جَامعة المَنارة
MANARA UNIVERSITY

كلية الهندسة قسم المعلوماتية

بنى معطيات 1

Data Structure 1

ا. د. علي عمران سليمان

محاضرات الأسبوع الثالث
بنى معطيات و أنماط البيانات المجردة
Data Structures and ADT

الفصل الثاني 2024-2025

محتوى الفصل

مدخل إلى بنى المعطيات و أنماط البيانات المجردة	
1- مقدمة	
2- تحقيق بنى المعطيات و أنماط البيانات المجردة	
3- أنماط البيانات البسيطة	
4- المصفوفات	
5- السجلات	
6- البرمجة الإجرائية	
7- الأصناف	
8- السلاسل المحرفية كأنماط بيانات مجردة	

References

- Deitel & Deitel, Java How to Program, Pearson; 10th Ed(2015)

- د.علي سليمان، بنى معطيات بلغة JAVA، بنى معطيات بلغة C++، بنى معطيات بلغة Pascal جامعة تشرين 2014، 2007، 1998

1- مقدمة:

إن مرحلة التصميم التي تمر بها عملية بناء البرمجيات تتضمن :

1. وصف كيفية تنظيم البيانات المستخدمة في المسألة
2. كيفية تصميم الخوارزميات التي تعمل على هذه البيانات (تخزين storage، استرجاع retrieval ومعالجة manipulation) .

إن البيانات المراد معالجتها يجب أن تنظم بحيث تعكس العلاقة بين عناصر البيانات وتسمح بمعالجة هذه البيانات بفعالية .
إن عملية تحديد كيفية تخزين عناصر البيانات وعملية تصميم الخوارزميات التي تؤدي العمليات عليها لا يمكن أن تنفصلا ويحب أن تتما بالتوازي .

نعرف ونبين في هذا الفصل بنى المعطيات وكيفية تحقيقها ، ونبين كيف تم تحقيق أنماط البيانات المسبقة التعريف في لغة C++وكيف يمكن أن تستخدم لتحقيق أنماط بيانات جديدة .

1- تحقيق بنى المعطيات و أنماط البيانات المجردة1

لنفرض أن شركة نقل تقوم بتشغيل وسيلة نقل واحدة تتسع لعشرة مسافرين ، هذه الشركة ترغب بتحديث عملياتها وكخطوة أولى قررت بناء برمجية تحدد في كل رحلة ما هي المقاعد غير المحجوزة التي يمكن حجزها .

من الواضح في هذه المسألة أن الغرض الرئيسي هو عبارة عن مجموعة المقاعد العشرة ومن أجل كل مقعد إشارة فيما إذا كان محجوزاً أم لا . يجب أن نكون قادرين على إجراء العمليات التالية :

- اختبار مجموعة المقاعد العشرة لمعرفة أي منها غير محجوز .
- حجز مقعد ما .
- إلغاء عملية حجز مقعد ما .

لإنجاز هذه العمليات ، من المناسب التفكير بأن هذه المقاعد منظمة في لائحة ، وتعريف العمليات اللازمة عليها ، هناك العديد من البنى الممكنة فعلى سبيل المثال يمكن تعريف نمط تعدادي (enumeration) كما يلي :

```
enum SeatStatus(OCCUPIED,UNOCCUPIED);
```

وتمثيل لائحة المقاعد من خلال 10 متحولات بسيطة من النمط SeatStatus :

```
SeatStatus seat1,seat2,.....,seat10;
```

انطلاقا من هذه البنية يمكن تعريف العمليات الأساسية كما يلي : خوارزمية عرض لائحة بالمقاعد الفارغة :

```
Algorithm to list unoccupied seats
1-for number ranging from 0 to MAX_SEATS-1 do the following :
        If  seat[number] is UNOCCUPIED
                Display number .
```

1- تحقيق بنى المعطيات و أنماط البيانات المجردة2

خوارزمية حجز مقعد :

Algorithm to reserve a seat
1-read number of seat to be reserved
2-if seat[number] is UNOCCUPIED
Set seat[number] to OCCUPIED
Else
Display a message that the seat having this number has already been assigned .

خوارزمية إلغاء حجز مقعد شبيهة تماماً بخوارزمية حجز مقعد .

إن هذا العمل البسيط يتطلب تحديد مجموعة عناصر البيانات و العلاقات الأساسية بينها والعمليات التي تنفذ عليها.

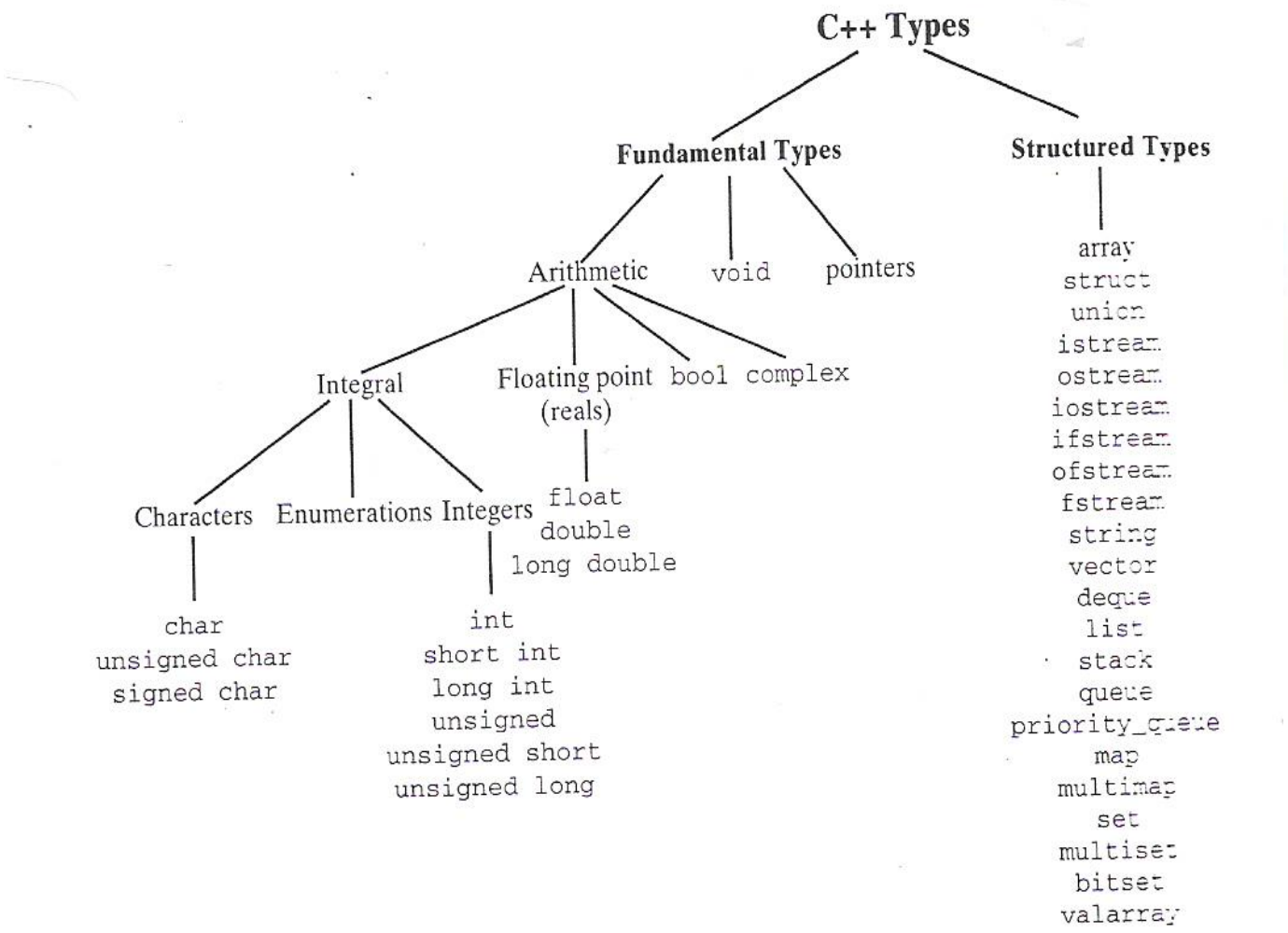
إن هذه المجموعة مع العمليات والعلاقات تدعى نمط بيانات مجرد (Abstract Data Type وتدعى اختصاراً ADT) .

