Technikum Łączności

im. Obrońców Poczty Polskiej

w Gdańsku

PRACA DYPLOMOWA

nr: 1998/1999/VD/7

Temat:

Baza danych dla biblioteki w języku C++.

konsultant:	autor:
mgr inż. Dorota Blicharz	**Jacek Marcin Jaworski**

1 Spis treści

Spis treści

2 Wstęp

Komputerowe bazy danych są niezbędnym narzędziem do sprawnego funkcjonowania każdej współczesnej instytucji. Począwszy od niewielkich hurtowni, a kończąc na międzynarodowych koncernach i urzędach państwowych, nikt nie może się bez nich obejść.

Bazy danych umożliwiają wydobycie potrzebnych informacji szybciej, niż pracownik instytucji zdąży wstać z krzesła by udać się do archiwum na ich poszukiwanie. Dziś czas ma większą wartość niż kiedykolwiek, więc jego oszczędność jest niezwykle opłacalna. Jednak czas ogranicza również samych twórców baz danych. Dlatego ważne jest poznanie odpowiednich narzędzi umożliwiających wydajne ich tworzenie. Współczesne programy umożliwiają naprawdę szybkie robienie baz danych. Nie znaczy to jednak, że jest to zadanie łatwe. Dzięki dostępnym dziś narzędziom proces powstawania bazy danych można skrócić z kilku miesięcy do kilku tygodni lub pracę zespołu może wykonać jeden człowiek w tej samej jednostce czasu. Dzięki temu mogłem zaryzykować i podjąć próbę stworzenia bazy danych do obsługi biblioteki. Według pierwotnych założeń moja baza miała nie ustępować produktom komercyjnym pod względem funkcjonalnym. Swoją bazę danych (i projekt) nazwałem „Bibliotekarz”. W opisie starałem się ograniczyć do minimum ilość terminów związanych z bazami danych i systemem C++ Builder żeby ułatwić i tak trudną lekturę.

3 Założenia projektu Bibliotekarz

3.1 Przez realizację projektu chciałem poznać

- Język wysokiego poziomu C++;
- Zasady i problemy związane z tworzeniem aplikacji w C++ pod Windows 95;
- Zagadnienia z zakresu tworzenia baz danych;
- Technologię RAD na przykładzie systemu C++ Builder (jego zalety i wady);
- Możliwości i zastosowanie strukturalnego języka zapytań SQL.

3.2 Program Bibliotekarz miał umożliwiać

- Obsługę biblioteki miejskiej lub szkolnej;
- Katalogowanie książek biblioteki;
- Katalogowanie czytelników biblioteki;
- Grupowanie czytelników w klasy;
- Ewidencjonowanie wypożyczania (zwracania) książek z (do) biblioteki;
- Wybiórcze udostępnianie książek czytelnikom;
- Określanie górnych limitów wypożyczeń dla czytelników;
- Tematyczne wyszukiwanie książek wg haseł;
- Edycję wcześniej wprowadzonych danych;
- Tworzenie i udostępnianie danych o wypożyczeniach;
- Sporządzanie raportów.

3.3 Bibliotekarz miał być prototypem programu

jak najbardziej zbliżonym do produktów komercyjnych jednak sam nigdy nie miał się stać programem komercyjnym. Wynika to z badawczego charakteru całego projektu.

4 Podstawy teoretyczne projektu Bibliotekarz

4.1 Ogólnie o C++

C++ jest językiem programowania wysokiego poziomu. Powstał w drodze ewolucji języka C. C++ (w porównaniu do C) został , przede wszystkim, wzbogacony o obsługę klas, dzięki czemu stał się językiem obiektowym.

Wcześniej pisano programy linia pod linią od początku do końca, mniej więcej tak jak się potem wykonywały (programowanie liniowe). Później zaczęto wyłączać z ciała programu **funkcje** (programowanie proceduralne). Następnie zaczęto łączyć **dane** (różnych typów) w bloki tworzące - struktury. W końcu połączono struktury i funkcje w logiczną całość i powstały **klasy**.

4.2 Klasy w C++

Pisząc program programista zazwyczaj chce odzwierciedlić jakieś zjawiska zachodzące w rzeczywistości. W C++ może to zrobić lepiej, dzięki temu, że można umieścić w jednej paczce (klasie) dane które opisują fizyczne parametry zjawiska i funkcje opisujące jego przebieg, np.:

samochód ma takie dane jak: pojemność zbiornika paliwa, temperatura oleju, prędkość, (...),

jego funkcje to: włączenie silnika, wycieraczek, ogrzewanie wnętrza, (...).

Klasy nie odzwierciedlają żadnego konkretnego obiektu, podobnie jak słowa: ptak, samochód, które mówią jedynie o rodzaju danego przedmiotu; nie znamy jego wagi, koloru czy nazwy (nie ma przecież zwierzęcia „ptak”). Dopiero obiekty klas zawierają dane na temat np. kruka czy Fiata 126p.

Po wcześniejszym zadeklarowaniu klasy można tworzyć jej obiekty tak jak zwykłe zmienne typów wbudowanych (int - liczba całkowita, char - znak, float - liczba rzeczywista, ...). Klasy dają możliwość ukrycia wybranych danych lub funkcji wtedy nie można się do nich odwoływać z poza klasy. Oczywiście nie wszystkie funkcje w klasie mogą być ukryte.

Istniejące klasy można modyfikować bez naruszenia ich kodu! Umożliwia to mechanizm **dziedziczenia**. Po prostu tworzymy nową klasę, która dziedziczy zawartość innej. Do tej nowej klasy możemy dodawać nowe dane i funkcje. Można dodawać dane i funkcje o tych samych nazwach co już istniejące (w klasie podstawowej), jednak wtedy stare będą przesłonięte przez nowe i nie będzie można się do nich odwoływać w zwykły sposób (ale w ogóle jest to możliwe). Dziedziczenie może być **wielopokoleniowe** - dzięki temu producenci oprogramowania oszczędzają mnóstwo pieniędzy. Dziedziczenie wielopokoleniowe jest jednym z podstawowych filarów systemu C++ Builder.

4.3 Wskaźniki

Są bardzo ważnym elementem C++ ponieważ umożliwiają tworzenie obiektów (i zmiennych) w czasie pracy programu - do tego celu wykorzystuje się zapas pamięci operacyjnej. Wskaźnik zawiera adres obiektu w pamięci i informacje o typie obiektu na który wskazuje (prawdopodobnie tylko jego wielkość). Wskaźniki są kolejnym filarem C++ Buildera.

