

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

Pointers and functions, arrays of pointers, pointers to classes

Comp Sci 1575 Data Structures



Computer Science

Pointers and
functionsPointers as
parameters to
functions

Pointers and const

Void pointers

Pointers returned
from functionsPointers to functions
themselvesArrays of
pointersPointers to
structsPointers to
classes

- First assignment is posted. If you have questions after trying it out this weekend, please come to office hours early this week (office hours on the site).

Pointers about arrays: arrays are important

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

Q: Why did the programmer get fired from his job?

Pointers about arrays: arrays are important

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

Q: Why did the programmer get fired from his job?

A: Because he didn't get arrays.

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

1 Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

2 Arrays of pointers

3 Pointers to structs

4 Pointers to classes

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

- ① **Pointers and functions**
 - Pointers as parameters to functions
 - Pointers and const
 - Void pointers
 - Pointers returned from functions
 - Pointers to functions themselves

- ② Arrays of pointers

- ③ Pointers to structs

- ④ Pointers to classes

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

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

```
void setValto50(long *par){
    *par = 50;
    return;
}
```

```
int main(){
    long val;
    setValto50(&val);
    cout << val << endl; // 50
```

```
    long intArr[5] = {1000, 2, 3, 17, 50};
    setValto50(intArr); // can pass array too!
    return 0;
} // What is value of intArr now?
```

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

```

int x;
const int z = 10;
int *p = &z; // fails
const int *pz = &z; // succeeds
int y = 10;
const int *p = &y; // succeeds

x = *p; // gets contents of p
// *p = x; // can't modify contents of p

// ++(*p); // ?
++p; // ?
  
```

- Function receiving a pointer to a non-const as a parameter can modify the value passed as argument
- Function that takes a pointer to a const as parameter can't modify the value

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

```
int x;
```

```
    int *p1 = &x; // non-const ptr to non-const int
```

```
const int *p2 = &x; // non-const ptr to const int
```

```
int const *p2 = &x; // non-const ptr to const int
```

```
    int *const p3 = &x; // const ptr to non-const int
```

```
const int *const p4 = &x; // const ptr to const int
```

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

- Void pointers point to a value that has no type, with undetermined length and undetermined dereferencing
- Can point to any data type (int, float, char, etc)
- Data pointed to can't be directly dereferenced
- Void pointer needs to be transformed into some other pointer type that points to a concrete data type before being dereferenced

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

Pass generic parameters to a function

```

#include <iostream>
using namespace std;

void nextElem(void *data, int psize){
    if( psize == sizeof(char) )
        { char *pchar; pchar=(char*)data; ++(*pchar); }
    else if( psize == sizeof(int) )
        { int *pint; pint=(int*)data; ++(*pint); }
    return ;
}

int main(){
    char a = 'x';
    int b = 1602;
    nextElem(&a, sizeof(a));
    nextElem(&b, sizeof(b));
    cout << a << ", " << b << '\n'; // y, 1603
    return 0;
}

```

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

① Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

② Arrays of pointers

③ Pointers to structs

④ Pointers to classes

Pointers returned from functions

- C++ does not allow returning an entire array as an argument to a function.
- Can return a pointer to an array by specifying the array's name without an index.

```
int * myFunction(int myArray []) {
    return myArray;
}
```

```
int main() {
    int myArray [4];
    int *p;
    p = myFunction(myArray); // no &

    // p can now be used like the array
}
```

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

Pointers returned from functions

- C++ does not allow returning an entire array as an argument to a function.
- Can return a pointer to an array by specifying the array's name without an index.

```
int * myFunction(int *myArray){
    return myArray;
}
```

```
int main(){
    int myArray [4];
    int *p;
    p = myFunction(myArray); // no &

    // p can now be used like the array
}
```

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

```

int * getRandom(){
    srand((unsigned)time(NULL)); // set seed
    static int r[10];
    for (int i = 0; i < 10; ++i){
        r[i] = rand();
        cout << r[i] << endl;
    }
    return r;
}

int main(){
    int *p;
    p = getRandom();
    for( int i = 0; i < 10; i++ ){
        cout << "(p+" << i << "):" << *(p+i) << endl;
    }
    return 0;
}

