Schemat konstrukcja pliku Makefile

Bogdan Kreczmer

bogdan.kreczmer@pwr.wroc.pl

Zakład Podstaw Cybernetyki i Robotyki
Instytut Informatyki, Automatyki i Robotyki
Politechnika Wrocławska

Kurs: Programowanie obiektowe

Copyright©2008 Bogdan Kreczmer

Niniejszy dokument zawiera materiały do wykładu dotyczącego programowania obiektowego. Jest on udostępniony pod warunkiem wykorzystania wyłącznie do własnych prywatnych potrzeb i może on być kopiowany wyłącznie w całości, razem z niniejszą stroną tytułową.

Niniejsza prezentacja została wykonana przy użyciu systemu składu LaTeX *oraz stylu* `beamer`, *którego autorem jest Till Tantau.*

Strona domowa projektu `Beamer`:
`http://latex-beamer.sourceforge.net`

Stan początkowy

glowny.cpp

```
#include "modul.hh"

int main()
{
  Wyswietl("Hejka!!");
}
```

modul.hh

```
#ifndef MODUL_HH
#define MODUL_HH

void Wyswietl(const char*);

#endif
```

modul.cpp

```
#include <iostream>
#include "modul.hh"
using namespace std;
void Wyswietl(const char* Napis)
{
  cout << Napis << endl;
}
```

Program składa się z modułu głównego i pomocniczego.

Stan początkowy

glowny.cpp

```
#include "modul.hh"

int main()
{
  Wyswietl("Hejka!!");
}
```

modul.hh

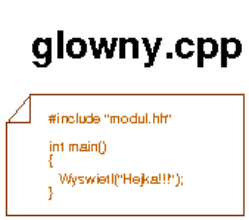
```
#ifndef MODUL_HH
#define MODUL_HH

void Wyswietl(const char*);

#endif
```

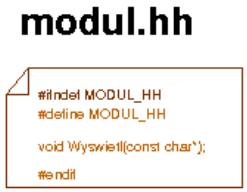
modul.cpp

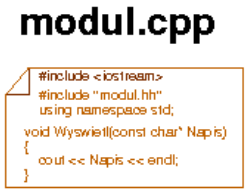
```
#include <iostream>
#include "modul.hh"
using namespace std;
void Wyswietl(const char* Napis)
{
  cout << Napis << endl;
}
```

Moduł główny i pomocniczy korzystają z pliku ¨modul.hh¨.

