

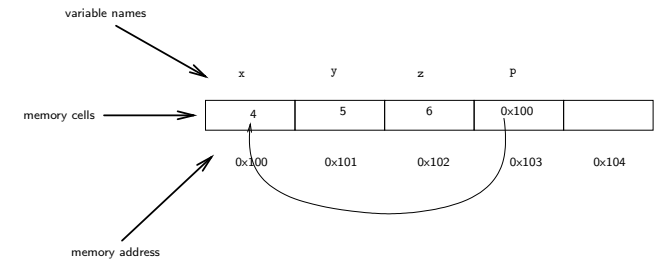
Pointers

C and C++ allow the value of a variable to be the memory address of another variable

Such variables are called *pointers*.

Available operations on pointers:

- Declaring them:
`int *p;` declares the variable `p` to be a pointer to a memory address which stores an integer
- Referencing a pointer:
If `p` is a pointer, `*p` is the memory cell `p` points to.
- Obtaining the address of a memory cell:
If `x` is a variable, `&x` is a pointer which points to the memory address of `x`.



Pointers and Function Arguments

C/C++ passes arguments to functions by value.

Hence if variable passed as argument and argument modified in

function, calling variable is *not modified*

debugger obeys scoping rules when displaying variables

Modification of calling variables may be achieved by passing pointers.



Arrays

Definition and assignment to elements of an array works as in Java.

Note: *array boundary limitations not enforced!*

Array boundary not necessary when arrays passed as argument.

In C, strings are character arrays

Each string terminated with `'\0'`.

A string constant "abcde" is an array of characters, terminated with `'\0'`.

Note: *This array is immutable*



Arrays and Pointers

value of an array is in fact pointer to first element
⇒ can use in programs name of array as pointer to first element

Important: array name is not a variable
⇒ assignment to it illegal.

When array is passed as argument of a function, in fact pointer to first element is passed
Declarations

```
void swap (int xs[], int ys[], int i)
```

and

```
void swap (int *xs, int *ys, int i)
```

have same effect

Address arithmetic

Pointers are memory addresses
C/C++ allows to use arithmetic operations on pointers (memory addresses)
Hence `xs[1]` and `*(xs + 1)` are the same.

Strings

In C, strings are character arrays
Each string terminated with `'\0'`.

A string constant `"abcde"` is an array of characters, terminated with `'\0'`.

Important: Have to be sure that space has been allocated
Declaration of pointer *does not* allocate any space

Space may be allocated by

- string constant:
`char *s = "abcde";`
- declaration of array

```
char s[6] = "abcde";
```

```
s[0] = 'c'
```

will work

- Explicit allocation: Have function `malloc` which requests memory from the OS

Space allocated for string constants *immutable*:

```
char *s = "This is a string";
```

```
*s = 'c';
```

will not work. Reason: Space allocated within immutable code section

Pointers to functions

Pointers may also be used to pass functions as arguments

Some functions use generic pointer `void *` as argument

This pointer needs to be cast to a pointer to whatever type the argument really is