Láthatósági szabályok

Pataki Norbert

Programozási Nyelvek és
Fordítóprogramok Tanszék

Programozási Nyelvek (C++)

Fogalmak

- Láthatóság
- Élettartam
- Memória modell

Láthatóság fogalma

```
void f()
{
  int x = 0;
  ++x;
}
```

C vs. C++

Láthatóság fogalma

```
void f()
{
  int y = 1;                    y
  int x;                        y x
  std::cin >> x;                y x
                                y x
  if ( x > 10 )                 y x
  {                             y x
    ++x;                        y x
    int t = y;                  y x t
    ++y;                        y x t
  }                             y x
  ++x;                          y x
  ++y;                          y x
}
```

Azonos nevű változók

```
// a.cpp:
void f()
{
  int i = 2;
  int i = 3;
}

$ g++ -W -Wall -pedantic -ansi a.cpp
a.cpp: In function ‘void f()’:
a.cpp:4:7: error: redeclaration of ‘int i’
   int i = 3;
       ^
a.cpp:3:7: note: ‘int i’ previously declared here
   int i = 2;
       ^
```

Azonos nevű változók

```
void f()
{
  int x;                          x
  std::cin >> x;                  x
  if ( x > 0 )                    x
  {                               x
    std::cout << x                x
              << std::endl;       x
    int x = 2;                        x
    ++x;                              x
  }                               x
  ++x;                            x
  std::cout << x                  x
            << std::endl;         x
}
```

Globális változók

```
int x;                          x
int y;                          x y

                                x y
void f()                        x y
{                               x y
  int i = 0;                    x y i
  ++i;                          x y i
}                               x y

                                x y
void g()                        x y
{                               x y
  int i = x;                    x y    i
  int j = x + y;                x y    i j
}                               x y
```

Globális változók több fordítási egységben

```
// a.cpp:
int x;
extern int y;
static int s;

void f()
{
  // ...
}

void g()
{
  // ...
}
```

```
// b.cpp:
int y;

void h()
{
  // ...
}
```

```
// n.cpp:
extern int y;
// ...
```

Globális változók elérésének csökkentése

```
// x.cpp:
static int k;

namespace
{
  int q;
}
```

Globális/lokális

```
int x = 0;

void f()
{
  int x = 1;
  if ( x > 0 )
  {
    int x = 2;
    ++x;
    ::x = x;
  }
  ++x;
}
```

Azonos nevű változók

```
int x;

void f()
{
  if ( x > 0 )
  {
    ++x;
    int x = 2;
    ++x;
    if ( x != 0 )
    {
      int x = 1;
      ++x;
    }
  }
}
```

Névterek

```
// ns.cpp:
namespace A
{
  int x;     // A::x
}

namespace B
{
  int x;     // B::x
}
---
$ g++ -c ns.cpp
$ nm ns.o
0000000000000000 B _ZN1A1xE
0000000000000004 B _ZN1B1xE
```

C vs C++

Névtér

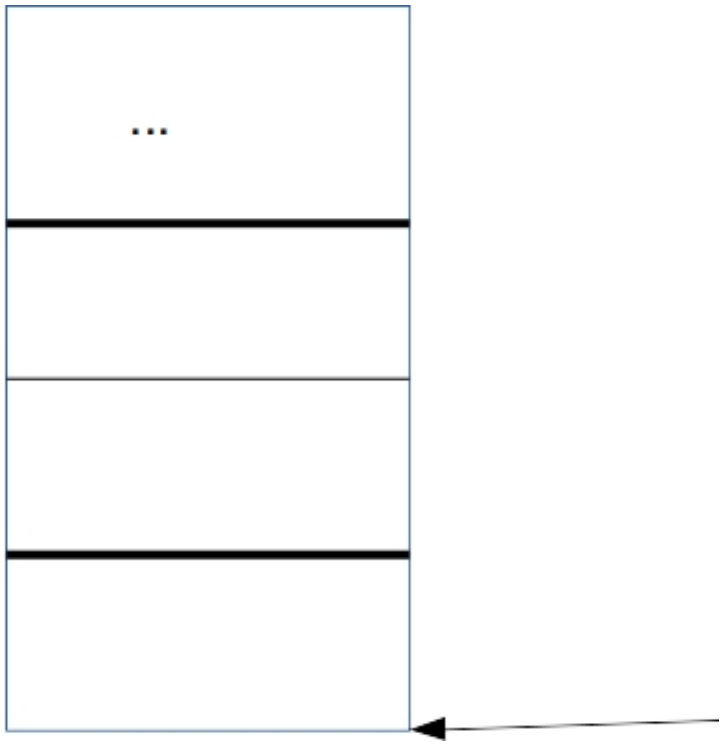
```
int x;

namespace A
{
  int x;        // A::x

  void f();
}
// ...
void A::f()
{
  ++x;          // A::x
}
```

