

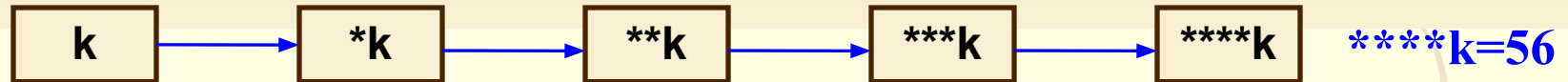
АЛГОхитрости. УКАЗАТЕЛИ

C / C++

Практические занятия

C / C++

- Сколько занимают в памяти указатели чисел типа **double** и **long double**?
- Пусть **a** и **b** - указатели. Всегда ли ***a == *b**, если **a == b**?
- Верно ли обратное, т.е. Всегда ли **a == b**, если ***a == *b**?
- В чем разница ***a** и **a[0]**?
- Допустимо ли выражение ******k=56**? Что оно делает?



Создание цепочки указателей

```
#include <iostream.h>
// Создание цепочки указателей
main ()
{ int *i;
  i = new int;
  int **j = &i;
  int ***n = &j;
  int ****k = &n;
  ****k = 56;
  cout << "\n значение i= " << *i << "\n адрес i=" << i << "\n адрес j=" << j
  << "\n адрес n=" << n << "\n адрес k=" << k;
  cout << "\n значение ****k= " << ****k << "\n значение ***k=" << ***k << "\n
  значение **k=" << **k << "\n значение *k=" << *k << "\n адрес k=" << k;
  delete i; return 0;}
```

АЛГОхитрости. УКАЗАТЕЛИ

C / C++

Практические занятия
Указатели на константы

C / C++

Чем отличается *int * const* от *int const **. Возможен ли указатель *int const * const* ?

Форма *int * const* означает константность указателя: указатель объявленный с этим определением, не может менять адрес, на который он указывает.

```
#include <iostream.h>
main ()
{ int number = 5;
  int var = 10;
  int *const prt = &number;
  *prt = 100;    // Допустимо
  prt = &var;    // Ошибка
return 0; }
```

Форма *int const ** означает константность самого значения, на которое адресует указатель: адрес, на который указывает указатель можно менять, но изменить значение содержимого адреса при помощи указателя невозможно.

```
#include <iostream.h>
main ()
{ int number = 5;
  int var = 10;
  int const * prt = &number;
  prt = &var;    // Допустимо
  var = 15;
  *prt = 100;    // Ошибка
return 0; }
```

Если модификации не должен подвергаться ни сам указатель, ни значение, на которое он указывает, то допускается использование типа *int const * const*

```
#include <iostream.h>
main ()
{ int number = 5;
  int var = 10;
  int const *const prt = &number;
  *prt = 100;    // Ошибка
  prt = &var;    // Ошибка
return 0; }
```

АЛГОхитрости. УКАЗАТЕЛИ

C / C++

Практические занятия

C / C++

Найти ошибку

```
#include <iostream.h>
// Переменная var получает целое значение и используется
// для инициализации указателя prt
main ()
{ int var, *prt;
  var = 10;
  *prt = var;
  cout << "\n Значение prt -> " <<* prt);
  return 0; }
```

Ошибка заключается в следующем:

Указатель не инициализирован, он не указывает ни на одну переменную, под которую выделена статическая или динамическая память. Программа будет работать, но ошибка может проявиться сбоями программы, причём не сразу, а через некоторое время после начала работы.

НАДО:

```
#include <iostream.h>
// Переменная var получает целое значение и используется
// для инициализации переменной, на которую указывает prt
main ()
{ int var, *prt;
  prt = new int;
  var = 10;
  *prt = var;
  cout << "\n Значение переменной, на которую
  указывает prt -> " <<* prt);

  delete prt;
  return 0; }
```

Ввести текстовую строку и вывести её на экран по 4-е символа в одной строке

```
#include <iostream.h>
#include <stdio.h>
main ()
{
  char input [80];
  char *current;
  int i = 1;

  cout << "Введите строку ->";
  gets (input);

  for (current=input; *current;
      current++, i++)
  {
    cout << *current;
    if (! (i % 4))
      cout << endl;
  }
  return 0;
}
```

АЛГОхитрости. УКАЗАТЕЛИ

C / C++

Практические занятия

C / C++

Разбить текстовую строку по символу пробела

Без использования указателей

```
#include <iostream.h>
#include <stdio.h>
#include (string.h)
int main()
{
    char input[80];
    char current[80];
    int i, j;
    cout << "Введите строку -> ";
    gets(input);

    for(i = 0; i < strlen (input); i++)
    {
        for(j = 0; input[i] != ' ' && input[i];
            j++, i++)
        {
            current[j] = input[i];
        }
        current[j] = '\0';
        cout << current << endl;
    } return 0;
}
```

C использованием указателей

```
#include <iostream.h>
#include <stdio.h>
#include (string.h)
int main()
{
    char input[80];
    char current[80];
    char *prt, *prt_cur;
    cout << "Введите строку -> ";
    gets(input);
    prt = input;
    while(*prt)
    {
        prt_cur = current;
        while(*prt != ' ' && *prt)
        {
            *prt_cur = *prt;
            prt_cur++;
            prt++;
        }
        // Пропускаем пробел
        if(*prt) prt++;
        *prt_cur = '\0';
        cout << current << endl;
    }
    return 0;
}
```

