كلية الهندسة – قسم المعلوماتية

مقرر برمجة 2

أ. د. علي عمران سليمان

المحاضرة الخامسة

الفصل الاول 2023-2024

محتوى الفصل

1. Introduction.
2. Operator overloading, concept and implementation.
3. Restrictions on operator overloading.
4. Operator overloading using methods as Class Members vs. as friend methods
5. **Overloading Stream-Insertion and Stream-Extraction Operators,**
6. **Overloading Unary, Binary Operators.**
7. **Overloading ++ and --.**
8. **Overloading new and delete**
9. **Case Study,**

1. مقدمة
2. التحميل الزائد للمعاملات، مفهومه وتحقيقه.
3. القيود على التحميل الزائد للمعاملات.
4. التحميل الزائد للمعاملات باستخدام التوابع الأعضاء والصديقة.
5. **التحميل الزائد للمعاملات الدخل (الحشر) والخرج (الاستخراج).**
6. **التحميل الزائد للمعاملات الاحادية والثنائية,**
7. **التحميل الزائد للمعاملات ++ , -- .**
8. **التحميل الزائد للمعاملين new & delete**
9. **دراسة حالة.**

المحاضرة من المراجع :

[1]- Deitel & Deitel, C++ How to Program, Pearson; 10th Edition (February 29, 2016)

[2]- د.علي سليمان, البرمجة غرضية التوجه في لغة C++ 2009-2010

4-2 – التحميل الزائد للمعاملات الدخل والخرج 1

Overloading Operator Stream-Insertion and Stream-Extraction Operators

- قمنا في المثال المطروح في الفقرة الأولى من هذا الفصل بتعريف تابعين Time_IN و Time_OUT لإنجاز عمليات الإدخال والإخراج لأغراض من النمط Time، وقد آثرنا في الفقرات الثالثة والرابعة تجاهل مناقشة حالة التحميل الزائد لمعاملات الإدخال والإخراج لكي نفرد فقرة خاصة لمناقشة هذه الحالة.
- كما نعلم، فقد حملت المعاملات >> و << تحميلاً زائداً للقيام بعمليات الدخل والخرج لأنواع البيانات المسبقة التعريف في لغة C++. يمكن القيام أيضاً بالتحميل الزائد لهذه المعاملات بحيث تستطيع إجراء عمليات دخل و خرج للأنماط المعرفة من قبل المستخدم.
- تستخدم المعاملات << و >> مجاري streams خاصة لتحقيق عمليتي الإدخال والإخراج حيث يتم تسمية معامل الخرج >> باسم معامل الحشر insertion operator وذلك لأنه يقوم بحشر المعطيات من الذاكرة إلى المجرى، وبشكل مشابه، معامل الدخل << يدعى معامل الاستخراج extraction operator لأنه يخرج المعطيات من المجرى إلى الذاكرة. وتدعى التوابع التي تقوم بالتحميل الزائد لمعاملات الحشر والاستخراج باسم حاشرات inserters و مستخرجات extractors على التوالي.

الصيغة العامة للحاشر:

```
ostream &operator<<(ostream &stream, class_type obj)
{
   // body of inserter
   return stream;      }
```

- لاحظ أن التابع يعيد مرجعاً إلى مجرى من النوع ostream، كما أن البارامتر الأول للتابع هو مرجع إلى مجرى الخرج أما البارامتر الثاني فهو الغرض المراد حشره. إن الحاشر يجب أن يقوم قبل إنهائه بإعادة stream.

Overloading Operator Stream-Insertion and Stream-Extraction Operators

4-2 – التحميل الزائد للمعاملات الدخل والخرج 2

- إن المستخرجات مكملة للحاشرات. الشكل العام للتابع المستخرج:

```
istream &operator>>(istream &stream, class_type &obj)
{
   // body of extractor
   return stream;
}
```

- تعيد المستخرجات مرجعاً إلى مجرى من النوع istream وهو مجرى دخل. البارامتر الأول يجب أن يكون مرجعاً لمجرى الدخل istream. البارامتر الثاني يجب أن يكون مرجعاً لغرض من الصنف الذي يقوم المستخرج بتحميله. ويمكن تعديل الغرض من خلال عملية الإدخال.

