 0;
}
```

Определение суммарной стоимости товара

Определение максимальной стоимости товара

Сохранение названия товара

2.6 Объединения

union <Имя объединения>

{<Список элементов объединения>}

[<Список переменных [и значений]>];

Пример:

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