4.4 Zastosowanie C++

Ze względu na swoje ogromne możliwości C++ jest obecnie podstawowym językiem programowania. Występuje na wszystkich platformach sprzętowych. Czysty kod w C++ nie jest związany z konkretną platformą sprzętową (tak jak np. asembler) co umożliwia dość łatwą (czyli tanią) implementację programów na inne platformy sprzętowe (i programowe – jest wiele systemów operacyjnych). Inną zaletą tego języka jest to, że człowiek uczący się C++ może poznawać go stopniowo i w zależności od potrzeb opanowywać kolejne jego możliwości. W procesie tłumaczenia kodu programu na język procesora następuje drobiazgowa analiza kodu programu w trakcie której zostają wyłapane wszystkie błędy składniowe popełnione przez programującego. W miejscach potencjalnych błędów logicznych generowane są ostrzeżenia. Są to dwie bardzo pomocne cechy języków wysokiego poziomu oszczędzające czas. C++ generuje szybkie programy w przeciwieństwie do wielu innych języków wysokiego poziomu. Jednak spadek prędkości w porównaniu do asemblera jest bardzo duży. W C++ powstają wszystkie możliwe programy od systemów operacyjnych po gry.

Z powyższych powodów znajomość języka C++ uważam za bardzo cenną.

4.5 Ogólnie o bazach danych

Komputerowe bazy danych służą do przechowywania dużych zbiorów danych. Ze względu na sposób zarządzania dzielimy je na analityczne i operacyjne.

Analityczne bazy danych służą do przechowywania danych historycznych, które wykorzystuje się np. jako źródło danych o tendencjach rynkowych. Dane w takiej bazie nie podlegają zmianom.

Operacyjne bazy danych zawierają ciągle aktualizowane dane odzwierciedlające np. obecną zawartość magazynu hurtowni. Bibliotekarz zalicza się do tej grupy baz danych ponieważ zawiera aktualne informacje na temat książek i czytelników w bibliotece, jednak przechowuje również dane historyczne o wypożyczeniach.

4.6 Tablice

Każda baza danych składa się z tablic. Każda tablica jest dwuwymiarowa - posiada wiersze i kolumny. Wiersze nazywa się **rekordami** a kolumny **polami**. W polach przechowuje się np. tytuły książek, numery inwentarzowe, itp., a w rekordach dane na temat jednego elementu tablicy np. jednej książki. Jeżeli przynajmniej w dwóch tablicach część pól się powtarza to mówimy, że baza danych zawiera **nadmiarowe dane** - idealna baza danych nie zawiera ich wcale. Fizycznie tablice mogą być przechowywane na dysku twardym w postaci jednego lub wielu plików.

Przykładowa tablica książek może wyglądać tak:

Tytuł	Nr inwentarzowy	Sygnatura	...
Matematyka	325454	836254	
Biały Kieł	569207	234884	
Sejm Czteroletni	837545	145782	
...			

Objaśnienie:

Tytuł - pole przechowujące tytuły książek,

Matematyka - rekord przechowujący dane książki pod tytułem „Matematyka”,

[...] - pozostałe pola i rekordy.

4.7 Relacyjny model bazy danych

Przez wiele lat starano się wypracować optymalny model bazy danych. Optymalny czyli taki który nie zawierał by nadmiarowych danych, byłby uniwersalny i jednocześnie wystarczająco szybki by mógł obsługiwać duże zbiory danych. Wczesne bazy zawierały nadmiarowe informacje a realizacja bardziej wyrafinowanych powiązań między tablicami była skomplikowana. Jednak te bazy były dość szybkie co miało niebagatelne znaczenie.

W 1970 roku matematyk dr E. F. Codd opublikował książkę w której zaprezentował założenia relacyjnego modelu baz danych. Założenia te były następujące:

1. Każda tablica składa się z rekordów oraz pól;
2. Fizyczna kolejność pól i rekordów w tablicy jest bez znaczenia;
3. Każdy rekord jest wyróżniony przez jedno pole zawierające unikatową wartość.

Dzięki takiemu rozwiązaniu użytkownik nie musiał już znać fizycznego położenia wiersza do którego chce się odwołać - nastąpiła fizyczna i logiczna niezależność bazy danych - zmiany w projekcie logicznym nie pociągały za sobą zmian w fizycznej implementacji tablic (i na odwrót). W istotny sposób został uproszczony dostęp do danych.

W punkcie 3. jest mowa o polu wyróżniającym rekord - nazywamy je **kluczem podstawowym**. Może to być w zasadzie dowolne pole z zastrzeżeniem, że zawartość tego pola będzie w każdym rekordzie inna. Skutecznym rozwiązaniem jest dodanie specjalnego pola liczbowego gwarantującego niepowtarzalność. W każdym nowym rekordzie przyjmuje ono wartość o jeden większą od dotychczas największej. Maksymalna możliwa wartość tego pola jest równocześnie maksymalną liczbą rekordów w tablicy. Klucze jednych tablic przechowywane w innych tablicach nazywa-

my **kluczami obcymi** i nie tworzą one nadmiarowych danych gdyż są niezbędnym elementem umożliwiającym realizację powiązań między tablicami (np. między tablicami książek i autorów). Pola które są kluczami dla wygody można wyróżnić przedrostkiem ID.

Przykładowa tablica książek w relacyjnym modelu bazy danych może wyglądać tak:

IDKsiążki	IDAutora	Tytuł	Nr inwentarzowy	Sygnatura	...
1	5	Matematyka	325454	836254	
2	7	Biały Kieł	569207	234884	
3	3	Sejm Czteroletni	837545	145782	
...					

Objaśnienie:

IDKsiążki - pole klucza podstawowego,

IDAutora - pole klucza obcego.

W podanym przykładzie aby dowiedzieć się kto jest autorem książki (np. „Matematyka") należy pobrać, z rekordu tej książki, wartość pola klucza obcego IDAutora (tu: 5) i odnaleźć w tablicy autorów rekord w którym pole klucza podstawowego IDAutora ma taką wartość (5).W znalezionym rekordzie będą pola z imieniem i nazwiskiem szukanego autora.

4.8 Typy relacji

Powiązania między tablicami nazywamy relacjami zarządzanie nimi jest jedną z podstawowych funkcji bazy danych. W relacyjnej bazie danych, między tablicami, występują trzy typy relacji:

- jeden-do-jednego: zachodzi wtedy gdy jeden rekord z tablicy A jest powiązany z dokładnie jednym rekordem z tablicy B;
- jeden-do-wielu: jeden rekord z tablicy A jest powiązany z kilkoma rekordami z tablicy B;
- wiele-do-wielu: jeden rekord z tablicy A jest powiązany z kilkoma rekordami z tablicy B i na odwrót. Ten typ relacji realizuje się wykorzystując tablicę pośrednią i podwójna relację jeden-do-wielu. Tablica pośrednia zawiera oprócz własnego pola klucza podstawowego dwa pola - klucze obce, które są kluczami podstawowymi tablic A i B. Wtedy można jednemu rekordowi z tablicy A przypisać dowolna liczbę rekordów z tablicy B i na odwrót.

Przykład tablicy pośredniej (książki-autorzy) między tablicami książek i autorów (jeden autor mógł napisać kilka książek i książka mogła mieć kilku autorów):

IDKsiążkiAutorzy	IDKsiążki	IDAutora	...
1	1	5	
2	1	4	
3	13	5	
...			

