المركز الوطني للمتميزين
NATIONAL CENTER FOR THE DISTINGUISHED
/NCDSyria

تقرير حلقة بحث بعنوان :

حاويات STL

تقديم الطالب : محمود قره فلاح
الصف: الحادي عشر
تاريخ : ١٤-١-٢٠١٥
اشراف: أمجد طه

ملخص

يقدم هذا البحث شرحاً لأشهر الحاويات الموجودة في مكتبة القوالب المعيارية STL.

الفهرس

فهرس الصور

مقدمة

تعتبر لغة C++ واحدة من أكثر لغات البرمجة انتشاراً و استخداماً في الوسط الطلابي الأكاديمي و في وسط تطوير البرمجيات و إنتاجها. وهي ما تزال تحتل مواقع الصدارة مقارنة مع غيرها من لغات البرمجة، رغم ظهور لغات أخرى Java و C#.

جرى بناء هذه المكتبة على يد السيدين Alexander Stepanov و Merg Lee من شركةHewlen Packard اعتماداً على خبراتهما في مجال البرمجة النمطية. كما ساهم السيد David Musser بكثير من الإضافات. وسوف نرى أنَّ هذه المكتبة مصممة بشكل ذكي من أجل المساهمة في تحسين الأداء و لتحقيق مرونة عالية عند الاستخدام.

تتضمنSTLالعديد من بنى المعطيات الفعالة والقابلة لإعادة الاستخدام إضافة إلى الخوارزميات اللازمة للتعامل مع هذه البنى.

تحوي STL ثلاث مكونات أساسية: الحاويات و التكرارات و الخوارزميات. وسوف نتحدث عن معظم الحاويات و أشهرها في STL.

و كما نعلم، فإن العديد من المبرمجين يواجهون صعوبات و تحديات كثيرة تعترض طريقهم البرمجي. لذلك، سعى العديد من المبرمجين لحل مثل هذه المشاكل، إلى أن وصلوا إلى مكتبة القوالب المعيارية Standard Template Library (STL).

وكما يبدو، فإن المشاكل البرمجية كثيرة، والحلول التي تنهي لنا مشاكلنا لا تزال مبهمة عند أغلب المبرمجين. ومن ضمن تلك الحلول مكتبة القوالب المعيارية STL.

هذا بالإضافة إلى أن العديد من الناس يواجهون صعوبات في التعامل مع STL، لذلك قمنا بشرح بشكل مبسّط و شيّق عن أهم و أشهر الحاويات الموجودة في STL.

الحاويات Containers:

الحاويات: هي بنى معطيات قادرة على تخزين الأغراض من أي نمط. وسوف نرى لاحقاً أنه يوجد ثلاثة فئات من الحاويات: حاويات السلاسل، حاويات التجميع، وحاويات التحويل.

تُعْرَف حاويات السلاسل بأنها بنى معطيات خطية، أما حاويات التجميع، فهي عبارة عن بنى معطيات غير خطية التي يمكن فيها تخزين أزواج من القيم.

و تشكل حاويات السلاسل و حاويات التجميع ما يسمّى بالحاويات من الدرجة الأولى.

كما يرتبط بكل حاوية من حاويات STL عدد من التوابع الأعضاء. كما جرى تعريف مجموعة جزئية من هذه التوابع الأعضاء مع جميع الحاويات التابعة لـ STL. كما في توابع vector و list و deque.

وتقسم حاويات الدرجة الأولى إلى حاويات السلاسل و حاويات التجميع. كما يوجد أربعة أنماط إضافية من الحاويات، و تعرف بأشباه الحاويات كالمصفوفات و سلاسل الحروف string، والبنية bitset للتعامل مع مجموعة من قيم الأدلة flag values والبنية valarray لتنفيذ عمليات رياضية عالية السرعة على الأشعة و المصفوفات.

تعتبر هذه الأنماط الأربعة أنماطاً لأشباه الحاويات بسبب إظهارها لقدرات مشابهة لتلك المتوفرة مع حاويات الدرجة الأولى، لكنها لا تمتلك جميع مقدرات هذا النوع من الحاويات.

حاول مصممو STL تصميم الحاويات بحيث أن توفر وظائف متشابهة، حيث هناك العديد من التوابع (كالتابع العضو size) الذي نستطيع استخدامه في كافة الحاويات. ولكن هناك بعض التوابع التي لا يمكن تطبيقها إلا على بعض الحاويات المتشابهة.

أنواع الحاويات:

كما ذكرنا سابقاً، فإن الحاويات تقسم إلى حاويات السلاسل، وحاويات التجميع، وحاويات التحويل.

وسوف نتحدث عن حاويات السلاسل و حاويات التحويل.