```

Remember: array indexing `p[i]` also works

- Do not create a stack variable in a local scope and return a pointer to it in the global scope! The local stack variables are “destroyed” after the scope finishes. Heap enables this - more later!

```
int * myFunction ()
{
    int array [2] {1, 2};
    int *pArray = array;
    return pArray;
}
```

```
int main ()
{
    int *p;
    p = myFunction ();
}
```


Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

① Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

② Arrays of pointers

③ Pointers to structs

④ Pointers to classes

Pointers to functions themselves

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

Often for passing a function as an argument to another function

```

int addition(int a, int b) {return (a+b);}
int subtraction(int a, int b) {return (a-b);}
int oper(int x,int y,int (*funct)(int ,int)){
    int g;
    g = (*funct)(x,y);
    return (g);
}

int main(){
    int m,n;
    m = oper(7, 5, addition);
    int (*minus)(int ,int) = subtraction; //alias
    n = oper(20, m, minus);
    cout <<n;
    return 0;
} // *addition or addition work above, why?
  
```

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

- ① Pointers and functions
 - Pointers as parameters to functions
 - Pointers and const
 - Void pointers
 - Pointers returned from functions
 - Pointers to functions themselves

- ② Arrays of pointers

- ③ Pointers to structs

- ④ Pointers to classes

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

```
// array of NUM pointers to int
```

```
int *ptr [NUM];
```

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

```

#include <iostream>
using namespace std;
const int NUM = 3;

int main(){
    int var[NUM] = {10, 100, 200};
    int *ptr[NUM];
    for(int i = 0; i < NUM; i++){
        ptr[i] = &var[i];           // ??
    }
    for(int i = 0; i < NUM; i++){
        cout << "ptr[" << i << "]=";
        cout << *ptr[i] << ", "; // two deref??
    }
    return 0;
}
// out: var[0]=10, var[1]=100, var[2]=200
  
```

NOT like `ptr = var;` where both will have the same address.

Name of variable	Storage address	Value
<code>var[0]</code>	0x7ffcb158c140	10
<code>var[1]</code>	0x7ffcb158c144	100
<code>var[1]</code>	0x7ffcb158c148	200
	0x7ffcb158c14c	
<code>ptr[0]</code>	0x7ffcb158c150	0x7ffcb158c140
<code>ptr[1]</code>	0x...	0x7ffcb158c144
<code>ptr[2]</code>	0x...	0x7ffcb158c148
	0x...	
<code>var</code>	0x...	0x7ffcb158c140
<code>ptr</code>	0x...	0x7ffcb158c150

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

- ① Pointers and functions
 - Pointers as parameters to functions
 - Pointers and const
 - Void pointers
 - Pointers returned from functions
 - Pointers to functions themselves

- ② Arrays of pointers

- ③ Pointers to structs

- ④ Pointers to classes

Pointers to structs: arrow operator (`->`)

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

```
struct person{
    string name;
    int age;
};
```

```
person aPerson;
person *pPerson;
pPerson = &aPerson;
pPerson->age = 23;
cout << pPerson->age << endl; // out: 23
/*pPerson.age = 25; // won't work, op. order
(*pPerson).age = 25; //works
cout << pPerson->age << endl; // out: 25
```

```
person personArray[4]; // array of people
personArray[2].age = 22; // works
cout << personArray[2].age << endl; // out: 22
```


Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

- 1 Pointers and functions
 - Pointers as parameters to functions
 - Pointers and const
 - Void pointers
 - Pointers returned from functions
 - Pointers to functions themselves
- 2 Arrays of pointers
- 3 Pointers to structs
- 4 Pointers to classes

Pointers and functions

Pointers as parameters to functions

Pointers and const

Void pointers

Pointers returned from functions

Pointers to functions themselves

Arrays of pointers

Pointers to structs

Pointers to classes

```
#include <iostream>
```

```
class Rectangle{
    int width, height;
public:
    Rectangle(int x, int y): width(x), height(y){}
    Rectangle(){ width=5; height=4;}
    int area(void) {return width * height;}
};
```

```
int main(){
    Rectangle aRect(3, 4);
    Rectangle *pRect = &aRect;
    std::cout << pRect->area() << endl; //12

    Rectangle arrRect[4];
    std::cout << arrRect[2].area() << endl; //20
```