Objaśnienie:

IDKsiążkiAutorzy - pole klucza podstawowego,

IDKsiążki, IDAutorzy - pola kluczy obcych.

W tym przypadku rolę tablic A i B pełnią tablice książek i autorów. Aby poznać autorów danej książki należy pobrać z tablicy książek wartość jej pola klucza podstawowego [IDksiążki] i znaleźć w tablicy pośredniej książki-autorzy wszystkie rekordy których pola [IDksiążki] są równe pobranej wartości. Wtedy w znalezionych rekordach pola [IDAutora] będą zawierać „numery autorów". Następnie w tablicy autorów w polu klucza [IDAutora] odszukujemy kolejno rekor-

dy o identycznej wartości co znalezione „numery autorów” w tablicy pośredniej. W ten sposób uzyskamy zbiór autorów książki.

W relacyjnych bazach danych fizycznym aspektem działania bazy zajmuje się **motor bazy danych**. To on zajmuje się obsługą plików w których znajdują się dane - jest to znaczne ułatwienie gdyż formatów takich plików jest bardzo wiele (i co nie mniej ważne na poszczególnych systemach operacyjnych mogą zachodzić w ich obsłudze istotne różnice). Relacje można zrealizować tradycyjnymi metodami programistycznymi pisząc w C++, jednak jest to czasochłonne i nieperspektywiczne znacznie lepszym rozwiązaniem jest zastosowanie strukturalnego języka zapytań - **SQL**.

4.9 SQL

SQL podobnie jak C++ jest standardem przemysłowym. Powstał w celu standaryzacji obsługi baz danych - głównie pod kątem rozległych systemów sieciowych opartych na architekturze klient - serwer (dane są zgromadzone na jednym komputerze i w razie potrzeby udostępniane innym). SQL jest nie zależny od platformy sprzętowej. Polecenia w SQL są nazywane zapytaniami i mają postać zwykłych znaków ASCII - dzięki czemu nie są związane z żadnym systemem operacyjnym i mogą być przesyłane dowolną drogą między komputerami. Aby SQL w ogóle zadziałał potrzebny jest motor bazy danych który zdekoduje zapytanie i za pomocą własnych narzędzi je wykona. Wynikiem zapytania jest tablica danych. Jest to chwilowa tablica utworzona wg kryteriów podanych w zapytaniu, czyli nie odzwierciedla żadnej fizycznej tablicy. Z takiej tablicy można korzystać tak długo jak jest to konieczne. Przykładowe zapytanie wygląda tak:

```
Select * from Autorzy Where Autorzy.Imie = Ryszard Order By Autorzy.Nazwisko ASC
```

Objaśnienie:

Select * - wybierz całe rekordy;

from Autorzy - z tabeli Autorzy;

Where Autorzy.Imie = Ryszard - w których pole Imie ma wartość Ryszard;

Order By Autorzy.Nazwisko ASC - ustaw: alfabetycznie, wg pola Nazwisko, rosnąco.

Czyli chodzi nam o wszystkich autorów o imieniu Ryszard uporządkowanych wg nazwisk alfabetycznie i rosnąco.

W praktyce zapytania są o wiele bardziej złożone. Rzadko wybiera się rekordy wraz z wszystkimi polami (Select *). Często wybiera się po kilka pól z kilku tablic. Klauzula Where zawiera zwykle cały szereg warunków. Select jest podstawowym poleceniem, jednak nie jedynym. Takich poleceń jest kilkadziesiąt i umożliwiają one przeprowadzanie wszystkich operacji na bazie danych od dodawania (usuwania) rekordów i tablic po realizację relacji.

Powyższe cechy sprawiają, że SQL jest bardzo potężnym narzędziem. Jednak ceną za ogromne możliwości SQL jest dramatyczny spadek prędkości - zdekodowanie zapytania i utworzenie tablicy wynikowej trwa bardzo długo co przy intensywnej pracy z SQL na dużych zbiorach danych może doprowadzić do uciążliwych przestojów. Dlatego serwery danych raczej nigdy nie będą za szybkie.

4.10 Windows 95

Windows 95 jest systemem operacyjnym. Cechą Windows jest graficzny interfejs użytkownika i jego standaryzacja. W DOSie programy działały głównie w trybie tekstowym, a jeśli wykorzystywały tryb graficzny to każdy program miał inny interfejs użytkownika. Windows rozwiązuje problemy związane z wyświetlaniem obrazu i działaniem urządzeń peryferyjnych. Przez co przy pomocy odpowiednich (wizualnych) narzędzi tworzenie programów staje się prostsze. W odróżnieniu do DOSa w którym program musiał śledzić stan klawiatury i myszy by odczytać intencje użytkownika, w Windows system operacyjny sam informuje program np. o naciśnięciu klawisza czy ruchu myszy. Robi to za pomocą tzw. **zdarzeń**.

4.11 Zdarzenia

Napływają do programu nie przerwanym ciągiem, to na jakie zdarzenia ma program reagować zależy od intencji programisty. Większość zdarzeń programy ignorują.

4.12 Ocena Windows 95

Windows 95 można uznać za system wygodny i łatwy w użyciu dla przeciętnego człowieka, nie trzeba się martwić o konfigurację nieznanych parametrów - wszystko dzieje się automatycznie. Jednak założona kompatybilność z

DOSem wraz z niską kulturą programowania MicroSoftu powoduje, że Windows 95 jest systemem wolnym i nie stabilnym o wygórowanych wymaganiach sprzętowych. Znacznie lepszym systemem operacyjnym jest Linux - odmiana Unixa. Jest to system bijący na głowę produkty MicroSoftu pod każdym względem: szybszy, mniejszy, super stabilny, bezpieczny, uniwersalny, występuje na wszystkich platformach sprzętowych i jest całkowicie darmowy. Jednak Linux jest trudniejszy w konfiguracji i obsłudze, mimo to bardzo chciałbym poznać dokładnie ten system.

4.13 Borland C++Builder

Wybrałem go gdyż korzystam od lat z innych programów tej firmy (Turbo Assembler 5.0, C++ 5.0). System Borland C++ Builder umożliwia tworzenie 32-bitowych programów w języku C++ pracujących pod kontrolą systemu operacyjnego Windows 95. Największą zaletą Buildera jest wizualny proces projektowania interfejsu użytkownika. Wizualny, czyli taki w którym kolejne elementy interfejsu wybieramy z dostępnej listy i umieszczamy na formularzu - oknie (oknach) przyszłego programu. Dzięki temu rozwiązaniu nie trzeba pisać interfejsu użytkownika i można się skupić na realizacji funkcji programu. Elementy interfejsu użytkownika dostępne na wspomnianej liście nazywane są komponentami.