Memória szegmensek

- Végrehajtási verem, execution stack, stack
- Statikus tárterület:
  - Globális változók
  - "Hello";
  - Kódszegmens
- Heap memória

Példa

```
void f( int x )
{
  int y = 0;
  if ( x == y )
  {
    ++y;
    int a = y;
  }
  ++y;
}
```



Végrehajtási verem működése

- Stack frame, aktivációs rekord
- BP, SP, EBP, ESP, RBP, RSP
- Függvényhívások
- Előnyök
- Limitációk

Stack trace

Estado HTTP 500 - java.lang.NullPointerException

type Informe de Excepción

mensaje java.lang.NullPointerException

descripción El servidor encontró un error interno que hizo que no pudiera rellenar este requerimiento.

excepción

```
javax.servlet.ServletException: java.lang.NullPointerException
        org.glassfish.jersey.servlet.WebComponent.service(WebComponent.java:392)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:382)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:345)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:220)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
        org.netbeans.modules.web.monitor.server.MonitorFilter.doFilter(MonitorFilter.java:393)
```

causa raíz

```
java.lang.NullPointerException
        service.AbstractFacade.findAll(AbstractFacade.java:42)
        service.ClienteREST.findAll(ClienteREST.java:68)
        sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        java.lang.reflect.Method.invoke(Method.java:498)
        org.glassfish.jersey.server.model.internal.ResourceMethodInvocationHandlerFactory$1.invoke(ResourceMethodInvocationHandlerFactory.java:81)
        org.glassfish.jersey.server.model.internal.AbstractJavaResourceMethodDispatcher$1.run(AbstractJavaResourceMethodDispatcher.java:151)
        org.glassfish.jersey.server.model.internal.AbstractJavaResourceMethodDispatcher.invoke(AbstractJavaResourceMethodDispatcher.java:171)
        org.glassfish.jersey.server.model.internal.JavaResourceMethodDispatcherProvider$TypeOutInvoker.doDispatch(JavaResourceMethodDispatcherProvider.java:195)
        org.glassfish.jersey.server.model.internal.AbstractJavaResourceMethodDispatcher.dispatch(AbstractJavaResourceMethodDispatcher.java:104)
        org.glassfish.jersey.server.model.ResourceMethodInvoker.invoke(ResourceMethodInvoker.java:402)
        org.glassfish.jersey.server.model.ResourceMethodInvoker.apply(ResourceMethodInvoker.java:349)
        org.glassfish.jersey.server.model.ResourceMethodInvoker.apply(ResourceMethodInvoker.java:106)
        org.glassfish.jersey.server.ServerRuntime$1.run(ServerRuntime.java:259)
        org.glassfish.jersey.internal.Errors$1.call(Errors.java:271)
        org.glassfish.jersey.internal.Errors$1.call(Errors.java:267)
        org.glassfish.jersey.internal.Errors.process(Errors.java:315)
        org.glassfish.jersey.internal.Errors.process(Errors.java:297)
        org.glassfish.jersey.internal.Errors.process(Errors.java:267)
        org.glassfish.jersey.process.internal.RequestScope.runInScope(RequestScope.java:318)
        org.glassfish.jersey.server.ServerRuntime.process(ServerRuntime.java:236)
        org.glassfish.jersey.server.ApplicationHandler.handle(ApplicationHandler.java:1010)
        org.glassfish.jersey.servlet.WebComponent.service(WebComponent.java:373)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:382)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:345)
        org.glassfish.jersey.servlet.ServletContainer.service(ServletContainer.java:220)
        org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:52)
        org.netbeans.modules.web.monitor.server.MonitorFilter.doFilter(MonitorFilter.java:393)
```

nota La traza completa de la causa de este error se encuentra en los archivos de diario de Apache Tomcat/8.0.27.

Apache Tomcat/8.0.27

Stack overflow

```
// so.cpp:
int fac( int n )
{
  return n * fac( n - 1 );
}

int main()
{
  fac( 3 );
}

$ g++ -W -Wall -pedantic -ansi so.cpp
$ ./a.out
Segmentation fault (core dumped)
```

Dinamikus memóriakezelés

- Heap szerepe
- `malloc, new, new []`
- `free, delete, delete []`
- Problémák:
  - Memória-szivárgás, memory leak
  - Dupla free
  - Dangling pointerek
- Garbage collection
- `valgrind` (demo)

Szükség esetén használjuk

Dinamikus memóriakezelés C++-ban

```
void f( int i )
{
  int *p = new int( i );
  // ...
  delete p;
}
```

Dinamikus memóriakezelés C++-ban

```
void g( int i )
{
  int *q = new int[ i ];
  // ...
  delete [] q;
}
```

Mi kerül a heapre?

