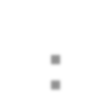
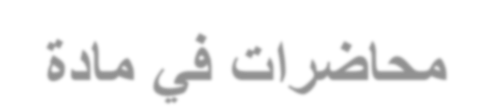
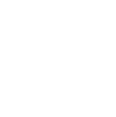
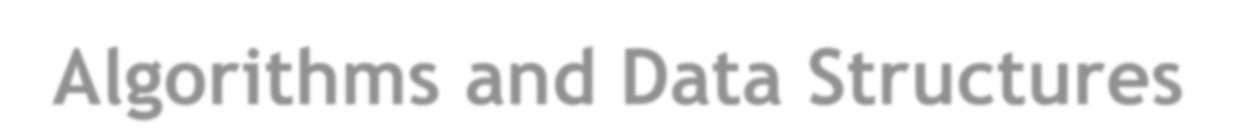
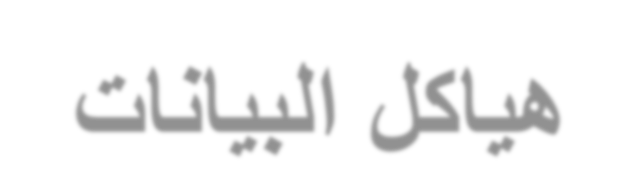
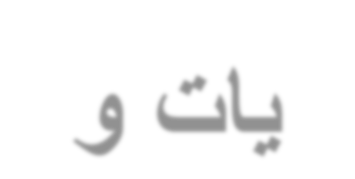
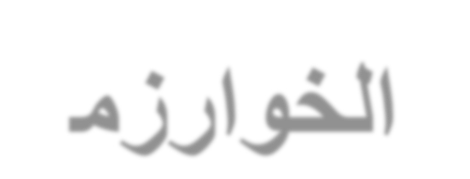
1



**د/ قسم السيد إبراهيم**

**جامعة أفريقيا العالمية**

Course Contents:

(1)IntroductiontoC++

(2)PointersandDynamicObjects

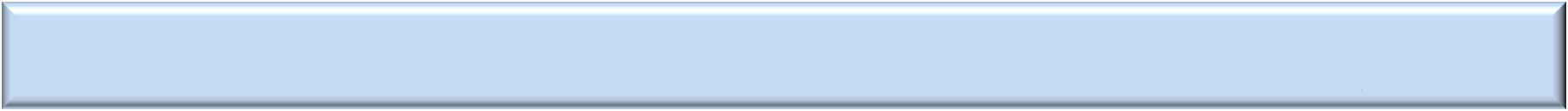
(3)LinkedLists

(4)Stacks (5)Queues (6)Trees (7)Sorting

(8)Searching

(9)Graphs

References:



“DataStructuresandAlgorithmAnalysisinC++”,byMarkAllenWeiss,

Addison-Wesley,2006.

“DataStructuresandAlgorithmsinC++”,byAdamDrozdek,2nded.,

2008.

“C++PlusDataStructures”,byNellDale,JonesandBartlettPublishers,

1999.

4

**Lecture# 1**

Introductionto C++

Outline

•C++basicfeatures

Statementsyntax.

•Classdefinitions

Data members,methods,constructor,destructor.

Pointers,arrays,andstrings.

Parameterpassinginfunctions.

BasicC++

•InheritallC syntax

Primitivedatatypes

Supported data types: **int**, **long**, **short**, **float**,

**double**, **char**, and **bool**.

Basic expressionsyntax

Definingtheusualarithmeticandlogical operations

suchas: **+**, **-**, **\***, **/**, **%**,**&&**, **||**,and **!**.

Basic statementsyntax

**If-else**, **switch**, **for**, **while**,and**do-while**.

BasicC++(cont…)

•Addanewcommentmark:

**//**.… : for1 line comment.

**/\***.…**\*/** : for agroupofline comment.

•*const*support for constantdeclaration, justlikesC.

ClassDefinitions

•AC++class consistsof*datamembers* and *methods(member functions).*

**classIntCell**

**{**

Memberfunctions

Initializerlist:usedtoinitializethedata

membersdirectly.

**public:**

**IntCell(intinitialValue=0 )**

**: storedValue(initialValue) {}**

**intread()const**

|  |  |  |
| --- | --- | --- |
| **{** | **return** | **storedValue;}** |
| **void** | **write(** | **int x)** |

**{ storedValue= x;}**

**private:**

**intstoredValue;**

**}**

Indicates that the member’s invocation does

Not change any of the data members.

