

## 77 Multiple TYPES of Data in a SINGLE VARIABLE in C++

#将不同类型的数据存储在单变量中系列2/3

本节要讲的是如何在一个变量中存储多种类型的数据，这是C++17新标准库中给我们的类。

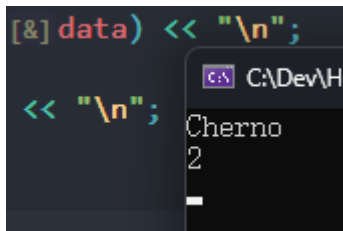
### 1. std::variant

这个和 `std::optional` 很像，它的作用是让我们不用担心处理的确切数据类型，只有一个变量放在那儿，我们之后再考虑它的具体类型。它允许你列出所有可能的类型，然后你可以决定它将是什么，如果你想的话可以把它重新赋值给任意类型，这也是你创建可能有多类型的变量的一种方式。

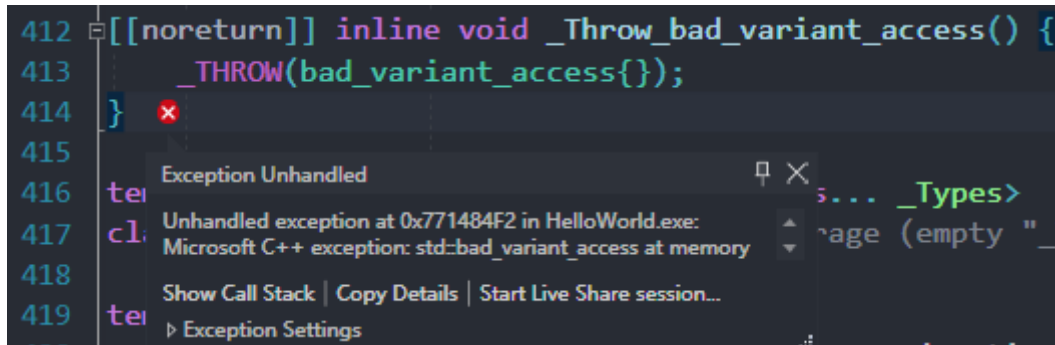
```
#include <variant>

int main()
{
    std::variant<std::string, int> data;
    data = "Cherno";
    std::cout << std::get<std::string>(data) << "\n";
    data = 2;
    std::cout << std::get<int>(data) << "\n";

    std::cin.get();
}
```



如果我们用错误的类型访问会发生什么？



`std::get` 函数会抛出一个 `bad variant access` 异常，可以用 `try catch` 来处理这个异常，不过还有其它方法而完全不使用异常就可以判断数据类型是否和我们想的一样。

首先是 `index` 函数：

C++

```
if(data.index()==0) // 索引是否为0, 即是否是类型中的第一个std::string
```

一个更好的方法, 就是使用 `get_if` 函数:

C++

```
if (auto value = std::get_if<std::string>(&data))
{
    std::string& v = *value;
}
else // 或else if来处理另一种类型
{
}
}
```

在这里的代码中, `std::get_if` 是一个用于访问 `std::variant` 的成员的函数。它会尝试以 `std::string` 的类型来访问 `data` 的值。如果 `data` 的当前类型是 `std::string`, 那么 `get_if` 将返回一个指向该值的指针, 否则它将返回 `nullptr`。然后 `if` 语句会检查 `value` 是否为 `nullptr`。如果 `value` 是 `nullptr`, 那么 `if` 语句的条件将评估为 `false`, 所以 `if` 语句块内的代码将不会执行, 反而会跳转到 `else` 语句块中去执行。

## 2. 和union对比

可参考[67 Unions in C++](#)

现在有很多关于 `variant` 的讨论说它本质上就像类型安全的 `union` (联合体), 但其实它们是非常不同的。

C++

```
std::variant<std::string, int> data;
std::cout << sizeof(data) << std::endl;
```

如果 `data` 是一个 `union` 的话, 它的大小就会是其中最大的类型的大小。  
而 `variant` 只是将所有可能的数据类型存储为单独的变量作为单独的成员:

```
std::cout << sizeof(int) << std::endl;
std::cout << sizeof(std::string) << std::endl;
std::cout << sizeof(data) << std::endl;
```

它相当于为你创建了一个结构体或者类, 只是将这两种数据类型存储为那个结构体或类中的成员。

因此, 从技术上讲, `union` 仍然是效率更高的更好的选择。

`variant` 则是更加类型安全, 不会造成未定义行为, 除非你从事的是底层优化要把内存大小保持在一个较低的位置, 比如CPU处理, 否则你应该使用它。

比如说你在桌面平台上开发, 你可以自由地使用更多的内存和更多处理能力, 那么 `std::variant` 在技术上更安全。

## 3. 其它作用

enums相关知识见[24 枚举类型](#)

根据上节课, `optional` 一般用来表示数据可能存在或不存在, 而 `variant` 可以使函数返回不同类型的值, 让我们读取文件时可以不仅仅返回一个空的可选值, 而是可以发现哪里出现问题, 并以这种方式来处理它:

```
enum class ErrorCode
{
    None = 0, NotFound = 1, NoAccess = 2
};

std::variant<std::string, ErrorCode> ReadFileAsString()
{
    if (error_occurred)
    {
        return ErrorCode::NotFound; // 返回 ErrorCode...
    }
    else
    {
        return "File contents"; // 返回文件内容
    }
}
```

这是另一个使用 **variant** 的好例子，比返回一个布尔值更详细一些，可以给我们更多的信息。