حاويات السلاسل:

يتوفر في STL ثلاثة حاويات للسلاسل وهي: list, deque, vector. تعتمد كل من vector و deque على المصفوفات. أما list فتقوم ببناء قائمة مرتبطة مشابهة للصف List الذي لسنا بصدد ذكره الآن.

وسوف نتحدث في هذا الفصل عن vector و deque:

vector:

يعد vector من أشهر الحاويات، كما يعتبر إعادة لتعريف Array. كما يمكن له أن يتغير ديناميكياً.
يوفر vector بنية معطيات متراصة ضمن الذاكرة. مما يمكّننا من الوصول إلى أي عنصر من العناصر باستخدام [ ]. حيث يمكن التعامل معه كما نتعامل مع المصفوفات.

و عندما يستهلك vector الذاكرة المتاحة له، فإنه يقوم بحجز ذاكرة أكبر، ويقوم بنسخ القيم الأصلية للعناصر إليها.

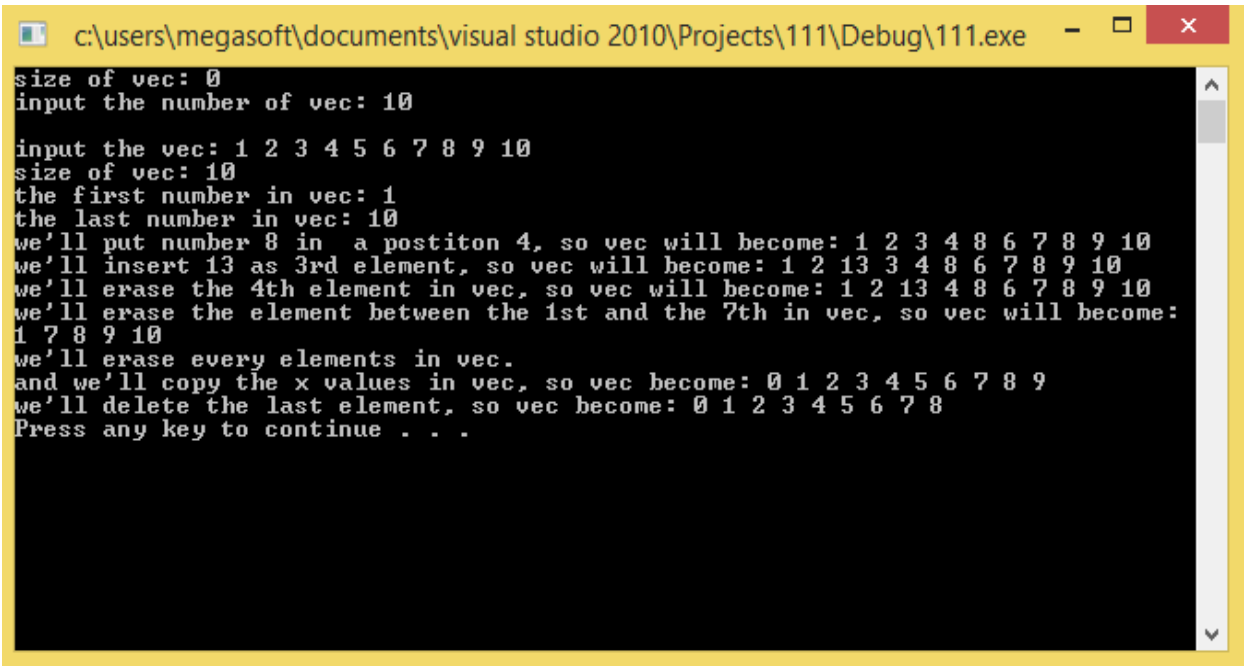
و لدينا المثال الآتي الذي يوضّح بعض أهم النقاط المتعلقة بـ vector:

```
1. #include<iostream>
2. #include<vector>

3. using namespace std;
4. int main(){
   4.1.    int x[]={0,1,2,3,4,5,6,7,8,9};
   4.2.    std::vector<int>vec;
   4.3.    cout<<"size of vec: "<<vec.size()<<endl;
   4.4.    cout<<"input the number of vec: ";
   4.5.    int n,v;
   4.6.    cin>>n;
   4.7.    cout<<"\ninput the vec: ";
   4.8.    for(int i=0;i<n;i++){
   4.9.    cin>>v;
```

```
4.10.  vec.push_back(v);
4.11.  }
4.12.  cout<<"size of vec: "<<vec.size()<<endl;
4.13.  cout<<"the first number in vec:
   "<<vec.front()<<"\nthe last number in vec:
   "<<vec.back()<<endl;
4.14.  cout<<"we\'ll put number 8 in  a postiton 4,
   so vec will become: ";
4.15.  vec[4]=8;
4.16.  for(int i=0;i<n;i++)
  4.16.1.  cout<<vec[i]<<" ";
4.17.  cout<<endl;
4.18.  cout<<"we\'ll insert 13 as 3rd element, so vec
   will become: ";
4.19.  vec.insert(vec.begin()+2,13);
4.20.  for(int i=0;i<n+1;i++)
  4.20.1.  cout<<vec[i]<<" ";
4.21.  cout<<endl;
4.22.  cout<<"we\'ll erase the 4th element in vec, so
   vec will become: ";
4.23.  vec.erase(vec.begin()+3);
4.24.  for(int i=0;i<n;i++)
  4.24.1.  cout<<vec[i]<<' ';
4.25.  cout<<endl;
4.26.  cout<<"we\'ll erase the element between the
   1st and the 7th in vec, so vec will become: ";
4.27.  vec.erase(vec.begin()+1,vec.begin()+6);
4.28.  for(int i=0;i<n-5;i++)
  4.28.1.  cout<<vec[i]<<" ";
4.29.  cout<<endl;
4.30.  cout<<"we\'ll erase every elements in vec.\n";
4.31.  vec.erase(vec.begin(),vec.end());
4.32.  cout<<"and we\'ll copy the x values in vec, so
   vec become: ";
4.33.  if(vec.empty()==1){
  4.33.1.  vec.insert(vec.begin(),x,x+10);
  4.33.2.  for(int i=0;i<10;i++)
    4.33.2.1.  cout<<vec[i]<<' ';
  4.33.3.  cout<<endl;
```

```
    }
    else
        return 0;
    cout<<"we\'ll delete the last element, so vec
     become: ";
    vec.pop_back();
    for(int i=0;i<9;i++)
        cout<<vec[i]<<' ';
    cout<<endl;
    return 0;
}
```