إن كلمة مجرد Abstract تشير إلى حقيقة أن البيانات والعمليات الأساسية عليها والعلاقات فيما بينها تمت دراستها بشكل مستقل عن كيفية تحقيقها .

إن تحقيق نمط بيانات مجرد (ADT) تتضمن بنى تخزينية storage structures تدعى عادة بنى معطيات data structures لتخزين عناصر البيانات ، وخوارزميات algorithms للعمليات والعلاقات الأساسية .

إن أهمية وجود أنماط البيانات المجردة (ADT) تكمن في أنك تستطيع استخدامها ودراستها بدون أن تكون معنياً بتفاصيل بنيتها وتحقيقها .

الأنواع الأكثر شيوعاً بلغة C++



2- أنماط البيانات البسيطة 1

Simple Data Types

في وسائط التخزين للنظم الحاسوبية تكون البايتات مرقمة والرقم المعطى للبايت يدعى عنوانه address ، يتم الوصول إلى محتوى كل بايت من خلال عنوانه .

أنماط المعطيات الأساسية في لغة C++ مثل int,float,double,char,bool تدعى أنماط البيانات البسيطة وذلك لأن القيمة في هذه الأنماط تكون عنصرية atomic أي تتكون من وحدة واحدة لا يمكن تجزئتها ومع ذلك يمكن النظر إليها على أنها نمط معطيات مجرد (ADT) .

البيانات المنطقية Boolean Data : وتتألف من قيمتين فقط هما false و true . وهي معرفة في لغة C++ بنمط المعطيات bool . إذا تم ترميز القيمة false بـ 0 والقيمة true بـ غير الصفر عندئذ كل ما نحتاجه لتخزين قيمة منطقية هو بت واحد ، عادة إذا استخدم byte أو word لتمثيل قيمة منطقية فإنه يكون ممثلاً للقيمة false إذا كانت جميع بتاته مساوية للصفر وممثلاً للقيمة true فيما عدا ذلك .

البيانات المحرفية Character Data : يقوم التمثيل الداخلي للبيانات الحرفية على أساس إسناد الترميز الرقمي للمحارف ، وقد تم تطوير العديد من مخططات الترميز مثل ASCII(American Standard Code for Information Interchange) و EBCDIC(Extended Binary Coded Decimal Interchange Code) و Unicode . تم التعبير عن المحارف في لغة C++ باستخدام النمط char وتخزن في بايت واحد (8bit) . على سبيل المثال الترميز ASCII للمحرف c هو 99 أي 01100011_2 وتخزن في بايت واحد ،

صمم الترميز Unicode للاستخدام مع أغلب لغات البرمجة وبالتالي يجب أن يتيح ترميز العديد من المحارف ، وبما أننا باستخدام بايت واحد نستطيع ترميز 256 حرف ، فإن الترميز Unicode استخدم 2 بايت (16bit) لترميز أكثر من 65000 محرف وبالتالي فإن المحرف c يرمز على النحو التالي : 0000000001100011_2 تستخدم لغة C++ **النمط المحرفي الواسع** (wide character type) **ويدعى** wchar_t **لتخزين المحارف الكبيرة للترميز** Unicode .

Simple Data Types

2- أنماط البيانات البسيطة 2

يمكن أن تعالج القيم الحرفية في لغة C++ كقيم صحيحة كمايلي :

int(char_value)=the numeric code of char_value.

ونميز من النمط char النمطين التاليين :النمط unsigned char ويأخذ القيم ضمن المجال من 0 إلى 255 والنمط signed charويأخذ القيم ضمن المجال 127- إلى 127 . يخزن تتالي من المحارف في بايتات متعاقبة ، بايت لكل محرف في السلسلة ، على سبيل المثال الكلمة CODE تخزن كما يلي :

0	1	1	0	0	0	1	1	0	1	1	0	1	1	1	1	0	1	1	0	0	1	0	0	0	1	1	0	0	1	0	1
C								O								D								E							

البيانات الصحيحة Integer Data : البيانات الصحيحة بدون إشارة unsigned integers هي مجوعة الأعداد الصحيحة غير السالبة أي {0,1,2,...} ويشار إليها في C++ بأنماط المعطيات التالية unsigned short ، unsigned و unsigned long (وهي اختصار لـ unsigned short int ، unsigned int و unsigned long int) تخزن القيم unsigned short في 2 بايت والقيم unsigned long في 4 بايت وبالتالي العدد $58=0000000000111010_2$

البيانات الصحيحة بإشارة signed integers : وهي عبارة عن مجموعة الأعداد التالية {.....,-3,-2,-1,0,1,2,3,...}والمعرفة في لغة C++ بالأنماط التالية short (أو short int) ، int ، long (أو long int) .

