

Mi registro técnico de liquidación de deudas

Programación (26)

diario (4)

pensamientos (8)

libro (3)

hogar

programación

[42Seúl] Módulo CPP 06 - Conversión de tipo

Aggrodonk

2022. 9. 11. 17:02

Introducción

success

100

100

Solo – about 7 hours – 0 XP

subject.pdf

Retry

youjeon's CPP Module 06

youjeon's group

100%

This team was locked 5 days ago and closed 4 days ago

GIT REPOSITORY

git@vogosphere.42seoul.kr: vogosphere/intra-uuid-4711ad46-dd2c-422e-8818-1003ed8b02d5-4387949-youjeon

EVALUATIONS

Peer evaluations (2/2)

EVALUATED BY JINHOKIM 4 DAYS AGO

100%

제가 처음 보는 과제인데 친절하게 설명해주셔서 좋았습니다. 제가 이 과제하게 되었을 때 도움이 될 것 같습니다! 수고 많으셨습니다.

YOUR FEEDBACK, 4 DAYS AGO:

복합하시고 cpp 평가가 처음이라고 하셔서 걱정했는데, 다행히 자바를 사용해보신적이 있으셔서 객체지향에 대한 설명을 오래 하지는 않아도 되서 좋았습니다. ex00에서 평가의 장점을 설명해드렸고 평가표를 보면서 평가를 완료했습니다. 평가 감사드립니다!

★★★★★

4 / 4

4 / 4

4 / 4

4 / 4

EVALUATED BY PHILANTHROPIST HAKO 4 DAYS AGO

100%

평가 전에 서브젝트를 읽어보고 cpp를 써봤음에도 무슨 과제인지 알쏭달쏭했는데 피평가자분께서 과제 관련 장점과 각 과제의 학습목표 등을 차근차근 설명해주셔서 이해하기 쉬웠습니다. 제가 나중에 cpp 06과제를 할 때 많은 도움 될 것 같습니다. 고생하셨습니다!

YOUR FEEDBACK, 4 DAYS AGO:

실작 늦으셨으나 5분전에 미리 늦는다고 말씀해주셔서 기다리면서 마음 졸이지 않았습니다. 감사드립니다. cpp평가를 받을때 가장 어려운 점이 점정 설명해야할게 많아져서 과제 설명을 시작하기 전에 한참 시간을 쓰게된다는 점인데, 다행히 평가자께서 c++를 사용해보셔서 레퍼런스 등을 바로 이해하셔서 평가받기에 정말 편했습니다. 평가 감사드립니다!

★★★★★

4 / 4

4 / 4

4 / 4

4 / 4

Esta es literalmente una tarea para realizar la conversión de tipos al estilo C++.

Una regla adicional es que cada ex debe usar un elenco específico para resolver, y esa elección debe incluirse en la defensa.

ex00

Cree un programa que convierta explícitamente una cadena a un tipo de datos general y la genere. Los tipos de datos comunes son char, int, float y double.

ejemplo de carácter: 'c', 'a'... Si es una cadena que no se puede imprimir, no intente imprimirla, simplemente proporcione la información relevante.

ejemplo entero: 0 -42 42...

Ejemplo de flotador: 0.0f, -4.2f, 4.2f... Además, se deben aceptar -inff, +inff y nanf.

doble ejemplo: 0.0, -4.2, 4.2... Además, se deben aceptar -inf, +inf y nan.

Después de detectar el carácter, debe convertirse al tipo más correcto y luego convertirse a tres tipos y salidas diferentes. Si hay un problema con la conversión o se produce un desbordamiento, se muestra un mensaje relacionado. Se pueden utilizar todos los encabezados relacionados con los límites numéricos o valores especiales.

Esta es una tarea para usar static_cast. Es responsable de la conversión de tipos básica y de la conversión de tipos que genera errores en el momento de la compilación.

Para hacer esto más fácil, podrías usar funciones de cadena como stoi, pero no importa cuánto lo pensé, no pude entender que 'Cualquier función para convertir de una cadena a un int' significaba que una función C++11 puede ser usada.

Aún así, para hacerlo más fácil, puedes simplemente cambiarlo a double y luego procesarlo nuevamente, pero quería seguir el dicho: conviértelo de cadena a su tipo real, luego conviértelo explícitamente a los otros tres tipos de datos.