و سيكون الخرج على الشكل الآتي:



خرج برنامج vector

و في هذا البرنامج، وضحنا أهم النقاط الأساسية في vector كالتابع Insertو push_back و غيرها من التوابع.

ففي السطرين ٤.٣ و ٤.١٢ استخدمنا التابع size لمعرفة حجم vector، والتابع push_back في السطر ٤.١٠لاقحام عنصر في vector، كما هناك التابعان front و back في السطر ٤.١٣ اللذان يعيدان أول و آخر عنصر. وفي السطر٤.١٩ التابع insert الذي له نمطان، حيث في النمط الأول يقوم بحشر عنصر معين داخل vector عند تمرير رقم الخانة المراد وضع العنصر فيها ثم العنصر. حيث يقوم هذا التابع ضع العنصر في

المكان المراد ثم يقوم بإزاحة كل العناصر على يمين هذا العنصر خانة واحدة إلى اليمين.
و في النمط الثاني، يقوم هذا التابع بحشر قيم من مصفوفة معينة في vector بتمرير رقم أول خانة في الحشر ثم اسم المصفوفة ثم اسم المصفوفة+عدد العناصر كما في السطر ٤.٣٣.١.
كما هناك التابع المنطقيempty الذي يعيد القيمة ١ إذا كان vector فارغاً، ويعيد ٠ إذا لم يكن فارغاً. وهناك التابع pop_back الذي يقوم بحذف آخر عنصر من vector كما في ٤.٣٧

:deque

ومن ميزات هذه الحاوية أنها تجمع العديد من الفوائد التي رأيناها في
وكلمة deque هي اختصار لـ "Double-ended queue" أي صف الانتظار ذو النهايتين.
بُنِيَتْ deque لتوفّر طريقة وصول مفهرسة (باستخدام الأدلة) على عناصره بشكل مشابه لـ vector. كما تم بناؤه للقيام بعمليات الإدخال والحذف من بداية الصف أو نهايته بطريقة تشبه list.