من الملاحظ أن عملية استدعاء تابع معامل الدخل أو الخرج يكون من الشكل التالي:

cin>>obj; **cout<<obj;**

- إن الغرض على يسار العملية هو من النمط istream أو ostream أما الغرض على يمين العملية فهو من نمط مختلف، إن هذا يوجب استخدام توابع غير أعضاء (صديقة أو مستقلة) للتحميل الزائد للمعاملات << و >>.

الأمثلة الثلاثة التالية تبين إنجاز هذا الأمر:

4-2 – التحميل الزائد للمعاملات الدخل والخرج 3

استخدام توابع صديقة:

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
using namespace std;
class Time   {
       friend ostream &operator<<(ostream &, Time );
       friend istream &operator>>(istream &, Time &);
private:     int hour;          // 0 - 23 (24-hour clock format)
             int minute;              // 0 - 59
             int second;              // 0 - 59
             char sep;                 //:
             }; // end clas Time
ostream &operator<<(ostream &out, Time T)
{out << " H  M  S "<<endl; out<< setfill('0')<< setw( 2) << T.hour<<T.sep
      << setw( 2)<<T.minute<<T.sep << setw( 2)<<T.second<<endl;
return out; } // must return out
```

4-2 – التحميل الزائد للمعاملات الدخل والخرج 4

```
istream &operator>>(istream &in, Time &T)
{cout << "Enter hours: ";          in >> T.hour;
cout << "Enter minutes: ";        in >> T.minute;
cout << "Enter seconds: ";        in >> T.second;
cout << "Enter seperator: ";      in >> T.sep;
return in;}
int main()
{Time t; cin>>t; cout<<t; system("pause"); return 0;} // end main
```

الخرج:

```
Enter hours: 17
Enter minutes: 55
Enter seconds: 46
Enter seperator: :
 H  M  S
17:55:46
Press any key to continue . . .
```

4-2 – التحميل الزائد للمعاملات الدخل والخرج 5

استخدام توابع مستقلة في صنف يحتوي على توابع وصول:

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
using namespace std;
class Time {
public:
inline void setHour() { cin>>hour; }
inline void setMinute() { cin>>minute; }
inline void setSecond() { cin>>second; }
inline void setSep() { cin>>sep; }
inline int getHour() { return hour; }
inline int getMinute() { return minute; }
inline int getSecond() { return second; }
inline char getSep() { return sep; }

private:
```

```
    int hour;                   // 0 - 23 (24-hour clock format)
    int minute;                 // 0 - 59
    int second;                 // 0 - 59
   char sep;                    //:
}; // end clas Time
ostream &operator<<(ostream &out, Time T)
{    out << " H  M  S "<<endl; out<<setfill( '0') << setw( 2) << T.getHour()
     <<T.getSep()<< setw( 2)<<T.getMinute() <<T.getSep()<<setw( 2)
     <<T.getSecond() <<endl;
    return out; // must return stream
}
istream &operator>>(istream &in, Time &T)
{cout << "Enter hours: ";          T.setHour();
cout << "Enter minutes: ";        T.setMinute();
cout << "Enter seconds: ";        T.setSecond();
cout << "Enter seperator: ";      T.setSep();
system("pause"); return in;       }
```

4-2 – التحميل الزائد للمعاملات الدخل والخرج 7

```
int main()
{       Time t;
        cin>>t;
        cout<<t;
        system("pause");return 0;
} // end main
```

الخرج:

```
Enter hours: 10
Enter minutes: 20
Enter seconds: 30
Enter seperator: :
 H  M  S
10:20:30
Press any key to continue . . .
```

من الملاحظ أن توابع المعاملات المحملة تحميلاً زائداً هي توابع مستقلة عن الصنف (غير أعضاء وغير أصدقاء) وقد استخدمت توابع وصول إلى البيانات الأعضاء الخاصة،

4-2 – التحميل الزائد للمعاملات الدخل والخرج 8

استخدام توابع مستقلة في صنف بياناته الأعضاء عامة:

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
using namespace std;
class Time {
public:
   int hour;              // 0 - 23 (24-hour clock format)
   int minute;            // 0 - 59
   int second;            // 0 - 59
  char sep;               //:
}; // end clas Time
ostream &operator<<(ostream &out, Time T)
{      out << " H  M  S "<<endl; out <<setfill('0')<< setw( 2) << T.hour
          <<T.sep<< setw(2)<<T.minute<<T.sep << setw( 2)<<T.second<<endl;
return out; // must return stream
}
```

4-2 – التحميل الزائد للمعاملات الدخل والخرج 9

Overloading Operator Stream-Insertion and Stream-Extraction Operators

