Politechnika Śląska
Wydział Informatyki, Elektroniki i Informatyki

Programowanie Komputerów 2

Środowisko programistyczne

Autor	Dominik Uszok
Prowadzący	mgr inż. Maciej Długosz
Rok akademicki	2019/2020
Kierunek	Informatyka
Rodzaj studiów	SSI
Semestr	2
Termin laboratorium	piątek, 12:00 – 13:30
Sekcja	21
Termin oddania sprawozdania	2020-09-16

---

1 Treść zadania

Proszę napisać program do zarządzania strukturą plików środowiska programistycznego. Każdy z plików znajduje się w dokładnie jednym projekcie. Projekty zgrupowane są w solucji.

Program powinien umożliwiać:

1. przechowywanie powyższej struktury w pamięci oraz jej modyfikację (dodawanie i usuwanie elementów),
2. import oraz eksport struktury z/do pliku binarnego,
3. utworzenie struktury katalogów i plików na dysku we wskazanej przez użytkownika lokalizacji, odpowiadającej strukturze solucji,
4. utworzenie struktury na bazie istniejącej struktury katalogów i plików we wskazanej przez użytkownika lokalizacji.

Uwagi techniczne Sugerowana struktura danych: lista projektów, której każdy element zawiera podwieszoną listę plików. W środowisku Windows do pobierania zawartości katalogu oraz tworzenia katalogów można wykorzystać funkcje WinAPI (niestandardowa biblioteka windows.h).

2 Analiza zadania

Zagadnienie przedstawia problem tworzenia list i odczytu konkretnych danych z tych list do celów przedstawionych w treści zadania.

2.1 Struktury danych

W programie wykorzystano strukturę solucji która zawiera listę jednokierunkową projektów które zawierają listę jednokierunkową plików.

Taka struktura umożliwia łatwe przyporządkowanie projektów solucji i plików projektom.

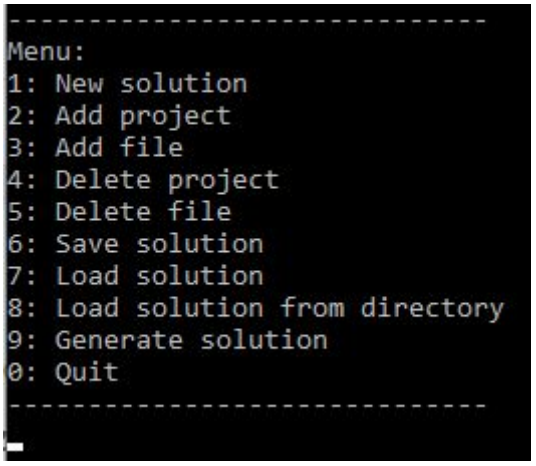
2.2 Algorytmy

Program wyszukuje określone elementy solucji porównując szukaną nazwę elementu do nazwy elementu znalezionego i jeśli pasują wykonuje określoną operację np. stworzenie pliku w znalezionym projekcie. Jeśli nie jest to szukany element to przechodzi do następnego poprzez wskaźnik next. Algorytm kończy się na wskaźniku NULL.

Usuwanie solucji realizowane jest przez rekurencyjne przejście przez listę.

3 Specyfikacja zewnętrzna

Program uruchamiany jest z pliku devenv.exe po uruchomieniu którego ukazuje się menu z 10 opcjami.



Opcje 1-5 zmieniają strukturę solucji zapisaną tylko w pamięci programu.

Szósta opcja - Save solution zapisuje strukturę solucji w pliku o rozszerzeniu .save w miejscu uruchomienia programu. Opcja Load solution wczytuje solucje z lokacji uruchomienia programu o nazwie podanej przez użytkownika.

Opcja ósma i dziewiąta oczekują od użytkownika podanie ścieżki do wczytania lub generacji solucji.

4 Specyfikacja wewnętrzna

Program został zrealizowany zgodnie z paradygmatem strukturalnym. W programie rozdzielono interfejs (komunikację z użytkownikiem) od logiki aplikacji - modyfikacji list, generowania i wczytywania plików binarnych, generowania katalogów i wczytywania z już istniejących.