Datamember(s)

InformationHidingin C++

•Twolabels: *public*and*private*

Determinevisibilityofclassmembers.

Amemberthat is *public*canbeaccessedbyanymethod in anyclass.

Amemberthat is *private*only canbeaccessedby methodsinitsclass.

•Informationhiding

Data membersaredeclared*private,*thusrestricting accessto internal detailsoftheclass.

Methodsintendedfor general usearemade*public*.

Constructors

•A*constructor*isaspecial methodthatdescribes howaninstanceof theclass(called*object*) is constructed.

•Wheneveraninstanceoftheclassis created, itsconstructoris called.

•C++providesa*default constructor* foreach class, whichis a constructorwithno parameters.

But,onecandefinemultipleconstructorsfor thesame class,

and may evenredefinethedefaultconstructor.

Howtodistinguishthem?

Destructor

•Typically, the*destructor* is used tofreeupanyresourcesthat wereallocatedduring theuse oftheobject.

•A*destructor* is called when anobjectisdeletedeitherimplicitly, orexplicitly(usingthe *delete*operation).

Thedestructoris called wheneveranobjectgoesout of scopeor issubjectedto a*delete.*

•C++ providesa*default destructor* for eachclass.

Thedefaultsimply applies thedestructoron each data member.

Butwe canredefinethedestructorofaclass.

•AC++classcanhave onlyonedestructor.

Pointers

•A*pointer*is a variablewhichcontainsaddress ofothervariable.

•Accessingthedataatthecontainedaddressiscalled“dereferencinga

pointer”or“followinga pointer”.

**x**

**(4104)**

pointer

***y***

***(4100)***

**n**

**(4096)**

***4096***

**7**

APointer Example

Thecode

BoxdiagramMemoryLayout

**voiddoubleIt(int x, int \*p)**

**{**

**main**

**\*p= 2 \* x;**

**}**

**a 16**

***p 8192***

**intmain()**

**{**

**inta = 16;**

**doubleIt(9,&a);**

**cout<<"a= "<< a;**

**return0;**

**}**

**doubleIt**

**x 9**

***(8200)***

**x9**

**(8196)**

**a16**

**(8192)**

**doubleIt**

**main**

**Output:**

a = 18 ***p***



ObjectPointerDeclaration

•Declaration:

**ClassName\*PointerName;**

Example:

**IntCell\**p*;**

**//definesa pointer to an object of class *IntCell***

The**\***indicatesthat*p* isapointervariable; it isallowedto point atanIntCellobject.

The*value*of*p*is theaddressof theobjectthat itpointsat.

*p*isuninitializedat this point.

DereferencingPointers

•Dynamic objectcreation

*p*=*new* IntCell;

p 8888

8888



InC++*new*returns a pointerto thenewly createdobject.

•Garbagecollection

C++does nothavegarbagecollection.

Whenanobjectthat**isallocated by *new***is no longer

referenced, the*delete* operationmust beapplied totheobject. delete*p;*

DereferencingPointers(cont…)

•Usingapointer

Wecanaccessaclassmemberof theobjectpointedatby a

pointerbyusingoperator‟**->**‟.

***p*=newIntCell;**

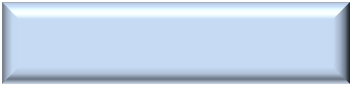
**int*b*;**

**……**

***b*=*p*->read();//thevalueofthedatamember**

**//*storedValue*oftheobjectpointedat**

**//by*p*isassignedto *b***



**cout<<*b*<<endl;**

**p->write(50);**

**cout<<*p*->read()<<endl;**

Theoutputis:

0

50

Array of Objects

•An*array*of objectsis a collectionofobjectswith same typestored consecutivelyinmemory.

•**Declaringstatic array:**

**ClassNameArrayName[size];**

Example:

**arr**

**0 1 2 3 4 5 6 7 8 9**

**IntCell*arr*[10];//anarrayconsistingof10IntCellobjects**

Thesizeof thearraymust beknownat compiletime.

*arr*actually is aconstantpointer.

Thevalueof*arr***cannot**bechanged.

The*(i+1)-*stobjectin thearray*arr*canbeaccessedeitherby using***arr[i]***,or by***\*(arr+i)***.

Arrayof Objects(cont…)

•**Declaringdynamicarray:**

**ClassName\*PointerName;**

**PointerName=newClassName[size];**

Example:

**IntCell\**p*=newIntCell[10];**

***arr=p;* //invalid**

**0 1 2 3 4 5 6 7 8 9**

**p**

The*(i+1)-*stobjectin thearray*p*canbeaccessedeither

by using***p[i]****,*orby***\*(p+i)***.

