# CS 31 Introduction to CS Discussion 2H

# Pointers

# Pointers

- It is a method of locating variables (data) in your program (in memory)
- Pointers store address in memory
  - not the value

# Declaring a pointer

- It is not the same thing to point to a char and to point to an integer or a float
- Examples:
    - int* number;
    - char* character;
    - float* greatnumber;

# * Means different things

- Similar to arrays
  - [] in variable declaration means size
  - [] in code execution means position in array
- With pointers
  - * in variable declaration means it is a pointer
  - * in code execution means get value by address

# Reference operator

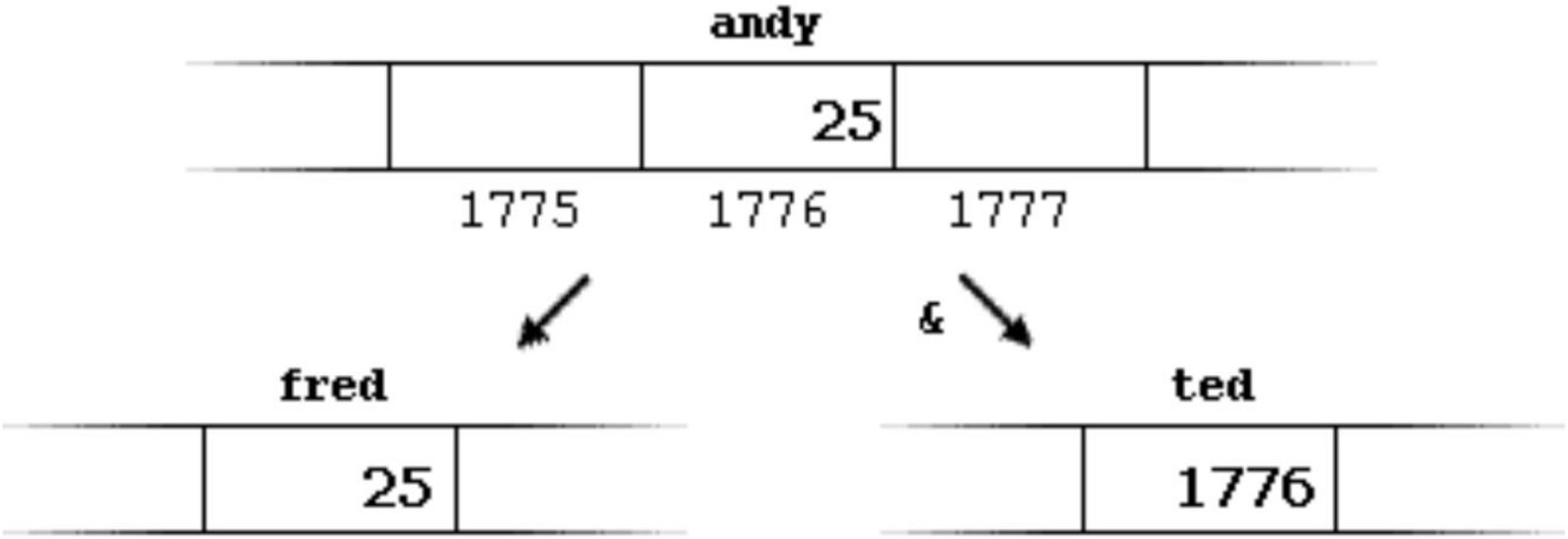
- By reference people understand “by address”
- By address people understand “by reference”
- Operator **&** does
  - Returns the address of a given variable name
  - With every program run this address changes
  - You **cannot** hardcode addresses

# Example

andy = 25;

fred = andy;

ted = &andy;