4.14 Komponenty w C++ Builder

Dzielą się na widzialne i niewidzialne (podczas pracy programu). Komponenty widzialne wykorzystuje się przy tworzeniu interfejsu użytkownika. Komponenty niewidzialne dotyczą funkcjonowania aplikacji, powstały głównie w celu usprawnienia tworzenia baz danych w środowiskach sieciowych. Komponenty są po prostu klasami [patrz: III.2.], które opisują np. wygląd i zachowanie się okna, klawisza itp. Gdy wstawiamy komponent do formularza w tle działa mechanizm dziedziczenia w skutek czego nasz np. klawisz jest obiektem klasy pochodnej od tej widocznej na liście komponentów. Wszystkie użyte przez nas komponenty, w czasie pracy programu, tworzone są dynamicznie - za pomocą wskaźników [III.3.]. Bardzo spodobało mi się to rozwiązanie. Po wstawieniu komponentu programista ma dostępną listę podstawowych zdarzeń [III.11.], które dany komponent może obsłużyć. I gdy chcemy żeby program wykonał jakieś działanie (np. w momencie naciśnięcia klawisza myszy nad tym komponentem) wystarczy ciele funkcji wywoływanej do obsługi tego zdarzenia wpisać kod który to działanie wykona.

4.15 Bazy danych w C++ Builder

System C++ Builder jest silnym narzędziem do tworzenia baz danych. Wynika to z dużej ilości komponentów bazodanowych (zarówno widzialnych jak i niewidzialnych). Za ich pomocą można stworzyć rozbudowane systemy sieciowe działające w oparciu o dowolny protokół transmisji. Obsługą bazy danych w Builderze zajmuje się motor bazy danych BDE. Na liście komponentów dostępne są dwa komponenty (niewidzialne) służące do obsługi tablic są to: Table i **Query**. **Table** służy do działań na rzeczywistej tablicy. Query służy do obsługi zapytań SQL [III.9.] - wynik zapytania zwraca tablicę do której mamy dostęp. Obsługa tej chwilowej tablicy jest (niestety tylko) prawie taka sama jak przez Table. Query zapewnia dostęp do chwilowej tablicy tak długo, aż nie zostanie wykonane następne zapytanie lub program nie zakończy pracy. Table i Query mogą być źródłem danych dla komponentów widzialnych.

4.16 C++ Builder a Delphi

Mówiąc o Builderze nie sposób pominąć systemu Delphi. Delphi jest obiektową wersją języka Pascal. Wszystkie komponenty dostępne na liście komponentów w Builderze pochodzą z Delphi. Firma Borland chcąc zaoszczędzić pieniądze (i czas) zdecydowała się na rozszerzenie języka C++ tak aby możliwe stało się wykorzystanie komponentów z Delphi. Zmiany dotyczą głównie deklaracji i definicji klas opisujących komponenty. Oczywiście nie trzeba korzystać z komponentów (z Delphi) - można wszystko napisać w C++, ale wtedy rezygnujemy z technologii RAD (błyskawiczne projektowanie aplikacji).

4.17 Ocena systemu C++ Builder

Borland C++ Builder jest wydajnym narzędziem umożliwiającym szybkie tworzenie aplikacji dla Windows (technologia RAD). Programista może się skupić na realizacji problemu zamiast „walczyć” z interfejsem użytkownika przyszłej aplikacji.

Jednak nie ma róży bez kolców. Najpoważniejszą wadą Buildera jest zastosowanie bibliotek (komponentów) z Delphi motywowane jedynie względami oszczędnościowymi a nie technicznymi. Panuje powszechne przekonanie, że biblioteki Delphi spowalniają tworzony program, zwiększają jego objętość i zapotrzebowanie na pamięć. Ponadto w ciągu kilku miesięcy intensywnej pracy systemem C++ Builder ujawniły się liczne błędy w jego działaniu.

Najbardziej irytującym błędem było wyświetlanie komunikatu o zmianie daty pliku za każdym razem gdy zapisałem wprowadzone w nim zmiany. Występowało to zawsze w momencie powrotu do systemu C++ Builder po chwilowym jego opuszczeniu.

Innym niemniej uciążliwym błędem było „niewidzenie" nowego komponentu zaraz po jego wstawieniu do formularza (i użyciu opcji Zapisz). Jedynym skutecznym rozwiązaniem tego problemu było zamknięcie systemu Builder i ponowne jego uruchomienie.

Oba omówione błędy nie występowały od początku, ich występowanie nasilało się w trakcie realizacji projektu. Tych błędów nie dało się wyeliminować za pomocą opcji, nie pomogło również przejście na nowszą wersję systemu C++ Builder.

Kłopotów dostarczały również same komponenty. Po prostu część z nich nie działała, choć w świetle dostępnych materiałów działać powinna. Interesujące było to, że niektóre komponenty „nie chciały" wyświetlać pola jednej tablicy, a pole innej wyświetlały bez problemów (przy zachowaniu identycznych warunków). Przyczyny tych błędów mogły być dwie: błąd w komponencie lub braki w dostępnej dokumentacji Buildera (b. prawdopodobne). Pewnym problemem okazało się przejście z C++ Builder w wersji 1.0 na wersję 3.0. Ponieważ zmieniono format pliku projektu (opisuje on projekt każdej tworzonej aplikacji) w momencie jego otwarcia Builder 3.0 próbował automatycznie przekształcić plik do nowszej wersji. Ta operacja przy każdej próbie kończyła się niepowodzeniem. Z problemem poradziłem sobie w ten sposób, że w C++ B. 3.0 utworzyłem nowy projekt aplikacji, następnie włączyłem do niej wszystkie formularze (okna) z wcześniejszego projektu (w C++ B. 1.0).

5 Realizacja projektu Bibliotekarz

5.1 Przygotowania

Kilka miesięcy przed rozpoczęciem roku szkolnego 1998/99 rozpocząłem gromadzić i czytać książki o języku C++. Po poznaniu asemblera (język programowania niskiego poziomu) chciałem opanować C++. Głównie z powodu jego olbrzymiej popularności i równie wielkich możliwości. W dalszej perspektywie miałem zamiar poznać bazy danych gdyż gromadzenie, przechowywanie, przetwarzanie i udostępnianie danych jest jednym z podstawowych zadań systemów komputerowych. Poszukując tematu pracy dyplomowej chciałem aby był on związany z moimi zainteresowaniami. Gdy tylko „Baza danych dla biblioteki w języku C++." jako temat pracy dyplomowej został zaakceptowany nabyłem książki na temat C++ Buildera i baz danych.

Po zapoznaniu się z zasadami projektowania bazy danych, stosując się do wytycznych, udałem się „w teren". Przeprowadziłem serię rozmów z pracownikami bibliotek: szkolnej i miejskiej - chciałem uzyskać informacje o sposobie ich działania i ew. różnicach w funkcjonowaniu.

Zebrawszy wszystkie niezbędne informacje zacząłem projektować (na kartce) logiczny rdzeń bazy danych: tablice i relacje między nimi. Po zakończeniu tego etapu mogłem przystąpić do właściwego tworzenia bazy danych.