Stan początkowy i cel

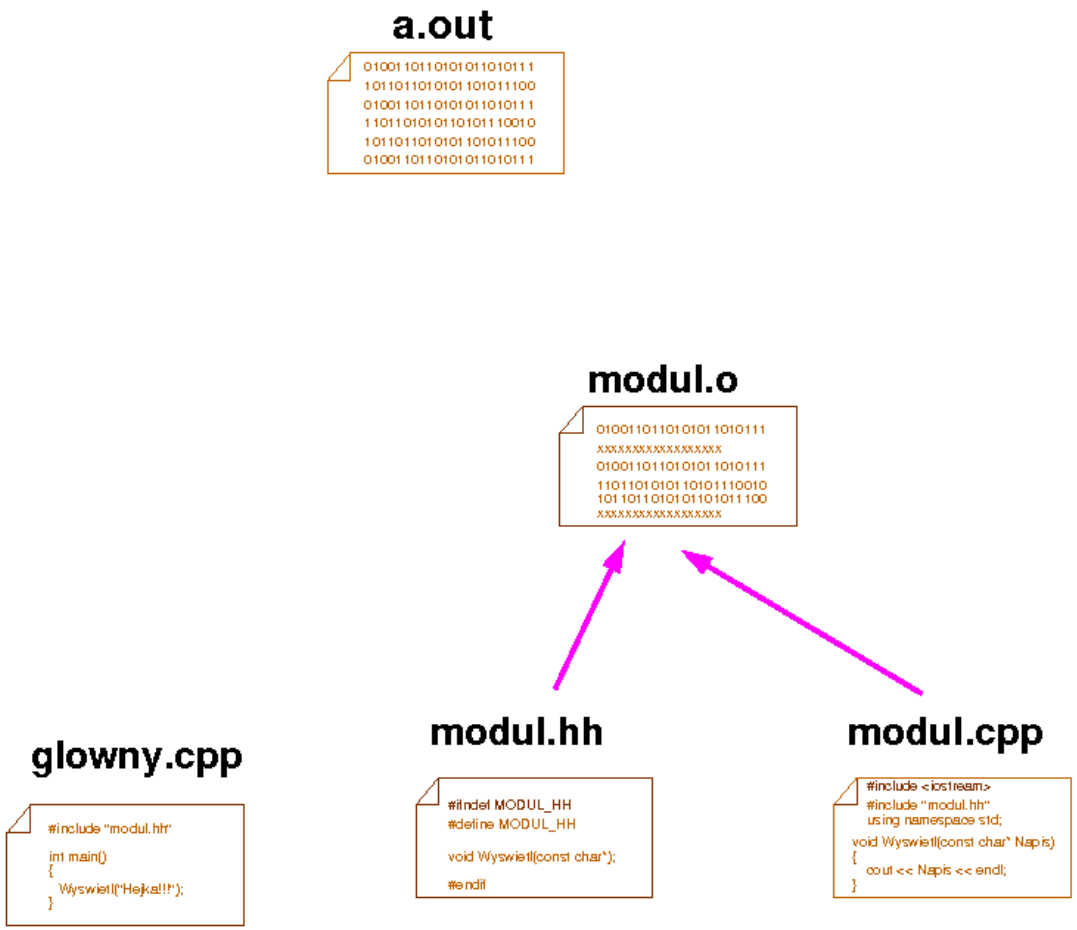






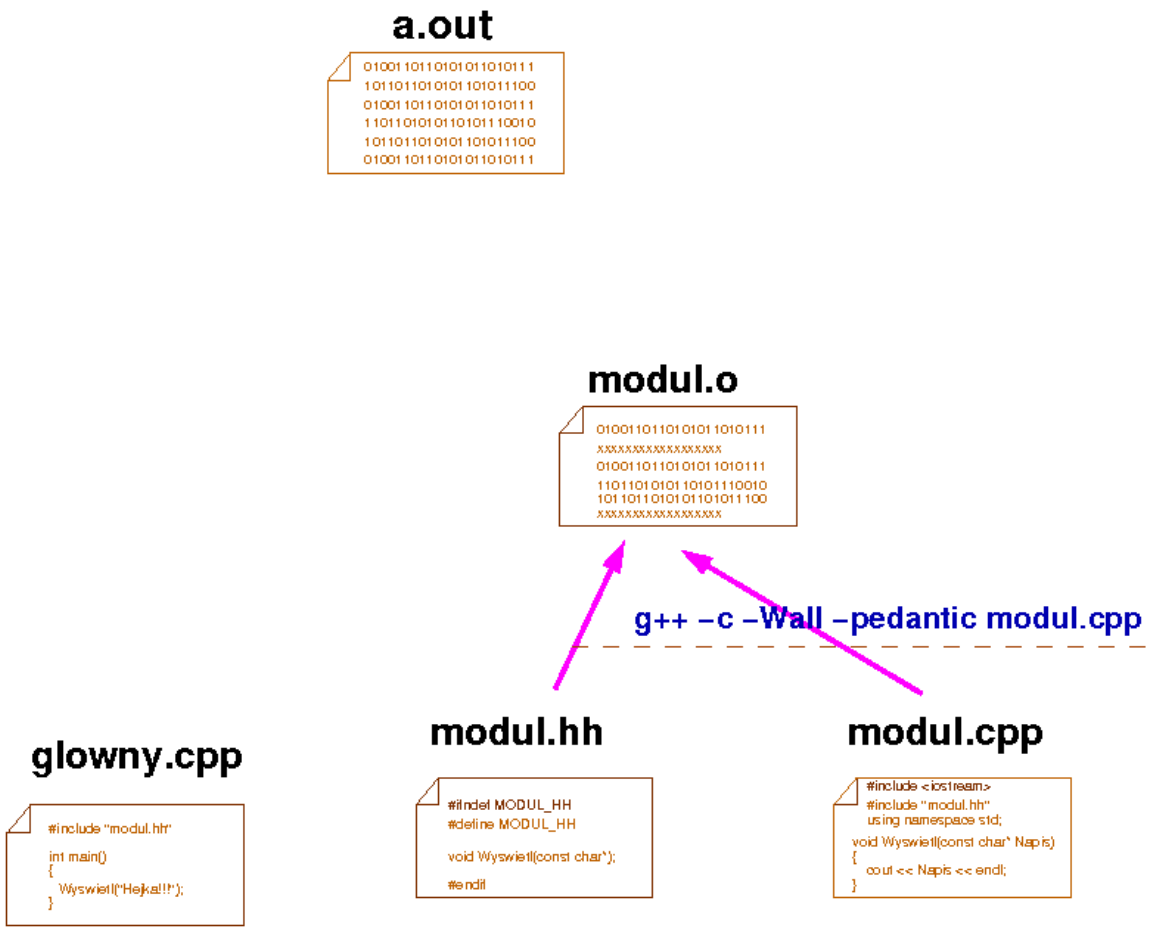
Celem jest zbudowanie pliku wykonywalnego ¨`a.out`¨.

Etap pośredni – kompilacja modułu



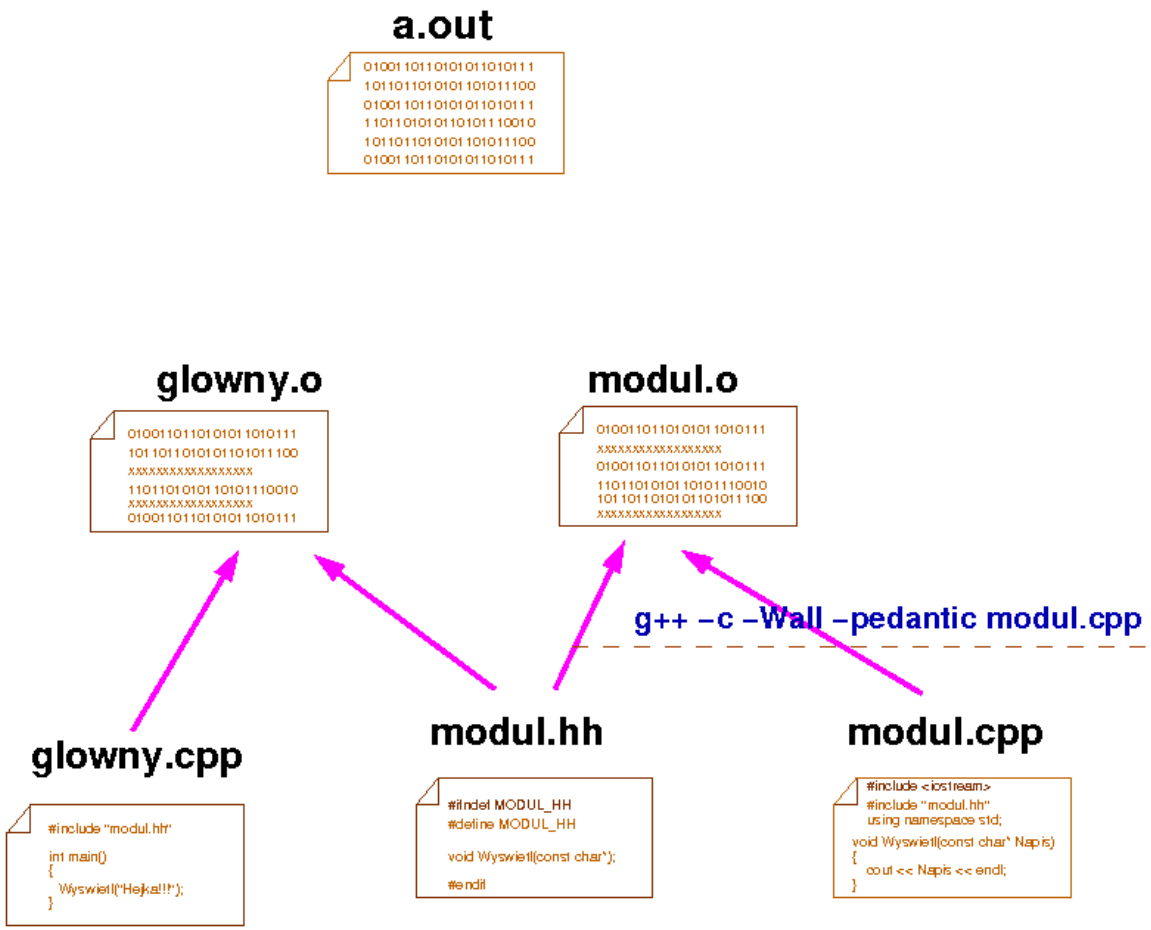
W tym celu należy utworzyć plik ¨`modul.o`¨