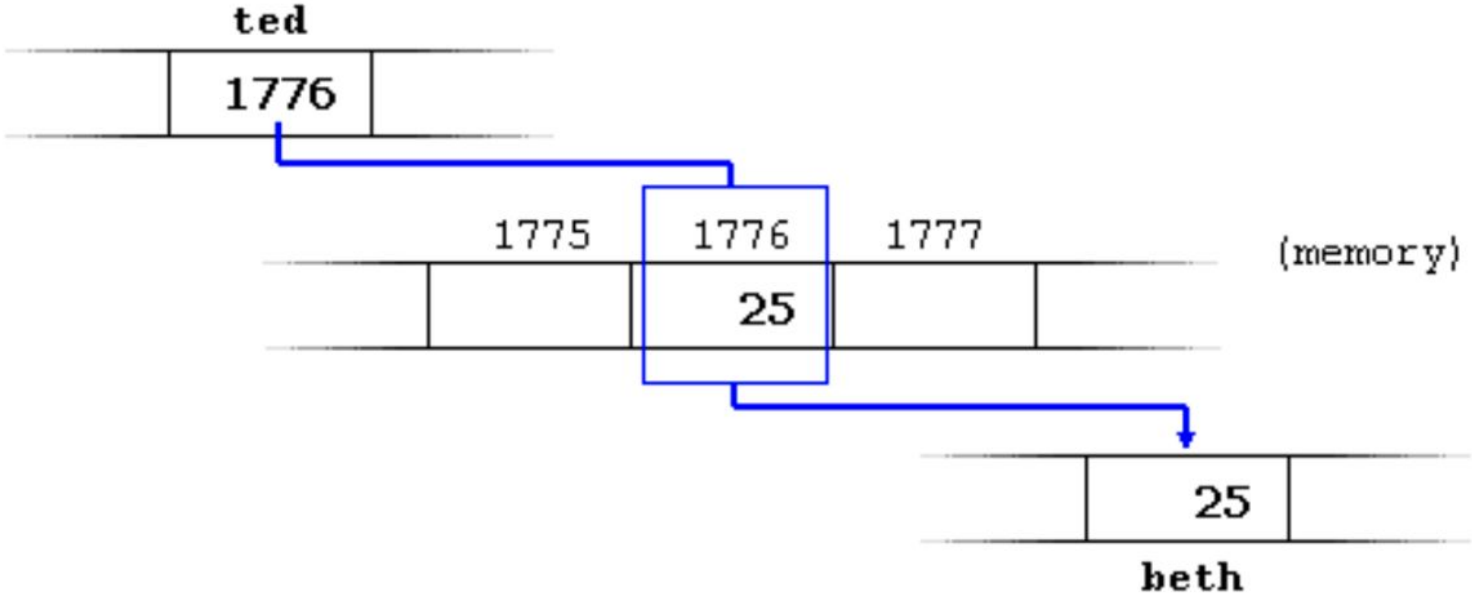


# Dereference operator

- Gets the value of a variable by a given address
- Dangerous operation:
  - Results in many bugs
  - The type of the variable must match the type of the variable at the specified address

# Example

- beth = *ted;



# Summary

- & is the reference operator and can be read as "address of"

- * is the dereference operator and can be read as "value pointed by"

# Time for practising Worksheet

1) What does the following program output?

```
int main() {
      int a = 100, b = 30;
      cout << a + b << endl;          // (1)

      int* ptr = &a;
      cout << *ptr + b << endl;       // (2)

      *ptr = 10;
      cout << *ptr + b << endl;       // (3)

      ptr = &b;
      *ptr = -12;
      cout << *ptr + 2*b << endl;          // (4)

      int c = a + *ptr;
      cout << c << endl;              // (5)

      b = -5;
      cout << a + b << endl;          // (6)

      int arr[5] = {4, 5, 10, 11, -1};
      ptr = arr + 1;
      cout << *arr + *ptr << endl;    // (7)

      int cs;
      int& pic = cs;
      ptr = &pic;
      pic = 31;
      cs++;
      cout << *ptr << endl;           // (8)
```