```
istream &operator>>(istream &in, Time &T)
{
cout << "Enter hours: ";
in>>T.hour;
cout << "Enter minutes: ";
in>>T.minute;
cout << "Enter seconds: ";
in>>T.second;
cout << "Enter seperator: ";
in>>T.sep;
return in;
}
int main()
{      Time t;       cin>>t;       cout<<t;      system("pause"); return 0;
} // end main
```

الخرج نفس الحاله السابقة (نكرر: طريقة غير مستحبة لانتهاك نوصيات هندسة البرمجيات).

Overloading Unary, Binary Operators.

5-2 – التحميل الزائد للمعاملات الاحادية والثنائية 1

هنا العديد من المعاملات تحتاج إلى معامل أو معاملين أو ثلاث.

- معامل واحد (~ ,>> ,<< ,! ,- - ,++)
- بعمل مع واحد ومع اثنين (<< ,>> ,+، -)
- مع ثلاث : ? لايحمل تحميلاً زائداً
- الباقي يعمل مع اثنين(حسابية، علائقية، منطقية، خانة ثنائية و اسناد).

	Operator	Type
Unary operator	++ , --	Unary operator
Binary operator	+, -, *, / , %	Arithmetic operator
	<, <=, >, >=, ==, !=	Rational operator
	&&, ||, !	Logical operator
	&, |, <<. >>, ~, ^	Bitwise operator
	=, +=, -=, *=, /=, %=	Assignment operator
Ternary operator	? :	Ternary conditional operator

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 1

- تتيح لغة C++ القيام بشكل صريح بإنشاء نسخ سابقة ولاحقة لمعاملات الزيادة والإنقاص.
- لإنجاز التحميل الزائد لمعامل الزيادة بشكليه السابق prefix واللاحق postfix يجب تعريف نسختين من تابع المعامل operator ++() وذلك لإن نفس العامل يسلك سلوكين مختلفين حسب وجودة سابق كان أم لاحق يأخذان الشكل العام التالي:

```
// Prefix increment
type &operator++( )
            {     // body of prefix operator  }

// Postfix increment
type &operator++(int x)
             {    // body of postfix operator }
```

- إذا سبق المعامل ++ الوسيط، فإن التابع operator++() يتم استدعاؤه، وإذا تلا ++ الوسيطه فإن التابع operator++(int x) يتم استدعاؤه ويأخذ x القيمة صفر.
- لإنجاز التحميل الزائد لمعامل الإنقاص بشكليه السابق prefix واللاحق postfix يجب تعريف نسختين من تابع المعامل operator --( ) يأخذان الشكل العام التالي:

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 2

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
using namespace std;
class Time {
   friend ostream &operator<<(ostream &, Time );
   friend istream &operator>>(istream &, Time &);
public:
Time &operator ++();
Time &operator ++(int );
Time &operator --();
Time &operator --(int );
private:     int hour;              // 0 - 23 (24-hour clock format)
             int minute;            // 0 - 59
             int second;            // 0 - 59
             char sep;               //:
}; // end clas Time
```

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 3

```
ostream &operator<<(ostream &out, Time T)
{ out<< setfill('0')<< setw( 2) << T.hour<<T.sep<< setw( 2)<<T.minute<<T.sep
<< setw( 2)<<T.second<<endl; return out;
}

istream &operator>>(istream &in, Time &T)
{cout << "Enter hours: ";          in >> T.hour;
cout << "Enter minutes: ";        in >> T.minute;
cout << "Enter seconds: ";        in >> T.second;
cout << "Enter seperator: ";      in >> T.sep;          return in;
}

Time &Time::operator ++()
{       second++;
        if (second>59) { second=second%60          minute++;
        if (minute>59) {minute=minute%60;          hour++;
        if (hour>23)hour=hour%24; } }              return *this;
}
```

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 4

```
Time &Time::operator ++(int x)
{       Time temp=*this; second++;
        if (second>59) {second=second%60; minute++;
        if (minute>59) {minute=minute%60; hour++;
        if (hour>23) hour=hour%24; } }     return temp;}
Time &Time::operator --()
{       second--;
        if (second<0) { second=second+60; minute--;
        if (minute<0) { minute=minute+60; hour--;
        if (hour<0) hour=hour+24; } }      return *this;}

Time &Time::operator --(int x)
{       Time temp=*this;    second--;
        if (second<0) { second=second+60; minute--;
        if (minute<0) { minute=minute+60; hour--;
        if (hour<0) hour=hour+24;}}return temp;}
```

