Data Structure using C++

Lecture 03

Reading Material

Data Structures and algorithm analysis in C++ Chapter. 3 3.1, 3.2, 3.2.1

Summary

- Strings
- Structures
- Nested Structures

Strings

A structure graph consists of a set of elements that are a lesson for The Alphabet symbols and numbers and special symbols that are located on the keyboard and declared in the following way:

```
1- char  name - of - string[size];
2- char  *name;
Ex:-
     1)  char   name[35];
     2)  char  *name;
```

The most prominent functions that apply on strings Which are located within the library **<string.h>**.

*ابرز الدوال التي تطبق على الخيوط الرمزية والتي تتواجد ضمن المكتبة <string.h> :

1- strcpy(st1,st2); : دالة لاستنساخ خيط رمزي معين من خيط آخر ، وممكن تحديد عدد الرموز المراد استقطاعها وبالشكل التالي: strncpy(st1,st2,n); تمثل n عدد الرموز المستقطعة.

2- strcmp(st1,st2); : دالة للمقارنة بين الخيوط الرمزية.

3- strlen(st); : دالة لحساب طول الخيط الرمزي .

4- strcat(st1,st2); : دالة لدمج خيطين رمزيين بحيث تكون النتيجة في الخيط الرمزي الاول.وممكن تحديد عدد الرموز التي تستقطع من الخيط الرمزي الثاني وتدمج الخيط الاول بواسطة: strncat(st1,st2,n); .

**ابرز الدوال التي تطبق على الخيوط الرمزية والتي تتواجد ضمن المكتبة <ctype.h> :

1- isalnum(ch); : دالة لمعرفة الرمز اذا كان عبارة عن حرف ابجدي أو رقم ، ترجع هذه الدالة قيمة صفرية اذا كان الرمز لا يساوي قيمة ابجدية ولا رقم.

2- isalpha(ch); : تختبر الرمز اذا كان حرف ابجدي او لا.

3- islower(ch); : تختبر الرمز فيما اذا كان صغير ترجع قيمة والا ترجع صفر.

4- isupper(ch); : تختبر الرمز اذاكان كبير فانها ترجع قيمة.

5- isdigit(ch); : تختبره اذا كان رقم ترجع قيمة.

6- struper(); : تحول الحرف من صغير الى كبير.

7- strlwr(); : تحول الحرف من كبير الى صغير.

8- strinv(); : تعكس الخيط الرمزي.

Definition of Strings

- Generally speaking, a string is a sequence of characters
- Examples: “hello”, “high school”, “H2O”.
- Typical desirable operations on strings are:
  - Concatenation: “high”+“school”=“highschool”
  - Comparisons: “high”<“school” // alphabetical
  - Finding/retrieving/modifying/deleting/inserting substrings in a given string
- C++ has a <string> library
- Include it in your programs when you wish to use strings: #include <string>
- In this library, a class string is defined and implemented
- It is very convenient and makes string processing easier than in C

Declaration of strings

- The following instructions are all equivalent. They declare x to be an object of type string, and assign the string “high school” to it:
  - string x(“high school”);
  - string x= “high school”;
  - string x; x=“high school”;

Operations on strings (Concatenation):

- Let x and y be two strings
- To concatenate x and y, write: x+y

```
string x= “high”;
string y= “school”;
string z;
z=x+y;
cout<<“z=“<<z<<endl; z =z+“ was fun”;
cout<<“z=“<<z<<endl;
```

```
Output:
z=highschool
z= highschool was fun
```

Concatenation of Mixed-Style Strings:

- In `s=u+v+w;` where s is of type **string**,
  - u can be
    - ➢ A **string** object, or
    - ➢ a C-style string (a char array or a char pointer),

   - a C-style char
   - or a double-quoted string,
   - or a single-quoted character.
- Same with v and w.
- At least u or v or w must be a **string** object

Example of Mixed-Style Concatenation:

```
string x= “high”;
char y[]= “school”;
char z[]= {‘w’,’a’,’s’,’\0’};
char *p = “good”;
string s= x+y+’ ‘+z+” very”+” “+p+’!’;
cout<<“s=“<<s<<endl;
cout<<“s=“+s<<endl;
```

```
Output:
s=highschool was very good!
s=highschool was very good!
```

Comparison Operators for string Objects:

- We can compare two strings x and y using the following operators: ==, !=, <, <=, >, >=
- The comparison is alphabetical
- The outcome of each comparison is: **true** or **false**
- The comparison works as long as at least x or y is a **string** object. The other string can be a **string** object, a C-style string variable, or a double-quoted string.