5.2 Implementacja

W trakcie implementacji programu zaszły poważne zmiany pierwotnych założeń wynikające z napotykania nieprzewidzianych trudności oraz poznawania i opanowywania kolejnych narzędzi.

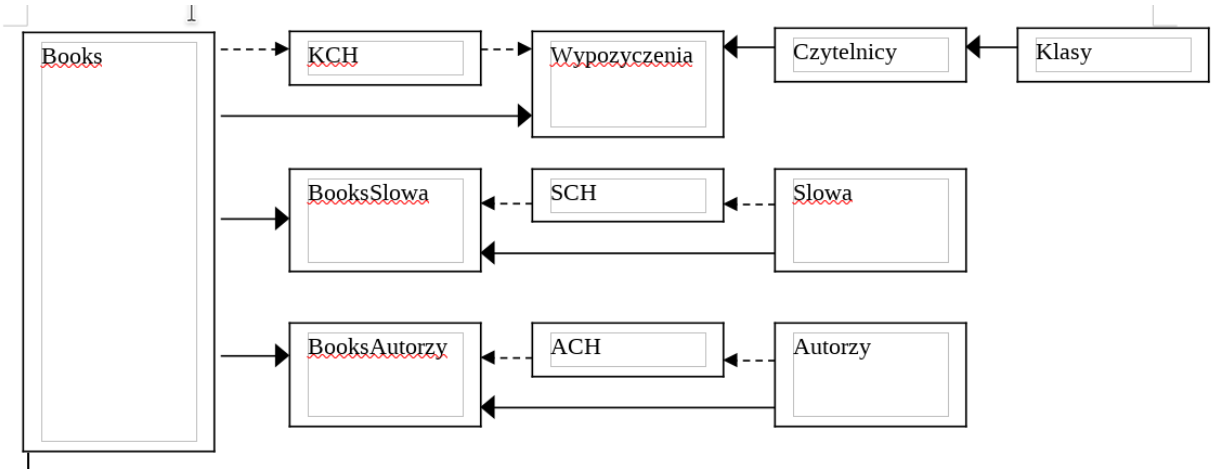
5.2.1 Tablice

Pierwszy etap implementacji polegał na utworzeniu tablic. Do tego celu użyłem programu Database Desktop wchodzącego w skład systemu C++ Builder. Każda tablica przechowywana jest w oddzielnym pliku, w formacie Paradox 7.0. Po zainstalowaniu programu Bibliotekarz pliki te znajdują się w katalogu Data. Jak nietrudno zauważyć w skład niektórych tablic wchodzi więcej niż jeden plik (kilka plików o tej samej nazwie z różnymi rozszerzeniami). W takich przypadkach właściwa tablica znajduje się w pliku z rozszerzeniem DB, natomiast w pozostałych plikach znajdują się dane pomocnicze (np. umożliwiające przeszukiwanie tablicy wg pól innych niż pole klucza - tzw. indeksy).

Pierwotnie do obsługi tablic planowałem korzystać wyłącznie z komponentu Table. Miał on być źródłem wyświetlanych danych, z jego pomocą chciałem modyfikować tablice. Pierwszym problemem okazało się wyświetlanie

danych pola w kolejności alfabetycznej. Zostając przy komponencie Table musiałbym napisać funkcję sortującą. Nie jest to problem jednak w ten sposób stracił bym mnóstwo czasu i nie zapoznałbym się z możliwościami systemu Builder, który musiał oferować jakieś automatyczne narzędzie do sortowania. Wtedy zacząłem poważnie myśleć o komponencie Query i SQLu. Po kilku eksperymentach podjąłem decyzję o zakupie kolejnej książki - „SQL". Natychmiast w jednej z książek o Builderze wykryłem poważny błąd w konstrukcji zapytania (które oczywiście nie działało). Błędy w zapytaniach są dość irytujące i (b.) trudne do wykrycia ponieważ zapytanie w momencie wywołania jest wykonywane albo nie - w przypadku błędu nie są generowane żadne komunikaty, które mogły by wskazać na jego źródło. Pomimo tej wady SQL umożliwił wygodne wyświetlanie zawartości tablic.

Wszystkie relacje w programie są typu jeden-do-wielu (wiele-do-wielu to dwukrotne użycie jeden-do-wielu). Po licznych przeróbkach logiczny model bazy danych Bibliotekarz przyjął następujący kształt (stosuję oryginalne, robocze nazwy tablic ich opis jest poniżej):



Objaśnienie:

- - - - → - oznacza możliwe drogi przepływu danych (kluczy – III.7.) między tablicami, przy tworzeniu relacji wiele-do-wielu z wykorzystaniem tablic dodatkowych (np. KCH),

→ relacja jeden do wielu, początek strzałki oznacza tablicę z której jeden rekord odpowiada kilu rekordom w tablicy wskazywanej grotem strzałki (np. jedna klasa ma wielu uczniów – czytelników, jednemu czytelnikowi odpowiada wiele wypożyczeń, itd.).

Nazwy tablic dla użytkownika programu nie mają znaczenia. Służą one jedynie twórcy bazy danych i programowi, który je wykorzystuje. Biorąc powyższe pod uwagę dobierałem nazwy tablic tak aby były one jasne, krótkie i możliwe do wykorzystania przez program (bez polskich znaków, spacji, itp.).

Opis tablic bazy danych Bibliotekarz:

1. **Tablice Podstawowe**

Books: Główna tablica programu, zawiera wszystkie dane o książkach (wymagane przez biblioteki) plus informacje o aktualnym stanie książki i o jej dostępności, bez informacji o autorach,

Autorzy:Imiona i nazwiska autorów,

Czytelnicy: Zawiera informacje wymagane przez biblioteki na temat czytelników, plus uwagi na temat każdego czytelnika i przysługujący mu limit książek;

Klasy: Nazwy klas i liczba uczniów należących do każdej z nich;

Slowa: Zbiór haseł (słów kluczowych) umożliwiających tematyczne przeszukiwanie księgozbioru.

2. **Tablice Pośrednie**: konieczne do realizacji relacji wiele-do-wielu (III.8.c):

BooksAutorzy: Między tablicami książek (Books) i autorów (Autorzy), plus informacja o tym czy dany autor książki jest jej pierwszym autorem. Jest to istotne przy wyświetlaniu księgozbioru;

Wypozyczenia: Między tablicami książek (Books) i czytelnikami biblioteki (Czytelnicy), plus data wypożyczenia i zwrotu oraz sposób zwrotu książki do biblioteki;

BooksSlowa: Między tablicami książek (Books) i słowami kluczowymi (Slowa) zawiera informacje której książce odpowiada jakie hasło.

