ФГБОУ ВПО ЧГУ им. И.Н. Ульянова
факультет радиотехники и электроники
кафедра «телекоммуникационные системы и технологи»

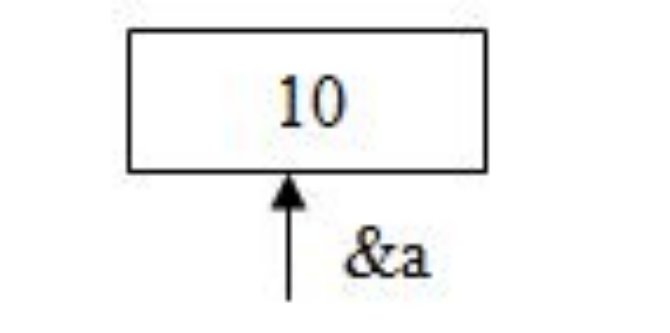
УКАЗАТЕЛИ И ДИНАМИЧЕСКИЕ МАССИВЫ

Лекция 2.4.

ст.преп. Васильева Л.Н.

Указатель переменная, значением которой является адрес памяти, по которому хранится объект определенного типа.

```
int a=10;
```



Указатели предназначены для хранения адресов памяти.

Указатель не является самостоятельным типом, он всегда связан с каким-то другим типом!

Указатели делятся на две категории:

- **указатели на объекты**
- указатели на функции.

В простейшем случае объявление указателя имеет вид:

```
тип *имя;
```

Примеры:

```
int *i;        double *f,*ff;      char *c;
int**a; // указатель на указатель
```

Указатель может быть константой или переменной, а также указывать на константу или переменную.

Примеры:

```
int i;        //целая переменная
const int ci=1;  //целая константа
int *pi=&i;    //указатель на целую переменную
const int *pci =&ci; //указатель на целую константу
```

Способы **инициализации** указателя **существующего** объекта:

1. с помощью операции получения адреса

```
int a=5;
int *p=&a; или int p(&a);
```

2. с помощью проинициализированного указателя

```
int *r=p;
```

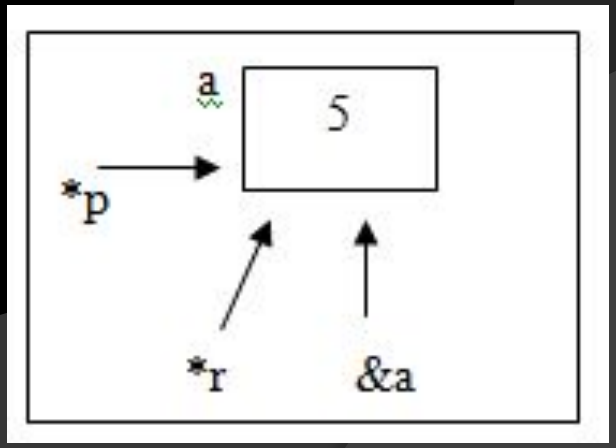
3. адрес присваивается в явном виде

```
char*cp=(char*)0x B800 0000;
```

,где 0xB8000000 – 16-ая константа,
(char*) – операция приведения типа.

4. присваивание пустого значения:

```
int*N=NULL;
int *R=0;
```



Динамическая память и динамические переменные

- **Динамическая память** – это память, выделяемая программе для ее работы за вычетом сегмента данных, стека.
- . С помощью **указателей** осуществляется доступ к участкам динамической памяти, которые называются **динамическими переменными**

Для создания динамич. переменных используют операцию **new**

```
указатель = new имя_типа[инициализатор];
```

Пример:

```
int*x=new int(5);
```

- Динамические переменные существуют либо до конца работы программ, либо до тех пор, пока не будут уничтожены с помощью специальных функций или операций.

Для удаления динамич. пер-ых используется операция **delete**

```
delete указатель;
```

Пример:

```
delete x;
```

В языке Си определены **библиотечные функции** для работы с динамической памятью, они находятся в библиотеке **<stdlib.h>**:

1. **`void*malloc(unsigned s)`** возвращает указатель на начало области динамической памяти длиной s байт, при неудачном завершении возвращает NULL;
2. **`void*calloc(unsigned n, unsigned m)`** – возвращает указатель на начало области динамической для размещения n элементов длиной m байт каждый, при неудачном завершении возвращает NULL;
3. **`void*realloc(void *p,unsigned s)`** –изменяет размер блока ранее выделенной динамической до размера s байт, p – адрес начала изменяемого блока, при неудачном завершении возвращает NULL;
4. **`void *free(void *p)`** – освобождает ранее выделенный участок динамической памяти, p- адрес начала участка.