C

Функция calloc

```
#include <stdlib.h>
#include <stdio.h>
#define N 100000
float *get_mem(void)
{
    float *p;
    p = calloc(N, sizeof(float));
    if (!p)
        { printf("????\n");exit(1); }
    return 0; }

main ()
{ get_mem();
  .....
  return 0; }
```

Программа возвращает указатель на динамический блок памяти для массива из N чисел типа float.

```
printf("Ошибка при
распределении памяти\n");
```

Практическая работа:
что делают программы

Функции malloc и free

```
#include <stdlib.h>
#include <stdio.h>
int main (void)
{ char *str[10]; int i;
  for (i=0; i<10; i++);
    { if ((str[i] =
(char*)malloc(10))==NULL)
      { printf("???????\n");
        exit(1); }
      printf("???????\n");
      gets (str[i]); }
  // ??????????
  for (i=0; i<10; i++);
    free (str[i]);
  return 0; }
```

Программа распределяет блок памяти для строк, а затем освобождает этот блок.

```
printf("Ошибка при распределении
памяти\n");
printf ("Введите строку
символов\n");
// Освобождение блока памяти
```

C

Функция realloc

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
int main (void)
{ char *p;
  p = (char*) malloc(16);
  if (!p)
    { printf("???????\n"); exit(1);}
    strcpy (p,"Это-15 символов\n");
  p=(char*)realloc (p < 17);
  if (!p)
    { printf("???????\n"); exit(1);}
    strcat (p < ".");
    printf("\n"); printf(p);
    free(p); return 0; }
```

Программа распределяет блок памяти для 16 символов, копирует в них строку "Это-15 символов", а затем увеличивает размер блока до 17 символов, чтобы разместить в конце точку и, наконец, освобождает этот блок.

АЛГОхитрости. УКАЗАТЕЛИ.

Pascal

Примеры:

Что делает программа?

```
Program MatrPointer;
Const  SizeOfReal = 6; (* Длина переменной типа REAL *)
      N = 100; (* Количество столбцов *) M = 200; (* Количество строк *)
Var  i, j : integer; PtrStr : array [1..N] of pointer; s : real;
Type  RealPoint = ^real;
(*-----*)
Function AddrR (i, j : word) : RealPoint;
(* По сегменту i и смещению j выдает адрес вещественной переменной *)
Begin  AddrR := ptr(seg(PtrStr[i]^), ofs(PtrStr[i]^) + (j-1)*SizeOfReal);
end (* AddrR *);
(*-----*)
Function GetR (i, j : Integer) : Real;
(* Выдает значение вещественной переменной по сегменту i и смещению j её адреса *)
Begin  GetR := AddrR (i, j)^;
end (* GetR *);
(*-----*)
Procedure PutR (i, j : Integer; x : Real);
(* Помещает в переменную, адрес которой имеет сегмент i и смещение j, вещественное значение x *)
Begin  AddrR (i, j)^ := x;
end (* PutR *);
(*-----*)
begin (* Main *)
(* Программа создает в памяти матрицу NxM случайных чисел и вычисляет их среднее значение *)
  for i := 1 to N do begin
    GetMem (PtrStr[i], M*SizeOfReal); for j:= 1 to M do PutR (i, j, Random); end;
s := 0; for i := 1 to N do for j := 1 to M do s := s + GetR (i, j);
writeln ('Среднее значение = ', s / (N * M) : 12:10); end (* Main *).
```

АЛГОхитрости. УКАЗАТЕЛИ.

Pascal

Лучше сделать так!!!

```
Program MatrPointer;
(* Const SizeOfReal = 6; (* Длина переменной типа REAL *) *)
N = 100; (* Количество строк *) M = 200; (* Количество столбцов *)
Var i, j : integer; PtrStr : array [1..N] of pointer; s : real;
Type RealPoint = ^real;
(*-----*)
Function AddrR (i, j : word) : RealPoint;
(* По сегменту i и смещению j выдает адрес вещественной переменной *)
Begin AddrR := ptr(seg(PtrStr[i]^), ofs(PtrStr[i]^) + (j-1)*SizeOf(Real));
end (* AddrR *);
(*-----*)
Function GetR (i, j : Integer) : Real;
(* Выдает значение вещественной переменной по сегменту i и смещению j её адреса *)
Begin GetR := AddrR (i, j)^;
end (* GetR *);
(*-----*)
Procedure PutR (i, j : Integer; x : Real);
(* Помещает в переменную, адрес которой имеет сегмент i и смещение j, вещественное значение x *)
Begin AddrR (i, j)^ := x;
end (* PutR *);
(*-----*)
begin (* Main *)
(* Программа создает в памяти матрицу NxM случайных чисел и вычисляет их среднее значение *)
for i := 1 to N do begin
    GetMem (PtrStr[i], M*SizeOf(Real)); for j:= 1 to M do PutR (i, j, Random); end;
s := 0; for i := 1 to N do for j := 1 to M do s := s + GetR (i, j);
writeln ('Среднее значение = ', s / (N * M) : 12:10); end (* Main *).
```

Pascal

C/C++

Индивидуальные домашние задания на указатели и динамическую память

Написать программы на Pascal и C