3. **Tablice Dodatkowe**: przechowują dane wyłącznie w czasie pracy programu:

ACH: Przy wprowadzaniu nowej książki użytkownik ma możliwość wybrania jej autorów, dopóki ostatecznie nie potwierdzi chęci wprowadzenia nowej książki informacje o jej autorach będą przechowywane w tej tablicy (następnie są przepisywane do tablicy BooksAutorzy i kasowane, lub tylko kasowane);

SCH: Zawiera informacje dotyczące słów kluczowych z którymi ma być powiązana nowo wprowadzana książka, (tak jak Ach) jej zawartość jest przepisywana i kasowana dopiero po ostatecznym zaakceptowaniu nowo wprowadzonej książki;

KCH: Przy wypożyczeniach, zawiera listę książek przeznaczonych do wypożyczenia czytelnikowi, po realizacji wypożyczenia jej zawartość jest kasowana.

5.2.2 Relacje

W wielu przypadkach do wyświetlania tablic wystarczał komponent Table. Problem pojawiał się w momencie gdy chciałem wyświetlić tylko niektóre rekordy tablicy lub chciałem wyświetlić jednocześnie pola pochodzące z dwóch tablic (wynik relacji). Realizacja relacji tradycyjnymi metodami, dla poznającego dopiero C++, byłaby kłopotliwa i przede wszystkim (b.) czasochłonna. Ta droga również doprowadziła mnie do SQLa.

5.2.3 SQL

Gdy pierwsze moje zapytania zaczęły działać ukazał się przede mną wielki, nieznany świat SQLa. Jego potęga i prostota zdumiała mnie. Zanim po raz pierwszy musiałem zrealizować relację wiele-do-wielu część projektu już zrealizowałem bazując wyłącznie na komponencie Table. Jednak wkrótce wypracowałem następujący system: do wyświetlania danych stosowałem jako źródło danych komponent Query (SQL), a do modyfikowania tablic komponent Table. W sumie mogłem wszystko zrealizować za pomocą Query (SQL), ale wtedy miałem jeszcze problemy z uruchamianiem nowych zapytań SQL, poza tym teoretycznie Table działa szybciej.

Początkowo moje zapytania były najprostsze. Z czasem ich wielkość rosła, aż w końcowej fazie realizacji projektu zastosowałem zapytania dynamiczne - ich treść jest ustalana w czasie pracy programu przez użytkownika. Wszystko odbywa się w tle, a użytkownik widzi jedynie końcowy efekt. Jaki był to skok jakościowy niech zobrazuje porównanie sposobów wyświetlania listy książek w oknach „Księgozbiór" i „Wypożyczanie książek". Okno „Wypożyczanie książek" bazuje na zapytaniach statycznych w skutek czego do każdej zmiany kryteriów wyświetlania potrzebna jest nowe zapytanie i funkcja je wywołująca. W tym oknie są aż cztery takie funkcje. Natomiast w oknie „Księgozbiór" zapytanie jest tworzone dynamicznie, w fragmentach, a przed wywołaniem zapytania poszczególne jego części są łączone w całość. W ten sposób dając użytkownikowi możliwość ustalania parametrów wyświetlania w znacznie szerszym niż dotychczas zakresie (okno Konfiguracja) udało mi się ograniczyć liczbę funkcji do jednej.

Zastosowanie SQLa spowodowało, że część pól w tablicach pozostała nie wykorzystana lub została zastosowana do innych celów niż pierwotnie zakładałem.

Część zapytań wyłączyłem z ciała programu i umieściłem w plikach z rozszerzeniem SQL. Znajdują się one w tym samym katalogu co program Bibliotekarz po zainstalowaniu. Można obejrzeć ich zawartość dowolnym edytorem tekstu. W przypadku zapytań dynamicznych w plikach (QCShow.SQL, QDBShow.SQL, QDBShow1.SQL) znajdują się jedynie początkowe fragmenty zapytań - reszta jest dodawana w czasie pracy programu.

5.2.4 Zdarzenia

Programowanie w C++ Builderze (i w ogóle w Windows) sprowadza się głównie do reakcji na zachodzące w systemie zdarzenia. Początkowo w ciele funkcji reagującej na zdarzenie umieszczałem kod wszystkiego co miało się

wykonać w wypadku zajścia tego zdarzenia. Prowadziło to do powielania kodu w sytuacjach gdy jedną rzecz można zrobić na kilka sposobów lub efekty działania miały być podobne (np. pozycję z listy można wybrać na trzy sposoby). Właśnie również z tego powodu w oknie „Wypożyczenia książek" są 4 funkcje wyświetlające tabelkę z książkami. Stopniowo zacząłem dodawać własne funkcje do klas przedstawiających okna programu i umieszczać w nich kod, który mógł się znaleźć przynajmniej w dwóch funkcjach opisujących zdarzenia. W końcowej fazie realizacji projektu operowałem prawie wyłącznie na własnych funkcjach wywoływanych przez różne zdarzenia. To umożliwiło (m. in.) efektywne zastosowanie dynamicznych zapytań SQLa.

5.3 Problemy

Większą część problemów przedstawiłem oceniając C++ Buildera 4.17 i opisując przebieg implementacji 5.2. Teraz przedstawię jaki wpływ miały te problemy na ostateczny kształt bazy danych Bibliotekarz.

Kłopoty z komponentami objawiły się zubożeniem interfejsu użytkownika w porównaniu do pierwotnego projektu. Np. planowałem intensywnie wykorzystywać pole edycji z listą rozwijaną, niestety tylko w jednym przypadku ten komponent zadziałał - w oknie „Nowy czytelnik" po zaznaczeniu „Uczeń" można wybrać klasę do której on należy. Nie możność użycia tego komponentu ograniczyła ilość tablic - planowałem użyć tablice zawierające: nazwy działów, rodzaje książek, rodzaje dokumentów towarzyszących, sposób nabycia (zbycia) książki przez bibliotekę, sposób zwrotu książki przez czytelnika, itd. Zastosowanie tych tablic wraz z polem edycji z listą rozwijaną by ułatwiło i usprawniło obsługę programu. Podczas realizacji projektu mocno odczułem brak książki o najnowszej wersji systemu C++ Bulider. W programie zaowocowało to całkowitym brakiem raportów, które w pierwotnych założeniach miały być. Brak raportów jest spowodowany tym, że komponenty służące do ich tworzenia dostarczane są przez jakąś nie zależną firmę, która całkowicie zmieniła filozofię ich użycia w nowej wersji C++ Buildera. Ta firma dostarcza również opis swoich komponentów, jednak jest on całkowicie nie przydatny do niczego.

Podczas pisania programu wystąpił jeszcze jeden poważny problem. Dotyczył on komponentu Table.