Example of String Comparisons:

```
string x= “high”;
char y[]= “school”;
char *p = “good”;
If (x<y)
      cout<<“x<y”<<endl;
If (x<“tree”)
      cout<<“x<tree”<,endl;
If (“low” != x)
      cout<<“low != x”<<endl;
if( (p>x)
      cout<<“p>x”<<endl;
Else
      cout<<“p<=x”<<endl;
```

```
Output:
x<y
x<tree
low != x
p>x
```

Example: program to read strings, and then calculate the number of symbols that is a numbers.

```
#include<iostream.h>
#include<string.h>
void main()
{char st[100],l;
cout<<"Enter st: "<<endl;
cin>>st;
l=strlen(st);
int c=0;
for(int i=0;i<l;i++)
if((st[i]>='0')&&(st[i]<='9'))
c++;
cout<<c;
}
```

Example: Program to read strings, and then calculate the character who is in the middle.

```
#include<iostream.h>
#include<string.h>
void main()
{int i,l;
char st[30];
cout<<"enter string: ";
cin>>st;
l=strlen(st);
i=l/2;
cout<<st[i];
}
```

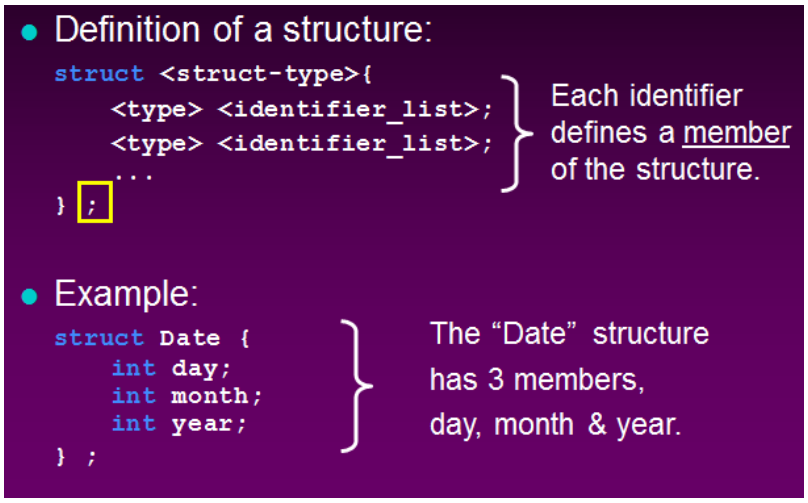
Example: Program to read String, and then convert the great character to lowercase.

```
#include<iostream.h>
#include<string.h>
#include<ctype.h>
void main()
{
 int i,l;
 char st[10];
 cout<<"enter your string: "<<endl;
 cin>>st;
 l=strlen(st);
 for(i=0;i<l;i++)
 {
  if(isupper(st[i])!=0)
  strlwr(st);
  }
   for(i=0;i<l;i++)
 cout<<st[i];
 cin>>" ";
}
```

Structures

- A **Structure** is a collection of related data items, possibly of different types.
- A structure type in C++ is called **struct**.
- A **struct** is **heterogeneous** in that it can be composed of data of different types.
- In contrast, **array** is **homogeneous** since it can contain only data of the same type.
- Structures hold data that belong **together**.
- Examples:
    - Student record: student id, name, major, gender, start year, …
    - Bank account: account number, name, currency, balance, …
    - Address book: name, address, telephone number, …
- In database applications, structures are called records.
- Individual components of a struct type are called **members** (or **fields**).
- Members can be of **different types** (simple, array or struct).
- A struct is named as a whole while individual members are named using field identifiers.
- Complex data structures can be formed by defining **arrays of structs**.