1) What does the following program output?

```
int main() {
      int a = 100, b = 30;
      cout << a + b << endl;          // (1)

      int* ptr = &a;
      cout << *ptr + b << endl;       // (2)

      *ptr = 10;
      cout << *ptr + b << endl;       // (3)

      ptr = &b;
      *ptr = -12;
      cout << *ptr + 2*b << endl;          // (4)

      int c = a + *ptr;
      cout << c << endl;              // (5)

      b = -5;
      cout << a + b << endl;          // (6)

      int arr[5] = {4, 5, 10, 11, -1};
      ptr = arr + 1;
      cout << *arr + *ptr << endl;    // (7)

      int cs;
      int& pic = cs;
```

```
ptr = &pic;
pic = 31;
cs++;
cout << *ptr << endl;              // (8)
```

(1) 130 (2) 130 (3) 40 (4) -36
(5) -2 (6) 5 (7) 9 (8) 32

2) Write statements that declare a variable of each of the following types:
   - pointer to char
   - array of 10 pointers to char
   - pointer to const int

2) Write statements that declare a variable of each of the following types:
   - pointer to char
   - array of 10 pointers to char
   - pointer to const int

```
char* c_ptr;
char* c_ptr_arr[10];
const int* i_ptr;
```

3) Consider the following function that loops through the characters of a C-string and prints them one by one. What are some possible inputs to the function that could break it?

```
void printChars(const char* str) {
  int i = 0;
  while (*(str + i) != '\0') {
    cout << *(str + i) << endl;
    i++;
  }
}
```

3) Consider the following function that loops through the characters of a C-string and prints them one by one. What are some possible inputs to the function that could break it?

```
void printChars(const char* str) {
  int i = 0;
  while (*(str + i) != '\0') {
    cout << *(str + i) << endl;
    i++;
  }
}
```

An uninitialized pointer, a null pointer, or a pointer to an array of characters containing no zero byte.

4) Write a function that reverses a C string without using the `[]`. It takes in a pointer to the beginning of the array holding the C string.

Function header: `void reverse(char* arr)`

```
char arr[6] = "hello";
reverse(arr);
// now arr should be "olleh"

char arr[5] = "ucla";
reverse(arr);
// now arr should be "alcu"

char arr[6] = "kayak";
reverse(arr);
// now arr should be "kayak"
```

```
void reverse(char *arr) {
    // Initialize a pointer to point to the last char before
\0
    char *first = arr;
    char *second = arr;
    while (*second != '\0')
        second++;
    second--;
    while (second > first) {
        int temp = *first;
        *first = *second;
        *second = temp;
        second--;
        first++;
    }
}
```

5) Write a function with the following header:

```
void descSort(int* nums, int len)
```

Given an unsorted array of integers *nums* of length *len*, this function should sort the elements of nums in descending order. Avoid using square brackets in the *descSort* definition.

Example:

```
int a[10] = {3, 1, 4, 0, -1, 2, 3, 4, 1, 2};
descSort(a, 10);
//a = [4, 4, 3, 3, 2, 2, 1, 1, 0, -1]
```

```
void descSort(int* nums, int len) {
  for (int x = 0; x < len; x++) {
    int max_num = *(nums + x);
    int max_ind = x;
    for (int y = x + 1; y < len; y++) {
      int curr_num = *(nums + y);

      if (curr_num > max_num) {
        max_num = curr_num;
        max_ind = y;
      }
    }
    *(nums + max_ind) = *(nums + x);
    *(nums + x) = max_num;
  }
}
```

6) Write a function minMax that takes in an int array `arr`, size of array `n`, and two int pointers, `min` and `max`. The function should set the min and max pointers to point to the min and max numbers in the array. Every number i in the array is constrained as follows: -1000 < i < 1000. Try to write this function without accessing any element of the array more than once.

