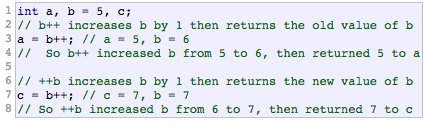
* Try not to use global variable (variable declared outside of the )
* Void swap (int&a, int &b)
* {
* int temp=a;
* a=b;
* b=temp;
* }
* &x: generate a pointer ti x, address of x
* \*p : follow the pointer p, the objct p points to, deference p
* when put;
* isdigit(k), the compiler doesn’t recongnize the function isdigit.
* If a pointer point to the array, if add one on the value of the pointer, it means point to the next element.
* &a[i]+j=&a[i+j]------it also works for subtraction.
* That’s how dp++ works
* When dp=&a[0]
* \*p=3.5, not storing 3.5 in p, but look at p and follow the pointer and put the value in.
* #include <iostream>
* #include <string>
* using namespace std;
* int main() {
* //0 1 234 56 7 89....
* string str = "HELLO WORLD!";
* string result;
* for (size\_t k = 0; k != str.size(); k++)
* result += tolower(str[k]);
* cout << result << endl;
* cout << result.substr(0,5) << endl;
* cout << result.substr(6,6) << endl; }
* function overloading…
* you can do this: int x[] = {1,2,3,4,5};
* #include <iostream>
* using namespace std;
* int main()
* {
* const int SIZE = 5; --if it’s not const, then it doesn’t work
* int sum=0;
* int x[SIZE] = {1,2,3,4,5}; // ??? int sum = 0;
* for(int i = 0 ; i < 5 ; i++)
* sum += x[ i ];
* cout << "sum of 1 to 5 is: " << sum << endl;
* return 0;
* }
* #include <iostream>
* using namespace std;
* int main() {
* int sampleArray[ 10 ];
* int index;

* for(int index = 1; index < 10; index++ )
* sampleArray[index] = 3\*index;




* cout << sampleArray[index] << endl; }----index is not initialized here.
* #include <iostream>
* #include <cmath>
* #include <math.h>
* using namespace std;
* int main ()
* {
* int arr[4]={1,2,2,3};
* int x=sizeof(arr);
* cout << x <<endl;
* }
* it would return 16, because each is 4 bits, and together it is 16.
* #include <iostream>
* #include <cmath>
* #include <math.h>
* using namespace std;
* int main ()
* {
* char year[]="2013";
* int y1=atoi(year);
* cout << y1<<endl;
* }
* if (str[]==”/0”)
* over running array is the number one cause of malware.
* Therefore, you wanna check if the array is too big:
* Say int score[10] creates an array for customizeation.
* Then if user puts 100, then it’s bad.
* We should do “int max\_number\_of\_scores”
* The name of the array is the first element of the array.
* If “p” is a character pointer then “p++” will increment “p” by one byte (typically), if “p” were an integer pointer its value on “p++” would be incremented by 2 bytes (typically).
* “int a[10];” “int\* p;” then the assignment “p = a;” is perfectly valid
* Also “\*(a+4)” and “a[4]” are equivalent as are “\*(p+4)” and “p[4]” .
* The only difference between the two is that we can change the value of “p” to any integer variable address whereas “a” will always point to the integer array of length 10 defined.
* Innitliiza a pointer: int \*p1=0;
* After initialize it, one should first assign an address first.
* char\* a = “Hello”;
* a -> gives address of ‘H’
* \*a -> gives ‘H’
* a[0] -> gives ‘H’
* a++ -> gives address of ‘e’
* \*a++ -> gives ‘e’
* a = &b; where b is another char variable is perfectly LEGAL. However “char a[100];” “a =&b;” where b is another char variable is ILLEGAL.
* 
* Ofstream fout (“fileName”);
* Fout <<
* Fout.close();
* Ifstream fin(“fileName”);
* While (infile.get(ch))
* Char c;
* Infile.get(c);

cons tint MAX=200;

char line[MAX];

infile.getline(line,MAX);

line[0]==’D’;

line[1]==’O’;

if(line[i]==’\0’)

if(line[i]<lineLength)

dynamic allocated array

two position

one for the input file, the other one for the output file.