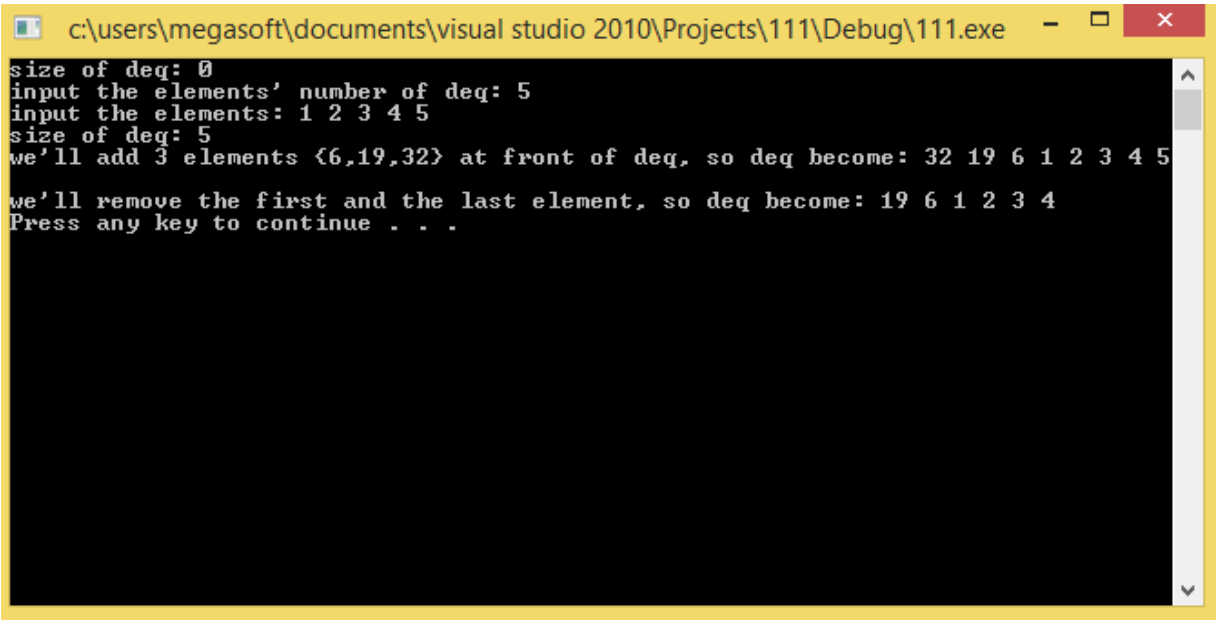
يمكن حجز أماكن تخزين إضافية من أجل deque وإضافتها في البداية أو النهاية. وعلى اعتبار أنه لا يوجد أية ضرورة لحجز حيز من الذاكرة متراص لبناء deque فإن تكرارات deque هي أكثر ذكاءً من المؤشرات التي تستخدم للتجول ضمن vector وغيرها من البنى المعتمدة على المؤشرات.

و لدينا المثال الآتي الذي يوضح أهم الأمور المتعلقة بـ deque:

```
1. #include<iostream>
2. #include<deque>

3. using namespace std;
4. int main(){
     4.1.1.    std::deque<int>deq;
     4.1.2.    int n,q;
     4.1.3.    cout<<"size of deq:
        "<<deq.size()<<endl<<"input the elements\'
        number of deq: ";
     4.1.4.    cin>>n;
     4.1.5.    cout<<"input the elements: ";
     4.1.6.    for(int i=0;i<n;i++){
```

```
            4.1.6.1.1.    cin>>q;
            4.1.6.1.2.    deq.push_back(q);
        4.1.7.    }
        4.1.8.    cout<<"size of deq: "<<deq.size()<<endl;
        4.1.9.    cout<<"we\'ll add 3 elements {6,19,32} at
           front of deq, so deq become: ";
        4.1.10.   deq.push_front(6);
        4.1.11.   deq.push_front(19);
        4.1.12.   deq.push_front(32);
        4.1.13.   n+=3;
        4.1.14.   for(int i=0;i<n;i++)
            4.1.14.1.1.   cout<<deq[i]<<' ';
        4.1.15.   cout<<"\n";
        4.1.16.   cout<<"we\'ll remove the first and the
           last element, so deq become: ";
        4.1.17.   deq.pop_back();
        4.1.18.   deq.pop_front();
        4.1.19.   n-=2;
        4.1.20.   for(int i=0;i<n;i++)
            4.1.20.1.1.   cout<<deq[i]<<" ";
        4.1.21.   cout<<endl;
        4.1.22.   return 0;
5. }
```

و سيكون الخرج على الشكل الآتي:



خرج برنامج deque

يوفر deque نفس العمليات الأساسية الموجودة لدى vector ولكن مع إضافة تابعين، هما pop_front و push_front للسماح بتنفيذ عمليات الدخل و الحذف من بداية deque (أي أننا لا نستطيع إدخال أو حذف عنصر في بداية vector لعدم وجود هذين التابعين فيه).

حاويات التحويل:

يتوفر في STL ثلاثة حاويات تحويل وهي stack, queue, priority_queue وهي ليست من حاويات الدرجة الأولى لأنها لا تقوم فعلياً بتقديم بنى المعطيات الحقيقية، ولا تتعامل مع التكرارات. تفيد هذه الحاويات كصفوف تحويل يمكن للمبرمج اختيار بنية المعطيات الملائمة له من خلالها. و سنعرض ذلك في الفقرات التالية:

:stack

أو ما يسمى المكدس. ويفيد في إدخال المعطيات وحذفها من طرف واحد (أي يعمل وفق مبدأ LIFO اختصاراً لـ Last in First Out). ونستطيع تشبيهها بعلبة pringles حيث تكون أول آخر قطعة بطاطس هي أول قطعة تخرج.