Etap pośredni – kompilacja modułu



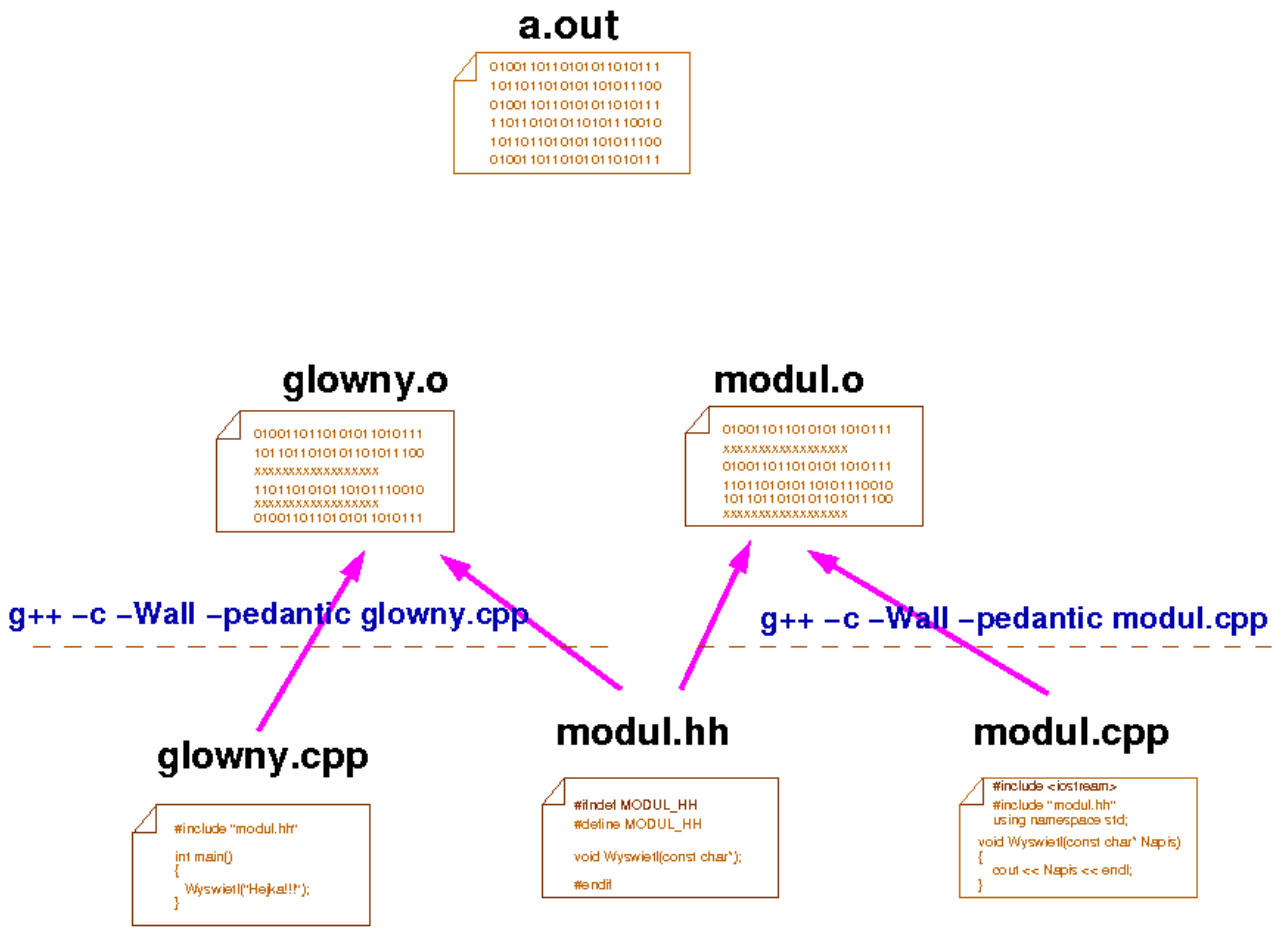
Przepisem na utworzenie tego pliku jest kompilacja.

Etap pośredni – kompilacja modułu głównego



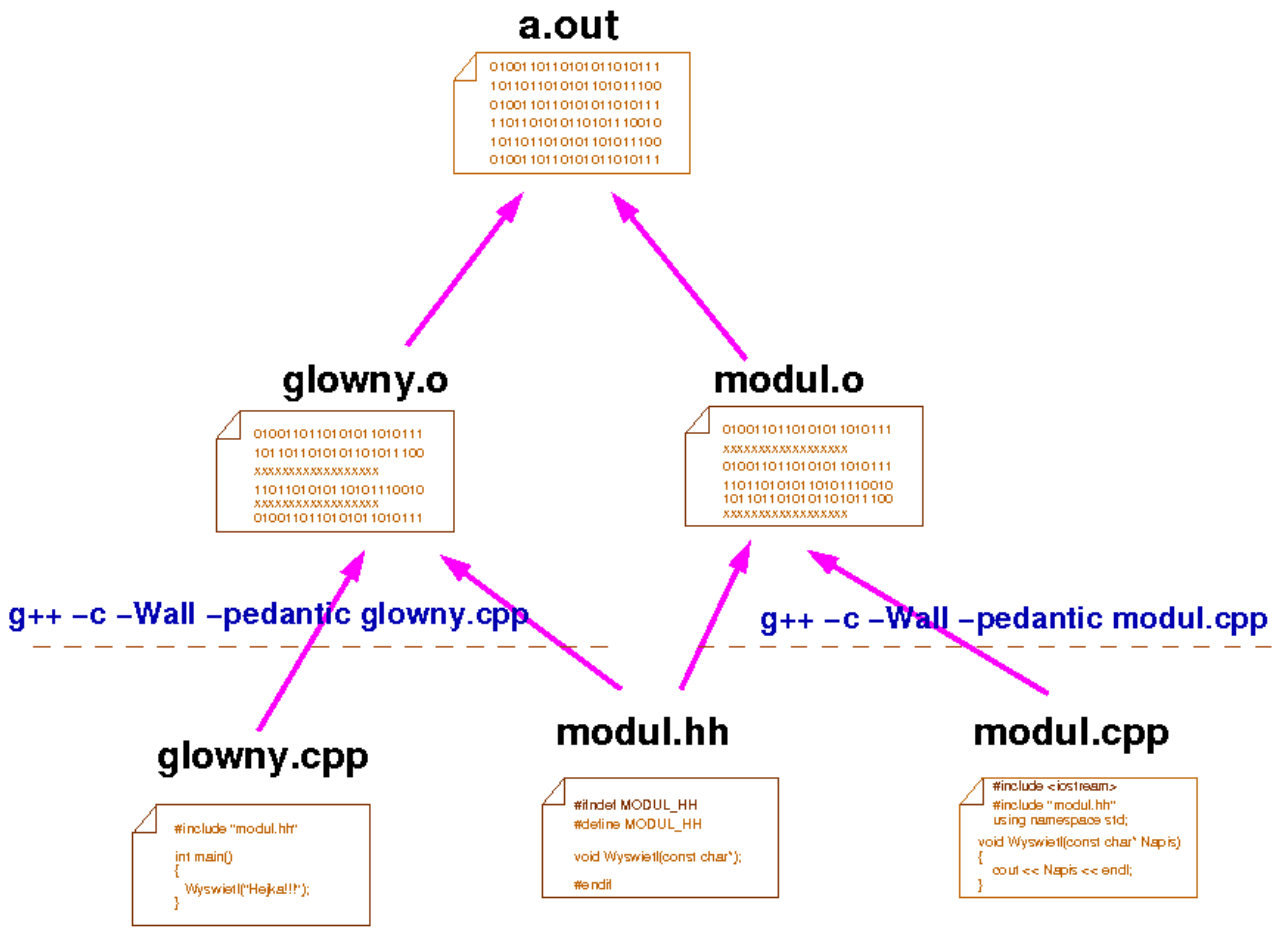
Podobnie należy utworzyć plik ¨`glowny.o`¨