If use the int \*ptr=new int;

Remember to do:

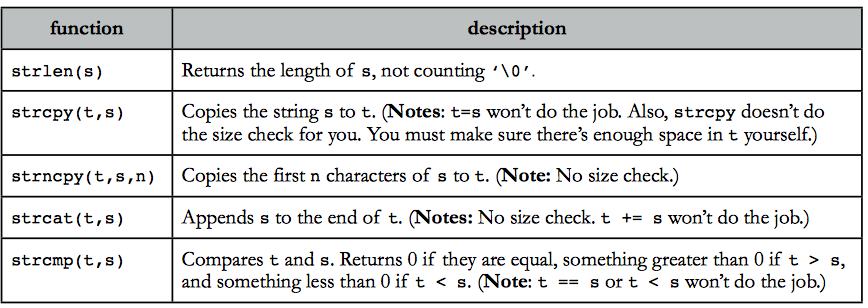
Delete ptr --deallocating the memory

Delete[] ptr; // if the pointer points to an array.

afterwards.

If ptr is a pointer for an array,

Ptr-=2 goes back to two spots in the array.

* For (; ;)
* Fruits[0][i]=tolower(fruits[0][i])
* Change every character in “Apple” to lower case.
* Investigate the property of arrays:
* int main ()
* {
* int str[3]={11,12,13};
* cout << str <<endl;
* cout << &str[0] << " " << &str[1] <<endl;
* cout << \* str << endl;
* cout << str[0] << " " << str[1] <<endl;
* }
* \*(str+1) –one more unit of that data type.
* \*str[0]—doesn’t work so well in the above case.
* int variable=10;
* int \*pointer1=&variable;
* int & reference1=variable;
* cout << reference1 <<endl;
* #include <iostream>
* using namespace std;
* int main() {
* const int SIZE = 3;
* int a[SIZE][SIZE] = { {1,2,3}, {4,5,6}, {7,8,9} };
* for(int i = 0 ; i < SIZE ; i++ ) {
* for(int j = 0 ; j < SIZE ; j++ ) cout << a[ i ][ j ];
* cout << endl; }
* return 0; }
* void swap(int x[],int N) {
* int temp;  
  for(int i = 0; i < N/2 ; i++ ) {
* temp = x[ i ];  
  x[ i ] = x[ N-i-1 ]; x[ N-i-1 ] = temp;
* // cout << "Swapping between " << i << " and " << N-i-1 << endl; }
* }
* #include <iostream>
* #include <cstring>
* using namespace std; int main()
* {
* char str1[14] = "CS31"; char str2[14] = "CS32";
* cout << "str1 has length: " << strlen(str1) << endl; cout << "str2 has length: " << strlen(str2) << endl;
* strcat(str1,str2);
* cout << str1 << endl << str2 << endl;
* strcpy(str1,str2);
* cout << str1 << endl << str2 << endl;
* if(strcmp(str1,str2) == 0)
* cout << "string 1 and string 2 are equal!\n";
* else
* cout << "string 1 and string 2 are not equal!\n";
* return 0; }
* result:
* **str1 has length: 4**
* **str2 has length: 4**
* **CS31CS32**
* **CS32**
* **CS32**
* **CS32**
* **string 1 and string 2 are equal!**
* #include <iostream>
* using namespace std; int mystrlen(char \*p) {
* int len=0;  
  while(\*p++ != '\0') len++; return len;
* }  
  int main() {
* char str1[] = "CS31";  
  char str2[] = "Pointers are very powerful!"; cout << mystrlen(str1) << endl;  
  cout << mystrlen(str2) << endl;
* }
* / more pointers
* #include <iostream>
* using namespace std;
* int main ()
* {
* int numbers[5];
* int \* p;
* p = numbers; \*p = 10;
* p++; \*p = 20;
* p = &numbers[2]; \*p = 30;
* p = numbers + 3; \*p = 40;
* p = numbers; \*(p+4) = 50;
* for (int n=0; n<5; n++)
* cout << numbers[n] << ", ";
* return 0;
* }
* result: 10, 20, 30, 40, 50,
* // my first pointer
* #include <iostream>
* using namespace std;
* int main ()
* {
* int firstvalue, secondvalue;
* int \* mypointer;
* mypointer = &firstvalue;
* \*mypointer = 10;
* mypointer = &secondvalue;
* \*mypointer = 20;
* cout << "firstvalue is " << firstvalue << '\n';
* cout << "secondvalue is " << secondvalue << '\n';
* return 0;
* }
* \*\*\*conversion from string literal to char array will be deprecated. \*\*\*
* if (!s.empty()) cout << “not empty”;
* s.substr(i, n) takes a substring of length n, starting from the i-th character
* s.replace(i, n, s2)--replaces a substring of length n starting at i with another string s2, and sets s with a new string
* int x = ‘3’ - ‘0’; x=3;
* for comparison of strings, you need to declare them and then compare:
* string str1="abc";
* string str2="bcd";
* A string comparison is done based on the lexicographical order, i.e. the order in which they may appear in the dictionary. Exceptions are that uppercase letters are ALWAYS less than lowercase letters are ALWAYS greater than uppercase letters. For instance, a is greater than A, B, ..., Z, and Z is less than a, b, ..., z. Digits are less than alphabets, and ties are broken by letters followed by them.
* How to declare an array
* int array[5];
* How to initialize an array?
* int array[5] = {1, 2, 3, 4, 5}; or int array[] = {1, 2, 3, 4, 5};
* char s[10] = "HOWAREYOU"; --a c-string 
* /\* strcpy example \*/
* #include <stdio.h>
* #include <string.h>
* int main ()
* {
* char str1[]="Sample string";
* char str2[40];
* char str3[40];
* strcpy (str2,str1);
* strcpy (str3,"copy successful");
* }
* const double pi=3.14….
* \*pointer=pi ---doesn’t work because the pointer and pi should be the same type and therefore the pointer should be constant.
* The following sets the value of p so that p points to the variable v:
* p = &v;
* \*p produces the variable pointed to by p, so after the previous assignment, \*p and v refer to the same variable.
* cin >> \*p1;
* \*p1 = \*p1 + 7;
* cout << \*p1;
* The new operator produces a new, nameless variable and returns a pointer that points to this new variable. (as long as \*p1 is declared)
* **You can use this to declare pointers and give them values as well:**
* int \*p1, \*p2;
* p1=new int;
* \*p1=9;
* the following program, why is p1 not NULL?
* int main ()
* {
* int \*p1;
* int \*p2;
* if (p1==NULL)
* cout << "yolo";


