CS101 Introduction to computing

# Array and Pointer

A. Sahu and S. V .Rao

Dept of Comp. Sc. & Engg.

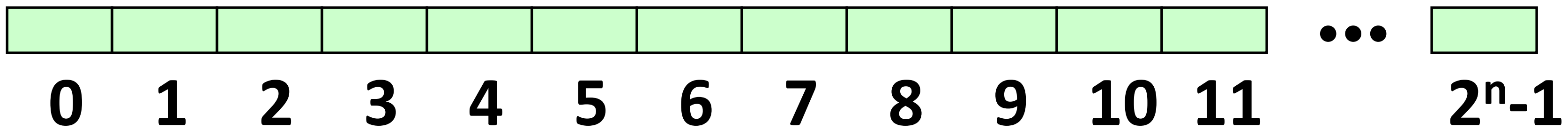
Indian Institute of Technology Guwahati

## Pointers

- Special case of bounded-size natural numbers
  - Maximum memory limited by processor word-size
  - $2^{32}$ bytes = 4GB, $2^{64}$ bytes = 16 exabytes
- A pointer is just another kind of value
  - A basic type in C

```
int *ptr;
```

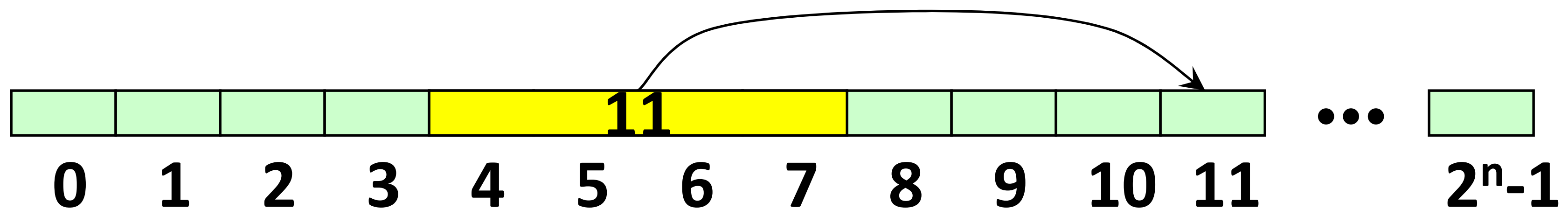
The variable “ptr” stores a pointer to an “int”.



## Recall: Memory Organization

- All modern processors have memories organized as sequence of *numbered bytes*
  - Many (but not all) are linear sequences
- Definitions:–
  - *Byte:* an 8-bit memory cell capable of storing a value in range 0 … 255
  - *Address:* number by which a memory cell is identified

## Definition – *Pointer*



- A *value* indicating the *number* of (the first byte of) a data object
  - Also called an *Address* or a *Location*
- Usually 2, 4, or 8 bytes, depending upon machine architecture

## Pointer Operations in C

- Creation

  & *variable*  Returns variable's memory address

- Dereference

  * *Pointer*  Returns contents stored at address

```
  int A, B;
  int *ptr;
  ptr=&A;   // Creation
  B=*(ptr); //Dereference
```

## Declaring Pointers in *C*

**`int *p;`** //a pointer to an **`int`**

**`double *q;`** // a pointer to a **`double`**

**`char *r;`** // a pointer to **`char`**

- *type* **`*s;`** — a pointer to an object of

## Declaring Pointers in *C* (continued)

- Pointer declarations:–read from *right* to *left*
- `const int *p;`
  - **p** is a pointer to an integer constant
  - I.e., pointer can change, thing it points to cannot

