Arrays, Pointers, and Struct

Yaping Jing

CS270 – Computer Science II

Pass by Reference

```
void getOrder(int &donuts){
    cout << "How many doughnuts do you want? ";
    cin >> donuts;
}

int jellyDonuts;
getOrder(jellyDonuts);
```

How Arrays Are Passed

```
int sumup(int tmp[], int size){
   int sum = 0;
   for (int i = 0; i<size; i++){
       sum+= tmp[i];
   }
   return sum;
}

int foo[5] = {53, 85, 100, 102, 103};
cout << sumup(foo, 5) << endl;
```

Pointers

```
int x = 25;

int *intptr = &x;

cout << *intptr << endl;

cout << intptr << endl;
```

Arrays and Pointers

int foo[8] = {92, 85, 75, 88, 79 };

Arrays and Pointers

int foo[8] = {92, 85, 75, 88, 79 };

int *bar = foo; // bar points to foo[0]

Arrays and Pointers

```
int foo[8] = {92,  85,   75,   88,   79 };
int *bar = foo;      // bar points to foo[0]
cout << *bar;        // print out foo[0]
cout << bar[0];      // print out foo[0]
```

Arrays and Pointers

```
int foo[8] = {92,  85,   75,   88,   79 };
int *bar = foo;      // bar points to foo[0]
cout << *bar;      // print out foo[0]
cout << bar[0];      // print out foo[0]
cout << foo[1];
```

Arrays and Pointers

```
int foo[8] = {92,  85,   75,   88,   79 };
int *bar = foo;      // bar points to foo[0]
cout << *bar;      // print out foo[0]
cout << bar[0];      // print out foo[0]
cout << foo[1];
cout << *(bar+2);      // print out foo[2]
```

Arrays and Pointers

```
int foo[8] = {92,  85,   75,   88,   79 };
int *bar = foo;      // bar points to foo[0]
cout << *bar;      // print out foo[0]
cout << bar[0];      // print out foo[0]
cout << foo[1];
cout << *(bar+2);      // print out foo[2]
cout << foo      // print out the beginning address of array foo
```

Pointer Arithmetic

```
int vals[] = {5, 6, 11};

int *valPtr  = vals;

valPtr++; // address in valPtr + (1* size of an int)
valPtr--; //points at 5
cout <<*(valPtr+2); // 11
```

Pointer Initialization

```
int x = 5;
int *xPtr = &x;

int vals[] = {5, 6, 11};
int *valPtr  = vals;

int *yPtr = NULL;

if(!yPtr){
   //
}
```

Comparing Pointers

```
int x = 5;
int *ptr1 = &x;

int y = 6;
int *ptr2 = &y;

if(ptr1 == ptr2){...} //compare address

if(*ptr1 == *ptr2){...} // compare contents
```

Pointers as Function Parameters

```
void getNum(int *ptr); // works like reference variable

void getNum(int *ptr){
  cin >>*ptr;
}

// What does this function call mean?
int num = 10;
getNum(&num);
```

Pointers as Function Parameters

```
void swap(int &x, int &y){
  int temp = x;
  x = y;
  y = temp;
}

void swap(int *x, int *y){
  int temp = *x;
  *x = *y;
  *y = temp;
}

int n1 = 2, n2 = 3;
swap(&n1, &n2);
```

Dynamic Memory Operators

new

delete

Dynamic Memory Operators

```
const int SIZE = 30;
int *x = new int[SIZE];

for(int i = 0; i<SIZE; i++){
  x[i] = 0;
}

delete [] x;
```

Dynamic Memory Operators

```
const int SIZE = 30;
double *x = new double[SIZE];

for(int i = 0; i<SIZE; i++){
  x[i] = 0.0;
}
delete [] x;
```

Dynamic Memory Operators

```
int *x = new int;

*x = 10;

delete x;


double *y = new double;
*y = 3.2;
delete y;
```

Dynamic Array Example

```
char* buffer = new char[500];

buffer[3] = 'a';

delete [ ] buffer;
```

Dynamic Variable Example

```
struct Passenger = {
    string      name;
    MealType    mealPref;
    bool        isItalianCuisian;
    string      mealNo;
};


  Passenger *p;

  p = new Passenger;
```

Accessing Pointer Member Variables

```
struct Passenger  {
    string      name;
    MealType    mealPref;
    bool        isItalianCuisian;
    string      mealNo;
};

  Passenger *p;

  p = new Passenger;
  p->name      = "Peter";
  p->mealPref = REGULAR;
  p->isItalianCuisian = true;
  p->mealNo = "1235";
```

Returning Pointers from Functions

```
int* getNumbers(int size){
  int *arr = NULL;
  if(size <=0){
    return NULL;
  }
  arr = new int[num];
  for (int i = 0; i<size; i++){
    arr[i] = 0;
  }
  return arr;
}
```