Etap pośredni – kompilacja modułu głównego



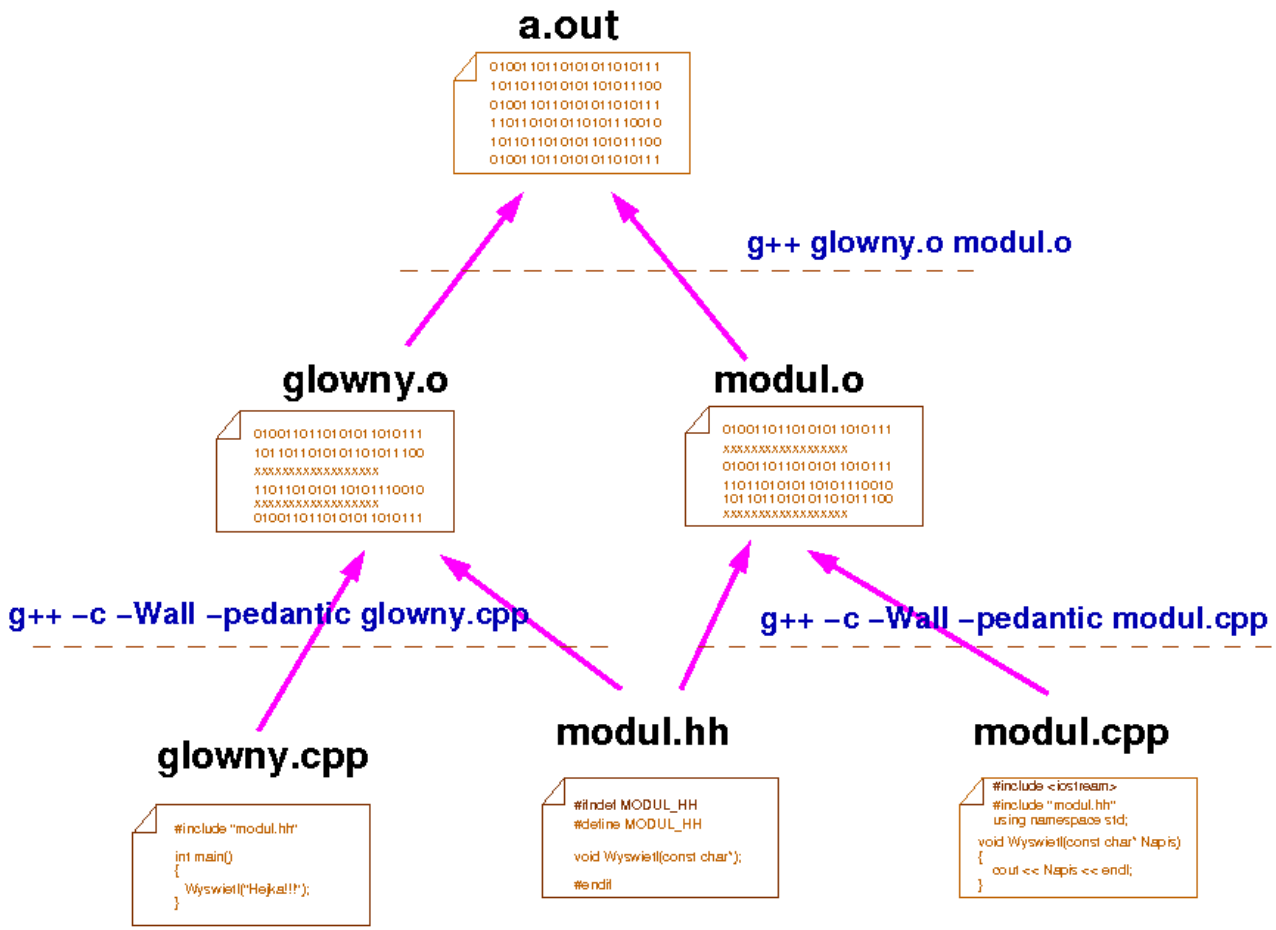
Należy więc skompilować plik ¨`glowny.cpp`¨.

Etap pośredni – konsolidacja programu



Teraz należy zbudować finalny plik.

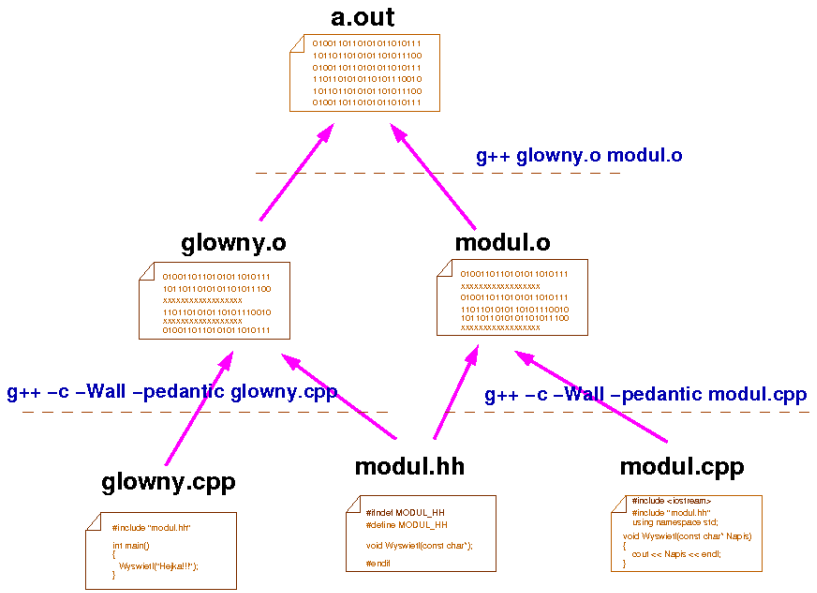
Etap pośredni – konsolidacja programu



Realizujemy to poprzez konsolidację plików typu *object*.

