
Subject: prototype not found

Posted by [Leander](#) on Sun, 09 Jun 2019 16:59:36 GMT

[View Forum Message](#) <> [Reply to Message](#)

Hi together.

Have got this code here:

```
#include <string>
#include <stdlib.h>
using namespace std;

template <typename T>
T* new_(int size=1, bool init=false);

template <typename T>
T* new_(int size, bool init)
{
    if (size < 1) {return NULL;}
    T *p = NULL;
    if (size==1) {p = new T;}
    else {p = new T[size];}
    if (!p) exit(-1);
    if (init) memset(p, 0, sizeof(T) * size);
    return p;
}

int main()
{
    std::string *pst;
    pst = (string*)new_();
    return 0;
}
```

The return "pst" should be enough to know what is a T.

But even the cast doesnt help (couldn't deduce...), see below.

Now I can call it with args or without and the MinGW will rant it (return.txt):

main.cpp: In function 'int main()':

main.cpp:28:22: error: no matching function for call to 'new_()'

main.cpp:28:22: note: candidate is:

main.cpp:14:4: note: template<class T> T* new_(int, bool)

main.cpp:14:4: note: template argument deduction/substitution failed:

main.cpp:28:22: note: couldn't deduce template parameter 'T'

And it suggests the only existing always, the upper prototype,
but doesn't accept it.
Someone has got a tip?

Martin

Subject: Re: prototype not found
Posted by [Novo](#) on Mon, 10 Jun 2019 14:55:49 GMT
[View Forum Message](#) <> [Reply to Message](#)

Type conversion operator will be able to deduct type in your case.
struct S {
 template <typename T>
 operator T() const;
};

Subject: Re: prototype not found
Posted by [Leander](#) on Tue, 11 Jun 2019 16:45:35 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Novo

is running, have declared now

```
S<std::string> s;
```

to get a string*, accepts the defaults now also.

Thank you
Leander

Subject: Re: prototype not found
Posted by [Novo](#) on Tue, 11 Jun 2019 18:17:59 GMT
[View Forum Message](#) <> [Reply to Message](#)

The code is supposed to look like below :roll:
`#include <Core/Core.h>`

`using namespace Upp;`

```
struct new_  
{  
    new_(int size=1, bool init=false)  
    : size(size)
```

```

, init(init)
{}

template <typename T>
operator T() const { NEVER(); return T(); }

template <typename T>
operator T*() const {
    using T0 = typename std::remove_const<T>::type;
    if (size < 1) {return NULL;}
    T0 *p = NULL;
    p = new T0[size];
    if (!p) exit(-1);
    if (init) memset(p, 0, sizeof(T) * size);
    return p;
}

private:
    const int size;
    const bool init;
};

CONSOLE_APP_MAIN
{
    const char* str1 = new_(12, true);
    std::string* pst = new_(12);
    delete [] pst;
    delete [] str1;
}

```

Subject: Re: prototype not found
 Posted by [Novo](#) on Tue, 11 Jun 2019 18:21:40 GMT
[View Forum Message](#) <> [Reply to Message](#)

You should even be able to write code like this:

```

new_ new12(12);
const char* str2 = new12;
std::string* pst2 = new12;

```

Subject: Re: prototype not found
 Posted by [Leander](#) on Thu, 13 Jun 2019 07:49:47 GMT
[View Forum Message](#) <> [Reply to Message](#)

Not bad. Going to try this soon.
Thanks for your effort.

Martin

Subject: Re: prototype not found
Posted by [mirek](#) on Fri, 21 Jun 2019 15:59:11 GMT
[View Forum Message](#) <> [Reply to Message](#)

Leander wrote on Sun, 09 June 2019 18:59Hi together.

Have got this code here:

```
#include <string>
#include <stdlib.h>
using namespace std;

template <typename T>
T* new_(int size=1, bool init=false);

template <typename T>
T* new_(int size, bool init)
{
    if (size < 1) {return NULL;}
    T *p = NULL;
    if (size==1) {p = new T;}
    else {p = new T[size];}
    if (!p) exit(-1);
    if (init) memset(p, 0, sizeof(T) * size);
    return p;
}

int main()
{
    std::string *pst;
    pst = (string*)new_();
    return 0;
}
```

The return "pst" should be enough to know what is a T.

No, it is not. Types only can get resolved as parameters. But, good news, you can specify it directly:

```
pst = new_<string>();
```

Subject: Re: prototype not found
Posted by [Leander](#) on Mon, 24 Jun 2019 19:46:56 GMT
[View Forum Message](#) <> [Reply to Message](#)

Hi Mirek

Ooops, never considered to write it like so, and it is working.
Thanks so much for the hint.

Leander