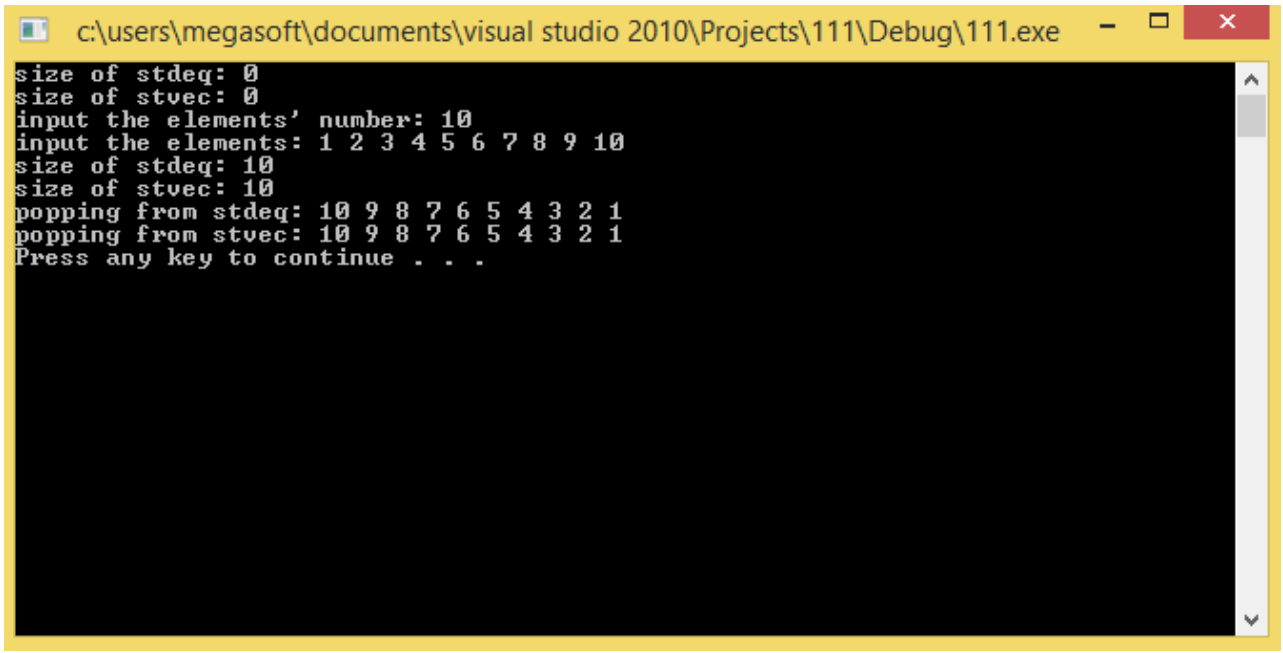
يمكن بناء stack باستخدام أي حاوية من حاويات السلاسل vector, list, deque. يجري بناء stack افتراضياً باستخدام الحاوية deque، أما بالنسبة للعمليات المرتبطة به، فهي push لإدخال العناصر واحداً تلو الآخر إلى المكدس، و pop لحذف العناصر واحداً تلو الآخر من قمة المكدس (أي أن آخر عنصر أُدْخِلَ في المكدس هو أول عنصر سوف يُحْذَف)، و top للحصول مرجع للعنصر المتواجد في قمة المكدس، و empty لتحديد فيما إذا كان المكدس فارغاً أم لا، و size لتحديد عدد العناصر المتواجدة ضمن المكدس.