Plik Makefile

Makefile

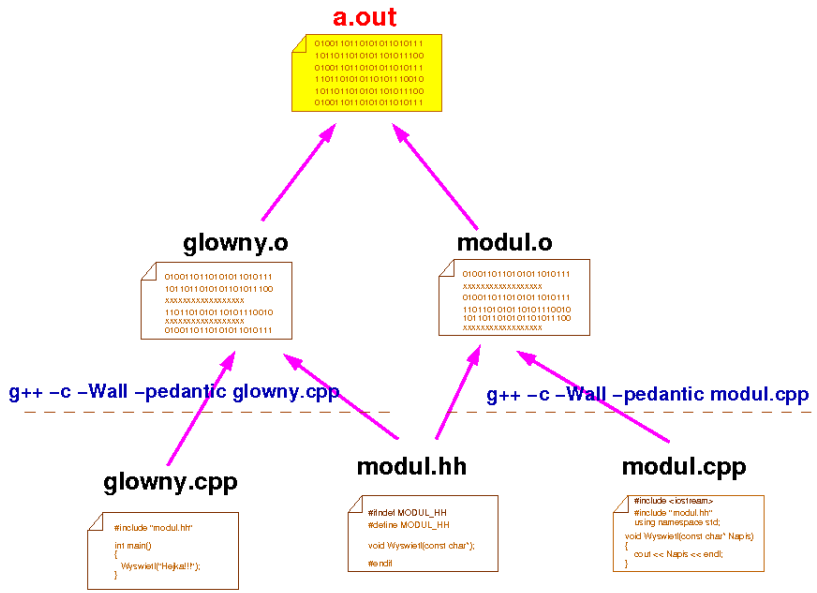


Na podstawie stworzonego schematu kolejnych operacji zbudujemy plik Makefile.

Plik Makefile

Makefile

```
a.out:
```

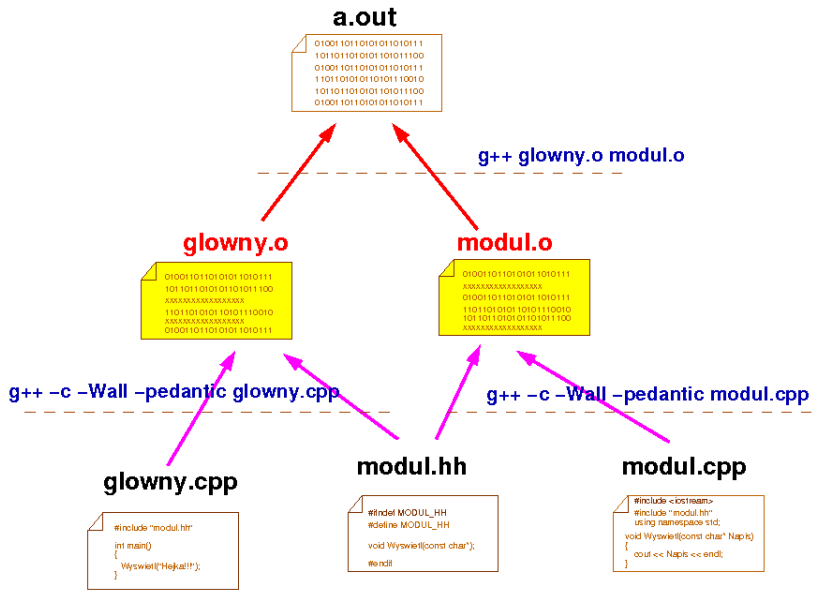


Podajemy zasadniczy cel, który ma być utworzony

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
```

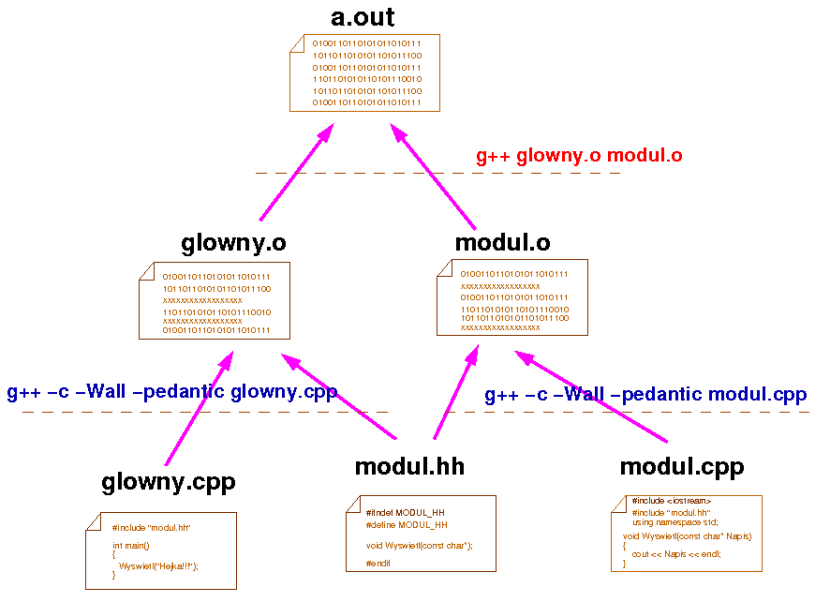


Wypisujemy podcele od których on zależy.

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
TAB g++ glowny.o modul.o
```

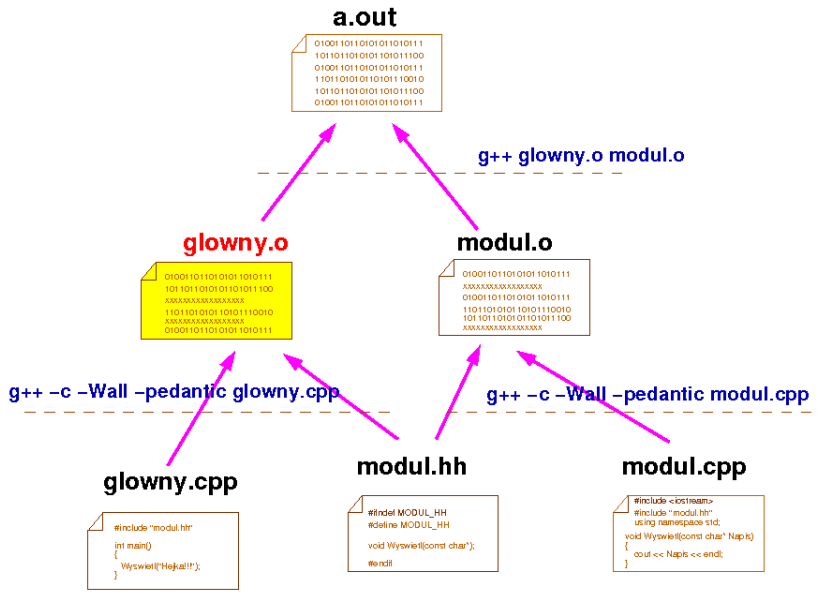


Podajemy przepis utworzenia (lub aktualizacji) celu ¨a.out¨. Należy pamiętać, że linię rozpoczynamy od **znaku tabulacji**.