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 5

```
int main()
{      Time t;       cin>>t;
cout<<"t++ = "<<t++;
cout<<"t-- = "<<t--;
system("pause"); return 0;
} // end main
```

```
cout<<"t   = "<<t;
cout<<"++t = "<<++t;
cout<<"--t = "<<--t;
```

الخرج:

```
Enter hours: 23
Enter minutes: 59
Enter seconds: 59
Enter seperator: :
t   = 23:59:59
t++ = 23:59:59
++t = 00:00:01
t-- = 00:00:01
--t = 23:59:59
Press any key to continue . . .
```

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 6

- إن تابعي الحالتين السابقة واللاحقة يملكان نموذجين مختلفين مكن المترجم التمييز بين الحالتين السابقة واللاحقة لكلا العمليتين.
- عند مصادفة المترجم العبارة t++ فإنه يقوم بتوليد الاستدعاء التالي للتابع العضو المرتبط بعملية الزيادة اللاحقة t.operator ++(0). القيمة صفر ليس لها معنى ويتم استخدامها كوسيط للتابع operator ++ وذلك للتمييز بين الزيادة السابقة واللاحقة.
- عندما يصادف المترجم العبارة التالية: t++ فإنه يقوم بتوليد الاستدعاء التالي للتابع المرتبط بعملية الزيادة السابقة t.operator ++( ). والأمر ينطبق على حالتي الإنقاص.
- كان بالإمكان إنجاز عملية التحميل الزائد لهذه المعاملات باستخدام التوابع الصديقة حيث تكون نماذج هذه التوابع من الشكل التالي:

```
friend Time &operator ++(Time &);
friend Time &operator ++(Time &,int );
friend Time &operator --(Time &);
friend Time &operator --(Time &,int );
```

وتكون تعريفاتها من الشكل:

```
Time &operator ++(Time &t)
{t.second++;
      if (t.second>59) { t.second=t.second%60; t.minute++;
```

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 7

```
        if (t.minute>59) { t.minute=t.minute%60; t.hour++;
        if (t.hour>23) t.hour=t.hour%24; } }    return t;
}

Time &operator ++(Time &t,int x)
{       Time temp=t; t.second++;
        if (t.second>59) { t.second=t.second%60; t.minute++;
        if (t.minute>59) { t.minute=t.minute%60; t.hour++;
        if (t.hour>23) t.hour=t.hour%24; } }    return temp;
}

Time &operator --(Time &t)
{       t.second--;
        if (t.second<0) { t.second=t.second+60; t.minute--;
        if (t.minute<0) { t.minute=t.minute+60; t.hour--;
        if (t.hour<0) t.hour=t.hour+24; } } return t;
}
```

Overloading increment and decrement operators

6-2 – التحميل الزائد لمعاملي الزيادة والإنقاص 8

```
Time &operator --(Time &t,int x)
{
        Time temp=t; t.second--;
        if (t.second<0) { t.second=t.second+60; t.minute--;
        if (t.minute<0) { t.minute=t.minute+60; t.hour--;
        if (t.hour<0) t.hour=t.hour+24;
        }
    }
return temp;
}
```

في هذه الحالة عند مصادفة المترجم العبارة t++ فإنه يقوم بتوليد الاستدعاء التالي operator ++(t,0).وعند مصادفة العبارة t++ فإنه يقوم بتوليد الاستدعاء التالي operator ++(t).
وبالمثل فإن هذا الأمر ينطبق على حالتي الإنقاص.