ولدينا المثال الآتي:

```
1. #include<iostream>
2. #include<stack>
3. #include<vector>

4. using namespace std;

5. void main(){
     5.1.1.   //stack with default underlying deque
     5.1.2.   std::stack<int>stdeq;
     5.1.3.   //stack with underlying vector
     5.1.4.   std::stack<int, std::vector<int>>stvec;
     5.1.5.   int n,q;
     5.1.6.   cout<<"size of stdeq:
        "<<stdeq.size()<<"\nsize of stvec:
        "<<stvec.size()<<endl;
     5.1.7.   cout<<"input the elements\' number: ";
     5.1.8.   cin>>n;
     5.1.9.   cout<<"input the elements: ";
     5.1.10.  for(int i=0;i<n;i++){
          5.1.10.1.1.   cin>>q;
          5.1.10.1.2.   stdeq.push(q);
          5.1.10.1.3.   stvec.push(q);
     5.1.11.  }
```

```
5.1.12.   cout<<"size of stdeq:
   "<<stdeq.size()<<"\nsize of stvec:
   "<<stvec.size()<<endl;
5.1.13.   cout<<"popping from stdeq: ";
5.1.14.   for(int i=0;i<n;i++){
     5.1.14.1.1.   cout<<stdeq.top()<<" ";
     5.1.14.1.2.   stdeq.pop();
5.1.15.   }
5.1.16.   cout<<endl;
5.1.17.   cout<<"popping from stvec: ";
5.1.18.   for(int i=0;i<n;i++){
     5.1.18.1.1.   cout<<stvec.top()<<" ";
     5.1.18.1.2.   stvec.pop();
5.1.19.   }

5.1.20.   cout<<endl;

6. }
```

و سيكون الخرج:



خرج برنامج stack

:queue

يسمح queue بإدخال المعطيات من نهاية بنية المعطيات، وبحذفها أو إخراجها من بدايتها و يسمى هذا المبدأ LIFO اختصاراً لـ (First in first out)، ويمكن تحقيق queue باستخدام الحاوية list أو deque من حاويات السلاسل في STL. ويجري استخدام الحاوية deque افتراضياً لبناء queue.

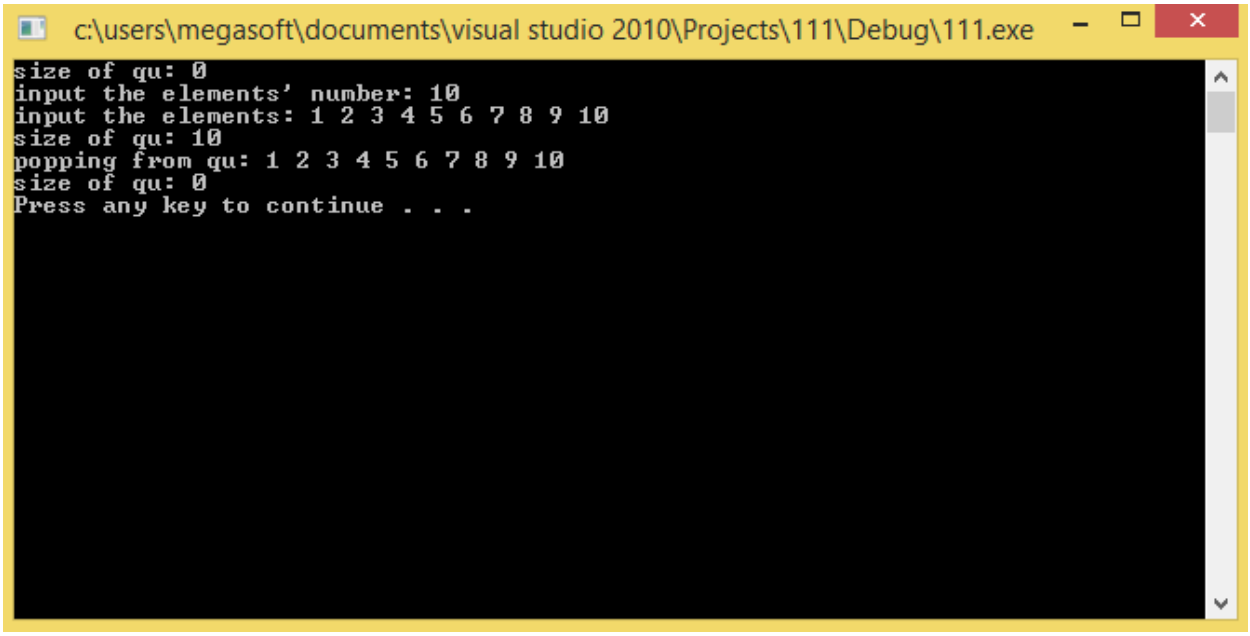
و من أشهر التوابع المستخدمة مع queue التابع push لإدخال عنصر عند نهاية queue. كما هناك التابع pop لإخراج عنصر من بداية queue. و front للحصول على مرجع نحو العنصر الأول. وback للحصول على مرجع نحو العنصر الأخير. ويوجد empty لتحديد فيما إذا كانت فارغة أم لا. و التابع size لتحديد عدد عناصر الحاوية.