ليكن لدينا العدد الصحيح 12345 ، في حال أردنا أن يعامل ويخزن كقيمة unsigned نستطيع إضافة اللاحقة U إليه أي (12345U) ، وفي حال أردنا أن يعامل ويخزن كقيمة long نستطيع إضافة اللاحقة L أي (12345L) وبالمثل (12345UL or 12345LU) للقيم unsigned long

2- أنماط البيانات البسيطة 3

Simple Data Types

تستخدم أغلب التمثيلات للأعداد الصحيحة خانة واحدة (bit) لتمثيل الإشارة ، إحدى الطرق للقيام بذلك هي إعطاء الخانة الواقعة في أقصى اليسار القيمة 1 إذا كان الرقم سالباً والقيمة 0 في حال كان موجباً واستخدام باقي الخانات لتمثيل القيمة ، مثلاً الرقم 6 أو 6- يمثل كما يلي:

على الرغم من أن تمثيل القيم الصحيحة سهل جداً ، فإن بعض العمليات على هذه الأعداد لا تتم بنفس البساطة ، على سبيل المثال جمع قيمتين صحيحتين مختلفتين بالإشارة .إحدى الأساليب الأكثر شيوعاً لإنجاز ذلك تتمثل باستخدام المتمم الثنائي two's complement ،

البيانات الحقيقية Real Data : القيم الحقيقية وأحياناً تدعى القيم ذات الفاصلة العائمة floating-point وتمثل في لغة C++ بالنمط ذو الدقة الأحادية float والنمط ذو الدقة المضاعفة double **والنمط ذو الدقة الموسعة** long double . النمط الافتراضي هو double وتستخدم اللاحقة F للدلالة على أن النمط هو float والرمز L للدلالة على أن النمط هو long double . في التمثيل العشري للقيمة الحقيقية فإن الخانات إلى يمين ويسار الفاصلة تكون معاملات لقوى العدد 10 .

هناك العديد من الأساليب المستخدمة لتخزين القيم الحقيقية في الذاكرة ، أحد تمثيلات الفاصلة العائمة تم تقييسها عام 1985 **من قبل** (IEEE the Institute for Electrical and Electronic Engineers) ، هذا التمثيل يحدد أن الأرقام الحقيقية يمكن أن تمثل بأحد شكلين : الدقة المفردة single precision وتستخدم 32bit والدقة المضاعفة double precision وتستخدم 64bit وهو أكثر عمومية من الدقة المفردة .

لتمثيل قيمة حقيقية وفق هذا التقييس نجد.

مثال :

$22.625_{10}=10110.101_2=1.0110101_2*2^4$

وبهذا الشكل يخزن العدد ذو الفاصلة العائمة في الذاكرة كما يلي :

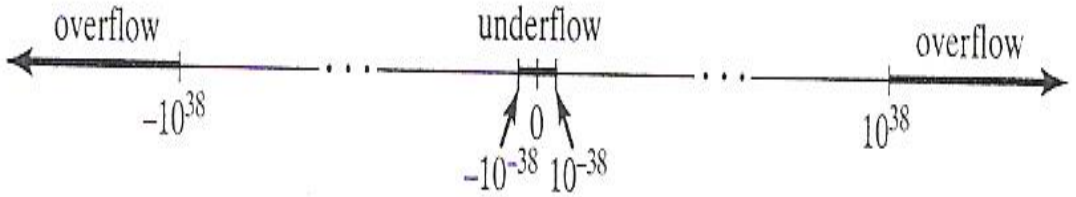
الخانة في أقصى اليسار هي للإشارة ، الخانات الثمانية التالية هي للأس ممثلة بالنمط bias والخانات الثلاث والعشرون في أقصى اليمين لتمثيل العدد أي :

0	1	0	0	0	0	0	1	1	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
+	exponent								Mantissa																							

أما في تمثيل القيم ذات الدقة المضاعفة فيستخدم 11bit للأس (بانحراف مقداره 1024) و53 خانة للقيمة .

وبما أن التمثيل الثنائي للأس قد يتطلب عدداً من الخانات أكبر من المتاح فإن هذا سيطرح أيضاً بالنسبة للأعداد الحقيقية مشكلة الطفحان overflow (في حال القيم ذات الأس الكبير جداً) وبالمثل في حال الأس السالب الصغير جداً نواجه حالة underflow .

في حالة القيم من النوع float (حيث تخصص 8bit لتمثيل الخانة) يحصل الطفحان للقيم الموجودة خارج المجال (-10^{38} to 10^{38}) ويحصل underflow للقيم الموجودة ضمن المجال (10^{-38} to -10^{-38}) ، يبين الشكل التالي هذه الحالة :



3- المصفوفات 1

Arrays

بالإضافة إلى أنماط المعطيات البسيطة ، تقدم أغلب لغات البرمجة عالية المستوى أنماط بيانات بنيوية أو تركيبية (structured data types) . في هذه الأنماط تكون القيم عبارة عن مجموعات من عناصر البيانات البسيطة أو التركيبة.

في هذه الفقرة ندرس أكثر هذه الأنماط شيوعاً وهو المصفوفة array . يمكن تعريف المصفوفة كنمط معطيات مجرد (ADT) كما يلي :

المصفوفة كنمط بيانات مجرد

مجموعة من عناصر البيانات : تتالي من العناصر ذو حجم ثابت (مجموعة مرقمة) وجميعها من نفس النوع .

العمليات الأساسية : الوصول المباشر direct access لكل مكان في المصفوفة وبالتالي فإن القيم يمكن الحصول عليها أو تخزينها في هذه الأماكن.

- إن للمصفوفة عدداً محدداً من العناصر أي ثابت إلا إذا عرفت ديناميكياً.
- نوعاً واحداً من المعطيات.
- عمليات التعديل عليها مثل الإضافة أو الحذف تتطلب الكثير من عمليات الإزاحة، إلا إذا كان في نهاية المصفوفة فهي سريعة جداً أي زمن ثابت بتعقيد O(1) .
- المصفوفات ليست أغراضاً قائمة بذاتها (self-contained objects) لأن العمليات على مصفوفة تتطلب على الأقل عنصري بيانات على الأقل (وهذا ما سنرى حلولاً له لدى دراسة البرمجة غرضيه التوجه OOP) ..

قد تكون ببعد أو عدة أبعاد:

المصفوفات أحادية البعد One-Dimensional Arrays :

التصريح في لغة C++ عن المصفوفة يجب أن يحدد صفتين أساسيتين للمصفوفة : عدد العناصر ونوع العناصر وطبعاً اسم لها ويتم كما يلي :

Arrays

3- المصفوفات 2

التصريح عن المصفوفات في C++

الشكل العام :

```
element_type  array_name[capacity];
element_type array_name[capacity]={initializer_list};
```

حيث :

element_type : هي أي نوع .

array_name : اسم المصفوفة المعرفة .

capacity : عدد القيم التي يمكن أن تحتويها المصفوفة ، وهو اختياري في الشكل الثاني .

initializer_list : مجموعة من القيم من النوع المحدد تفصل فيما بينها فواصل.

الهدف :

تطلب من المعالج القيام بحجز قطاع من مو اقع الذاكرة المتتالية لاحتواء capacity عنصر من النمط element_type وتسميتها بالاسم array_name ، تأخذ عناصر المصفوفة في هذه الحالة الدلائل من 0 وحتى capacity-1 .

في الشكل الثاني يتم إعطاء عناصر المصفوفة القيم الابتدائية المحددة في initializer_list . في حال كون عدد القيم الابتدائية المحددة أقل من حجم المصفوفة فإن باقي المو اقع تأخذ القيمة 0 ، أما إذا كان عدد القيم المحددة أكبر من حجم المصفوفة فإن ذلك يعتبر خطأ .