Gdy dodaję do tablicy nowy rekord (np. nowego autora) muszę dla niego ustalić nową, unikalną wartość pola klucza podstawowego. Normalnie robiłem to w ten sposób, że pobierałem z dotychczas ostatniego rekordu tablicy wartość pola klucza podstawowego - wychodziłem z założenia, że ma on maksymalną wartość. Następnie zwiększałem pobraną wartość o 1 i wstawiałem ją do (pola klucza podstawowego) nowego rekordu. To działało... do czasu. W pewnym momencie (przy testowaniu nowego fragmentu programu) pojawił się błąd. Byłem zdziwiony, że wystąpił przy użyciu mechanizmu wielokrotnie sprawdzonego. Wcześniej spotkałem się z tym rodzajem błędu - pojawiał się zawsze, gdy wartość pola klucza podstawowego została powtórzona przynajmniej w dwóch rekordach. Przekopałem program, książki, dokumentację systemu Builder i nic. Zacząłem linia po linii porównywać dwa pozornie identyczne fragmenty kodu (jeden działał, a drugi nie) - wstawiające nowe rekordy do tablic. Dostrzegłem drobną różnicę. Przed wywołaniem feralnego fragmentu kodu dokonywałem sprawdzenia czy użytkownik nie próbował wprowadzić autora o nazwisku identycznym z już istniejącym. W tym celu musiałem przeszukać tablicę autorów wg pola nazwisko. Przed przeszukiwaniem należy zaznaczyć (w komponencie Table), że chcemy przeszukać tablicę wg jakiegoś innego pola niż klucz podstawowy. Jeżeli w następnej kolejności zażądam przejścia do ostatniego rekordu w tablicy to Table istotnie przejdzie do jakiegoś rekordu jednak nie ostatniego (jego pole klucza nie będzie zawierać wartości maksymalnej). Rozwiązałem ten problem odznaczając (w komponencie Table) przeszukiwanie wg innego pola niż klucz podstawowy i wszystko znów działało poprawnie.

Później zastanawiałem się nad tym zjawiskiem. Doszedłem do wniosku, że jest to poszukiwany przeze mnie wcześniej sposób na automatyczne wyświetlanie zawartości tablicy wg wybranego pola w kolejności alfabetycznej. Po zakończeniu projektu Bibliotekarz sprawdziłem tą hipotezę, okazało się, że miałem rację. W tej sytuacji fizycznie ostatni rekord mógł się znajdować logicznie gdzieś w środku tablicy. Okazuje się, że komponent table jest czymś więcej niż tylko fizycznym odwzorowaniem tablicy.

5.4 Statystyka[1]

W tym miejscu przedstawię ilościowy wymiar projektu Bibliotekarz:

linii kodu: > 5500[2],

1 Prawdziwe statystyki wygenerowane d. 2022.11.11 umieściłem w Dodatek 2: Prawdziwe statystyki z 2022r.

tablic: 11,

okien w programie: 20.

2 Zaniżona wielkość szacunkowa; przy założeniu, że średnio 1 linia kodu w C++ to <30 bajtów, mój program ma ok. 6500 linii, ponieważ „szkielet” klas tworzy Builder zaniżyłem (z zapasem) tę liczbę o ok. 1000.

6 Wymagania sprzętowe

Wymagania sprzętowe programu Bibliotekarz nie są zbyt wygórowane, stanowią obecnie (1999) standard. Jedynym wymaganiem wybiegającym w przyszłość (jednak niezbyt odległą) jest rozdzielczość ekranu. Do sensownej pracy konieczna jest rozdzielczość ekranu 1024x768.

6.1 Minimalne

- Procesor Pentium[3];
- Windows 95;
- Rozdzielczość ekranu 800x600;
- Ok. 10 MB na dysku twardym.

6.2 Zalecane

- Procesor PentiumII (lub szybszy)[4];
- Windows 98[5];
- Rozdzielczość ekranu 1024x768;
- 32 MB RAM (lub więcej)[6];
- Jak najszybszy dysk twardy.

6.3 Uzasadnienie

- W przypadku dużych zbiorów danych do bez stresowej pracy konieczny jest wydajny sprzęt;
- Ilość wyświetlanych informacji powoduje, że w wyższej rozdzielczości pracuje się wygodniej;
- Windows 98 jest poprawioną wersją Windows 95.

3 Komentarz z 2022r: Tak na prawdę wystarczy nawet procesor klasy 386 późnych serii.
4 Komentarz z 2022r: Z Pentium 2 to oczywiście żart.
5 Komentarz z 2022r: Wersja Windows jest bez znaczenia.
6 Komentarz z 2022r: Biliotekarz jest maleńkim programem. Więcej pamięci mogłoby się przydać jedynie do buforowania odczytu danych z dysku.

7 Obsługa programu Bibliotekarz

Obsługa Bibliotekarza jest podobna do obsługi innych programów pod Windows 95. To sprawia, że każdy kto widział choćby „Worda" nie będzie miał problemów z Bibliotekarzem. W tym miejscu opiszę przebieg instalacji i usuwania Bibliotekarza oraz funkcje oferowane przez program.

7.1 Instalacja

Instalacja programu jest analogiczna do instalacji innych programów pod Windows 95. Aby zainstalować program Bibliotekarz należy wykonać następujące czynności

1. włożyć dyskietkę B1 do stacji;
2. uruchomić program „setup.exe" znajdujący się na dyskietce;
3. po wyświetleniu okna dialogowego należy kliknąć[7] kilkukrotnie klawisz „Next >" w celu rozpoczęcia kopiowania plików;
4. podczas kopiowania program poprosi kilka razy o zmianę dyskietki w stacji;
5. po zakończeniu kopiowania należy kliknąć klawisz „Finish" w celu zakończenia instalacji.

Uwaga: Po zakończeniu instalacji baza danych jest pusta – należy do niej wprowadzić dane by zaobserwować jej działanie. Pierwszą czynnością po zainstalowaniu Bibliotekarza powinno być utworzenie klasy (z menu Czytelnik ->Klasy ->Nowa klasa). Do poprawnego działania programu wymagane jest żeby każdy czytelnik należał do jakiejś klasy (nawet jeśli będzie się ona nazywać "-").

7.2 Funkcje programu – opis okien

Uwagi ogólne o obsłudze.

Po zakończeniu instalacji w menu „Start -> Programy" pojawi się nowa pozycja „Bibliotekarz". Należy ją kliknąć aby uruchomić program.

Program nie nakłada żadnych ograniczeń odnośnie ilości wprowadzanych danych. Umożliwia np. wprowadzenie książki o dowolnej liczbie autorów, powiązanie książki z dowolną liczbą haseł. Pod tym względem przewyższa (znane mi) programy komercyjne. Przewyższa je również pod względem prostoty obsługi i estetyki.

Dominującym elementem w programie są tabelki. W każdej tabelce znajduje się niebieski kursor. To on decyduje o tym, na jakim elemencie w tabelce będziemy działać. Aby skorzystać z tabelki należy na nią kliknąć. Wtedy można się w niej poruszać za pomocą kursorów. Element z tabelki można wybrać na 3 sposoby:

- Wciskając klawisz „Enter" na klawiaturze;
- Dwukrotnie klikając na wybranej pozycji w tabelce;
- Klikając odpowiedni klawisz w pobliżu tabelki.