ولدينا المثال الآتي:

```
1. #include<iostream>
2. #include<queue>

3. using namespace std;

4. void main(){
     4.1.1.   std::queue<int>qu;
     4.1.2.   cout<<"size of qu: "<<qu.size()<<"\ninput
        the elements\' number: ";
     4.1.3.   int n,w;
     4.1.4.   cin>>n;
     4.1.5.   cout<<"input the elements: ";
     4.1.6.   for(int i=0;i<n;i++){
          4.1.6.1.1.    cin>>w;
          4.1.6.1.2.    qu.push(w);
     4.1.7.   }
     4.1.8.   cout<<"size of qu: "<<qu.size()<<endl;
     4.1.9.   cout<<"popping from qu: ";
     4.1.10.  while(qu.empty()!=1){
          4.1.10.1.1.   cout<<qu.front()<<' ';
          4.1.10.1.2.   qu.pop();
     4.1.11.  }
```

```
    4.1.12.   cout<<endl;
    4.1.13.   cout<<"size of qu: "<<qu.size()<<endl;
5. }
```

و سيكون الخرج على الشكل الآتي:



خرج برنامج qeueu

priority_queue:

ويوفر هذا الصف إمكانية القيام بعمليات إدخال المعطيات أو حذفها. يمكن بناؤه كبنية معطيات ضمنية بشكل افتراضي.

عند إضافة عناصر إلى priority_queue فإنه يجري إضافة العناصر وفقاً لترتيب الأولويات حيث ستكون أول قيمة سيتم إخراجها من الصف هي القيمة العليا.

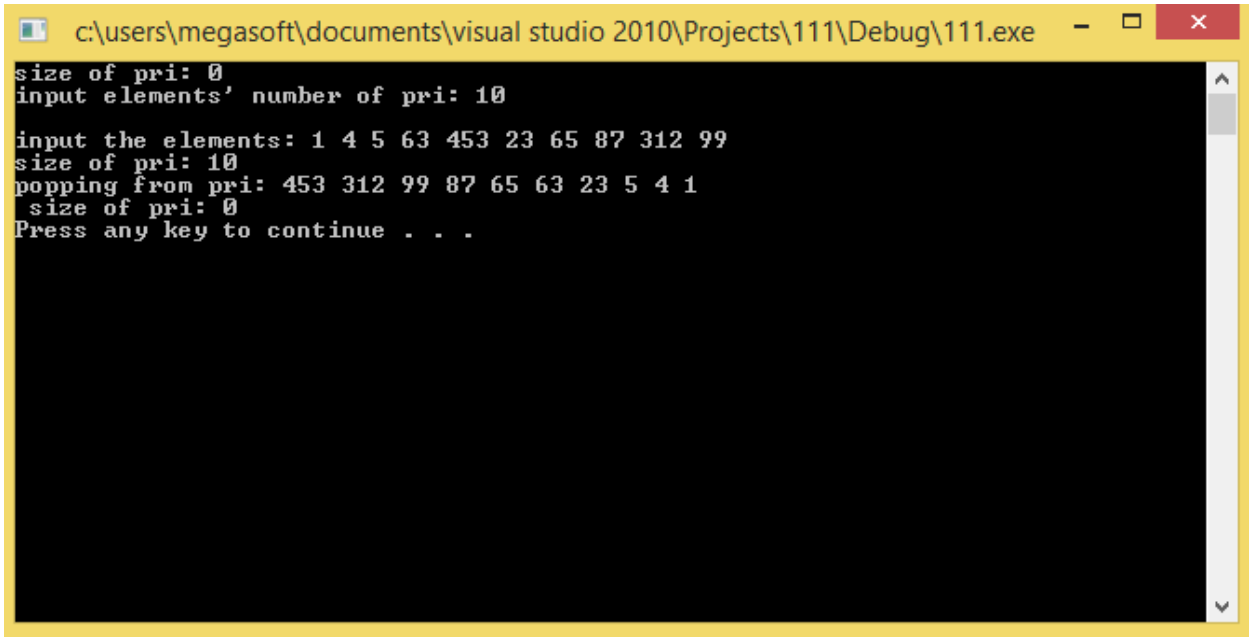
و يجري ذلك باستخدام اسلوب للفرز يسمى الفرز بالكومة Heapsort الذي يحدد دائماً أعلى قيمة من حيث الأولوية، ويضعها في بداية البنية.