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 1

- من الممكن القيام بالتحميل الزائد للمعاملين new و delete، قد ترغب بالقيام بذلك في حال رغبت باستخدام أسلوب خاص في تخصيص الذاكرة. فقد ترغب على سبيل المثال بجعل إجرائيات تخصيص الذاكرة تبدأ باستخدام ملف على القرص كذاكرة افتراضية أو ظاهرية عندما تكون ذاكرة الكومة heap ممتلئة.
- نبين فيما يلي الشكل العام للمعاملات new و delete المحملة بشكل زائد:

```
// Allocate an object.
void *operator new(size_t size)
{ /* Perform allocation. Throw bad_alloc on failure.
Constructor called automatically. */
return pointer_to_memory;
}
// Delete an object.
void operator delete(void *p)
{
/* Free memory pointed to by p.
Destructor called automatically. */
}
```

7-2 – التحميل الزائد للمعاملين new و delete 2

Overloading new and delete

- النمط size_t هو نمط معرف قادر على احتواء أكبر جزء متفصل من الذاكرة يمكن تخصيصه (size_t هو في الأساس من النوع unsigned int). البارامتر size سيحتوي على عدد البايتات اللازمة لاحتواء الغرض المراد تخصيصه. هذه هي كمية الذاكرة التي سيقوم التابع new الجديد بتخصيصها. يجب أن يعيد التابع new المحمل بشكل زائد مؤشراً إلى الذاكرة التي يقوم بتخصيصها أو يقوم بإظهار الاستثناء bad_alloc إذا حصل أي خطأ في عملية التخصيص. عندما تقوم بتخصيص غرض ما باستخدام التابع new (سواء بنسختك الخاصة أو لا) فإن التابع الباني للغرض يتم استدعاؤه تلقائياً.

- يتلقى التابع delete مؤشراً إلى مكان الذاكرة المراد تحريره، فيقوم عندئذ بتحرير الذاكرة المحجوزة سابقاً. يتم استدعاء التابع الهادم تلقائياً عند حذف غرض ما.

- يمكن للمعاملين new و delete أن يحملا تحميلاً زائداً بشكل شمولي globally حيث إن جميع الاستخدامات لهذه المعاملات تستدعي النسخة المحملة بشكل زائد، كما يمكن أن يحملا تحميلاً زائداً بالنسبة لصنف واحد أو أكثر.

- نقوم في المثال التالي بالتحميل الزائد للتابعين new و delete بالنسبة للصنف Time، ومن أجل التبسيط لن يستخدم أسلوب جديد في تخصيص الذاكرة، وإنما سيتم ذلك باستخدام توابع المكتبة القياسية malloc() و free().

- من أجل التحميل الزائد للمعاملين new و delete يمكن ببساطة جعل تابعي المعاملين تابعين عضوين في الصنف كما يلي:

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 3

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <new>
using namespace std;
class Time {
   friend ostream &operator<<(ostream &, Time );
public:      Time (int =0,int =0,int =0,char =':');
void *operator new(size_t size);
      void operator delete(void *p);
private:     int hour;              // 0 - 23 (24-hour clock format)
             int minute;            // 0 - 59
             int second;            // 0 - 59
             char sep;               //:
}; // end clas Time
```

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 4

```
// new overloaded relative to Time.
void *Time::operator new(size_t size)
{      void *p;        cout << "overloaded new.\n";   p = malloc(size);
       return p;
}
// delete overloaded relative to Time.
void Time::operator delete(void *p)
{      cout << "overloaded delete.\n";        free(p);      }
ostream &operator<<(ostream &out, Time T)
{      out<< setfill('0')<< setw( 2) << T.hour<<T.sep << setw(2)<<T.minute
       <<T.sep<< setw( 2)<<T.second<<endl;     return out;
}
Time::Time(int hr,int mn,int sc,char sp)
{      hour=(hr>23)?hr%24:hr;     minute=(mn>59)?mn%60:mn;
       second=(sc>59)?sc%60:sc;   sep=(sp==':')?sp:':';
}
```

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 5

```
int main()
{
Time *t;
t=new Time(10,20,30,':'); cout<<*t;
delete t;
cout<<*t;
system("pause"); return 0;
} // end main
```

الخرج:

```
overloaded new.
10:20:30
overloaded delete.
17891602-ى17891602-ى17891602-
Press any key to continue . . .
```

ملاحظة: افترضنا أن عملية التخصيص تتم دائماً بشكل سليم لذا لم نناقش حالة حصول استثناءات في عملية تخصيص الذاكرة وذلك توخياً للتبسيط.

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 6

لاحظ أن عملية إخراج قيمة المتحول بعد حذفه أعطت قيماً عشوائية مما يدل على إلغاء تخصيص الذاكرة للمتحول.
عند التحميل الزائد للمعاملين new و delete يستخدام توابع أعضاء ضمن الصنف فإن استخدامهما new و delete من أجل أي نوع بيانات آخر يسبب استخدام النسخة الأصلية من التابعين new و delete. ولا يتم استخدام المعاملين المحملين بشكل زائد إلا في حال استخدامهما من أجل الأنواع التي عرفا من أجلها.
يمكن تحميل new وdelete شمولياً وذلك بالتحميل الزائد لهذه المعاملات خارج التصريح عن أي صنف. عند القيام بذلك فإن المعاملين new و delete الافتراضيين في لغة C++ يتم تجاهلهما، ويتم استخدام المعاملات الجديدة في جميع عمليات التخصيص.
إذا احتوى صنف ما تقوم بتعريفه ضمن البرنامج على نسخ من المعاملين new و delete خاصين به، عندئذ فإن النسخ الخاصة بالصنف تستخدم لتخصيص الأغراض من ذلك الصنف، بمعنى آخر عندما يصادف المترجم للمعاملين new و delete فإنه يقوم بداية باختبار إذا كانا معرفين بالنسبة للصنف الذين يؤثران فيه، فإن كان كذلك فإن النسخة الخاصة بالصنف يتم استخدامها في التخصيص وإلا فإن لغة C++ تقوم باستخدام المعاملين المعرفين بشكل شمولي إن وجدت.
يبين المثال التالي استخدام مثل هذه الخاصية:

```
#include "stdafx.h"
#include <iostream>
#include <iomanip>
#include <stdlib.h>
#include <new>
using namespace std;
```

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 7

```
class Time {
   friend ostream &operator<<(ostream &, Time );
public:
Time (int =0,int =0,int =0,char =':');
private:
   int hour;              // 0 - 23 (24-hour clock format)
   int minute;            // 0 - 59
   int second;            // 0 - 59
  char sep;               //:
}; // end clas Time

Time::Time(int hr,int mn,int sc,char sp)
{      hour=(hr>23)?hr%24:hr;    minute=(mn>59)?mn%60:mn;
       second=(sc>59)?sc%60:sc;  sep=(sp==':')?sp:':';
}
```

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 8

```
ostream &operator<<(ostream &out, Time T)
{      out<< setfill('0')<< setw( 2) << T.hour<<T.sep << setw( 2)<<T.minute
       <<T.sep << setw( 2)<<T.second<<endl;    return out;
}
// new overloaded globally.
void *operator new(size_t size)
{       void *p;              cout << "overloaded new.\n";
       p = malloc(size);     return p;
}
// delete overloaded globally.
void operator delete(void *p)
{       cout << "overloaded delete.\n";          free(p);}

int main()
{Time *t;    t=new Time(10,20,30,':');  cout<<*t;
int *p;p=new int(3);cout<<*p<<endl;     double *q;q=new double(10.25);
```

Overloading new and delete

7-2 – التحميل الزائد للمعاملين new و delete 9

```
cout<<*q<<endl;     delete t;     delete p;     delete q;
system("pause"); return 0;} // end main
```

الخرج:

```
overloaded new.
10:20:30
overloaded new.
3
overloaded new.
10.25
overloaded delete.
overloaded delete.
overloaded delete.
Press any key to continue . . .
```

Case Study

8-2 – دراسة حالة

أولاً: صنف التاريخ (Date Class)

ليكن المطلوب

- **تطبيق الزيادة على صنف التاريخ من خلال زيادة يوم بشكل سابق ولاحق والذي قد ينعكس على الشهر وعلى العام.**
- **تعميم ذلك لزيادة عدد من الأيام.**

ثانياً: التحميل الزائد لبعض عمليات المصفوفات

نظراً لاستخدام المصفوفات كبديل للمؤشرات لذلك ينتج عن التعامل معها الكثير من المشاكل، ومثال ذلك

- **التجول في المصفوفة والخروج من مجالها نظرا لأن لغة C++.**
- **لا تتأكد من أن الدلائل لا تزال ضمن حدود المصفوفة.**
- **عدم إمكانية إدخال وإخراج مصفوفة دفعة واحدة بل المرور لكل عنصر بعنصر.**
- **المقارنة بين مصفوفتين، وكذلك عند تمرير مصفوفة لتابع يجب تمرير حجمها.**
- **لا يمكن إسناد مصفوفة إلى مصفوفة باستخدام عملية الإسناد.**

والمطلوب استخدام التحميل الزائد من أجل تأمين ذلك.

انتهت المحاضرة الخامسة