* }
* attempt move
* {
* if (dir==0)
* ……
* if(r-1<1 ) …… return false
* else if(a, getCellstatus==true)
* return false;
* forcemove==> attemptmove(m\_arena, dir, m\_row, mcol)
* {
* }
* zurt
* private :
* m\_health
* if(m\_zurts[i]🡪isdead)
* {
* m\_nzurts--;
* delete m\_zurst[i];
* }
* for (int i=0; i<m\_nzurts; i+1)
* {
* if (willfollow && m\_zurts[i]🡪color()==color)
* m\_zurts[i]🡪 forceMove();
* else
* m\_zurts[i]🡪move
* if
* (m\_zurts[i]🡪 isdead ())
* {
* delete m\_zurts[i];
* }

)

if((m\_zurts[i]🡪row()==m\_player🡪row())&& (m\_zurts[i]🡪col() == m\_player🡪col()) )

* 
* void getData(CDAccountV1& theAccount);
* *//Postcondition: theAccount.balance, theAccount.interestRate, and*
* *//theAccount.term have been given values that the user entered at the*
* *//keyboard.*
* struct PersonInfo
* {
* double height; //*in inches*
* int weight; //*in pounds*
* Date birthday;
* };
* A structure variable of type PersonInfo is declared in the usual way:
* PersonInfo person1;
* struct Date {
* int month;
* int day;
* int year;
* };
* struct PersonInfo
* {
* double height; //in inches
* int weight; //in pounds
* Date birthday;
* };
* Once the type Date is defined, you can declare and initialize a structure variable called dueDate as follows:
* Date dueDate = {12, 31, 2012}
* for class, a really simple example:
* class DayOfYear
* {
* public:
* void output();
* int month;
* int day;


* };
* int main ()
* {
* DayOfYear today, birthday;
* cin >> today.month;
* cin>> today.day;
* today.output();
* }
* void DayOfYear::output()
* {
* switch (month) {
* case 1:
* cout << "Januarary" ;
* break;
* default:
* cout << "default!!";
* break;
* }
* cout << day;
* }
* Once you make a member variable a private member variable, there is no way to change its value (or to reference the member variable in any other way) except by using one of the member functions.
* Structures are normally used with all member variables public and with no member functions. However, in C++ a structure can have private member variables and both public and private member functions.
* For a pointer \*p, you can use this to call p a function in p: p->eat(1);
* history += toupper(dir);
* if history is a string, then it just add char (dir) to the string every time it does +=.
* Delete a[1]. We are not deleting the pointer, but an object in the pointer.

|  |
| --- |
| * 1st - I don't want the cow6 to be replicated at the end. How could I delete that? |