Entonces, pensé en todas las excepciones y me ocupé de ellas, pero fue una tarea mucho más difícil de lo que pensaba, así que pasé varios días solo en ex00. Si encuentra un evaluador que procesó el método anterior, definitivamente hará clic en el error de compilación, pero si hay alguien que procesó el último método y no está en la tabla de evaluación, no debe estar en desacuerdo.

<https://www.ryugod.com/pages/ide/cc98>

RyuDios

IDE para comprobar que std=c++98 no funciona correctamente en Mac

[illegible]

Simplemente anoté de antemano las situaciones que requerían manejo de excepciones y las codifiqué una por una. Lo hice después de sentirme como un mini caparazón donde cada vez que arr

eglo algo, algo más se estropea, y fue útil.

```
void Convert::setValue(std::string s)
{
    std::string::size_type n;
    std::string::size_type f;

    try
    {
        this->value = s;
    }
    catch(const std::bad_alloc& e)
    {
        err = true;
        return ;
    }

    n = s.find('.');
    f = s.find('f', s.length() - 1);
    if (value == "nan" || value == "inf" || value == "+inf" || value == "-inf" || value ==
"nanf" || value == "inff" || value == "+inff" || value == "-inff")
    {
        if (value == "nan" || value == "nanf")
        {
            val_double = sqrt(-1.0);
        }
        else
        {
            if (value[0] == '-')
            {
                val_double = __DBL_MAX__ * -1000;
            }
            else
            {
                val_double = __DBL_MAX__ * 1000;
            }
        }
    }
    else if (s.length() == 1 && !std::isdigit(static_cast<char>(s[0])))
    {
        val_char = static_cast<char>(s[0]);
        val_int = static_cast<int>(val_char);
        val_float = static_cast<float>(val_int);
        val_double = static_cast<double>(val_int);
    }
    else if (n == std::string::npos)
    {
        val_int = atoi(s.c_str());

        std::stringstream ss;
        ss << val_int;
        std::cout << "len : " << s.length() << " size : " << s.size() << std::endl;
        if (ss.str() != value)
        {
            if (val_int > 0 && f == s.length() - 1)
            {
                val_char = static_cast<char>(val_int);
                val_float = static_cast<float>(val_int);
                val_double = static_cast<double>(val_int);
            }
            else
            {

```

```

        err = true;
    }
}
else
{
    val_char = static_cast<char>(val_int);
    val_float = static_cast<float>(val_int);
    val_double = static_cast<double>(val_int);
}
}
else
{
    const char *str = s.c_str();
    char *end = NULL;

    val_double = strtod(str, &end);

    if((*end && !(*end == 'f' && end == &str[s.length() - 1])))
    {
        err = true;
    }
    else
    {
        val_int = static_cast<int>(val_double);
        val_char = static_cast<char>(val_double);
        val_float = static_cast<float>(val_double);
    }
}
}
}

```

La primera cadena recuperada se almacena en el objeto. Si la cadena es demasiado grande, el programa puede detenerse debido a `bad_alloc`, por lo que se maneja una excepción para evitar que se detenga.

Al procesar `nan` o `inf`, primero verifique si se ha ingresado el carácter correspondiente y luego almacene el valor en una variable doble.

Si la cadena tiene un carácter pero no un número, guárdela como `char` y procese el valor transfiriéndolo `static_cast` a otra variable.

Usando `size_type` y buscar, `<`. Encuentre las posiciones de `>` y `<f>`. En el caso de `f`, solo verifica si está al final.

si . Si la posición de es `npos`, ese carácter tiene . Esto significa que no existe, por lo que se trata como un `int` y el resto se trata como un error.