Пример:

```
int *u=(int *)malloc(sizeof(int));
free(u); //освобождение выделенной памяти
```

Операции с указателями

1. **разыменование** (*);
2. **присваивание**;
3. **арифметические операции** (сложение с константой, вычитание, инкремент ++, декремент --);
4. **приведение типов.**

1) Операция **разыменования** предназначена для получения значения переменной или константы, адрес которой хранится в указателе.

Если указатель указывает на переменную, то это значение можно изменять, также используя операцию разыменования.

Пример:

```
int a; //переменная типа int
int*pa=new int; //указатель и выделение памяти под динамическую переменную
*pa=10; //присвоили значение динамической переменной, на которую указывает указатель
a=*pa; //присвоили значение переменной a
```

Присваивать значение указателям-константам запрещено!!!

2) Значение одного указателя можно **присвоить** другому. Если указатели одного типа, то один можно присваивать другому с помощью обычной операции присваивания(=).

Пример:

```
#include <stdio.h>
int main()
{
float PI=3.14159,*p1,*p2;
p1=p2=&PI;
cout<<"По адресу p1="<<p1<<" хранится "<<*p1);
cout<<"По адресу p1="<<p1<<" хранится "<<*p1);
}
```

По адресу p1=0012FF7C хранится *p1=3.14159

По адресу p2=0012FF7C хранится *p2=3.14159

3) Арифметические операции применимы только к указателям одного типа.

Пример:

```
double *p1;
float *p2;
int *i;
p1++
p2++;
i++;
}
```

Операция p1++ увеличивает значение адреса на 8, операция p2++ увеличивает значение адреса на 4, а операция i++ на 4.

Операции адресной арифметики выполняются следующим образом:

- операция увеличения приводит к тому, что указатель будет слаться на следующий объект базового типа (для p1 э то double, для p2 float, для i int);
- операция уменьшения приводит к тому, что указатель, ссылается на предыдущий объект базового типа.
- после операции p1=p1+n, указатель будет передвинут на n объектов базового типа;
- p1+n как бы адресует n-й элемент массива, если p1 адрес начала массива .

4) Приведение типов

На одну и ту же область памяти могут ссылаться указатели разного типа. Если применить к ним операцию разыменования, то получатся разные результаты.

Пиример:

```
int a=123;
int*pi=&a;
char*pc=(char*)&a;
float *pf=(float*)&a;
cout<<"\n"<<pi<<" "<<*pi;
cout<<"\n"<<pc<<" "<<*pc;
cout<<"\n"<<pf<<" "<<*pf;
```

```
66fd9c  123
66fd9c  {
66fd9c  123
```

Одномерные массивы и указатели

С помощью указателей в С++ можно выделить участок памяти (динамический массив) заданного размера для хранения данных определенного типа. Для этого необходимо выполнить следующие действия :

1. **Описать указатель** (например, переменную p) определенного типа.

2. Начиная с адреса, определенного указателем, с помощью функций calloc, malloc или операции new выделить участок памяти определенного размера. После этого p будет адресом первого элемента выделенного участка оперативной памяти (0-й элемент массива), p+1 будет адресовать следующий элемент в выделенном участке памяти (1-й элемент динамического массива), p+i является адресом i-го элемента.

Необходимо только следить, чтобы не выйти за границы выделенного участка памяти.

К i-му элементу динамического массива p можно обратиться одним из двух способов *(p+i) или p[i].