Function header:

```
void minMax(int arr[], int n, int*& min, int*& max)

int arr[5] = {0, 5, 7, -10, 2};
int* pmin;
int* pmax;
minMax(arr, 5, pmin, pmax);
// pmin should point to the -10
// pmax should point to the 7
```

```
void minMax(int arr[], int n, int*& min, int*& max) {
      int max_num = -1000;
      int min_num = 1000;
      for (int i = 0; i < n; i++) {
            if (*(arr + i) < min_num) {
                  min_num = *(arr+i);
                  min = arr + i;
            }
            if (*(arr + i) > max_num) {
                  max_num = *(arr + i);
                  max = arr + i;
            }
      }
}
```

9) Write a function, rotate, that takes in an array A of 6 integers and an integer n, and rotates A by n positions to the right. If n is negative, rotate |n| positions to the left. Square brackets "[]" may **not** be used anywhere in your solution. Here's the prototype and some examples:

```
void rotate(int* A, int n);
```

```
A = {1,2,3,4,5,6};
B = {1,2,3,4,5,6};
C = {1,2,3,4,5,6};
```

After calling `rotate(A, 4), A = {3,4,5,6,1,2}`
After calling `rotate(B, -1), B = {2,3,4,5,6,1}`
After calling `rotate(C, 8), C = {5,6,1,2,3,4}`

```
// helper function
void reverseSubArray(int* A, int startIndex, int endIndex) {
  while(startIndex < endIndex) {
    int temp = *(A + startIndex);
    *(A + startIndex) = *(A + endIndex);
    *(A + endIndex) = temp;
    startIndex++;
    endIndex--;
  }
}

void rotate(int* A, int n) {
  if (n < 0) {
    n = n * -1;
    n %= 6;
    n = (6 - n) % 6;
  } else {
    n %= 6;
  }

  reverseSubArray(A, 0, 5);
  reverseSubArray(A, 0, n - 1);
  reverseSubArray(A, n, 5);
}
```

**Question 1:**

**a) What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

int main() {
        char c[] = "CAT";
        char* helper = c;
        int len = strlen(c);

        for (int i = 0; i < len; i++) {
                cout << helper[0] << endl;
                helper++;
        }
}
```

**Question 1:**

**a) What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

int main() {
	char c[] = "CAT";
	char* helper = c;
	int len = strlen(c);

	for (int i = 0; i < len; i++) {
		cout << helper[0] << endl;
		helper++;
	}
}
```

**Solution:**

```
C
A
T
```

**b) Fill in the blanks. The output should be same as Q1a).**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
	char c[] = "CAT";

	for (_____ helper = _____; _______ != '\0'; _____) {
		cout << _______ << endl;
	}
}
```

**b) Fill in the blanks. The output should be same as Q1a).**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
        char c[] = "CAT";

        for (_____ helper = _____; _______ != '\0'; _____) {
                cout << _______ << endl;
        }
}
```

**Solution:**

```
int main() {
        char c[] = "CAT";

        for (char* helper = c; *helper != '\0'; helper++) {
                cout << helper[0] << endl;
        }
}
```

**Question 2: What will the output of this function be?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
	double d[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };
	double* ptr = d;

	cout << *ptr << endl;
	cout << *(ptr + 1) << endl;
	cout << ptr[2] << endl;

	ptr += 2;
	cout << ptr[2] << endl;
}
```

**Question 2: What will the output of this function be?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
	double d[] = { 1.1, 2.2, 3.3, 4.4, 5.5 };
	double* ptr = d;

	cout << *ptr << endl;
	cout << *(ptr + 1) << endl;
	cout << ptr[2] << endl;

	ptr += 2;
	cout << ptr[2] << endl;
}
```

**Solution:**

```
1.1
2.2
3.3
5.5
```

**Question 3: What will the output of this function be?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
	int i[] = { 1, 2, 3, 4 };
	int* p1 = i;
	int* p2 = (i + 2);

	cout << (p1 - p2) << endl;
	cout << (p2 - p1) << endl;
}
```

**Question 3: What will the output of this function be?**