3- المصفوفات 4

مصفوفات المحارف character arrays :

تخزن السلاسل المحرفية في C++ في مصفوفات من المحارف ، ويمكن تعريف هذه المصفوفات وتهيئتها بنفس الطريقة المتبعة في حالة مصفوفات الأرقام ، على سبيل المثال :

```
const int NAME_CAPACITY=10;
char name[NAME_CAPACITY]={'j','o','h','n',' ','D','o','e'};
```

ويمكن أيضاً تهيئتها باستخدام الشكل التالي :

```
char name[name_capacity]="john Doe";
```

كلا التعريفين يبنيان name كمصفوفة سعتها 10 وحجمها 8 .
إذا كان عدد القيم أقل من سعة المصفوفة فإن باقي المواقع المحجوزة في المصفوفة تملأ بالمحرف الصفري null character أو '\0' . تستخدم توابع معالجة السلاسل المحرفية هذا المحرف لإيجاد نهاية السلسلة كما يلي :

```
for(int i=0;name[i]!='\0';i++)
cout<<name[i];
```

أخطاء تجاوز المجال out-of-range errors : من المهم أن نذكر بأنه لا يتم إجراء أي عملية تحقق من أن الدلائل index تقع ضمن المجال المعرف من خلال التصريح عن المصفوفة وبالتالي فإن إعطاء الدليل قيمة تتجاوز حدود المصفوفة قد يولد نتائج مشوشة ،

3- المصفوفات 5

المصفوفات المتعددة الأبعاد multidimensional arrays :

التصريح عن المصفوفات متعددة الأبعاد في C++

الشكل العام :

element_type array_name[DIM_1][DIM_2].....[DIM_n]={initializer_list};

حيث :

element_type : هي أي نوع .

array_name : اسم المصفوفة المعرفة .

DIM_i : قيمة صحيحة ثابتة غير سالبة .

initializer_list:هي قائمة بالقيم الابتدائية التي يمكن تخزينها في مو اقع المصفوفة (اختياري).

الهدف :

تعريف كائن متعدد الأبعاد عناصره من النوع element_type يكون فيه DIM_1,DIM_2.....DIM_n عدد العناصر في كل بعد وتكون initializer_list القيم المخزنة في المصفوفة حسب الصفوف (سطراً سطراً) .

، لنأخذ بعين الاعتبار المثال التالي الذي يتعامل مع مصفوفة ثنائية البعد من ثلاثة أسطر وأربعة أعمدة عناصرها من النوع المحرفي مصرح عنها كما يلي :

3- المصفوفات 6

```
char t[3][4];
```

أو بشكل مكافئ :

```
typedef char Row[4];
typedef Row Table[3];
Table t;
```

ومخزن في هذه المصفوفة القيم التالية :

A	B	C	D
E	F	G	H
I	J	K	L

بما كل محرف يحتاج إلى 1byte لتخزينه ، فإن التصريح السابق يطلب من المنقح أن يحجز 12 موقع ذاكرة متجاورة لتخزين عناصر المصفوفة .

4- السجلات 1

- ماذا لو كانت أنماط البيانات Pre-defined data types غير ملائمة لطبيعة الأنماط في المسألة ؟.
- الحاجة لأنماط البيانات بحيث تملك خاصية إخفاء المعلومات Information hiding، أي إمكانية قيام الأغراض بالاتصال ببعضها البعض من خلال واجهات معرفة ومحددة بدون أمكانية معرفة كيف تم تطويرها (بمعنى آخر إخفاء التفاصيل المتعلقة بها عن الأغراض الأخرى)؟.
- المسألة تحتاج لعدة أنماط وليس لنوع واحد.
  - ✓ الحل يكون بإنشاء نمط بيانات جديد لنمذجتها.
  - ✓ الاعتماد على مفهوم البرمجة غرضية التوجه والذي يقوم على أساس الاهتمام بالأغراض Objects المأخوذه من السجلات struct أو الأصناف Classes .
  - ✓ إذا كان النمط المعرف متعدد الصفات ويملك عدة أفعال خاصة بالتعامل معه، يجب استخدام الأصناف في نمذجته والمطوره عن السجل struct في لغة C.

- السجل: مجموعة مرتبة من العناصر، قد تكون من أنواع محتلفة، وتعرف بعناصر السجل واحياناً أعضاء السجل members أو حقول السجل fields. ويمكن الوصول إلى كل عنصر من السجل بشكل مباشر وبالتالي يمكن التعامل مع القيم أو تخزينها في هذا العنصر.

- يختلف السجل عن المصفوفة كون عناصر المصفوفة يجب أن تكون من النوع نفسه.
- تعريف السجل: يمكن أن يتم تعريف السجل بالصيغة العامة المبسطة التالية:

```
struct structName
{    Declarations of members   };
```

- يبتدئ تعريف السجل بالكلمة المفتاحية **struct** متبوعة باسم السجل (والذي يمكن أن يكون أي اسم يختاره المبرمج وفقاً لقواعد تسمية المعرفات التي تعرفنا عليها في مساقات سابقة).
- تعريف عناصر السجل بين قوسي بداية ونهاية كتلة برمجية حيث يمكن للعناصر أن تكون من أي نوع.
- ينتهي تعريف السجل بفاصلة منقوطة.
- حيث يتم وضع هذا التعريف ضمن قسم التضمينات للبرنامج. كما يمكن تعريف السجل بصيغة أكثر تعقيداً وتركيباً كما يلي:

```
struct structName
{ private:     Declarations of private members
public:     Declarations of public members    };
```

- الكلمتان المفتاحيتان private و public محددا وصول access specifiers يعلنان أن ما هو معرف بعدهما يمكن الوصول إليه من قبل أجزاء الغرض نفسه (private) أو من قبل جميع أجزاء البرنامج (public)على الترتيب.
- بعد تعريف السجل فإنه يصبح نوع بيانات ومن الممكن البدء باستخدامه من خلال التصريح عن متحولات من هذا النوع، حيث يأخذ التصريح الشكل العام التالي:

1. structName objStruct_name;
2. structName objStruct_name={initializer_list};

في الشكل الأول، تم التصريح عن متحول يدعى objStruct_name من النمط الذي قمنا بتعريفه حيث يطلب إلى المنقح أن يقوم بحجز قطاع من مو اقع الذاكرة لتخزين الأغراض من الأنماط المحددة وإطلاق الاسم objStruct_name على هذا القطاع.

في الشكل الثاني، تكون أنماط القيم المحددة في لائحة القيم الابتدائية initializer_list متو افقة مع أنماط الأعضاء المقابلة لها في التصريح عن السجل وتستخدم لإعطاء قيم ابتدائية لعناصر البيانات للمتحول من نوع السجل .

