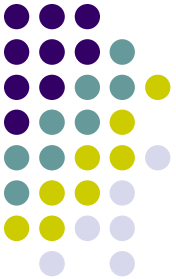
NIT



Показивач на void

- Показивач на неки тип може узети само адресу тог истог типа, у супротном компајлер јавља упозорење или грешку
- Али то не важи за показивач на воид

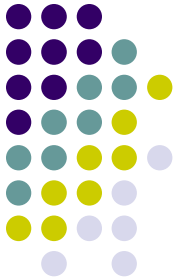
```
int var;
float* fptr;
void* vptr;

fptr = &var; /* compiler warning */
vptr = &var; /* OK */
```

- Ова могућност је некада потребна - пре свега као механизам превезилажења ограничења које намеће статичка типизираност

```
pthread_create(...,func1,(void*)&args1);
pthread_create(...,func2,(void*)&args2);

struct params1
{
  int handle;
  int value;
} args1 = {0x45689216, 57};

struct params2
{
  short id;
  char* ident;
} args2 = {17, "hd0"};
```

```
void* func1(void* param)
{
  struct params1* args;
  args = (struct params1*)param;
  printf("%x %d", args->handle, args->value);
}


void* func2(void* params)
{
  struct params2* args;
  args = (struct params2*)param;
  printf("%d %s", args->id, args->ident);
}
```

Показивачи и const квалификатор 1/2



- Показивач се може мењати али не и оно на шта он показује
- Кључна реч **const** мора бити лево од '*'

```
const type* ptr_variable;
type const* ptr_variable;
```

```
int var1 = 3;
int var2 = 5;
const int* ptr = &var1;
ptr = var2; /* OK */
*ptr = 7;   /* error */
```

- Експлицитна конверзија мора бити коришћена када вредност оваквог показивача желимо доделити обичном, неконстантном, показивачу

```
int* ptr;
int const* cptr;

ptr = cptr;  /* compiler warning or error */

ptr = (int*)cptr; /* OK */
```

Показивачи и const квалификатор 2/2

- Могуће је мењати оно на шта показивач показује, али не и сам показивач
- Кључна реч **const** мора бити са десне стране '*'

```
type* const ptr_variable;
```

```
int var1 = 3;
int var2 = 5;
int* const ptr = &var1;
ptr = &var2; /* error */
*ptr = 7;    /* OK */
```

- А могућ је и дупло константни показивач

```
const type* const ptr_variable;
type const* const ptr_variable;
```

```
int var1 = 3;
int var2 = 5;
int const* const ptr = &var1;
ptr = &var2; /* error */
*ptr = 7;    /* error */
```