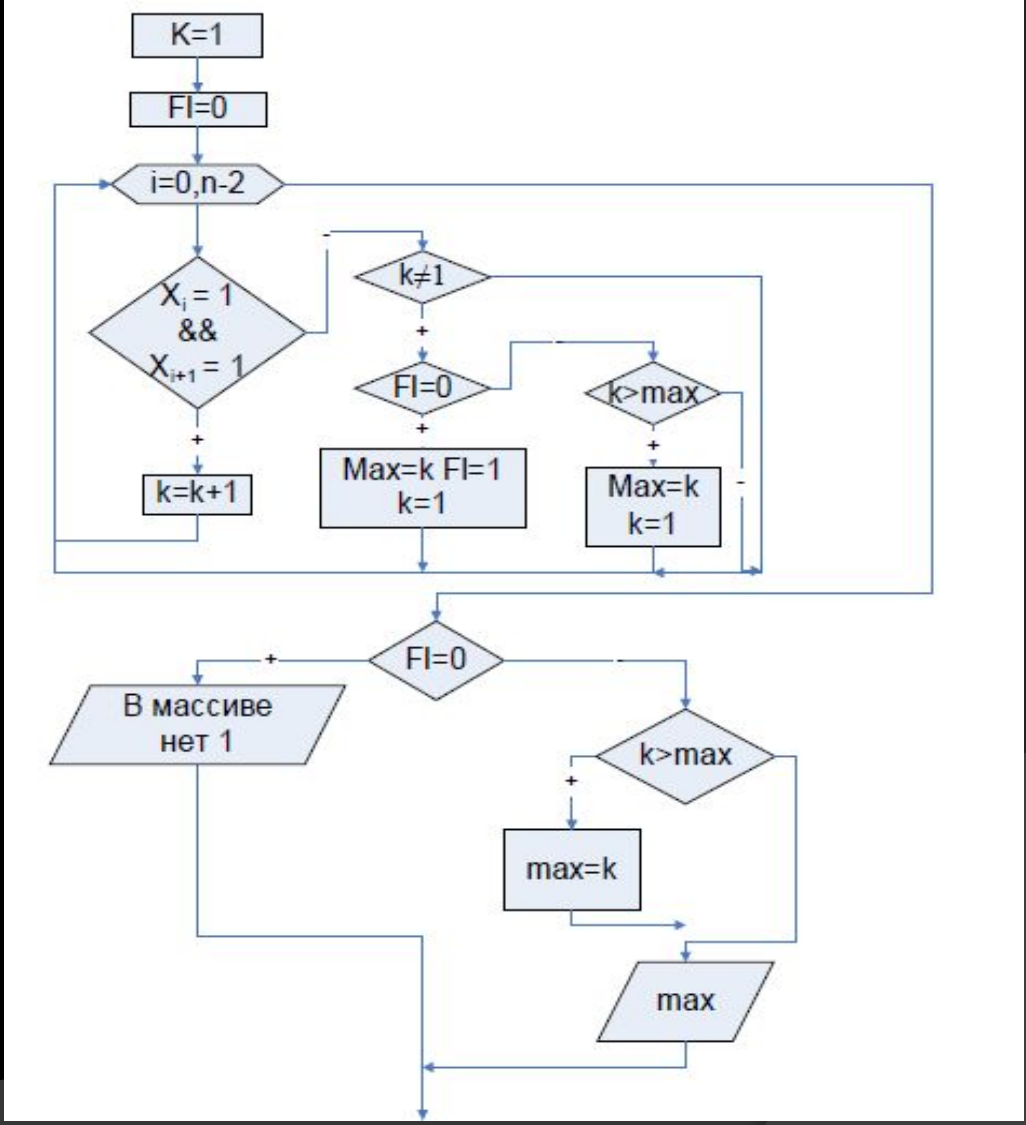
3. Когда участок памяти будет не нужен, его можно освободить с помощью функции free(), операции delete.

В заданном массиве найти длину самой длинной серии элементов, состоящей из единиц.

Переменная **Fl** принимает значение 1, если была серия элементов, состоящая из единиц, и 0 если такой серии не было.

В переменной **k** хранится длина текущей серии.

В переменной **max** количество элементов в самой длинной серии.



```
int main()
{int *x,max,i,k,fl,n;
cout<<"\n n=";
cin>>n;
x=new int[n];
for (i=0;i<n;i++)
{   cout<<"\n x["<<i<<”]=",i);
    cin>>x[i];
}
for (k=1,fl=0,i=0;i<n-12i++)
{   if (x[i]==1 && x[i+1]==1)      k++;
    else    if (k!=1)
        {  if (!fl)
                {  max=k; fl=1; k=1;}
           else if (k>max) max=k;
           k=1;
        }
}
if (fl==0) cout<<"V massive net seriy iz 1\n";
     else { if (k>max) max=k; cout<<endl<<max;}
delete [] x; return 0;
}
```

Контрольные вопросы

1. Какое общее назначение указателей в языке **C?**
2. Какие арифметические операции допускаются для указателей?
3. Какие унарные операторы используются с указателями?
4. Для каких типов данных может быть использован указатель?
5. Как осуществляется инициализация указателей на вещественные типы данных?
6. Как осуществляется инициализация указателей на символьный тип данных?
7. Какой смысл имеет значение указателя NULL?
8. Что произойдет, если применить к указателю со значением **NULL операцию разыменования?**
9. Как следует определять и инициализировать указателя на константу?
10. Как следует определять и инициализировать константный указатель?
11. Какое отличие константного указателя от указателя на константу?