4.1 Ogólna struktura programu

W funkcji głównej znajduje się instrukcja wyboru której fraza wywołana zależy od wyboru użytkownika. Fraza 1 wywołuje funkcję *new_solution*, fraza 2 funkcję *new_project*, fraza 3 funkcję *new_file*, fraza 4 funkcję *delete_project* a fraza 5 funkcję *delete_file*. Każda z fraz sprawdza czy nazwa solucji/projektu/pliku podana przez użytkownika jest poprawna.

Fraza 6 zawiera funkcję wytwarzającą plik binarny ze strukturą solucji - *saveSolution* która zwraca 0 gdy zostanie wywołana poprawnie lub 1 gdy nie uda się stworzyć pliku binarnego. Fraza 7 wczytuje zapisany plik binarny ze strukturą solucji przy użyciu funkcji *loadSolution* która zwraca NULL przy nieudanej próbie odczytu i właściwy wskaźnik przy udanej.

Fraza 8 tak samo jak fraza 1 wywołuje *new_solution* po czym wywołuje swoją własną funkcję - *loadSolutionAlreadyCreated* która zwraca 1 przy nieznalezieniu ścieżki i 0 przy znalezieniu.

Fraza 9 wywołuje funkcję *createSolutionDirectories* która tworzy katalogi solucji zapisanej w pamięci programu.

Fraza 0 usuwa z pamięci solucje wraz z jej projektami i plikami przy użyciu funkcji *delete_solution* po czym zamyka program zwracając 0.

4.2 Szczegółowy opis typów i funkcji

Szczegółowy opis typów i funkcji zawarty w załączniku.

5 Testowanie

Program został przetestowany wprowadzając różnej długości nazwy solucji/projektów/plików które może wprowadzić użytkownik. Zbyt długie nazwy zostają ucięte do ich maksymalnej długości. Niepoprawne ścieżki wyświetlają odpowiedni komunikat nieistnienia takiej lokacji. Próba wczytania solucji z nieistniejącego pliku binarnego wyświetla użytkownikowi odpowiedni komunikat o nieistnieniu takiego pliku. Zapisanie nieistniejącej solucji w pliku binarnym również kończy się odpowiednim komunikatem.

Program został sprawdzony pod kątem wycieków pamięci.

6 Wnioski

Program “Środowisko programistyczne” jest prostym programem który wykorzystuje nieskomplikowaną strukturę listy jednokierunkowej w liście jednokierunkowej. Jednak wymaga użycia dużej ilości funkcji przez wiele operacji wykonywanych na wyżej wymienionej strukturze. Trudne okazały się operacje na tekstowym typie danych przez konieczność użycia specyficznych funkcji dla języka C do modyfikacji tego typu danych. Problematyczne okazały się ścieżki wprowadzane przez użytkownika które zawierały znaki białe co wymagało rozważenie innej funkcji odczytującej dane.

Literatura

- https://docs.microsoft.com/en-us/windows/win32/fileio/directory-management-functions
- https://docs.microsoft.com/en-us/windows/win32/api/fileapi/
- https://stackoverflow.com/questions/2314542/listing-directory-contents-using-c-and-windows
- https://docs.microsoft.com/pl-pl/cpp/cpp/lvalue-reference-declarator-amp?view=vs-2019

Dodatek
Szczegółowy opis typów i funkcji

Środowisko programistyczne

Wygenerowano przez Doxygen 1.8.20

1 Indeks klas

1.1 Lista klas

Tutaj znajdują się klasy, struktury, unie i interfejsy wraz z ich krótkimi opisami:

2 Indeks plików

2.1 Lista plików

Tutaj znajduje się lista wszystkich udokumentowanych plików z ich krótkimi opisami:

3 Dokumentacja klas

3.1 Dokumentacja struktury files_

`#include <structs.h>`

Atrybuty publiczne

- char filename [100]
  nazwa pliku
- char file_in_projectname [100]
  nazwa projektu w którym znajduje się plik
- struct files_ * next
  wskaźnik na następny plik

