Różnice między C i C++

1 Cel zadania

Realizacja niniejszego zadania powinna unaocznić wybrane cechy, które są wspólne dla języków C i C++ oraz te, które je różnią.

2 Opis zadania

2.1 Różnice między C i C++ — wybrane przykłady

W tej części należy obejrzeć dostarczone przykładowe programy napisane w języku C. Następnie należy przeprowadzić ich kompilację i konsolidację, aby uzyskać program wykonywalny. Operację należy wykonać za pomocą kompilatora języka C (zalecane jest przeprowadzenie tej operacji zarówno za pomocą kompilatora cc jak też gcc). Następnie należy tę operację powtórzyć używając kompilatora języka C++. Jeżeli proces ten zakończył się powodzeniem, to należy porównać działanie programów.

Jeżeli proces kompilacji lub konsolidacji nie powiódł się, należy spróbować znaleźć przyczynę. Podobnie jeśli okaże się, że działanie programów różni się. Niezbędne informacje zamieszczone są w materiałach dostępnych pod adresem:
`http://rab.ict.pwr.wroc.pl/~kreczmer/kpo/zadania/zad-roznice-c-cpp/slides/roznice-c-cpp.pdf`

Poniżej opisany jest sposób postępowania dla czterech dostarczonych programów, których działanie należy wypróbować i porównać. Źródła programów znajdą się na serwerach panamint/diablo w podkatalogu: `~bk/edu/po/zad/z1`. Zalecane jest utworzenie analogicznej struktury podkartotek i przekopiowanie całej zawartości, np.:

```
mkdir -p ~/kpo/zad
cd  ~/kpo/zad
cp -r ~bk/edu/kpo/zad/z1 .
cd  z1
```

W podkatalogu `z1` znajdują się następujące podkatalogi: `test1, test2, test3, test4`. Zadanie dotyczy zawartości tych podkatalogów.

2.1.1 `z1/test1`

W podkatalogu tym znajduje się plik `rownanie.c`. Zawiera on kod programu obliczającego pierwiastki równania kwadratowego. Zgodnie ze wskazówkami osoby prowadzącej zajęcia laboratoryjne należy utworzyć program wykonywalny za pomocą kompilatora języka C i C++, a następnie porównać jego działanie.

2.1.2 `z1/test2`

W podkatalogu znajduje się plik `konwerter.c`. Należy przeprowadzić analogiczne operacje jak w punkcie poprzednim.

2.1.3 `z1/test3`

W podkatalogu znajduje się pliki: `prog.c` i `modul.c`. Należy przeprowadzić osobną kompilację i konsolidację za pomocą kompilatora/konsolidatora dla języka C i C++. Następnie należy porównać otrzymane wyniki. Przykładowy sposób postępowania:

Kompilacja i konsolidacja dla języka C

```
gcc -c -Wall -pedantic modul.c      # etap kompilacji modułu pomocniczego
gcc -c -Wall -pedantic program.c    # etap kompilacji modułu głównego
gcc -Wall program.o modul.o         # konsolidacja
```

Kompilacja i konsolidacja dla języka C++

```
g++ -c -Wall -pedantic modul.c      # etap kompilacji modułu pomocniczego
g++ -c -Wall -pedantic program.c    # etap kompilacji modułu głównego
g++ -Wall program.o modul.o         # konsolidacja
```

2.1.4 `z1/test4`

W podkatalogu, tak jak wcześniej, znajduje się pliki: `program.c` i `modul.c`. Należy przeprowadzić identyczne operacja jak we wcześniejszym podpunkcie i dokładnie w takiej samej kolejności.

2.1.5 Praca własna z dostarczonymi przykładami

Po zademonstrowaniu wszystkich przykładów dla każdego z przypadków, w którym pojawił się problem, trzeba będzie samodzielnie znaleźć przyczynę i usunąć źródło problemu.

2.2 Poprawność programów, plik `core`

W podkatalogu `z1/zly_program` znajduje się program `przetworz.c`. Zawiera on jeden istotny błąd. Aby go zidentyfikować, konieczne jest przetestowanie programu. Należy wspomniany program skompilować za pomocą kompilatora `gcc` i uruchomić otrzymany plik wynikowy.

```
/usr/bin/gcc -Wall -pedantic przetworz.c
./a.out
```

Program spowoduje błąd naruszenia pamięci. Należy umożliwić mu zrzut stanu pamięci procesu. Pojawi się on w postaci pliku `core`. Aby to było możliwe, należy wykonać polecenie:

```
ulimit -c unlimited
```

A następnie należy ponownie uruchomić program `./a.out`.

```
./a.out
```

Tym razem, oprócz informacji o błędzie, w kartotece bieżącej pojawi się plik `core`. Należy go wykorzystać i podjąć próbę zlokalizowania źródła problemu poprzez obejrzenie zawartości stosu wywołań funkcji. W tym celu należy uruchomić program `gdb`.

```
/usr/bin/gdb a.out core
```

Wyświetlenie stosu wywołań można uzyskać poprzez polecenie `bt`. Zakończenie współpracy z programem dokonujemy za pomocą polecenia `q`. Następnie należy powtórnie skompilować program w trybie generacji dodatkowej informacji dla debuggera (opcja `-g`).

```
/usr/bin/gcc -g -Wall -pedantic przetworz.c
```

Tak jak poprzednio należy uruchomić program wywołaniem `./a.out`, a następnie obejrzeć stos wywołań funkcji za pomocą debuggera `gdb`. Na podstawie otrzymanych informacji trzeba zlokalizować źródło błędu i wyjaśnić przyczynę dziwnego zachowania się programu.

W wyjaśnieniu problemu pomocne jest skompilowanie programu za pomocą kompilatora języka C++, a mianowicie `g++`. Należy zwrócić uwagę na ostrzeżenie, które wyświetli się w trakcie kompilacji. ***Jest to przykład pokazujący dlaczego nie należy lekceważyć ostrzeżeń***.