ملاحظة هامة: يمكن لأعضاء السجل أن تكون عناصر بيانات أو تو ابع خاصة بالتعامل مع عناصر البيانات تلك، إلا أننا سنقتصر في عرضنا المقتضب هذا لموضوع السجلات على تعريف سجلات أعضاؤها هي عناصر بيانات فقط، كما أننا سنقتصر على تعريف سجلات بدون استخدام محددات الوصول حيث يكون محدد الوصول الافتراضي في هذه الحالة هو public أي يمكن الوصول لأعضاء السجل من قبل كامل أجزاء البرنامج، وسيصار عند الانتقال لموضوع الأصناف التوسع أكثر في هذا الأمر.

4- السجلات 4

- كائن التاريخ له الشكل :

December	31	1999
month	day	year

بما أن مثل هذه الأغراض لا يمكن تمثيلها ببساطة باستخدام الأنماط البسيطة ، قدمت لغة البرمجة C++ مفهوم السجلات structures لإنشاء أنماط جديدة لها عدة صفات ، وبالتالي يمكن تعديل البند الأول من المنهجية المعروضة سابقاً لتصميم البرنامج كما يلي :

1. تحديد الأغراض في المسألة .
   a. إذا كانت الأنماط المسبقة التعريف غير ملائمة لنمذجة الغرض قم بإنشاء نمط جديد لنمذجتها .
   b. إذا كان الغرض متعدد الصفات ، قم ببناء سجل لمثيل الغرض من ذلك النمط .

يمكن تعريف السجل structure كنمط بيانات مجرد كما يلي :

السجل كنمط بيانات مجرد

<u>مجموعة من عناصر البيانات :</u>

تتالي ذو حجم ثابت (مجموعة مرتبة) من العناصر ، ليس بالضرورة أن تكون من نفس النوع ، عناصره تدعى أعضاء members أو حقول fields .

<u>العمليات الأساسية :</u>

الوصول المباشر إلى كل عضو في السجل وبالتالي فإن القيم يمكن الحصول عليها أو تخزينها في هذا العضو.

Structures

4- السجلات 5

- يمكن التصريح عن النمط Date للتعامل مع التواريخ كما يلي :

```
struct Date
{
  string month;   //name of month
  int day,year;   //day number and year number
};
```

- يمكن تعريف غرض نهاية القرن endofCentury من النوع Date وتمرير قيم ابتدائية لأعضائه كما يلي :

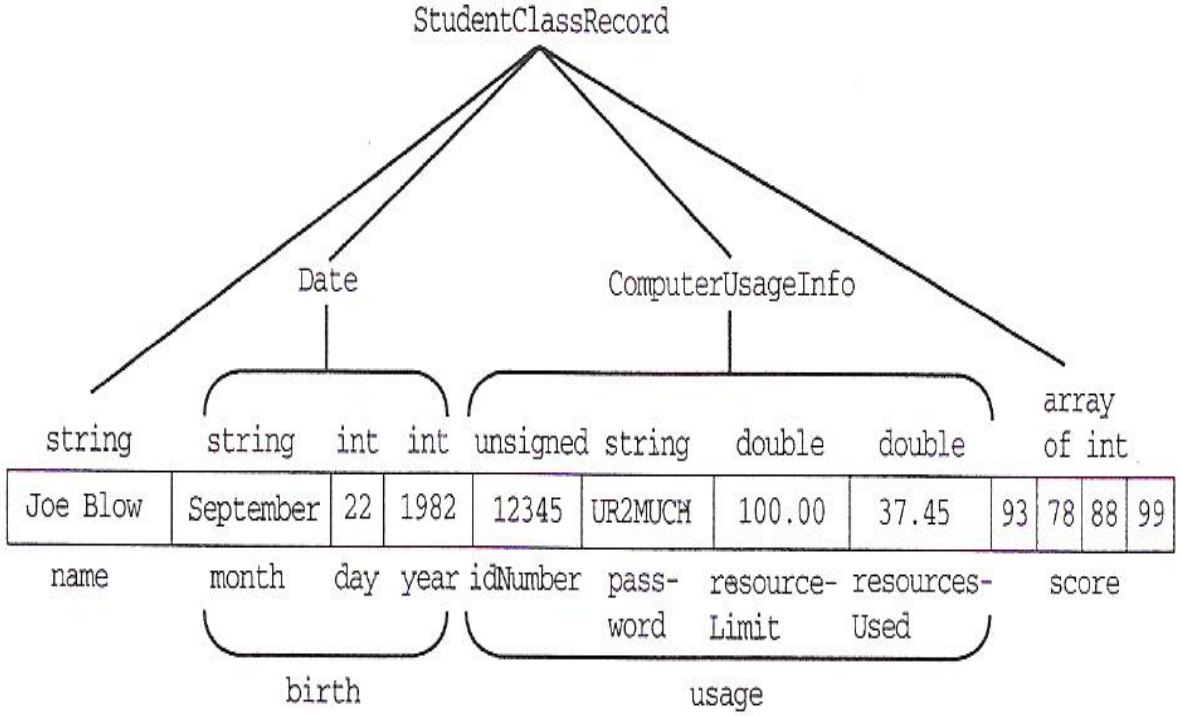
```
Date endofCentury ={"December",31,1999};
```

endofCentury	December	31	1999
	month	day	Year

- بما أن أعضاء السجل يمكن أن تكون من أي نمط ، فإنها ليس بالضرورة أن تكون من أنماط بسيطة (كما في جميع الأمثلة السابقة) وإنما يمكن أن تكون من أنماط بيانات بنيوية كالمصفوفات أو سجلات أخرى .
- تدعى السجلات التي تحتوي في تعريفها سجلات أخرى باسم السجلات المتداخلة nested structures **وتدعى أحياناً السجلات الشجرية** hierarchical structures

Structures

4- السجلات 6



العمليات على السجلات struct operations :

تتيح معاملات الوصول للأعضاء member access operators المعرفة في لغة C++ إمكانية الوصول إلى أعضاء السجل . أحد أهم هذه المعاملات معامل النقطة dot operator ويستخدم كما يلي :

```
Struct_object.member_name;
```

فيما يلي بعض الأمثلة :

- حساب المكافئ المئوي لـ temp :

```
Temperature tempCels;
tempCels.degrees=(temp.degrees -32.0)/1.8;
tempCels.scale='C';
```

4- السجلات 7

- إدخال قيم إلى أعضاء الغرض currentDate :

```
cin>>currentDate.month>>currentDate.day>>currentDate.year;
```

- مجموع درجات الاختبارات للغرض aStudent :

```
int sum=0;
for (int i=0;i<4;i++)
    sum+=aStudent.score[i];
```

- طباعة شهر الولادة والرقم المعرف للغرض aStudent

```
cout<<"birth month: "<<aStudent.birth.month<<endl;
cout<<"id number : "<<aStudent.usage.idNumber<<endl;
```

يمكن إجراء عملية نسخ سجل إلى آخر من خلال إجراء مجموعة من عمليات الإسناد المتتالية إلى أعضاء السجل ، كما يمكن إجراء هذه العملية ببساطة أكبر من خلال تعليمة إسناد وحيدة من الشكل :

```
struct_variable=struct_object;
```

5- الأصناف 1

Classes

تم تعريف السجلات structs في لغة البرمجة C لتعريف بنى معطيات مركبة من قبل المستخدم ، وقد أبقت لغة C++ السجلات ولكن أضافت أسلوباً جديداً لتحقيق هذا الهدف وهو الأصناف classes .