3.1.1 Opis szczegółowy

Struktura reprezentująca pliki w projekcie

3.2 Dokumentacja struktury projects_

`#include <structs.h>`

Atrybuty publiczne

- char projectname [100]
  nazwa projektu
- int numberOfFiles
  liczba plików w projekcie
- struct projects_ * next
  wskaźnik na następny projekt
- files * filelist
  wskaźnik na liste plików

3.2.1 Opis szczegółowy

Struktura reprezentująca projekty w solucji

3.3 Dokumentacja struktury solution_

`#include <structs.h>`

Atrybuty publiczne

- char solutionname [100]
  nazwa solucji
- int numberOfProjects
  liczba projektów w solucji
- projects * projectlist
  wskaźnik na liste projektów

3.3.1 Opis szczegółowy

Struktura reprezentująca solucje

4 Dokumentacja plików

4.1 Dokumentacja pliku functions.h

```
#include <stdio.h>
#include <string.h>
#include "structs.h"
```

Funkcje

- solution * new_solution (char *solutionname)
  Funkcja tworząca nową solucje.
- void new_project (solution *solution, char *name)
  Funkcja tworząca nowy projekt.
- int new_file (solution *solution, char *project_name, char *file_name)
  Funkcja tworząca nowy plik.
- int delete_file (solution *solution, char *project_name, char *file_name)
  Funkcja usuwająca plik.
- void delete_all_files (files **file)
  Usuwa wszystkie pliki. Wywoływana w delete_all_projects().
- void delete_all_projects (projects **project)
  Usuwa wszystkie projekty. Wywoływana w delete_solution().
- void delete_solution (solution **solution)
  Usuwa solucje wraz z jej projektami i plikami.
- int delete_project (solution *solution, char *name)
  Usuwa projekt o podanej nazwie wraz z plikami.
- void display_files (files *files)
  Wyświetla liste plików w konsoli. Wywoływana w display_projects().
- void display_projects (projects *pHead)
  Wyświetla liste projektów w konsoli wraz z jej plikami. Wywoływana w display_solution().
- void display_solution (solution *solution)
  Wyświetla solucje wraz z jej projektami i plikami.
- void saveFiles (files *file, FILE *fptr, int numberOfFiles)
  Zapisuje pliki projektu w pliku binarnym.

- void saveProjects (projects ∗project, FILE ∗fptr, int numberOfProjects)

  Zapisuje projekty solucji w pliku binarnym wraz z plikami projektu.
- int saveSolution (solution ∗solution)

  Zapisuje solucje w pliku binarnym wraz z jej projektami i plikami.
- void loadFiles (solution ∗solutionStructure, char ∗projectName, FILE ∗fptr, int numberOfFiles)

  Wczytuje pliki projektu z pliku binarnego i tworzy ich strukture w programie.
- void loadProjects (solution ∗solutionStrucutre, int numberOfProjects, FILE ∗fptr)

  Wczytuje projekty solucji z pliku binarnego wraz z plikami projektu i tworzy ich strukture w programie.
- solution ∗ loadSolution (char ∗savedSolutionName)

  Wczytuje solucje z pliku binarnego wraz z jej projektami i plikami oraz tworzy ich strukture w pamięci programu.
- void createFiles (projects ∗project)

  Tworzy pliki w folderze projektu.
- void createProjectsDirectories (projects ∗project)

  Tworzy foldery projektów w folderze solucji.
- void createSolutionDirectories (solution ∗solutionStructure, char ∗destination)

  Tworzy folder solucji wraz z jej projektami i plikami.
- void loadFilesAlreadyCreated (solution ∗solutionStructure, char ∗projectName, char ∗destination)

  Funkcja szuka plików w podanej lokacji i tworzy liste tych plików w projekcie.
- int loadSolutionAlreadyCreated (solution ∗solutionStructure, char ∗destination)

  Funkcja szuka folderów w podanej lokacji i tworzy liste projektów na podstawie znalezionych folderów.