## Declaring Pointers in C (continued)

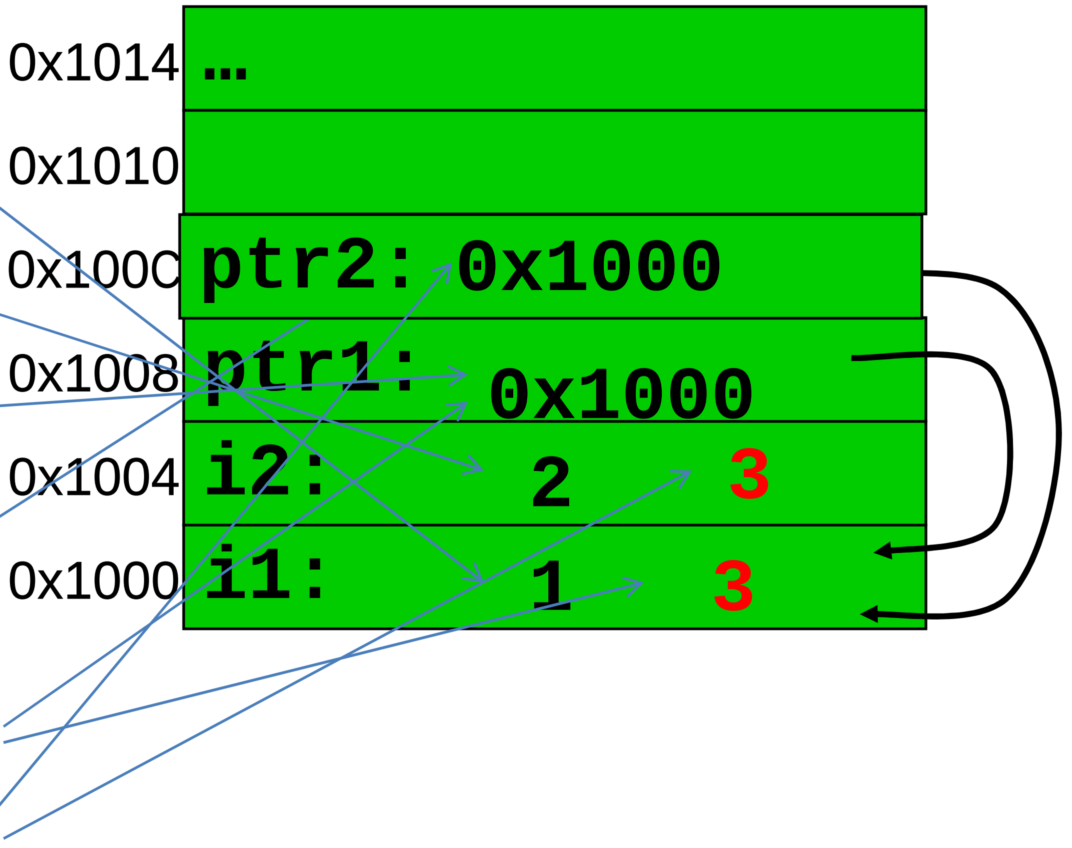
- `int * const q;`
  - **q** is a constant pointer to an integer variable
  - I.e., pointer cannot change, thing it points to can!
- `const int * const r;`
  - **r** is a constant pointer to an integer constant

# Using Pointers

```
int  i1, i2;
int *ptr1,*ptr2;

i1=1;
i2 = 2;
ptr1 = &i1;
ptr2 = ptr1;

*ptr1 = 3;
i2 = *ptr2;
```



# Using Pointers (cont.)

```
int  int1 = 1036; /* some data to point to */
int  int2  = 8;

int *int_ptr1 = &int1;  /* get addr of data  */
int *int_ptr2 = &int2;

*int_ptr1 = int_ptr2;
*int_ptr1 = int2;
```

**What happens?**

**Type check warning: int_ptr2 is not an int**

**int1 becomes 8**

## A Special Pointer in C

- Special constant pointer **`NULL`**
  - Points to no data
  - Dereferencing illegal – causes *segmentation fault*

## Generic Pointers

- **void** *: a "pointer to anything"

```
void    *p;
int      i;
char     c;
p = &i;
p = &c;
putchar(*(char *)p);
```

**type cast: tells the compiler to "change" an object's type (for type checking purposes – does not modify the object in any way)**

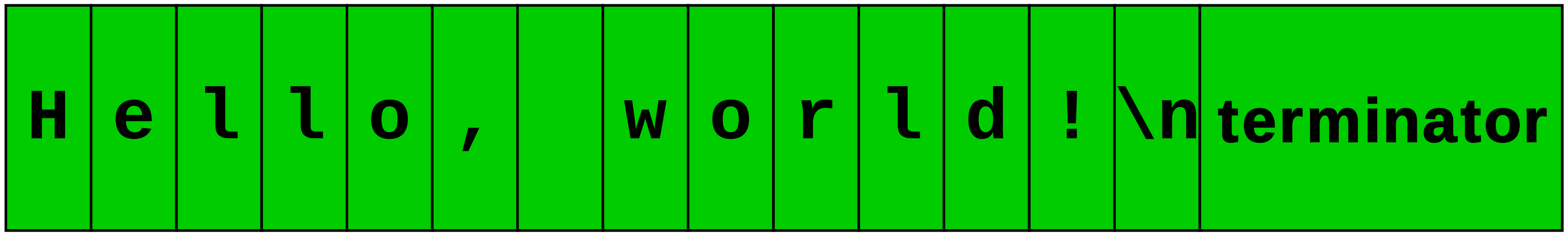
**Dangerous! Sometimes necessary…**

- Lose all information about what type of thing is pointed to
  - Reduces effectiveness of compiler's type-checking
  - Can't use pointer arithmetic

# Strings

- In C, strings are just an array of characters
  - Terminated with ‘\0’ character
  - Arrays for bounded-length strings
  - Pointer for constant strings (or unknown length)

```
char  str1[15] = “Hello, world!\n”;
char *str2     = “Hello, world!\n”;
```



**C terminator: ’\0’**

Thanks