هناك العديد من أوجه التشابه بين الأصناف والسجلات ، أهمها :

- كلاهما يمكن أن يستخدما لنمذجة الأغراض ذات الخصائص المختلفة والممثلة كعناصر بيانات . وبالتالي فهما يستخدمان لتنظيم مجموعات البيانات غير المتجانسة .
- كلاهما يملكان عموماً نفس الصيغة syntax .

أما أهم أوجه الاختلاف فهي :

- أعضاء الصنف تكون خاصة private (افتراضياً) أي لا يمكن الوصول إليها بالتوابع غير الأعضاء في الصنف ما لم يتم التصريح عنها بشكل صريح عن أنها عامة public . أما أعضاء السجل تكون عامة (افتراضياً) أي يمكن الوصول إليها من قبل التوابع غير الأعضاء ، وفي C++ يمكن التصريح عنها بشكل صريح لتكون خاصة .

إذاً السجلات والأصناف في لغة C++ هي عبارة عن توسيع لمفهوم السجلات التقليدي في لغة C . فقد أضافت لغة C++ إلى مفهوم السجل إمكانية تعريف التوابع كأعضاء للسجل (بالإضافة إلى البيانات) وذلك لتعريف العمليات على هذه البيانات ، وهذه الإضافة مهمة للأسباب التالية:

- تمنحنا طريقة لبناء أنماط المعطيات المجردة ADT's يكون فيها عناصر البيانات و توابع العمليات الأساسية مغلفة encapsulated في بنية واحدة
- تمهد لأسلوب جديد في البرمجة وهو الأسلوب الغرضي التوجة object-oriented بدلاً من الأسلوب الإجرائي procedural .

Classes

5- الأصناف 2

التصريح عن الأصناف class declarations :

الشكل الأول :

```
class  ClassName
{ public :
   declarations of public members
private:
   declarations of private members
};
```

الشكل الثاني :

```
class  ClassName
{ private:          //this keyword  is optional
   declarations of private members
public :
   declarations of public members
};
```

ملاحظات :

1. افتراضياً ، تكون الأعضاء خاصة private ، لذلك نرى في الشكل الثاني أن محدد الوصول (private:) اختياري ولكنه يذكر عادة من أجل الوضوح . في الشكل الأول نحتاج إلى المحدد (public :) لتحديد الجزء العام من الصنف والمحدد (private :) لتحديد نهاية الجزء العام وبداية الجزء الخاص .
2. يمكن أن يحتوي تعريف الصنف على عدة أجزاء خاصة وعدة أجزاء عامة (على الرغم من ذلك ليس شائع الاستخدام) ، في هذه الحالة الكلمة private تحدد بداية كل جزء خاص والكلمة public تحدد بداية كل جزء عام .
3. توضع البيانات الأعضاء عادة في الجزء أو الأجزاء الخاصة للصنف والتوابع الأعضاء في الجزء أو الأجزاء العامة .
4. يمكن الوصول إلى الأعضاء الخاصة من قبل التوابع الأعضاء في الصنف فقط (أو من قبل التوابع الأصدقاء friend functions كما سنرى لاحقاً).
5. يمكن الوصول إلى الأعضاء العامة من قبل كل من التوابع الأعضاء وغير الأعضاء في الصنف .

إن ترتيب الأجزاء العامة والخاصة ليس مهماً ،بعض المبرمجين يفضلون وضع الجزء العام أولاً البعض الآخر يفضلون الجزء الخاص أولاً . سنعتمد في هذا الكتاب الشكل الأول في بناء الأصناف .

توضع تصريحات الأصناف عادة في مكتبة ، وتدعى عندئذ مكتبة الصنف class library . تتألف مكتبة الصنف تقليدياً من ملف رأسي يدعى ClassName.h وملف التعريف ClassName.cpp .

تدعى النسخة (أو المثل instance) عن صنف ما بالاسم غرض object :

```
ClassName object_name;
```

Classes

5- الأصناف 4

يمكن الوصول إلى الأعضاء العامة ضمن الغرض باستخدام معامل النقطة dot operator :

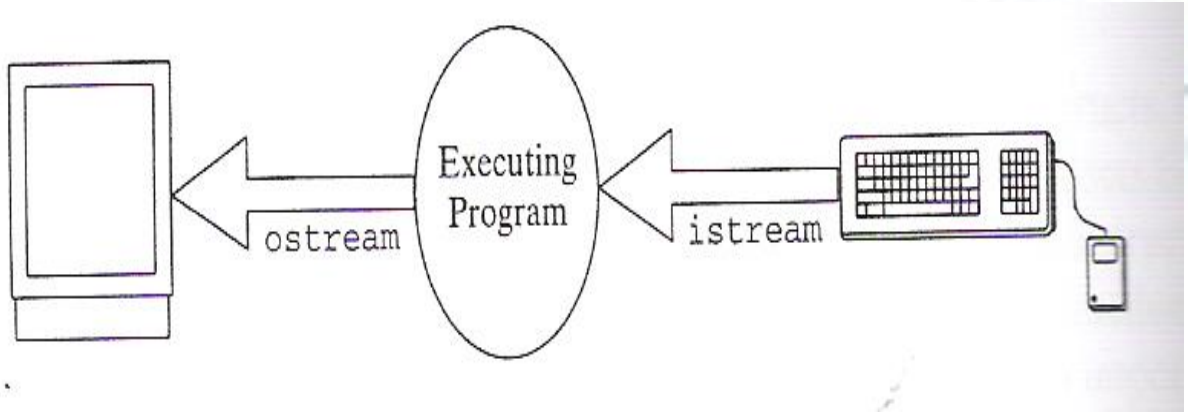
Object_name.member_name

أو من خلال السهم إذا عرف الغرض بالمرجع كما سنرى لاحقاً

مثال : أصناف الدخل/الخرج القياسية في C++ C++'s standard I/O classes

تحتوي المكتبة القياسية في لغة C++ العديد من الأصناف وقوالب الأصناف المختلفة . أكثر هذه الأصناف شيوعاً أصناف الدخل/الخرج أي istream ، ostream و iostream . هذه الأصناف هي عبارة عن نماذج عامة للدخل والخرج يستطيع المبرمجون استخدامها بدون أن يشغلوا أنفسهم بتفاصيل كيفية الحصول على البيانات من لوحة المفاتيح إلى البرنامج أو من البرنامج إلى الشاشة .

يمكن النظر إلى الغرض istream على أنه دفق من المحارف يتدفق من أي جهاز دخل إلى برنامج تنفيذي ، والغرض ostream كدفق من المحارف يتدفق من برنامج تنفيذي إلى أي جهاز خرج ، كما يبين الشكل التالي :



الصنف المضاف مؤخراً iostream يدعم كلاً من تدفق الدخل وتدفق الخرج .