4.1.1 Dokumentacja funkcji

4.1.1.1 createFiles() `void createFiles ( projects * project )`

Tworzy pliki w folderze projektu.

Parametry

project	wskaźnik na projekt w którym tworzone są pliki

4.1.1.2 createProjectsDirectories() `void createProjectsDirectories ( projects * project )`

Tworzy foldery projektów w folderze solucji.

Parametry

project	wskaźnik na liste projektów do stworzenia

4.1.1.3 createSolutionDirectories()

```
void createSolutionDirectories (
        solution * solutionStructure,
        char * destination )
```

Tworzy folder solucji wraz z jej projektami i plikami.

Parametry

solutionStructure	wskaźnik na solucje
destination	ścieżka na której zostanie stworzona solucja

4.1.1.4 delete_all_files()

```
void delete_all_files (
        files ** file )
```

Usuwa wszystkie pliki. Wywoływana w delete_all_projects().

Parametry

file	wskaźnik na wskaźnik listy plików

4.1.1.5 delete_all_projects()

```
void delete_all_projects (
        projects ** project )
```

Usuwa wszystkie projekty. Wywoływana w delete_solution().

Parametry

project	wskaźnik na wskaźnik listy projektów

4.1.1.6 delete_file()

```
int delete_file (
        solution * solution,
        char * project_name,
        char * file_name )
```

Funkcja usuwająca plik.

Parametry

solution	wskaźnik na solucje z której będzie usuwany plik
project_name	nazwa projektu z którego zostanie usunięty plik
file_name	nazwa usuwanego pliku

Zwraca

int - numer błędu. 0 - brak błędu 1 - nie znaleziono projektu 2 - nie znaleziono pliku

4.1.1.7 delete_project() `int delete_project (`

```
solution * solution,
char * name )
```

Usuwa projekt o podanej nazwie wraz z plikami.

Parametry

solution	wskaźnik na solucje z której zostanie usunięty projekt
name	nazwa usuwanego pliku

Zwraca

int - numer błędu. 0 - brak błędu 1 - nie znaleziono projektu

4.1.1.8 delete_solution() `void delete_solution (`

```
solution ** solution )
```

Usuwa solucje wraz z jej projektami i plikami.

Parametry

solution	wskaźnik na wskaźnik solucji

4.1.1.9 display_files() `void display_files (`

```
files * files )
```

Wyświetla liste plików w konsoli. Wywoływana w display_projects().

Parametry

files	wskaźnik na liste plików do wyświetlenia

4.1.1.10 display_projects() `void display_projects (`

```
projects * pHead )
```

Wyświetla liste projektów w konsoli wraz z jej plikami. Wywoływana w display_solution().

Parametry

pHead	wskaźnik na liste projektów do wyświetlenia

4.1.1.11 display_solution()

```
void display_solution (
        solution * solution )
```

Wyświetla solucje wraz z jej projektami i plikami.

Parametry

solution	wskaźnik na solucje do wyświetlenia

4.1.1.12 loadFiles()

```
void loadFiles (
        solution * solutionStructure,
        char * projectName,
        FILE * fptr,
        int numberOfFiles )
```

Wczytuje pliki projektu z pliku binarnego i tworzy ich strukture w programie.

Parametry

solutionStructure	wskaźnik na solucje do której zostaną wczytane pliki
projectName	wskaźnik projektu do którego zostaną wczytane pliki
fptr	wskaźnik na zmienną plikową
numberOfFiles	liczba plików w projekcie

4.1.1.13 loadFilesAlreadyCreated()

```
void loadFilesAlreadyCreated (
        solution * solutionStructure,
        char * projectName,
        char * destination )
```

Funkcja szuka plików w podanej lokacji i tworzy liste tych plików w projekcie.