Sin embargo, si la cadena guardada y el valor convertido son diferentes, se maneja una excepción porque es una cadena no numérica, un desbordamiento o un tipo flotante con una `f` al final.

```
std::ostream& operator<<(std::ostream &out, const Convert &b)
{
    std::stringstream ss;
    ss << b.getInt();

    try
    {
        out << "char: " << b.printChar() << std::endl;
    }
    catch(const std::exception& e)
    {
        std::cerr << e.what() << '\n';
    }

    try
    {
        out << "int: " << b.printInt() << std::endl;
    }
    catch(const std::exception& e)
    {
        std::cerr << e.what() << '\n';
    }

    try
    {
        if (static_cast<float>(b.getInt()) == b.getFloat())
        {
            if (ss.str().size() > 6)
            {
                out << "float: " << b.printFloat() << std::endl;
            }
            else
            {
                out << "float: " << b.printFloat() << ".0f" << std::endl;
            }
        }
        else
        {
            out << "float: " << b.printFloat() << "f" << std::endl;
        }
    }
    catch(const std::exception& e)
    {
        std::cerr << e.what() << '\n';
    }

    try
    {
        if (static_cast<double>(b.getInt()) == b.getDouble())
        {
            if (ss.str().size() > 6)
            {
                out << "double: " << b.printDouble() << std::endl;
            }
            else
            {
                out << "double: " << b.printDouble() << ".0" << std::endl;
            }
        }
        else
        {
            out << "double: " << b.printDouble() << std::endl;
        }
    }
}
```

```
    }  
}  
catch(const std::exception& e)  
{  
    std::cerr << e.what() << '\n';  
}  
  
return (out);  
}
```

En la parte de salida, se arrojó un error en la función de impresión para manejar situaciones de error o situaciones como nan inf.

En la salida de números reales, los números enteros con más de 7 dígitos se generan en una forma como 1.11111e+07 y todo al final. No pude procesarlo agregando 0f. Entonces, se hizo una excepción en los casos en que el número real era igual a un número entero y el tamaño de la cadena era mayor que 6, pero el problema era que no había certeza de que este fuera 100% mayor que 6.

Dado que la mantisa de IEEE 754 es de 23 bits y el valor correspondiente es de 7 dígitos (8388608), para indicar el valor correcto solo se indican hasta 6 dígitos en lugar de manejar una excepción en el medio... Lo supuse, pero es sólo mi suposición. No pude encontrar la razón exacta, pero parecía que estaba documentado que la precisión era 6. Entonces, cuando la longitud es mayor que 6, la procesé por separado, pero no sé si es el método correcto. Parecía que algo estaba definido como macro, pero no era así.

<https://hwan-shell.tistory.com/211>

C++] ¿Qué es static_cast?

Todos los idiomas tienen conversión de tipos. C++ proporciona varios objetos de conversión de tipos. 1. static_cast 2. dynamic_cast = ht...

hwan-shell.tistory.com

<https://www.delftstack.com/ko/howto/cpp/cpp-last-character-of-string/>

Obtener el último carácter de una cadena ...

Este artículo explica cómo obtener el último carácter de una cadena en C++.

www.delftstack.com

<https://ju3un.github.io/c++-new-excepiton/>

C ++ nuevo manejo de excepciones bad_...

Este es un código relacionado con la nueva asignación de memoria dinámica, que se ve comúnmente en el desarrollo de C++. char* pC...

ju3un.github.io

<https://dojang.io/mod/page/view.php?id=739>

Sello de codificación en lenguaje C: 85.5 ...

Aquí se explica cómo verificar el infinito en lenguaje C: float_infinity.c
#include #include // Archivo de encabezado donde se definen los v...

dojang.io

<https://blockdmask.tistory.com/39>

[C++] Limpiando la conversión de clases ...

Aprendamos sobre la conversión entre string, char * e int en C++. (Si desea ver el cambio de char* → int, [Atajo]) (Si desea ver el cambio...

blockdmask.tistory.com

https://mg729.github.io/c++/2019/11/03/C++_string_npos/

C++ - std::string::npos (qué significa npos)

C++ - Cadena cadena::npos

mg729.github.io

https://en.cppreference.com/w/cpp/io/basic_ios/init

std::basic_ios<CharT,Traits>::init - cppref...

Establece el búfer de flujo asociado en sb e inicializa el estado interno. Las condiciones posteriores son las siguientes: Esta función mie...

es.cppreference.com

<http://websites.umich.edu/~eecs381/handouts/formatting.pdf>

ex01

Implementar las siguientes dos funciones.

uintptr_t serialize(Data* ptr), que convierte el puntero a Datos en un tipo entero sin signo uintptr_t;

Data* deserialize(uintptr_t raw), que convierte un entero sin signo en un puntero a Datos;

Debes crear una estructura de datos cuyo valor no esté vacío.