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
	g++ glowny.o modul.o

glowny.o:
```

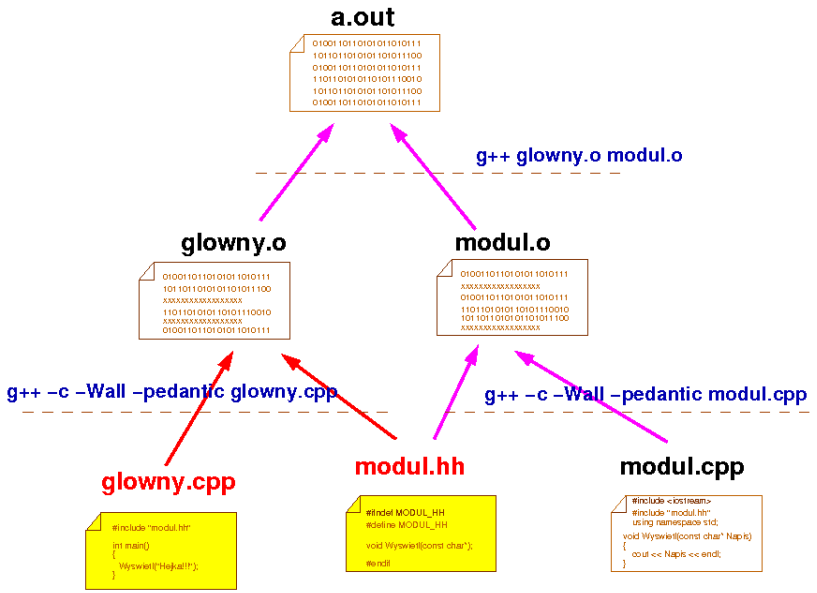


Podobnie robimy dla podcelu ¨glowny.o¨ i następnych podceli (w tym przypadku ¨modul.o¨)

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
```

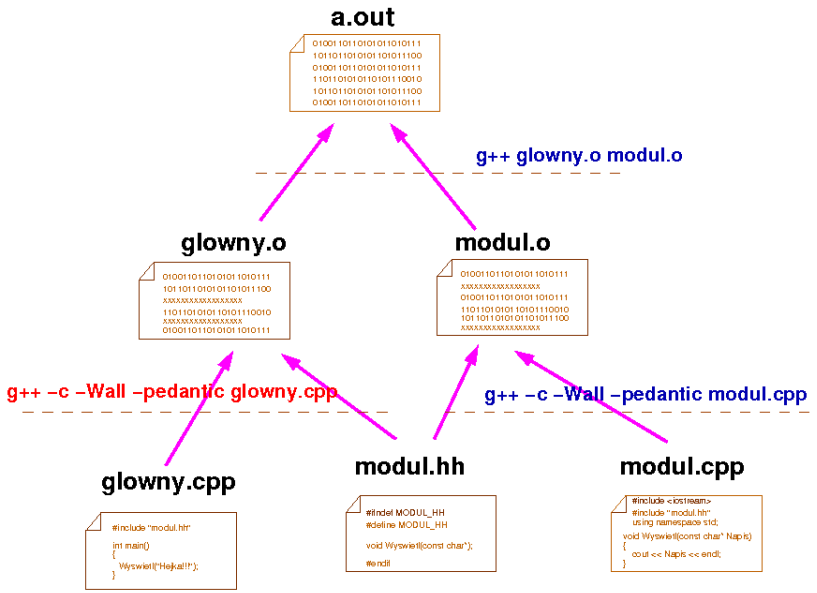


Wypisujemy podcele od których on zależy.

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
| TAB | g++ -c -Wall -pedantic glowny.cpp
```

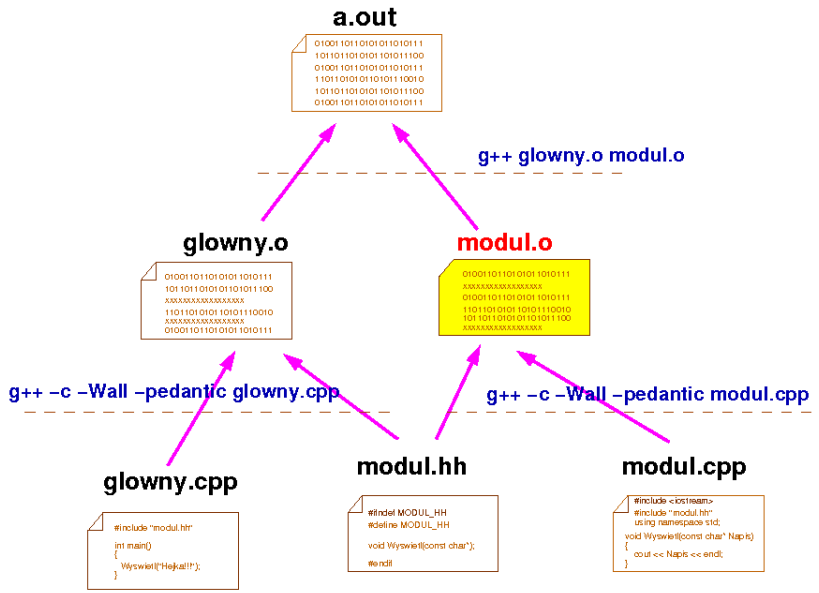


Podajemy przepis utworzenia (lub aktualizacji) celu ¨glowny.o¨.

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
        g++ -c -Wall -pedantic glowny.cpp

modul.o:
```