هذه الأصناف معرفة في المكتبة <iostream> التي تعرف أيضاً كائنات التدفق التالية :

Cout تدفق الخرج القياسي من النمط char .

Cerr تدفق الخرج القياسي غير المخزن unbuffered لرسائل الخطأ .

Clog تدفق الخرج القياسي المخزن buffered لرسائل الخطأ .

Cin تدفق الدخل القياسي من النمط char .

Classes

5- الأصناف 5

الصنف istream		الصنف ostream	
معامل الدخل .	>>	معامل الإخراج .	<<
لإدخال محرف واحد أو سلسلة من المحارف .	get( )	إخراج محرف واحد .	put( )
إدخال سطر .	Getline( )	إخراج عدد محدد من المحارف .	write( )
عدد المحارف المدخلة بواسطة آخر تابع إدخال .	gcount( )	إفراغ الـbuffer .	flush( )
تجاوز الحروف .	Ignore( )	تحديد الوقع الحالي .	seekp( )
النظر إلى المحرف التالي المراد قراءته .	peek( )	الحصول على الموقع الحالي .	tellp( )
إدخال محرف أو عدد محدد من المحارف .	read ( )	معامل الإخراج .	<<
إدخال عدد محدد من المحارف على الأكثر .	Readsome( )	معامل الإخراج .	<<
إعادة محرف إلى الـbuffer .	putpack( )	إخراج محرف واحد .	put( )
إعادة آخر محرف تمت قراءته .	unget( )		
تحديد الموقع الحالي .	seekg( )		
الحصول على الموقع الحالي .	tellg( )		

7 - السلاسل المحرفية كأنماط بيانات مجردة 1

السلاسل المحرفية كنمط بيانات مجرد

مجموعات من عناصر البيانات :

تتالي منتهي من المحارف المختارة من مجموعة معرفة من المحارف .

العمليات الأساسية :

- الدخل والخرج (input & output) : قراءة وعرض السلسلة المحرفية.
- الطول (length): معرفة عدد المحارف في السلسلة المحرفية .
- المقارنة (compare): تحديد فيما إذا كانت سلسلتان متطابقتين ، أو فيما إذا كانت إحداهما تسبق الأخرى في الترتيب .
- الدمج (concatenate) : جمع سلسلتين معاً .
- النسخ (copy) : نسخ سلسلة أو جزء من سلسلة إلى سلسلة أخرى .
- البحث (find) : تحديد موقع سلسلة أو محرف ضمن أخرى .
- إدراج أو حشر (insert) : إدراج أو حشر سلسلة ضمن أخرى .
- حذف (delete) : حذف جزء من سلسلة .
- استبدال (replace) : استبدال جزء من سلسلة بسلسلة أخرى .

والعمليات الأساسية التي يمكن تطبيقها واستنتاج باقي العمليات عليها هى:

الطول( )length, الفهرسة ( )Index,الدمج ( )Concatenate,النسق الفرعي ( ) SubString,

النمط string في C++: 1

the C++ string type

قامت لغة C++ بإصدار تقييس للصنف string يتضمن مجموعة كبيرة من التوابع المسبقة التعريف لإجراء العمليات على السلاسل المحرفية وضمنته في المكتبة <string> .

فيما يلي سنتعرف على أهم العمليات المعرفة في الصنف string .

التعريف والبناء definitions and constructors : هناك العديد من الطرق المختلفة لتعريف الغرض سلسلة محرفية ، وبالإضافة إلى الباني الافتراضي هناك العديد من التوابع البانية التي تقوم بتهيئة الغرض بسلسلة محرفية أخرى، فمثلاً التعاريف التالية :

string s;	بناء السلسلة s كسلسلة فارغة .
string s(str_ca);	تهيئة سلسلة s بنسخة من سلسلة أو مصفوفة محارف str_ca
string s(ca,n);	تهيئة السلسلة المحرفية s بنسخة من الـ n محرف الأولى من مصفوفة المحارف ca
string s(str,pos,n);	تهيئة السلسلة المحرفية s لاحتواء نسخة من n محرف من السلسلة المحرفية str بدءاً من الموقع pos ، إذا كانت n كبيرة جداً يتم نسخ المحارف بدءاً من pos وحتى نهاية السلسلة .
string s(n,ch);	تهيئة السلسلة المحرفية s لاحتواء n نسخة من المحرف ch .

النمط string في C++: 2

the C++ string type

بعض التحقيقات للسلاسل المحرفية تستخدم مصفوفات القيم من النوع char لتمثيل السلاسل المحرفية القصيرة وبنى التخزين الديناميكية dynamic storage structures للسلاسل المحرفية الأطول التي تتسع وتتقلص بسبب عمليات الإضافة والحذف المتكررة .

تحتوي المكتبة string على مجموعة من التوابع التي تتيح التعرف على البنية التخزينية للسلسلة والمعلومات المتعلقة بها ، نذكر منها :

s.capacity( )	يعيد السعة التخزينية للسلسلة المحرفية s .
s.size( ),s.length( )	يعيد طول السلسلة المحرفية .
s.empty( )	يعيد true إذا لم تحتوي s أي محرف .
s.max_size( )	تعيد الطول الأعظمي للسلسلة s .

حددنا العمليات الأساسية التي تدمج أو تعديل السلاسل المحرفية كما يلي : الدمج ، الحشر ، الحذف والاستبدال ..

فيما يلي سرداً بأهم العمليات حيث s,t,str من النوع string و ca من النوع مصفوفة محارف و str_ca هي سلسلة من مصفوفات المحارف و n,n1,pos1,pos2 هي أعداد صحيحة :

توابع تعديل السلاسل

: string modifiers

s+t , t+s	تعيد ناتج دمج السلستين s ، t . حيث t يمكن أن تكون سلسلة محرفية ، مصفوفة محارف أو محرف .
s.append(str_ca)	إلحاق str_ca في نهاية السلسلة s وإعادة s .
s.append(ca,n)	إلحاق الـ n محرف الأولى في ca بنهاية السلسلة s وإعادة s .
s.append(n,ch)	إلحاق n نسخة من المحرف ch إلى نهاية السلسلة s وإعادة s .
s.insert(pos,str)	حشر نسخة من السلسلة str في s عند الموضع pos وإعادة s .
s.insert(pos1,ste,pos2,n)	حشر نسخة من n محرف من السلسلة str بدءاً من الموضع pos2 إلى s في الموضع pos1 , إذا كانت n كبيرة جداً فإن المحارف ستنسخ فقط إلى نهاية السلسلة str . وإعادة s
s.insert(pos,ca,n)	حشر نسخة من الـ n محرف الأولى من caإلى s في الموضع pos ، في حال حذف n سيتم حشر كامل محارف السلسلة ca إلى s ، وإعادة s .
s.insert(pos,n,ch)	حشر n نسخة من المحرف ch إلى s في المرضع pos وإعادة s .
s.erase(pos,n)	حذف n محرف من s بدءاً من الموضع pos (افتراضياً 0) وإعادة s .
s.replace(pos1,n1,str)	استبدال سلسلة جزئية من s بطول n1 بدءاً من الموضع pos1 بالسلسلة str ، إذا كانت n1 كبيرة جداً ، فإن كافة المحارف إلى نهاية السلسلة ستستبدل . وإعادة s .
s.replace(pos1,n1,ca,n2)	استبدال سلسلة جزئية من s كما في السابق ولكن بالمحارف الـ n2 الأولى من ca وإعادة s .
s.swap(str)	التبديل بين محتويي s و str ولا يعيد أي شئ .
swap(str1,str2)	تبديل محتوى str1 و str2 ولا يعيد أي شئ .