•Arraysofobjects cannotbecopiedwith„**=**„.

•Arrays are*not* passedbycopy.Instead,theaddressof thefirst

elementis passedto thefunction.

**intsumOfArray(intvalues[],intnumValues)**

Strings

•Built-inC-stylestrings areimplementedas anarrayof characters.

•Each string ends withthespecial null-terminator„**\0**‟.

•Commonstring functions:

***strcpy***:usedto copy strings.

***strcmp***:used tocompare strings.

***strcat***: used tojoinstrings.

***strlen***:used todeterminethelengthof strings.

•Individualcharacters canbeaccessedbythearrayindexing operator.

Strings (cont…)

**0 1 2 3 4 5 6 7 8 9**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **f** | **o** | **o** | **l** | **\0** |  |  |  |  |  |

**s1**

**char*s1[]=*"*fool*"*;* char*s2[]=*"*fool*"*;* char*s[]=*"*abcdefg*"*;***

***if(strcmp(s1,s2)==0)***

***cout<<*"*Samestrings.*"*;***

***elsecout<<*"*Differentstrings.*"*; strcpy(s1,s);//copystos1 strcat(s2,s);//addstos2***

***cout<<strlen(s1)<<endl; cout<<strlen(s2)<<endl; cout<<strlen(s)<<endl;***

**0 1 2 3 4 5 6 7**

**s2**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **f** | **o** | **o** | **l** | **\0** |  |  |  |  |  |

**0 1 2 3 4 5 6 7**

**s**

**0 1 2 3 4 5 6 7**

**s1**

**0 1 2 3 4 5 6 7**

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **f** | **o** | **o** | **l** | **a** | **b** | **c** | **d** | **e** | **f** | **g** | **\0** |

**s2**

**0 1 2 3 4 5 6 7**

**s**

**8 9**

**8 9**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **e** | **f** | **g** | **\0** |  |  |

**8 9**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **e** | **f** | **g** | **\0** |  |  |

**8 9 10 11**

**8 9**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **a** | **b** | **c** | **d** | **e** | **f** | **g** | **\0** |  |  |

FunctionCallbyValue

**voidf(intx)**

**{ cout<<"valueof x= " <<x <<endl;**

**x =4; }**

|  |  |  |
| --- | --- | --- |
| **main()** |  |  |
| **{ intv =** | **5;** |
| **f(v);** |  |  |
| **cout<<** | **"value** | **of v=" << v<<endl;}** |

*Output:*

valueof x= 5

valueof v =5

Whenavariable*v*is passed *byvalue*toafunction*f*, its valueis copiedto thecorrespondingvariable*x* in*f*.

Anychanges tothevalueof*x* doesNOT affect thevalueof *v inthe*

*mainprogram.*

Call byvalueisthe defaultmechanismforparameter passing in C++.

FunctionCallbyReference

**voidf(int&x)**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **{cout<<** | **"value** | **of** | **x** | **=** | **"** | **<<** | **x** | **<<** | **endl;** |
| **x=4;** | **}** |  |  |  |  |  |  |  |  |

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **main()** |  | | | | | | | | |
| **{intv= f(v); cout<<** | **5;**  **"value** | **of** | **v** | **=** | **"** | **<<** | **v** | **<<** | **endl;}** |

*Output:*

value of x = 5

valueof v =4

Whena variable *v*ispassed*byreference*to a parameter *x* offunction

*f*, *v* andthecorrespondingparameter*x* refer tothesame variable.

Anychanges tothevalueof*x* DOES affect thevalue of *v*.

FunctionCallbyConstant Reference

**voidf(constint&x)**

**{ cout<<"valueofx="<<x<<endl;**

**x=4; //invalid**

**}**

**main()**

**{intv=5;**

**f(v);**

**cout<<"valueofv="<<v<<endl;}**

*Output:*

value of x =5

valueof v =5

Passingvariable*v by constantreference*toparameter*x* of*f*will**NOT**

allowany changetothevalueof*x.*

 Itisappropriateforpassinglargeobjectsthatshouldnotbe changed bythecalled function.

Usage of ParameterPassing

•*Callbyvalue* is appropriateforsmallobjectsthat shouldnotbe

changedbythefunction.

•*Callbyconstantreference*is appropriateforlargeobjectsthat shouldnotbechangedbythefunction.

•*Callbyreference*isappropriatefor allobjectsthat may be

changedbythefunction.