- Stack limitáció
- Alaptípus/beépített típusú adatot tárolhatunk a heap-en
- Felhasználói típusú objektumot tárolhatunk a stack-en
- Független kérdés: típus és memória

Felhasználói objektumok

```
class complex
{
private:
  double re, im;
// ...
};

//...

void f()
{
  complex c( 1.2, 3.4 );
  //
}
```

Problémák

```
void f( int i )
{
  int *p = new int[ i ];
  complex* c = new complex( 1.2, 3.4 );
  f( ... );
  g( ... );
  // ...
  delete [] p;
  delete c;
}
```

Ellenőrző kérdések

```
void question0( int i )
{
  int *p = new int[ i ];
  int *q = p;

  delete [] q;
  delete [] p;
}
```

Ellenőrző kérdések

Ellenőrző kérdések

```
void question1( int i )
{
  int *p = new int[ i ];
  int *q = p;

  delete [] q;
  p = 0;
}
```

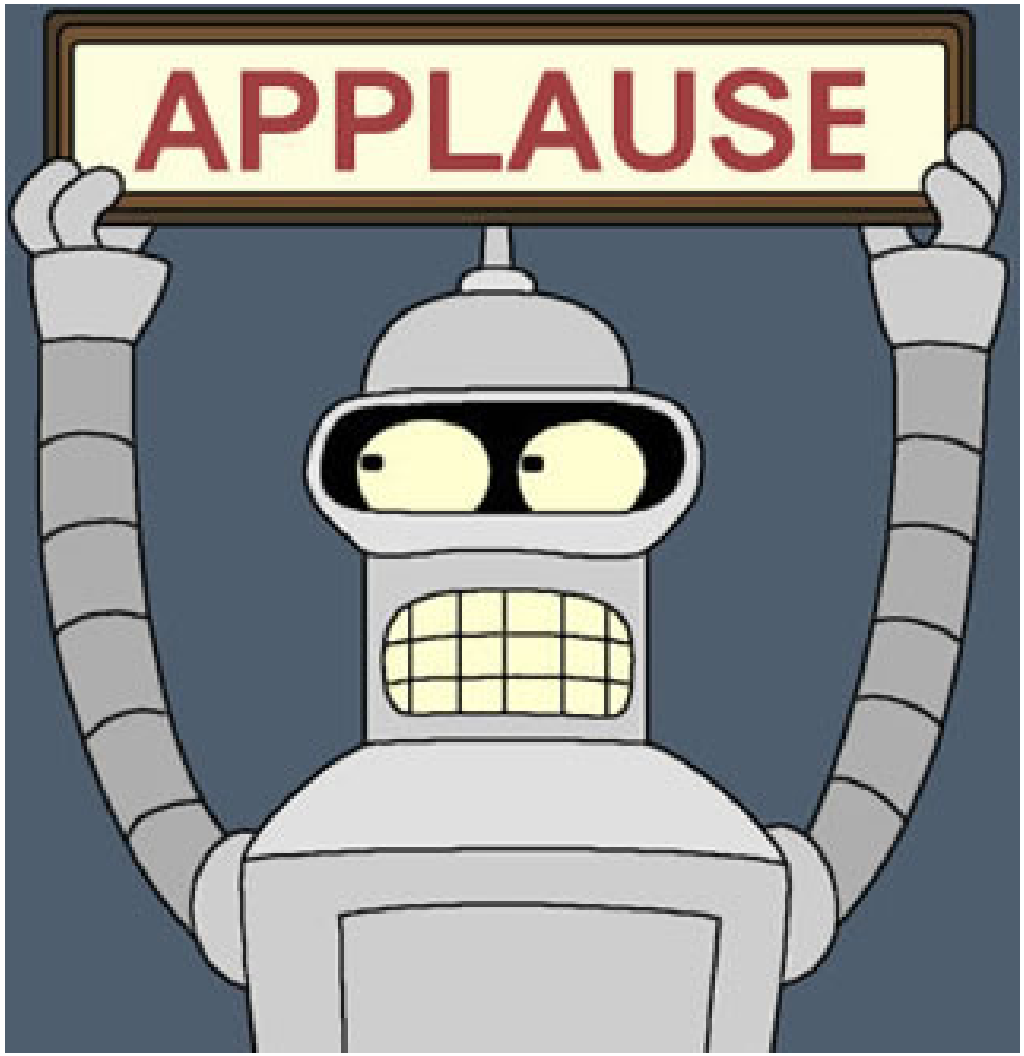
Ellenőrző kérdések

Ellenőrző kérdések

```
void question2( int i )
{
  int *p = new int[ i ];
  int *q = p;

  delete [] p;
  q[ 0 ] = 123;
}
```

Ellenőrző kérdések

Ellenőrző kérdések



RAII

- Erőforrás, resource
- C++: csomagoljuk be ezeket egy osztályba
- Konstruktor, destruktor
- Használunk saját típusú lokális változó objektumokat
- Élettartam szabályok (részletesen jövő héten)
- Stroustrup: „C++ is a resource-oriented programming language."

Resource acquisition is initialization

RAII példa

```
void f( int i )
{
  std::vector<int> v( i );
  // ...
}
```