Лекція 11

Тема лекції: Використання виключних ситуацій. Збудження виключних ситуацій. Синтаксис конструкцій, що використовуються для обробки виключних ситуацій

Виключні ситуації

Виключна ситуація - це умова виключної ситуації, що потребує спеціальної обробки. Виключні ситуації найкраще використовувати для обробки помилок, що виникають при виконанні, але їхнє застосування на цьому не обмежується.

Для збудження виключні ситуації оператор посилає об'єкт, що описує суть виключної ситуації. Об'єкт може бути літеральним значенням, рядком, об'єктом класу або будь-якого іншого об’єкту.

Для обробки виключної ситуації деякий оператор перехоплює умову, яка послана іншим процесом. Оператори, що перехоплюють виключні ситуації, називаються обробниками виключних ситуацій.

Програми готуються до перехоплення виключних ситуацій, випробуючи один або декілька процесів, що збуджують виключні ситуації. Для використання виключних ситуацій випробовується один або декілька операторів і перехоплюється будь-яка виключна ситуація, що збуджується цими операторами.

Виявивши стан помилки, функція може збудити виключні ситуації, що викликає такі наслідки:

- Функція оголошує, що виникла умова виключнї ситуації. Це може бути як помилкою, так і іншою обставиною, що потребує обробки.
- Функція запитує рішення проблеми обробки виключної ситуації. Обробник, якщо він існує, викликається автоматично у відповідь на посилку об’єкту виключної ситуації.

У програмі збуджується виключні ситуації за допомогою виконання

оператора *throw*, зазвичай всередині функції:

```
throw "Overflow";
```

В іншому місці програми обробник рядкових виключних ситуацій може перехопити і відобразити на екрані посланий об'єкт. В обробнику задається тип об’єкта у виразі:

```
catch(const char *message)
 {
  cout << "Error! " << message;
  ....
 }
```

Оператор *catch* ловить послані об'єкти рядкових виключних ситуацій і відображує їх оператором виведення в потік. Якщо подальші дії не визначені в блоці *catch*, то програма продовжить свою роботу після *catch*. Можна також аварійно перервати виконання програми, викликати іншу функцію або продовжити цикл для повторного виконання дій, що викликали проблему.

Виключні ситуації є механізмом для повідомлень і вживання заходів у випадку виникнення умови виключної ситуації. Виключні ситуації не нав'язують необхідних дій. Їхня обробка цілком залежить від програміста.

Функції виключних ситуацій не обмежені обробкою помилок. Наприклад, порожній обліковий об'єкт може повідомляти про те, що він порожній, шляхом збудження виключної ситуації. Відбулася помилка або щось інше, залежить тільки від того, як програміст призначить аварійне переривання виконання програм.

Але, оскільки використання виключних ситуацій може призвести до двозначності в програмах, їх найкраще використовувати для вживання заходів у випадку виникнення дійсних помилок, що можуть змусити програму перервати процес і призвести до аварійного завершення або невірних результатів.

Виключні ситуації представляються об’єктами, що дуже схожі на аргументи функцій. Це можуть бути об'єкти будь-якого типу. Найчастіше - це примірники класу. Наприклад, можна оголосити клас *Overflow*:

```
class Overflow
 {
  …
```

```
};
```

Можна послати примірник класу для збудження виключні ситуації:

```
throw Overflow;
```

У цьому операторі створюється об'єкт класу *Overflow*, що потім посилається назад, в місце виклику функції. У іншому місці програми можна перехопити виключну ситуацію за допомогою оператора *catch*:

```
catch(Overflow)
 {
  cout << "Повідомлення";
 }
```

Одна тільки присутність об’єкту *Overflow* свідчить про збудження виключної ситуації. Об'єкт не зобов'язаний починати якихось дій (хоча може). Можливий також такий запис:

```
catch (Overflow overObject){};
```

Об'єкт в операторі *catch* одержав ім'я *overObject*. В операторах всередині *catch* можна використовувати об'єкт *overObject* так само, як і параметр функції. Зазвичай в класі виключні ситуації реалізуються функціями-членами, які можна викликати у виразах всередині *catch*:

```
class Overflow
{
 void Report()
  {
   cout << "Error";
  }
};
```

У класі *Overflow* оголошується функція-член *Report()*, що відображує повідомлення про помилку. Функція може бути не тільки вбудованою, але й функцією, що викликається. В операторі *catch* можна викликати функцію *Report()* об’єкта виключної ситуації для відображення повідомлення про помилку:

```
catch(Overflow overObject)
 {
  overObject.Report();
 }
```

У функціях можна збуджувати об'єкти виключних ситуацій різних типів для підтримки різних умов виключних ситуацій:

```
int AnyFunction()
 {
   if(condition1) throw "Big !!! ";
   if(condition2) throw Overflow;
   return 123;
 }
```

Якщо умова *condition1* справедлива, функція збуджує рядкову виключну ситуацію з повідомленням *"Big !!!"*. Якщо виконується умова *condition2*, функція посилає об'єкт класу *Overflow*, створений, у даному випадку, за допомогою конструктора класу за замовчуванням. Якщо ж функція не виявила помилок, вона повертає значення 123 .