* You need to decrement 'length\_array' and 'length' (if that's something different).
* Line 1 should suffice. length\_array=length\_array--; // No need to assign
* 2nd - the object cow2 was deleted but the pointer \*cow2 is not deleted and it still exists (pointing to nowhere)If you mean that the memory of the array is still the same: Yes. If you want a smaller array that reflects the actual size then you have to alloc the new (smaller) array and copy the pointer to it.
* I'd recommend to use std::vector that does most of the work for you
* Delete target[i];
* Ntarget- -
* One of the function of pointer is to represent possessional relationship (whether it’s related), with a null pointer being not related.
* Destructor:
* Void f()
* {

pet.p1(“Fido”,20);

p1.addToy();

* }
* never takes argument, never call it, automatically called. Not return values.
* The destructor:whatever is necessary to make the object go away.
* Don’t ever call a dangling pointer.
* this works fine:
* #include <iostream>
* using namespace std;
* int main()
* {
* int a;
* int \*p = &a;
* cin >> \*p;
* cout << \*p << endl; return 0;
* }
* \*pt++
* which means "increment ptr to go to the element after the one it points at, then dereference its old value" (since postfix ++ hands back the value the pointer used to have).
* ++(\*ptr) meaning "increment the value pointed at by ptr,
* \*p++ returns \*p and then do p++
* \*(p++) is the same as \*p++
* (\*p)++ means adding 1 to (\*p) which is x[2] here.
* A class is a type. (like int, char, float, double )
* An object is an instance of a class (like int a; a is an instance)  
  Attributes are variables declared inside the class representing the state of the object. Methods are functions declared inside the class and can be called by its object to manipulate attributes inside the object.
* The differences between struct and class are:  
  (1) Members of a struct is ‘public’ by default while members of a class is ‘private’ by default.  
  (2) Inheritance between struct is public by default while inheritance between class is private by default. (CS32... maybe).
* employee emp1[2];
* //  An employee struct array with 2 cells (each cell is an employee structure)
* employee emp = {0,0,""}; // initialization  
  // same as: emp.ID = 0; emp.salary = 0; emp.jobtitle = "";
* employee emp[2];  
  // employee \*emp = new employee[2];
* cout << "Please enter employee information:\n";
* for(int i = 0 ; i < 4 ; i++ )
* emp[i] = emp[i + 1];
* ---it swaps the whole pointer, with its obejcts and everything.
* employee emp1[2];
* //  An employee struct array with 2 cells (each cell is an employee structure)
* \*\*\*ask about example 3\*\*
* is it a symbol for deconstructor class?
* ~employee() {
* cout << ID << " got laid off!!" << endl; }
* use the dot operator, when what you have on the left is an object or a reference to an object.
* Is the arrow operator -> when what you have on the left is a pointer to an object
* “overloaded function name”

if two function, different name, in c++:

as long as they are different somehow, it’s ok to have more than one function with the same name.

You delete [] when you newed an array type, and delete when you didn't.

* int main()
* {
* char a[80]="Hakun";
* char b[80]="amatata";
* strcat(a, b);
* for (int i=0; i<strlen(a);i++)
* {
* cout << a[i];
* }

}

* class myArray
* {
* public:
* myArray(int size): //constructor (no return type and same name as class)
* {
* arr = new int[size];

}

}

* #include <iostream>
* using namespace std; int main()
* {
* int arr[12] = {1,3,5,0,7,2,0,4,4,0,8,8}; int count = 0;
* for(int i=0;i<11;i++) {
* if(arr[i] = arr[i+1] ) count++;
* else
* count--;
* }

cout << count << endl; }

* void swap2(int\* a, int \*b) {
* int temp=\*a; \*a=\*b; \*b=temp; }

--the address never change, it only changes the value.

Define a constructor:

* **Goldfish::Goldfish(int capacity) {**
* **if (capacity < 1) m\_capacity = 3;**
* **else m\_capacity = capacity;**
* **m\_memory = new char[m\_capacity];**
* **m\_amount = 0; forget();**
* **}**

implement a function of a class:

void goldfish::remember(char c);

how to implement add zurt:

* bool Arena::addZurt(int r, int c, char color)
* {
* if (m\_nZurts == MAXZURTS)
* return false;
* m\_zurts[m\_nZurts] = new Zurt(this, r, c, color);
* m\_nZurts++;
* return true;
* }

if it’s a pointer use ->, if its not use .

you can do %=

the constructor do stuff first, for the hotel example it’s to empty the room.

* char str1[]="Sample string";
* cout << reverseString(&str1[0], &str1[10]);