```
#include <iostream>
#include <string>
using namespace std;

int main() {
	int i[] = { 1, 2, 3, 4 };
	int* p1 = i;
	int* p2 = (i + 2);

	cout << (p1 - p2) << endl;
	cout << (p2 - p1) << endl;
}
```

**Solution:**

```
-2
2
```

**Question 4: What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

int main() {
	char c[] = "CATDOG";
	char* cPtr = c;
	int len = strlen(c);

	for (int i = 0; i < len; i++) {
		if (cPtr < &c[3]) {
			cout << *cPtr << endl;
			cPtr++;
		}
	}
}
```

**Question 4: What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

int main() {
	char c[] = "CATDOG";
	char* cPtr = c;
	int len = strlen(c);

	for (int i = 0; i < len; i++) {
		if (cPtr < &c[3]) {
			cout << *cPtr << endl;
			cPtr++;
		}
	}
}
```

**Solution:**

```
C
A
T
```

**Question 5: What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

int main() {
	int i[] = { 5, 6, 7, 8, 9 };
	int* iPtr = i;
	int len = 3;

	for (int j = 0; j < len; j++) {
		if (iPtr < &i[3]) {
			cout << (&i[3] - iPtr) << endl;
			iPtr++;
		}
	}
}
```

**Question 5: What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

int main() {
	int i[] = { 5, 6, 7, 8, 9 };
	int* iPtr = i;
	int len = 3;

	for (int j = 0; j < len; j++) {
		if (iPtr < &i[3]) {
			cout << (&i[3] - iPtr) << endl;
			iPtr++;
		}
	}
}
```

**Solution:**

```
3
2
1
```

**Question 6: What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

char* secretDecoder(char c[], int i[], int n) {
	for (int j = 0; j < n; j++) {
		c[j] = c[*(i + j)];
	}
	c[n] = '\0';
	return c;
}

int main() {
	char c[] = "disproportionate";
	int i[] = { 3, 5, 1, 0 };
	char* d = secretDecoder(c, i, 4);
	cout << d << endl;
}
```

**Question 6: What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cctype>
using namespace std;

char* secretDecoder(char c[], int i[], int n) {
	for (int j = 0; j < n; j++) {
		c[j] = c[*(i + j)];
	}
	c[n] = '\0';
	return c;
}

int main() {
	char c[] = "disproportionate";
	int i[] = { 3, 5, 1, 0 };
	char* d = secretDecoder(c, i, 4);
	cout << d << endl;
}
```

**Solution:**

The output is poop.

The word goes through the following transformations in the function secretDecoder
disproportionate -> pisproportionate -> posproportionate -> pooproportionate ->
pooproportionate -> poop

**Question 7: What will the output of this function be?**

```
#include <iostream>
using namespace std;

int f(int x, int *py, int **ppz)
{
	int y, z;
	**ppz += 1;
	z = **ppz;
	*py += 2;
	y = *py;
	x += 3;
	return x + y + z;
}

int main()
{
	int c, *b, **a;
	c = 4;
	b = &c;
	a = &b;
	cout << f(c, b, a) << endl;
}
```

**Question 7: What will the output of this function be?**

```
#include <iostream>
using namespace std;

int f(int x, int *py, int **ppz)
{
	int y, z;
	**ppz += 1;
	z = **ppz;
	*py += 2;
	y = *py;
	x += 3;
	return x + y + z;
}

int main()
{
	int c, *b, **a;
	c = 4;
	b = &c;
	a = &b;
	cout << f(c, b, a) << endl;
}
```

**Solution:**

The output is 19.

**Question 8: What will the output of this function be?**

```
#include <iostream>
using namespace std;

int x;
void Q(int z)
{
	z += x;
	cout << z << endl;
}
void P(int *y)
{
	int x = *y + 2;
	Q(x);
	*y = x - 1;
	cout << x << endl;
}
int main()
{
	x = 5;
	P(&x);
	cout << x << endl;
}
```

**Question 8: What will the output of this function be?**