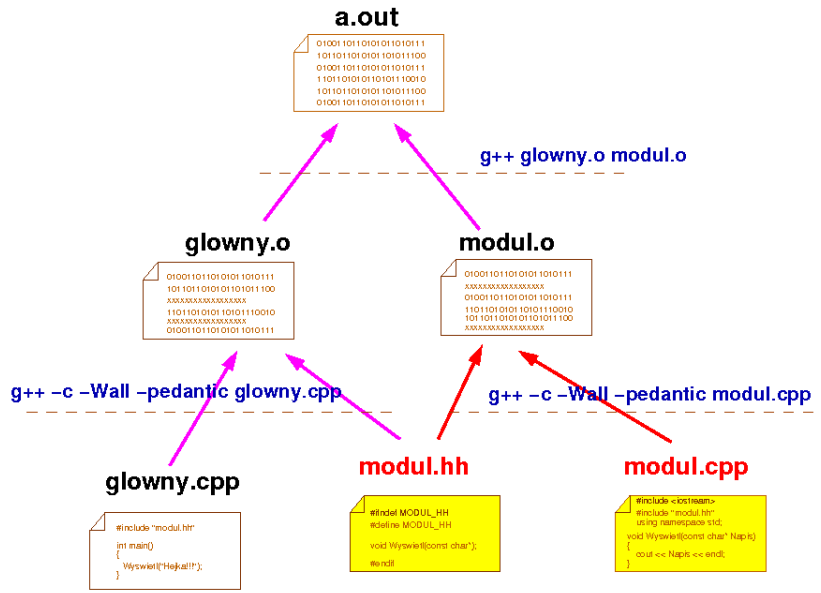


Analogicznie postępujemy dla podcelu ¨modul.o¨.

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
        g++ -c -Wall -pedantic glowny.cpp

modul.o: modul.hh modul.cpp
```

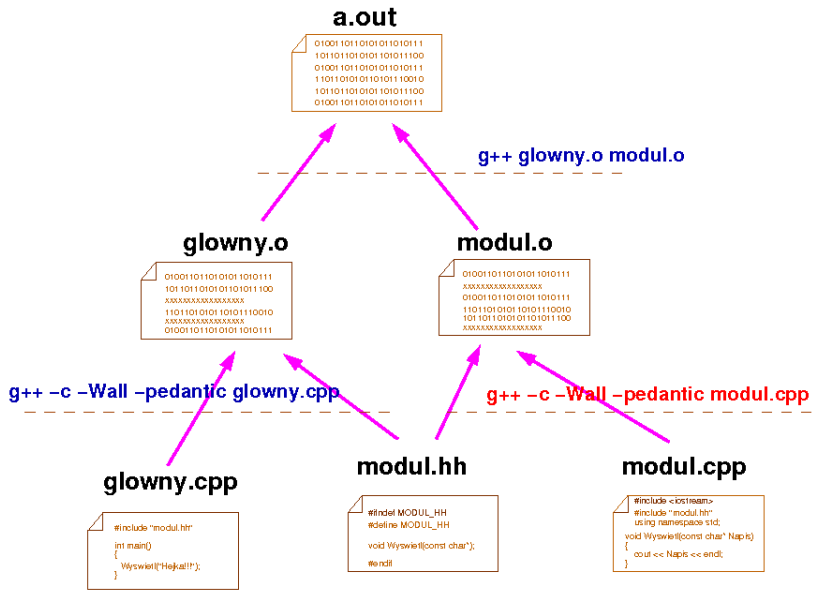


Wypisujemy podcele od których on zależy.

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
        g++ -c -Wall -pedantic glowny.cpp

modul.o: modul.hh modul.cpp
| TAB | g++ -c -Wall -pedantic modul.cpp
```

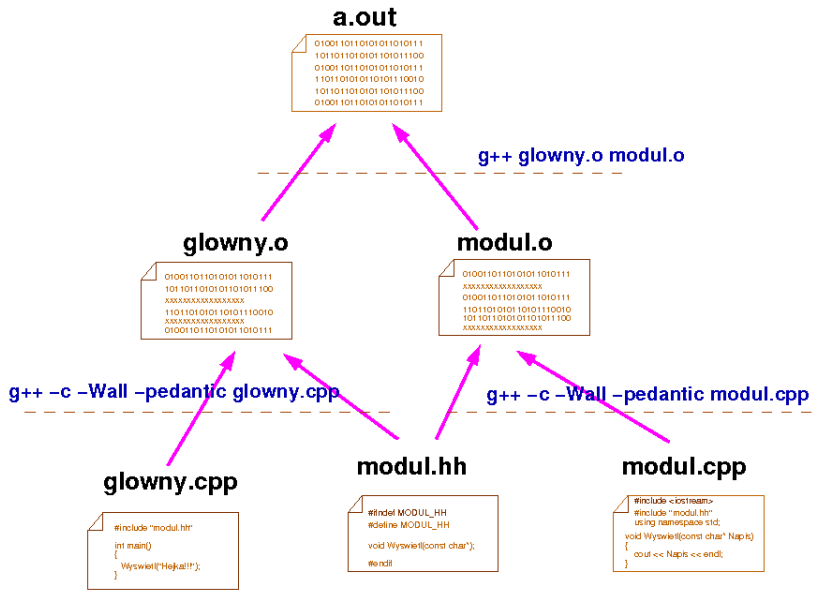


Podajemy przepis utworzenia (lub aktualizacji) celu ¨modul.o¨.

Plik Makefile

Makefile

```
a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
        g++ -c -Wall -pedantic glowny.cpp

modul.o: modul.hh modul.cpp
        g++ -c -Wall -pedantic modul.cpp
```

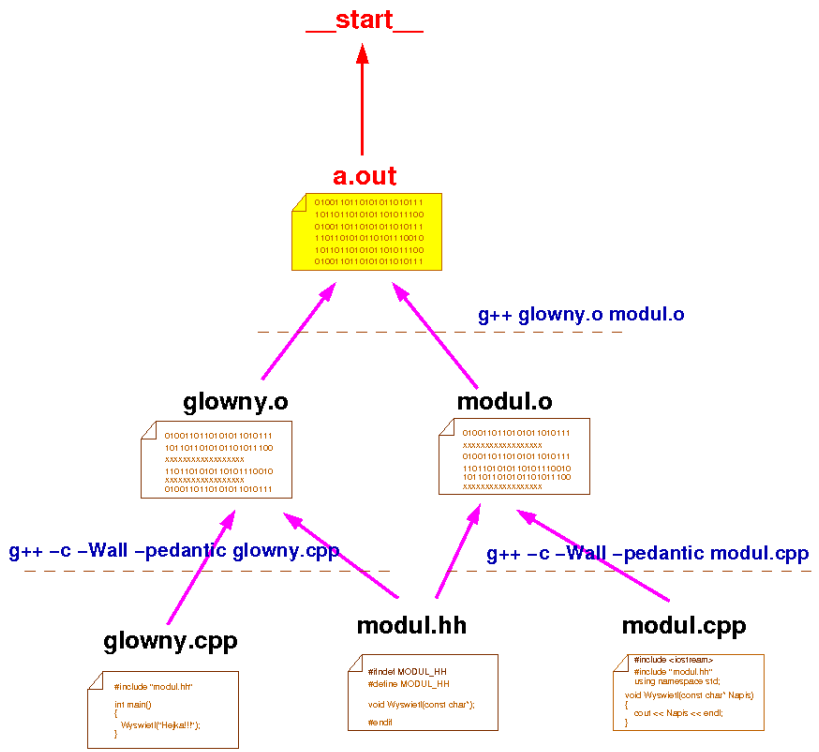


Tak napisany plik Makefile umożliwia automatyczne przeprowadzenie kompilacji i konsolidacji aplikacji. Jednak nie pozwala na jej uruchomienie.