Utilice serializar en la dirección del objeto de datos y deserializar el valor de retorno para verificar si el valor de retorno es el mismo que el valor inicial.

```
struct Data
{
    std::string name;
};
```

```

uintptr_t serialize(Data* ptr)
{
    return(reinterpret_cast<uintptr_t>(ptr));
}

Data* deserialize(uintptr_t raw)
{
    return(reinterpret_cast<Data *>(raw));
}

int main(int ac, char *av[])
{
    Data prev;
    Data *next;
    uintptr_t ptr;

    if (ac != 2)
    {
        std::cout << "argument count is not 2" << std::endl;
        return (1);
    }
    prev.name = av[1];
    std::cout << "prev : " << prev.name << std::endl;

    ptr = serialize(&prev);
    std::cout << "ptr : " << ptr << std::endl;

    next = deserialize(ptr);
    std::cout << "next : " << next->name << std::endl;

    return (0);
}

```

Esta es una tarea para usar reinterpret_cast.

reinterpret_cast es una conversión que se utiliza principalmente para punteros y es responsable de puntero → puntero, variable → puntero y puntero → variable.

Esto se usa principalmente en la serialización, como el nombre de la tarea y el nombre de la función. Cuando se envía información a través de la red, se debe enviar como una secuencia, pero, por supuesto, esto solo se puede enviar como un tipo de datos básico. .

Por lo tanto, para enviar información sobre objetos o punteros de la estructura o clase que se está utilizando, la información debe convertirse a un tipo de datos básico, lo que se llama serialización.

Por supuesto, no existe ninguna opción para la transmisión de red con la tarea anterior, pero puede considerarlo como convertir un objeto en un valor que puede enviarse a la red, recibirlo y convertirlo nuevamente en datos.

<https://hwan-shell.tistory.com/219>

C++] Acerca de reinterpret_cast...

Todos los idiomas tienen conversión de tipos. C++ proporciona varios objetos de conversión de tipos. 1. static_cast = [https://hwan-shell...](https://hwan-shell.tistory.com)

hwan-shell.tistory.com

<https://isocpp.org/wiki/faq/serialization>

C++ estándar

isocpp.org

<https://elecs.tistory.com/289>

Cómo comunicar y transmitir objetos de cl...

La mayor ventaja de usar un lenguaje orientado a objetos es que puedes usar constructores y herencia usando clases, lo que hace que...

elecs.tistory.com

ex02

Esta asignación no requiere el uso de la Forma Canónica Ortodoxa.

Crea una clase base con un solo destructor virtual. Luego crea tres clases vacías A, B y C.

Implementar las siguientes características

Base * generar(void), que devuelve un objeto de una clase aleatoria entre A, B y C;

void identificar(Base* p); imprime qué clase es el objeto; identificación nula (Base& p);

Sin embargo, void identificar(Base& p); No utilice punteros al implementar.

El encabezado typeid está prohibido y los evaluadores deben crear el suyo propio.

```
void identify(Base* p)
{
    if (dynamic_cast<A*>(p))
        std::cout << "pointer is A\n";
    if (dynamic_cast<B*>(p))
        std::cout << "pointer is B\n";
    if (dynamic_cast<C*>(p))
        std::cout << "pointer is C\n";
}

void identify(Base& p)
{
    try
    {
        A &a = dynamic_cast<A&>(p);
        std::cout << "reference is A\n";
        static_cast<void>(a);
    }
    catch (std::exception&) {}
    try
    {
        B &b = dynamic_cast<B&>(p);
        std::cout << "reference is B\n";
        static_cast<void>(b);
    }
    catch (std::exception&) {}
    try {
        C &c = dynamic_cast<C&>(p);
        std::cout << "reference is C\n";
        static_cast<void>(c);
    }
    catch (std::exception&) {}
}
```

Esta es una tarea para usar Dynamic_cast.