7 - السلاسل المحرفية كأنماط بيانات مجردة 1

strings as ADTs

يمكن وضع خوارزمية تستخدم هذه الأغراض والعمليات لتحرير النصوص كما يلي :

1. خذ inFileName ؛ أنشئ outFileName ؛ و افتح المجاري inFile و outFile .
2. اعرض COMMAND_MENU .
3. اقرأ line من inFile واعرضه على الشاشة .
4. كرر العمليات التالية :
   a. قم بإنهاء عملية التكرار إذا تم الوصول إلى نهاية الملف .
   b. تلق command ، حوله إلى محرف كبير (uppercase) وقم بالعملية المناسبة :

L : عرض طول الـ line .

P : إيجاد وعرض position لـ str ضمن line .

I : حشر str1 في line عند position .

D : حذف numChars محرف بدءاً بـ position-1 .

R : إيجاد str1 في line عند position واستبدله بـ str2 .

N : إخراج line إلى outFile وقراءة line جديد من inFile .

Q : إخراج line إلى outFile ، وتكرار قراءة line جديد من inFile وإخراج line إلى outFile لحين الوصول إلى نهاية inFile . وإلا طباعة رسالة خطأ .

   a. إذا لم يتم الوصول إلى نهاية الملف اعرض السطر المكتوب .
1. اعرض رسالة تفيد بانتهاء عملية التنسيق .

وبالتالي يكون ملف الشيفرة بلغة C++ **الذي ينجز هذه العملية :**

strings as ADTs

```
#include "stdafx.h"
#include<iostream>
#include<fstream>
#include<string>
using namespace std;
int main( ){    const char COMMAND_MENU[]=
        "editing commands are :\n"
        " L     : Determine the length of current line \n"
        "P str : Find position (counting from 0) of string str\n"
        " I str : Insert string str\n"
        " D p n : Delete n characters begining at position p\n"
        " R str : Replace substring str with another string \n"
        " N     : Get next line of text\n"
        " Q     : Quit editing \n";
```

```
string inFileName,outFileName;
cout<<"enter the name of the input file : ";
getline(cin,inFileName);
outFileName=inFileName;
outFileName.append(".out");
ifstream inFile(inFileName.data());
ofstream outFile(outFileName.data());
if (!inFile.is_open() || !outFile.is_open())
{cerr<<"error in opening files "; exit(-1); }
cout<<COMMAND_MENU<<“\nenter an edi. Com. Foll.each pro.>\n";
char command;
string line,str1,str2;
int position,numChars;
```

strings as ADTs

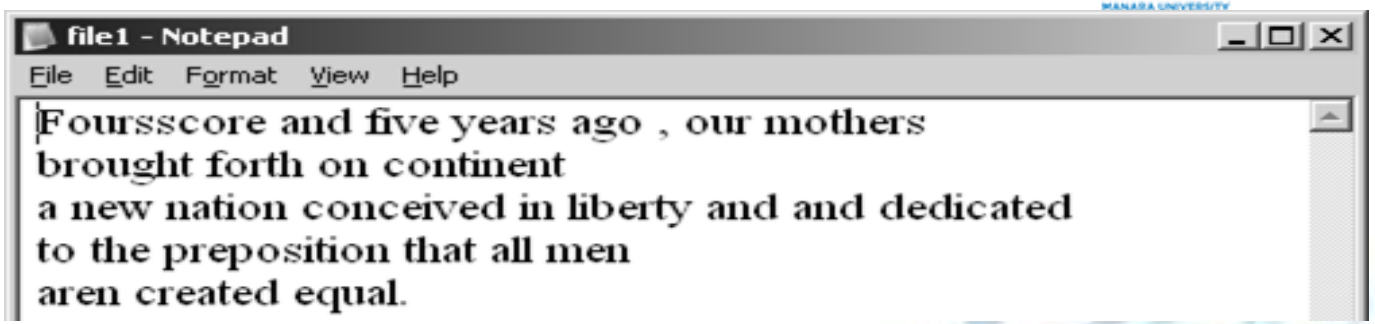
```
getline(inFile,line);
cout<<line<<endl;
for ( ; ; ){if (inFile.eof()) break;
      cout<<'>';                cin>>command;
      switch (toupper(command))
      {case 'L': cout<<"Length= "<<line.length()<<endl;break;
      case 'P':cin.ignore(1); getline(cin,str1);
            position=line.find(str1);
            if (position!=string::npos)
                  cout<<"position os "<<1+position<<endl;
            else  cout<<str1<<"not found\n"; break;
      case 'I': cin.ignore(1);getline(cin,str1);
            cout<<"insert where ? ";    cin>>position;
            line.insert(position-1,str1);break;
```

```
        case 'D':  cin>>position>>numChars;
        line.erase(position,numChars);   break;
    case 'R':  cin.ignore(1);  getline(cin,str1);
        position=line.find(str1);
        if (position==string::npos)
        {cout<<str1<<"not found\n";break;}
        cout<<"with what ?";
        getline(cin,str2);
         line.replace(position,str1.length(),str2); break;
    case 'N':outFile<<line<<endl;getline(inFile,line);
        cout<<"\nnext line :\n";   break;
    case 'Q':
        outFile<<line<<endl;
```

```
            for( ; ; )
                 {     getline(inFile,line);
                       if (inFile.eof()) break;
                       outFile<<line<<endl;
                 }           break;
            default:cout<< "***illegal command ***\n" <<
COMMAND_MENU << line << endl;          }
            if (!inFile.eof())
                  cout<<line<<endl;     }
      cout<<"\n*** editing complete ***\n";
system("pause");
      return 0;
}
```

strings as ADTs

7 - السلاسل المحرفية كأنماط بيانات مجردة 7



بتنفيذ هذا البرنامج على ملف الدخل file1.txt المبين في الشكل التالي :

نحصل على شاشة الخرج التالية :

```
enter the name of the input file : file1.txt
editing commands are :
 L    : Determine the length of current line
 P str : Find position (counting from 0) of string str
 I str : Insert string str
 D p n : Delete n characters begining at position p
 R str : Replace substring str with another string
 N    : Get next line of text
 Q    : Quit editing
enter an editing command following each prompt >
Foursscore and five years ago. our mothers
>
```

انتهت المحاضرة الثالثة