```
#include <iostream>
using namespace std;

int x;
void Q(int z)
{
	z += x;
	cout << z << endl;
}
void P(int *y)
{
	int x = *y + 2;
	Q(x);
	*y = x - 1;
	cout << x << endl;
}
int main()
{
	x = 5;
	P(&x);
	cout << x << endl;
}
```

**Solution:**

The output is 12 7 6.

**Question 9A: What will the output of this function be?**

```
#include<iostream>
using namespace std;

void mystery(int *ptra, int *ptrb)
{
	int *temp;
	temp = ptrb;
	ptrb = ptra;
	ptra = temp;
}

int main()
{
	int a = 2016, b = 0, c = 4, d = 42;
	mystery(&a, &b);
	if (a < c)
		mystery(&c, &a);
	mystery(&a, &d);
	cout << a << endl;
}
```

**Question 9A: What will the output of this function be?**

```
#include<iostream>
using namespace std;

void mystery(int *ptra, int *ptrb)
{
	int *temp;
	temp = ptrb;
	ptrb = ptra;
	ptra = temp;
}

int main()
{
	int a = 2016, b = 0, c = 4, d = 42;
	mystery(&a, &b);
	if (a < c)
		mystery(&c, &a);
	mystery(&a, &d);
	cout << a << endl;
}
```

**Solution:**

The output is 2016.

Note that a and d are not swapped as the function mystery() doesn't change values, but pointers which are local to the function.

**Question 10: What will the output of this function be?**

```
#include <iostream>
#include <string>
#include <cstring>
using namespace std;

struct Computer {
	char model[5];
	int processors;
	double processorSpeed;
};

int main() {
	Computer myAncientMac;
	cout << myAncientMac.model << endl;
	cout << myAncientMac.processors << endl;
	cout << myAncientMac.processorSpeed << endl;

	strcpy(myAncientMac.model, "C00L");
	myAncientMac.processors = 4;
	myAncientMac.processorSpeed = 2.6;
	cout << myAncientMac.model << endl;
	cout << myAncientMac.processors << endl;
	cout << myAncientMac.processorSpeed << endl;
}
```

**Question 11: What will the output of this function be?**

```
#include <iostream>
#include <cstring>
#include <string>
using namespace std;

struct Stuff {
	int x;
	string s;
};

int main() {
	Stuff s;
	s.x = 5;
	s.s = "CS31";
	Stuff* ptr = &s;
	cout << ptr->x << endl;
	cout << ptr->s << endl;
}
```

**Question 12: What will the output of this function be?**

```
#include <iostream>
#include <cstring>
#include <string>
using namespace std;

struct Stuff {
	int x;
	char c[6];
};

int main() {
	Stuff s[5];
	for (int i = 0; i < 5; i++) {
		s[i].x = 1;
		strcpy(s[i].c, "Hello");
	}
	cout << s[0].c << endl;
	cout << s[2].c << endl;
	cout << s[2].c[4] << endl;
}
```

**Question 13: What will the output of this function be?**

```
#include <iostream>
#include <cstring>
#include <string>
using namespace std;

struct stuff {
	int x;
	char c[6];
};

int main() {
	stuff s[3];
	for (int i = 0; i < 3; i++) {
		s[i].x = 1;
		strcpy(s[i].c, "hello");
	}
	stuff* ptr = s;
	strcpy((ptr + 1)->c, "hey!");

	for (int i = 0; i < 3; i++) {
		cout << (ptr + i)->c << endl;
	}
}
```

## Question 14: What will the output of this function be?

```
#include <iostream>
#include <cstring>
#include <string>
using namespace std;

struct Stuff {
	int x;
	char c[6];
};

int main() {
	Stuff s[3];
	for (int i = 0; i < 3; i++) {
		s[i].x = 1;
		strcpy(s[i].c, "hello");
	}
	Stuff* ptr = s;
	cout << *(ptr+1)->c << endl;
	cout << s[1].c << endl;
}
```

# Thank You!