Parametry

solutionStructure	wskaźnik na solucje na której stworzony zostanie projekt z plikami
projectName	wskaźnik na projekt do którego zostaną stworzone pliki
destination	ścieżka w której pliki będą wyszukiwane

4.1.1.14 loadProjects()

```
void loadProjects (
        solution * solutionStrucutre,
        int numberOfProjects,
        FILE * fptr )
```

Wczytuje projekty solucji z pliku binarnego wraz z plikami projektu i tworzy ich strukture w programie.

Parametry

solutionStrucutre	wskaźnik na solucje do której zostaną wczytane projekty
numberOfProjects	liczba projektów w solucji
fptr	wskaźnik na zmienną plikową

4.1.1.15 loadSolution()

```
solution* loadSolution (
        char * savedSolutionName )
```

Wczytuje solucje z pliku binarnego wraz z jej projektami i plikami oraz tworzy ich strukture w pamięci programu.

Parametry

savedSolutionName	nazwa solucji do wczytania

Zwraca

solution∗ wskaźnik na nowo stworzoną solucje

4.1.1.16 loadSolutionAlreadyCreated()

```
int loadSolutionAlreadyCreated (
        solution * solutionStructure,
        char * destination )
```

Funkcja szuka folderów w podanej lokacji i tworzy liste projektów na podstawie znalezionych folderów.

Parametry

solutionStructure	wskaźnik na solucje w której zostaną stworzone projekty
destination	ścieżka w której foldery będą wyszukiwane

Zwraca

int - numer błędu. 0 - brak błędu 1 - nie znaleziono ścieżki solucji

4.1.1.17 new_file()

```
int new_file (
        solution * solution,
        char * project_name,
        char * file_name )
```

Funkcja tworząca nowy plik.

Parametry

solution	wskaźnik na solucje do której będzie należał plik
project_name	nazwa projektu w którym zostanie stworzony plik
file_name	nazwa nowego pliku

Zwraca

int - numer błędu. 0 - brak błędu 1 - nie znaleziono projektu

4.1.1.18 new_project()

```
void new_project (
        solution * solution,
        char * name )
```

Funkcja tworząca nowy projekt.

Parametry

solution	wskaźnik na solucje do której będzie należał projekt
name	nazwa nowego projektu

4.1.1.19 new_solution()

```
solution* new_solution (
        char * solutionname )
```

Funkcja tworząca nową solucje.

Parametry

solutionname	nazwa nowej solucji

Zwraca

solution∗ wskaźnik na nowo stworzoną solucje

4.1.1.20 saveFiles() `void saveFiles (`

```
files * file,
FILE * fptr,
int numberOfFiles )
```

Zapisuje pliki projektu w pliku binarnym.

Parametry

file	wskaźnik na pierwszy plik projektu
fptr	wskaźnik na zmienną plikową
numberOfFiles	liczba plików w projekcie

4.1.1.21 saveProjects() `void saveProjects (`

```
projects * project,
FILE * fptr,
int numberOfProjects )
```

Zapisuje projekty solucji w pliku binarnym wraz z plikami projektu.

Parametry

project	wskaźnik na pierwszy projekt solucji
fptr	wskaźnik na zmienną plikową
numberOfProjects	liczba projektów w solucji

4.1.1.22 saveSolution() `int saveSolution (`

```
solution * solution )
```

Zapisuje solucje w pliku binarnym wraz z jej projektami i plikami.

Parametry

solution	wskaźnik na solucje

Zwraca

int - numer błędu. 0 - brak błędu 1 - nie udało się zapisać pliku binarnego

4.2 Dokumentacja pliku structs.h

Komponenty

- struct files_
- struct projects_
- struct solution_

Definicje typów

- typedef struct files_ files
- typedef struct projects_ projects
- typedef struct solution_ solution

4.2.1 Dokumentacja definicji typów

4.2.1.1 files `typedef struct files_ files`

Struktura reprezentująca pliki w projekcie

4.2.1.2 projects `typedef struct projects_ projects`

Struktura reprezentująca projekty w solucji

4.2.1.3 solution `typedef struct solution_ solution`

Struktura reprezentująca solucje

Indeks