Struct basics: 1-



- Definition of a structure:

```
struct <struct-type>{
    <type> <identifier_list>;
    <type> <identifier_list>;
    ...
} ;
```

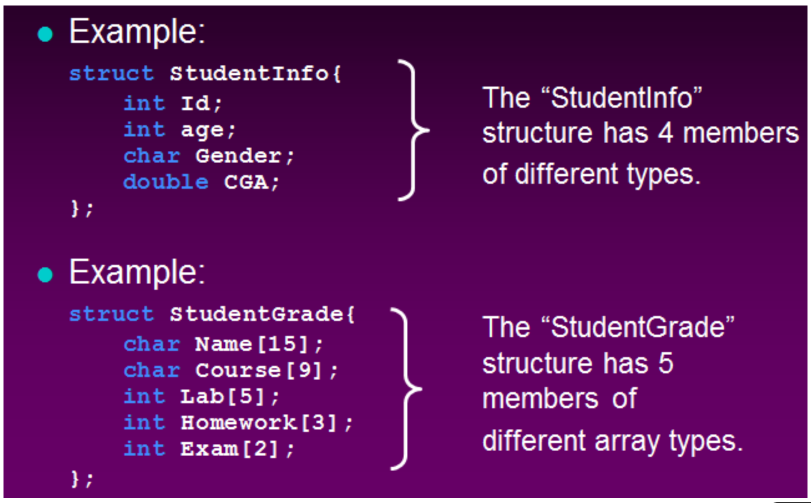
Each identifier defines a member of the structure.

- Example:

```
struct Date {
    int day;
    int month;
    int year;
} ;
```

The “Date” structure has 3 members, day, month & year.

Struct examples:



- Example:

```
struct StudentInfo{
    int Id;
    int age;
    char Gender;
    double CGA;
};
```

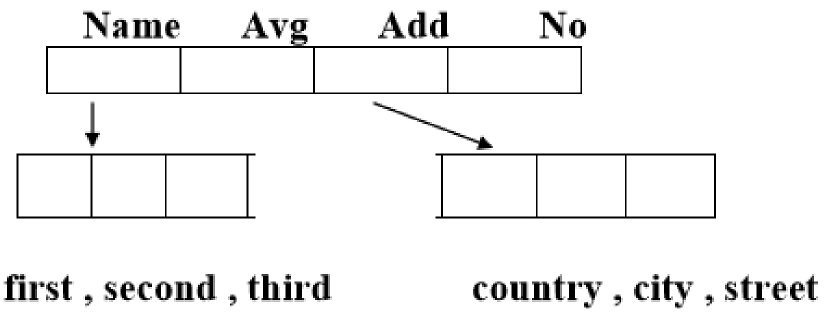
The “StudentInfo” structure has 4 members of different types.

- Example:

```
struct StudentGrade{
    char Name[15];
    char Course[9];
    int Lab[5];
    int Homework[3];
    int Exam[2];
};
```

The “StudentGrade” structure has 5 members of different array types.

Nested Structures



Declaration of Nested Structures:

```
struct  names
 {
   char  first[30],second[30],third[30];
  };
 struct  address
   {
   char  country[30],city[30],street[30];
    };
struct  student
  {
   names  name;
   float  avg;
   address   add;
   int  no;
  };
```

```
void  main()
{
 student  S;
 S.name.first;
 S.name.second;
 S.avg;
 S.address.city;
 S.address.street; }
```

الامثلة:

1- برنامج لقراءة قيود **N** من الطلبة ، كل قيد يتكون من الاسم (الاسم الأول ، اسم العائلة) ، والرقم ، والعنوان (المدينة ، الشارع) ، احسب عدد الطلبة الذين تبدأ مدنهم بحرف **B** ؟

```
 #include<iostream.h>
 #include<string.h>
 #include<conio.h>
 const int size=100;
  struct names
  {

char first[35],family[35];
};
struct address
{
 char city[35],street[35];
 };
 struct student
 {names name;
 address add;
 int no;
 };
void readrec(student[],int);
void countr(student[],int);
void main()
{
  clrscr();
  int n;
 student s[size];
 cout<<"Enter n ";
 cin>>n;
 readrec(s,n);
 countr(s,n);
}
void readrec(student s[size],int n)
{
int i;
for(i=0;i<n;i++)
 {
  cout<<"Enter informaiton: "<<endl;
cin>>s[i].name.first>>s[i].name.family>>s[i].no>>s[i].add.city>>s[i].add.street;
 }
 void countr(student s[size],int n)
 {
 int i,c=0;
 char ch;
  for(i=0;i<n;i++)
  {strncpy(ch,s[i].add.city,1);
  if(strcmp(ch,'b')!=0)
  c++;
  }
  cout<<c;
  cin>>"";
  }
```