Dynamic_cast es una conversión que se utiliza principalmente para abatir donde un puntero que apunta a un padre se cambia para que apunte a un hijo. Si la clase principal es una clase abstracta con una función virtual, el downcasting es posible porque la dirección del hijo se almacena por separado (se explica de pasada en la trampa de aplausos).

Sin embargo, si no se ha llamado al constructor del niño (si el puntero de la clase principal se llama al constructor de la clase principal), es imposible porque no hay una dirección a la que hacer referencia, y si no es una clase abstracta, Se debe usar `static_cast` en lugar de `dynamic_cast` (sin embargo, en este caso, la seguridad no está garantizada).

Si lanzas `dynamic_cast` de una clase secundaria a otra clase secundaria, fallará porque no habrá recibido el constructor de otra clase secundaria.

El método de implementación cambia dependiendo de si el argumento se recibe como un puntero o una referencia. Cuando `Dynamic_cast` falla, si es un puntero, se devuelve `NULL`, pero si es una referencia (ya que `NULL` no se puede usar), se produce una excepción.

Por lo tanto, cuando lo reciba como puntero, use la declaración `if else` y cuando lo reciba como referencia, use la declaración `try catch`.

<https://hwan-shell.tistory.com/213>

C++] Acerca de la transmisión dinámica...

Todos los idiomas tienen conversión de tipos. C++ proporciona varios objetos de conversión de tipos. 1. `static_cast` = <https://hwan-shell.tistory.com/213>

[hwan-shell.tistory.com](https://hwan-shell.tistory.com/213)

<https://blockdmask.tistory.com/241>

[C++]dynamic_cast (operador de encajillamiento)

Hola. Esto es BlockDMask. Esta vez, veremos el último de los cuatro operadores de encajillamiento (`static_cast`, `const_cast`, `reinterpret_cast`,...

[blockdmask.tistory.com](https://blockdmask.tistory.com/241)

<https://musket-ade.tistory.com/entry/CC-type-conversion-operator-staticcast-constcast-dynamic-cast-reinterpretcast>

[C++] Resumen de operadores de conver...

El operador de conversión de estilo C es un operador de conversión de tipo invencible (operador de conversión de estilo C antiguo), por...

mosquete-ade.tistory.com

<https://docs.microsoft.com/ko-kr/cpp/cpp/dynamic-cast-operator?view=msvc-170>

operador de transmisión dinámica

Descripción general del operador Dynamic_cast del lenguaje C++.

docs.microsoft.com

<https://stackoverflow.com/questions/7343833/srand-why-call-it-only-once>

srand(): ¿por qué llamarlo solo una vez?

Esta pregunta es sobre un comentario en esta pregunta ¿Forma recomendada de inicializar srand? El primer comentario dice que srand(...

stackoverflow.com

Uno

Suscribir

Otras publicaciones en la categoría ' **Programación** '

[42Seúl] Módulo CPP 08 - STL (1)

2022.09.11

[42Seúl] Módulo 07 del CPP - Plantilla (1)

2022.09.11

2/5/24, 11:46

[42Seúl] Módulo CPP 06 - Conversión de tipo

[42Seoul] Módulo 05 de CPP: reutilización y manejo de excepciones (3)	2022.09.10
[42Seoul] CPP Módulo 04 - Polimorfismo y clases abstractas (0)	2022.09.10
[42Seoul] Módulo CPP 03 - Herencia de clase (0)	2022.08.18

etiqueta

#42Seúl, #cpp

Artículos relacionados con 'Programación'

Mi registro técnico de liquidación de deudas
Este es el blog de Aggrodonk.

Suscribir

@

viajero alfa

2022.09.22 18:31

Informe

Editar/eliminar

dirección de comentario

Escribir comentario

¡Tus habilidades de escritura son excelentes! Disfruté leyendo el buen artículo. Volveré la próxima vez :)

nombre

contraseña

☐ Secreto

Por favor ingrese sus valiosos comentarios.

Deja un comentario

Mensajes recientes

- [42Seúl] ft_containers[4] - Mapa...
- [42Seúl] ft_containers[3] - TR...
- [42 Seúl] ft_containers[2] - Beck...
- [42Seúl] ft_containers[1] - S...
- [42Seúl] ft_containers[0] - y...

buscar

Por favor esc

visitantes totales

100.591

hoy	55
ayer	68