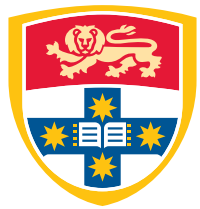
Introduction to Programming (Adv)

School of Computer Science, University of Sydney



Copyright Warning

Lecture 5: More on pointers and Testing

Moving, manipulating and testing memory

Review pointers

Given a memory address to a `char`, return the *value*

```
1 char get_element(char *input) {
2     char value = *input; // or input[0]
3     return value;
4 }
```

Given a memory address to a `char`, return the *address*

```
1 char * get_address(char *input) {
2     char *address = input; // easy
3     return address;
4 }
```

Pointer arithmetic

Given a memory address to an *array of* `char`, return the *value* of the 3rd element

```
1 char get_element(char *input) {
2     char value = input[2];
3     return value;
4 }
```

Given a memory address to an *array of* `char`, return the *address* to the 3rd element

```
1 char * get_address(char *input) {
2     char *address = input + 2; // &(input[2]);
3     return address;
4 }
```

Pointer arithmetic

Given a memory address to `void`, return the value of that memory when represented by the data type `int`.

```
1 int get_value(void *input) {
2     int *intptr = (int*)input;
3     int answer = *intptr;
4     return answer;
5 }
```

Pointer arithmetic (cont.)

Given a memory address to an array of `unsigned short`, return the 4th element

```
1 unsigned short get_address(unsigned short *input) {
2     unsigned short *address = input + 3; // easy...
3     return *address;
4 }
```

`sizeof(short)` is at least 2 and no more than `sizeof(int)`

Pointer arithmetic (cont.)

Given a memory address to `void`, treat it as an array of `unsigned short` and return the 4th element

```
1 unsigned short get_address(void *input) {
2     unsigned short *address = (unsigned short*)input;
3     unsigned short answer = address[3]; // or *(address +
          3)
4     return answer;
5 }
```

Need to perform casting

Pointer arithmetic (cont.)

Given a memory address to `void`, return the address of the beginning of the 4th element, where *every* element is *bw* bytes long.

```
1 void * get_address(void *input, int bw) {
2     char *charptr = (char*)input;
3     charptr += (3 * bw);
4     void *answer = (void*)charptr;
5     return answer;
6 }
```

void is a non-scalar type. Need to do the arithmetic ourselves.

Pointer arithmetic (cont.)

Given a memory address to `void`, return the address of byte offset 492.

```
1 void * get_address(void *input) {
2     char *charptr = (char*)input;
3     charptr += 492;
4     void *answer = (void*)charptr;
5     return answer;
6 }
```

More simply

```
1 void * get_address(void *input) {
2     return (void*) ( ((char *)input) + 492 );
3 }
```

Testing

Unit test

```
1  #include <assert.h>
2  int square(int x) {
3      return x*x;
4  }
5
6  int square_test_1() {
7      int in = 3;
8      int expected = 9;
9      int out = square(in);
10     assert(in == out);
11 }
12
13 int main() {
14     square_test_1();
15     return 0;
16 }
```

Testing

```
1  void square_test_generic(int in, int expected) {
2      int out = square(in);
3      assert(expected == out);
4  }
5
6  int main() {
7      int tests = 3;
8      int test_data[] = {
9          3, 9,
10         -3, 9,
11         0, 0
12     };
13     for (int i = 0; i < tests; ++i)
14     {
15         int *data = test_data + 2*i;
16         square_test_generic(data[0], data[1]);
17     }
18     return 0;
```

Testing (cont.)

```
19 }
```

String functions: Read only

Read operations using idioms: counting, searching...

`size_t strlen(const char *s);`
`strlen()` function calculates the length of the string pointed to by `s`, excluding the terminating null byte (`'\0'`).

`char *strchr(const char *s, int c);`
`strchr()` returns a pointer to the first occurrence of the character c in the string s.

String functions: Read only (cont.)

`int strncmp(const char *s1, const char *s2, size_t n);`
`strcmp()` compares the two strings `s1` and `s2`. It returns an integer less than, equal to, or greater than zero if `s1` is found, respectively, to be less than, to match, or be greater than `s2`. It compares only the first (at most) `n` bytes of `s1` and `s2`.

String functions: Read & write

Read and Write operations using pointer arithmetic

`char *strncpy(char *dest, const char *src, size_t n);`
`strncpy()` copies the string pointed to by `src`, including the terminating null byte (`'\0'`), to the buffer pointed to by `dest`. The strings may not overlap, and the destination string dest must be large enough to receive the copy. At most `n` bytes are copied.

String functions: Read & write (cont.)

`char *strncat(char *dest, const char *src, size_t n);`
`strncat()` appends the `src` string to the `dest` string, over? writing the terminating null byte (`'\0'`) at the end of `dest`, and then adds a terminating null byte. The strings may not overlap, and the dest string must have enough space for the result. At most `n` bytes are copied. If dest is not large enough, program behaviour is unpredictable;

String problems there is no function for

Given a string of words, separated by spaces, find how many words there are.

Which letters, among alphanumeric, have the highest frequency in a string?

Given a string of words, separated by spaces, produce a reverse ordering of the string such that the words appear in reverse order, but are preserved as words.
e.g.

```
char sentence[] = "This is a sunny day";

Input: This is a sunny day
Output: day sunny a is This
```

String problems there is no function for (cont.)

```
1 void mystrncat(char *dst,
2                 const char *src1,
3                 const char *src2, int n)
```

Test mystrncat

```
1   // we do not observe large input
2   char out[2048] = {0}; // initialise memory!
3
4   mystrncat(out, in1, in2, 2048);
5
6   // memory comparison options
7
8   // char by char
9   ...
10
11  // strcmp - dangerous!
12  ...
13
14  // memcmp
15  ...
16
17  return pass/fail
```

Summary

We covered the unit test. All the ingredients required for testing.

How to navigate across memory using pointer arithmetic

Practice with string processing functions and understanding the memory issues