و من توابع priority_queue التابع push لإدراج عنصر في المكان المناسب وفقا لأولويته، والتابع pop لحذف العنصر ذو الأعلى قيمة من الحاوية. بالإضافة للتابع top للحصول على العنصر الموجود في أعلى الصف priority_queue. والتابع empty المنطقي لمعرفة إذا كان priority_queue فارغاً أم لا. و size لمعرفة عدد العناصر الموجودة في priority_queue

و هذا المثال يوضح هذه التوابع و كيفية التعامل مع priority_queue:

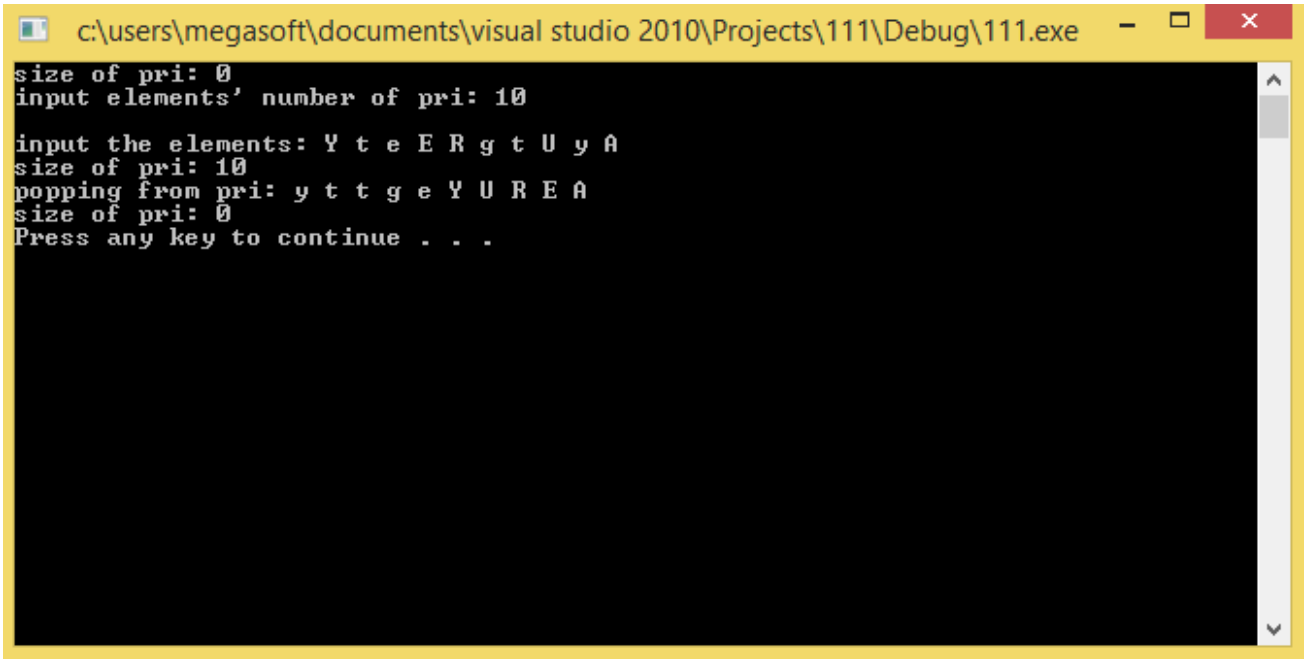
```
1. #include<iostream>
2. #include<queue>

3. using namespace std;
4. void main(){
     4.1.1.    std::priority_queue<int>pri;
     4.1.2.    cout<<"size of pri: "<<pri.size()<<endl;
     4.1.3.    cout<<"input elements\' number of pri: ";
     4.1.4.    int n;
     4.1.5.    int w;
     4.1.6.    cin>>n;
     4.1.7.    cout<<"\ninput the elements: ";
     4.1.8.    for(int i=0;i<n;i++){
          4.1.8.1.1.    cin>>w;
          4.1.8.1.2.    pri.push(w);
     4.1.9.    }
     4.1.10.   cout<<"size of pri:
        "<<pri.size()<<endl<<"popping from pri: ";
     4.1.11.   for(int i=0;i<n;i++){
          4.1.11.1.1.   cout<<pri.top()<<' ';
          4.1.11.1.2.   pri.pop();
     4.1.12.   }
     4.1.13.   cout<<"\nsize of pri: "<<pri.size()<<endl;
5. }
```

و سيكون الخرج:



خرج برنامج priority_queue(1)

و أيضاً إذا عرّفنا الحاوية على أنها حاوية محارف char ، سيكون الخرج:



خرج برنامج priority_queue(2)

الخاتمة

و هكذا، كانت ولا زالت إحدى أهم الركائز الأساسية في عالم البرمجة في . لما كان لها الدور الأكبر والفعال في اختصار العديد من البرامج C++، وتوفيرها الوقت والحجم على الذاكرة، واحتوائها على معظم العمليات التي يحتاجها كل مبرمج في حياته البرمجية...

STL

هــي

المراجع

١. كيف تبرمج بلغة C++ ، د. صلاح الدوجي ، دار شعاع للنشر والعلوم – سوريا.

٢. بنى البيانات في C++ ، المهندس ساري علي الحاج حسين ،سنة ٢٠١٠.