Отже:

• Функції можуть збуджувати одну або декілька виключних ситуацій різних типів, що представляють різні умови виключних ситуацій

• Збудження виключної ситуації негайно завершує виконання функції, у якій виконується оператор *throw*.

• Виключні ситуації забезпечують альтернативний механізм повернення для функцій.

Функція *AnyFunction()* зазвичай повертає ціле значення, проте, якщо виникає виключна ситуація, вона повертає рядок або об'єкт класу *Overflow*. Тільки оператори *catch* можуть сприйняти ці значення.

Можна підтримувати декілька типів об'єктів виключних ситуацій за допомогою серії операторів *catch*.

Для обробки виключної ситуації для класу *Error* і рядків можна написати:

```
catch(Error e){}
catch(const char* message){}
```

Введення до блоків try

Для того, щоб приймати об'єкти виключних ситуацій деякої функції, її

потрібно викликати всередині блоку *try*:

```
try
 {
  int x=AnyFunction();
 }
catch(Error e)
 {
  //Якщо функція збудила виключну ситуацію викликати
  e.Report();   //Report() об'єкта Error
  exit(-1);
 }
```

Блок *try* містить один або декілька операторів, виключні ситуації котрих потрібно перехоплювати. Один або більше операторів *catch* повинні випливати за блоком *try*.

Декілька блоків *try* і відповідних їм операторів *catch* можуть бути оголошені вкладеними, хоча синтаксис у цьому випадку може призвести до заплутаного коду.

```
try
 {
  Function();
   try
    {
     Function();
    }
catch(Error e)
 {
  e.Report();
 }
}
catch(Error e)
 {
  e.Report();
 }
```

Тут якщо у функції *Function()* збуджується виключна ситуація, програма цілком пропускає вкладений блок *try*.

У блоці *try* може бути декілька операторів:

```
try
 {
  cout << "Повідомлення";
  int x=AnyFunction();
  cout << x;
 }
```

У блоці *try* спочатку відображується повідомлення. Потім у ньому викликається функція *AnyFunction()*, результат якої привласнюється цілій змінній *x*. Якщо всередині функції *AnyFunction()* виникла виключна ситуація, то блок *try* негайно завершується. Таким чином, будь-яка умова виключної ситуації призводить до пропуску привласнення *x* і завершального оператора виведення.

За блоком *try* обов'язково повинні випливати один або декілька операторів *catch*:

```
catch(char* message)
{
 cout << "Error !!! " << message;
 exit(-1);
}
catch(Overflow)
{
 cout << "Overflow! ";
 exit(-2);
}
```

Спочатку випробується виклик функції *AnyFunction()*. Якщо функція завершилася нормально, її результат привласнюється змінній *x*, і виводиться. Якщо виключної ситуації не було, обидва оператори *catch* пропускаються, тому що немає об'єктів виключних ситуацій оголошених типів, які потрібно було б перехоплювати. Якщо функція *AnyFunction()* збуджує виключні ситуації, виконання блока *try* негайно переривається, послані об'єкти перехоплюються відповідними операторами *catch*. В прикладі в операторах *catch* викликається бібліотечна функція *exit()* для аварійного завершення програми. Вона необов'язкова. Два оператори *catch* обробляють виключні ситуації для рядків і

класу *Overflow*. Будь-які інші типи виключних ситуацій, не оброблені оператором *catch*, передаються нагору по ланцюжку викликів. Наприклад, припустимо, що функція *AnyFunction()* також збуджує виключні ситуації типу *NewException*. Якщо функція *g()* викликає код функції *AnyFunction(),* то об'єкт цієї виключної ситуації буде передаватися нагору, у *g()*, тому що в цьому коді обробляються тільки виключні ситуації типів *char** та *Overflow*. Якщо жодного з операторів *catch* потрібного типу не було знайдено у ланцюжку викликів, що ведуть до виключної ситуації, виконання програми переривається викликом спеціального обробника виконуючої системи.

Оголошення виключних ситуацій. За допомогою альтернативної форми оголошення функції можна задавати типи виключних ситуацій, які дозволяється збуджувати в даній функції. Наприклад, функція *AnyFunction()* може задати типи своїх виключних ситуацій таким чином:

```
void AnyFunction() throw(Error);
```

Вираз *throw(),* що йде за ім'ям функції і списком її параметрів вказує, що функція *AnyFunction()* може збуджувати виключні ситуації типу *Error*. Це оголошення вказує компілятору, що функції *AnyFunction()* не дозволяється збуджувати виключні ситуації інших типів. Для завдання функції, що збуджує виключні ситуації декількох типів, треба перерахувати типи даних локальних об'єктів виключних ситуацій:

```
void AnyFunction() throw(Error, char*, OtherType);
```