"Should array indices start at 0 or 1? My compromise of 0.5 was rejected without, I thought, proper consideration."
Stan Kelly-Bootle

http://www.devtopics.com/101-great-computer-programming-quotes/

David Notkin ● Autumn 2009 ● CSE303 Lecture 11

Upcoming schedule

Today 10/23	Monday 10/26	Wednesday 10/28	Friday 10/30	Monday 11/2
Finish-up Wednesday Some specifics for HW3 Social implications Friday	Memory management		Midterm review	Midterm

Wednesday: after class question

- Paraphrase: "If I overwrite memory outside the bounds of my process, can I hurt other processes or my computer?"
- No, you can't
- Indeed, although you can do almost anything within your process – and can make your life miserable doing so – Unix keeps everything you do within your own process (well, close enough)
- Indeed, that's why you get a segfault if you access memory outside of your virtual address space
- So, you can destroy your process, but other processes and your computer remain safe

Pointer: a memory address referring to another value

```
type* name;                 // declare
type* name = address;       // declare/initialize

int x = 42;
int* p;
p = &x;        // p stores address of x

printf("x  is %d\n",    x);   // x  is 42
printf("&x is %p\n", &x);     // &x is 0x0022ff8c
printf("p  is %p\n",  p);     // p  is 0x0022ff8c


int* p1, p2;    // int* p1; int p2;
int *p1, p2;    // int* p1; int p2;
int* p1, *p2;   // int* p1; int* p2
int *p1, *p2;   // int* p1; int* p2
```

Dereferencing: access the memory referred to by a pointer

```
*pointer                    // dereference
*pointer = value;           // dereference/assign

int x = 42;
int* p;
p = &x;        // p stores address of x

*p = 99;       // go to the int p refers to; set to 99

printf("x  is %d\n",    x);
```

Output: `x is 99`

* vs. &

- many students get `*` and `&` mixed up
  - `&` references (ampersand gets an address)
  - `*` dereferences (star follows a pointer)

```
int x = 42;
int* y = &x;
printf("x  is %d    \n",  x);   // x  is 42
printf("&x is %p\n", &x);       // &x is 0x0022ff8c
printf("y  is %p\n",  y);       // y  is 0x0022ff8c
printf("*y is %d    \n", *y);   // *y is 42
printf("&y is %p\n", &y);       // &y is 0x0022ff88
```

- What is `*x` ?

L-values and R-values

- L-value: Suitable for being on left-side of an = assignment -- a valid memory address to store into
- R-value: Suitable for right-side of an = assignment

```
int x = 42;
int* p = &x;
```

- L-values : `x` or `*p` (store into `x`), `p` (changes what `p` points to)
  - not `&x, &p, *x, *(*p), *12`
- R-values : `x` or `*p, &x` or `p, &p`
  - not `&(&p), &42`

Pass-by-value: copy parameters' values

- Cannot change the original ("actual") parameter variable

```
int main(void) {
    int a = 42, b = -7;
    swap(a, b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}

void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```

Pass-by-reference: point to parameters

- Can change the actual parameter variable using the "formal"

```
int main(void) {
    int a = 42, b = -7;
    swap(a, b);
    printf("a = %d, b = %d\n", a, b);
    return 0;
}

void swap(int a, int b) {
    int temp = a;
    a = b;
    b = temp;
}
```

#1 to know for HW3: arguments to main

```
#include <stdio.h>
#include <string.h>
int main(int argc, char *argv[]) {
  printf("%s#%d#%d\n",
  argv[1]           // print 1st argument as string
  strlen(argv[1]),  // show it's a string
  atoi(argv[1])+1); // convert it to an int
}
-------------------------------------------------
$ a.out 546
546#3#547
$
```


#2 to know for HW3

- printf("%4d%2d%3.1f%%\n"…)
- Printing fixed-width field
- Printing fixed number of decimal places
- Printing %

#3 to know for HW3

- Input functions to consider (depending on your approach)
  - **`scanf`** – read and convert values
  - **`getchar`** – read next character
- Depending on your approach, you may need to convert among data types [I didn't need any of these, except for one **`atoi`**]
  - **`atoi`** – ascii to int
  - **`sscanf`** – same as **`scanf`**, but from a string instead of a file (stream)
  - **`sprintf`** – same as **`printf`**, but into a string instead of to a file (stream)

#4 to know for HW3

- How to handle reaaallllllllllllyyyyyyyyyyyyyyy long integer entries?

```
2295368686771969123000270782186855260112447232907923 1
3076254225030127069205146053958616692729173275496 1
2992740239799128648962783773417918638518829638222731454
4648472980354018310183016787562378879453344121677932 3
9564780647927552813573378126620390479441956306440723
6449532773188769353973855869106683910338856730044 92
5864556331756430984733447871493906949524320067479 3
4870509135523888277884290923005671214081346015789923332
1545241701177578785195104730956315938884094630980713232
5354288503961524527117435531562370433428477356819923309
```

- Hint: think about the whole problem – not just how to handle input – before you code – it can be easier than you think, unless you don't think about it

Ethics

- Definition?

"the rules of conduct recognized in respect to a particular class of human actions or a particular group, culture, etc."

ethics. Dictionary.com. *The American Heritage® Dictionary of the English Language, Fourth Edition*. Houghton Mifflin Company, 2004. http://dictionary.reference.com/browse/ethics (accessed: October 23, 2009).

Codes of ethics

- http://ethics.iit.edu/ -- just two categories in their lists are
  - Sports and Athletics
    - American Football Coaches Association
    - Australian Sports Commission
    - Canadian Curling Association
    - Canadian Soccer Association
    - Club Cycliste Baconsfield
    - National Association of Sports Officials
    - National Basketball Association
    - United States Olympic Committee
  - Fraternal Social Organizations
    - Brookline Bird Club
    - Gamma Beta Phi
    - National Speleogical Society
    - Universal Autograph Collectors Club

Why have codes of ethics?

- Aren't laws enough?
- Isn't personal commitment enough?
- Aren't company policies enough?
- …

Why have engineering codes of ethics?

- And why have software engineering codes of ethics?

Software Engineering Code of Ethics and Professional Practice [short version]

- In accordance with their commitment to the health, safety, and welfare of the public, software engineers shall adhere to the following eight Principles:

1. Public. Software engineers shall act consistently with the public interest.
2. Client and employer. Software engineers shall act in a manner that is in the best interests of their client and employer, consistent with the public interest.
3. Product. Software engineers shall ensure that their products and related modifications meet the highest professional standards possible.
4. Judgment. Software engineers shall maintain integrity and independence in their professional judgment.
5. Management. Software engineering managers and leaders shall subscribe to and promote an ethical approach to the management of software development and maintenance.
6. Profession. Software engineers shall advance the integrity and reputation of the profession consistent with the public interest.
7. Colleagues. Software engineers shall be fair to and supportive of thei colleagues.
8. Self. Software engineers shall participate in lifelong learning regarding the practice of their profession and shall promote an ethical approach to the practice of the profession.

Questions?