Plik Makefile

Makefile

```
__start__: a.out


a.out: glowny.o modul.o
	g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
	g++ -c -Wall -pedantic glowny.cpp

modul.o: modul.hh modul.cpp
	g++ -c -Wall -pedantic modul.cpp
```

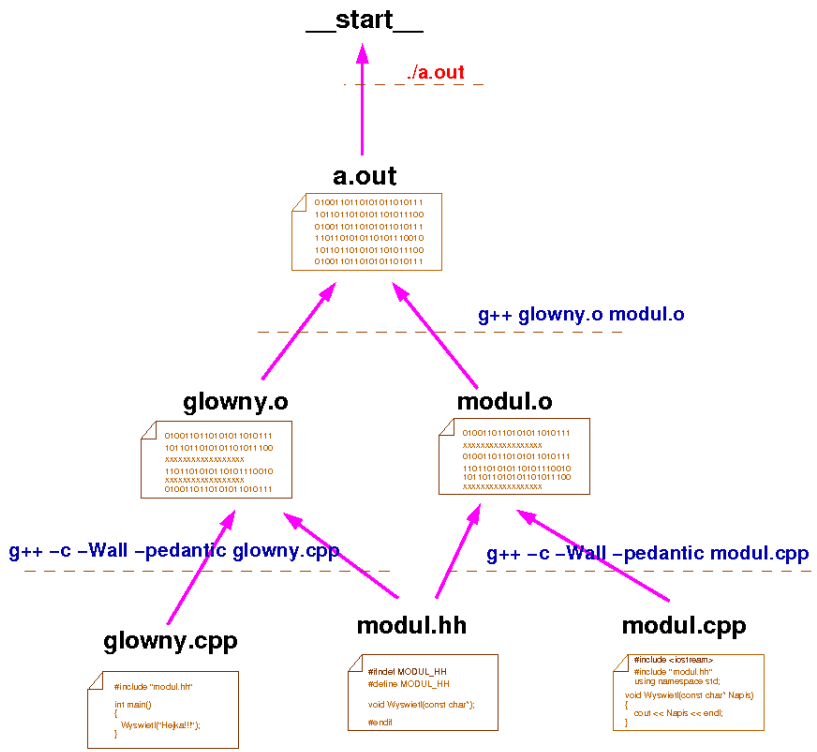


Dodanie nowego podcelu, który nigdy nie zostanie utworzony, pozwala wymusić uruchomienie aplikacji, gdy zostanie ona poprawnie zbudowana.

Plik Makefile

Makefile

```
__start__: a.out
    TAB    ./a.out

a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
        g++ -c -Wall -pedantic glowny.cpp

modul.o: modul.hh modul.cpp
        g++ -c -Wall -pedantic modul.cpp
```

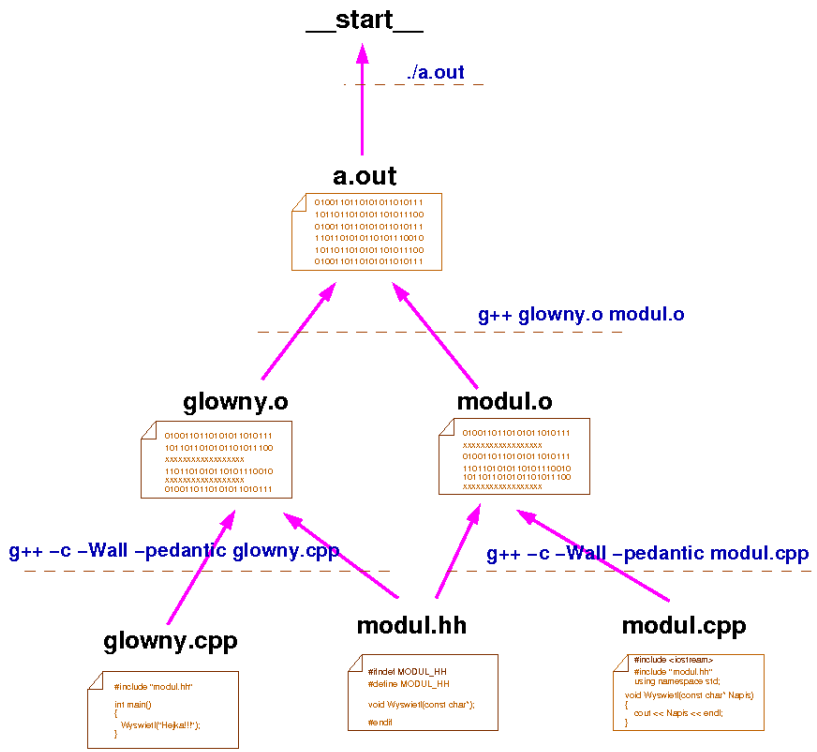


Jako przepis utworzenia celu `__start__` wpisujemy polecenie uruchomienia aplikacji.

Plik Makefile

Makefile

```
__start__: a.out
        ./a.out

a.out: glowny.o modul.o
        g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
        g++ -c -Wall -pedantic glowny.cpp

modul.o: modul.hh modul.cpp
        g++ -c -Wall -pedantic modul.cpp
```



Należy pamiętać, że ostatnią linię tekstu w pilku należy zakończyć znakiem przejścia do nowej linii (tzn. ostatnia linia w pliku musi być linią pustą).

Plik `Makefile` – postać finalna

```
__start__: a.out
	./a.out

a.out: glowny.o modul.o
	g++ glowny.o modul.o

glowny.o: glowny.cpp modul.hh
	g++ -c -Wall -pedantic glowny.cpp

modul.o: modul.hh modul.cpp
	g++ -c -Wall -pedantic modul.cpp
```

Koniec prezentacji