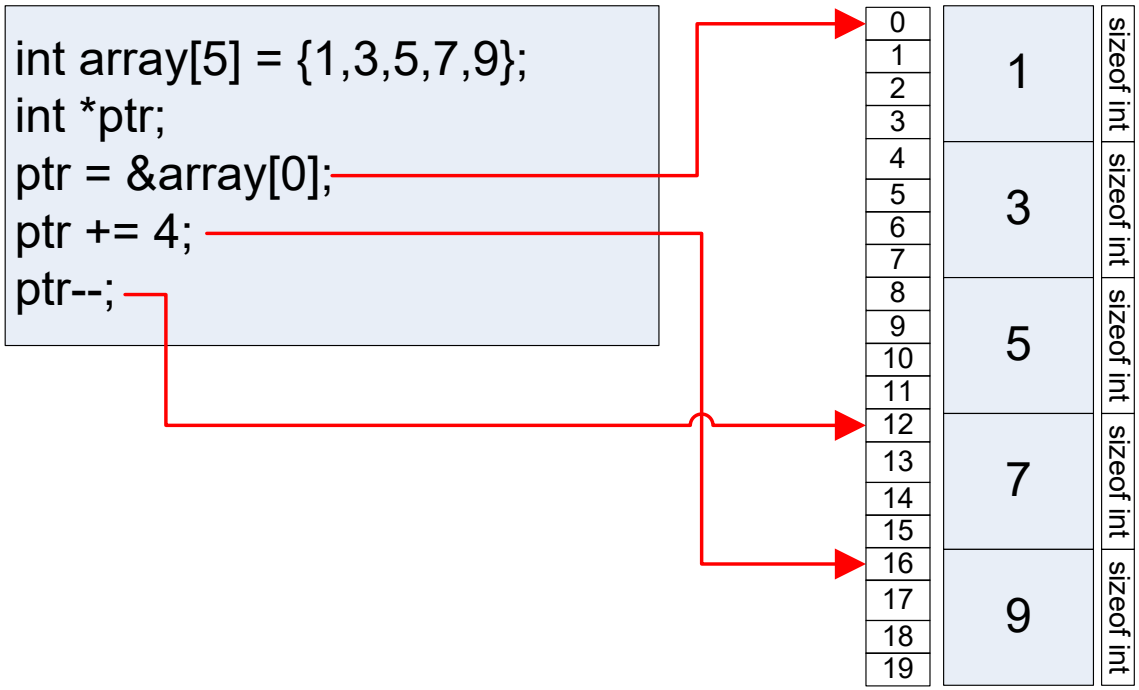
sizeof оператор

- Унарни оператор који срачунава величину типа и изражава је у бајтима (Подсетити се шта је бајт)
- Ради над променљивом или над типом
  - Када ради над променљивом враћа величину те променљиве, то јест величину типа ког је та променљива
  - Када ради над типом враћа његову величину
- По дефиницији sizeof(char) је 1

```
int* ptr;
size_t s;

s = sizeof(*ptr);
printf("%d\n", s);

s = sizeof(int);
printf("%d\n", s);

s = sizeof(ptr);
printf("%d\n", s);
```

Показивачи и операције над њима 1/4

- Сабирање и одузимање целог броја:
  - `data_type* ptr;`
    `ptr ± n <=> ptr ± n * sizeof(data_type)`
  - Исто важи и за унарне операторе ++/--



Показивачи и операције над њима 2/4

- Одузимање два показивача - само ако су истог типа

```
#include <stddef.h>
int array[5] = {1,3,5,7,9};
int* ptr1;
int* ptr2;
ptrdiff_t diff;

ptr1 = &array[1];
ptr2 = &array[4];
diff = ptr2 - ptr1;
```

diff = 3

- И има смисла само ако показују на адресе унутар истог парчета меморије

```
int array1[5] = {1,3,5,7,9};
int array2[5] = {2,4,6,8,10};
int* ptr1;
int* ptr2;
ptrdiff_t diff;

ptr1 = &array1[1];
ptr2 = &array2[4];
diff = ptr2 - ptr1;
```

Undefined result

Сабирање два показивача не може

Показивачи и операције над њима 3/4

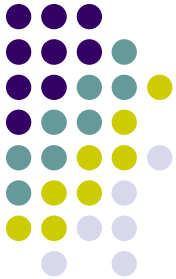
- Поређење показивача:
  - Могуће поредити само показиваче на објекте

```
int array[5] = {9,7,5,3,1};
int* ptr1;
int* ptr2;

ptr1 = &array[1];
ptr2 = &array[4];
if(ptr1 < ptr2)
  printf("Expected\n");
else
  printf("Unexpected\n");
```

Output: Expected

- Такође има смисла само ако показују на адресе унутар истог парчета меморије

Показивачи и операције над њима 4/4



- Оператор []

  `def: A[B] <=> *(A + B)`

A + B мора бити показивачког типа јер оператор * ради само са показивачем

Дакле, или A мора бити показивач, а B целобројног типа, или обрнуто!

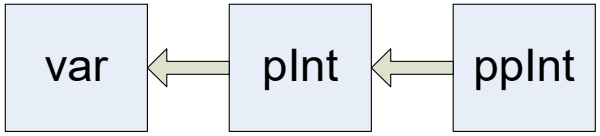
```
data_type* A; int B;
A[B] <=> *(A + B) <=> *(A + B * sizeof(data_type))
data_type* A; int B;
B[A] <=> *(B + A) <=> *(A + B * sizeof(data_type))
```

```
float* p;
float x;

/* neka je p 1000, tj. p pokazuje na adresu 1000 */

x = *p; // x je float vrednost sa adrese 1000
x = p[0]; // x je float vrednost sa adrese 1000
x = p[4]; // x je float vrednost sa adrese 1000 + 4*sizeof(float)
x = 4[p]; // x je float vrednost sa adrese 1000 + 4*sizeof(float)
```

Показивач на показивач 1/2

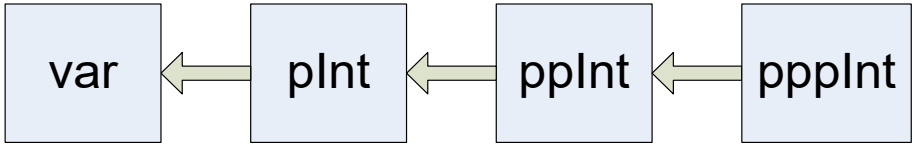
- Показивач може показивати на било који тип, па тако и на тип показивача

```
int var;
int* pInt = &var;
int** ppInt = &pInt;
```



- И то тако може у недоглед

```
int var;
int* pInt = &var;
int** ppInt = &pInt;
int*** pppInt = &ppInt;
```



Показивач на показивач 2/2

- Када нам то треба?
- 1. Прослеђивање показивача по референци

```
int g_var;

void bar(int** p)
{
   /* change pointer value */
   *p = &g_var;

   /* change value of variable to
    which pointer points to */
   **p = 39;
}
```

```
void foo()
{
   int var;
   int* ptr= &var;
   bar(&ptr);
   ...

}
```

- 2. Вишедимензионални низови...