Jeśli w pobliżu tabelki znajduje się pole edycyjne wspomagające wyszukiwanie, to klawisze kursorów i „Enter" w obrębie tego pola działają tak samo jak w obrębie tabelki.

Pole edycji wspomagające wyszukiwanie działa w ten sposób, że program automatycznie wyświetla tylko te elementy w tabelce, które zaczynają się na litery wpisywane w tym polu. Gdy chcemy przywrócić pełną zawartość tabelki, należy wyczyścić pole edycji (np. klawiszem „Backspace").

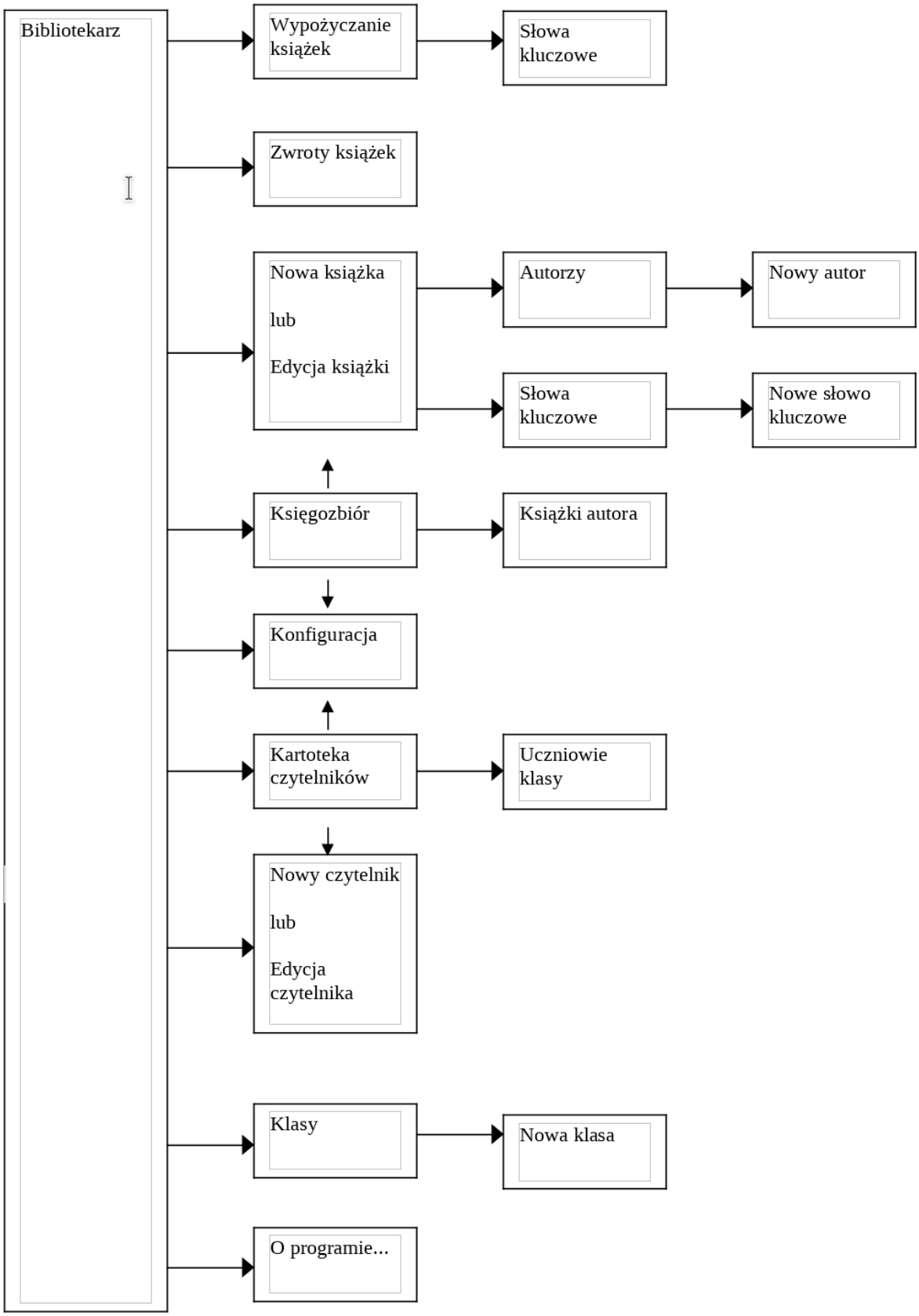
W każdym oknie mogą wystąpić następujące przyciski:

- OK - zatwierdza daną operację i zamyka okno;
- Zastosuj - tak jak OK., tylko nie zamyka okna;
- Zamknij - zamyka okno;
- Anuluj - rezygnacja z wprowadzonych zmian i zamknięcie okna.

7 Nacisnąć lewy klawisz myszy w momencie gdy jej kursor znajduje się nad omawianym obiektem.

Ponieważ Bibliotekarz działa w oparciu o szereg okien najpierw przedstawię ich logiczną organizację później opiszę ich funkcje.

Schemat logicznych powiązań między oknami programu Bibliotekarz:



Objaśnienie:

→ - pokazuje możliwe drogi wywołania okna; zamknięcie okna zawsze oznacza powrót do okna wywołującego.

Opis funkcji okien programu Bibliotekarz:

7.2.1 Okno główne programu „Bibliotekarz”

To okno składa się z dwóch elementów:

1. Menu rozwijanego, które jest podzielone tematycznie:

- Plik: ma tylko jedną pozycję „Zamknij program”;
- Książka: grupuje okna służące do obsługi książek, zawiera następujące;

pozycje:

- Wypożyczenie: przejście do okna „Wypożyczanie książek”;
- Zwrot: jw. „Zwrot książek”;
- Nowa: jw. „Nowa książka”;
- Przeglądaj księgozbiór: jw. „Księgozbiór”;

Czytelnik - grupuje okna służące do obsługi Czytelnika, składa się z następujących pozycji:

- Nowy: przejście do okna „Nowy czytelnik”;
- Przeglądaj kartotekę: jw. „Kartoteka czytelników”;
- Klasy: jw. „Klasy”;

Opcje

- Konfiguracja: przejście do okna „Konfiguracja”;

Pomoc

- O programie: przejście do okna „O programie”.

2. Klawiszy szybkiego dostępu

Klawisze szybkiego dostępu są rozmieszczone w jednej linii poniżej menu rozwijanego. Umożliwiają szybkie przejście do określonego okna z pominięciem menu rozwijanego. Przy intensywnej pracy z programem jest to bardzo pomocne. Od lewej, klawisze umożliwiają przejście do okien:

- „Nowa książka”;
- „Nowy czytelnik”;
- „Wypożyczanie książek”;
- „Zwroty książek”;
- „Księgozbiór”;
- „Kartoteka czytelników”.

Aby sobie przypomnieć do czego służy dany klawisz szybkiego dostępu wystarczy najechać kursorem na klawisz i poczekać kilka sekund – pojawi się krótka podpowiedź.

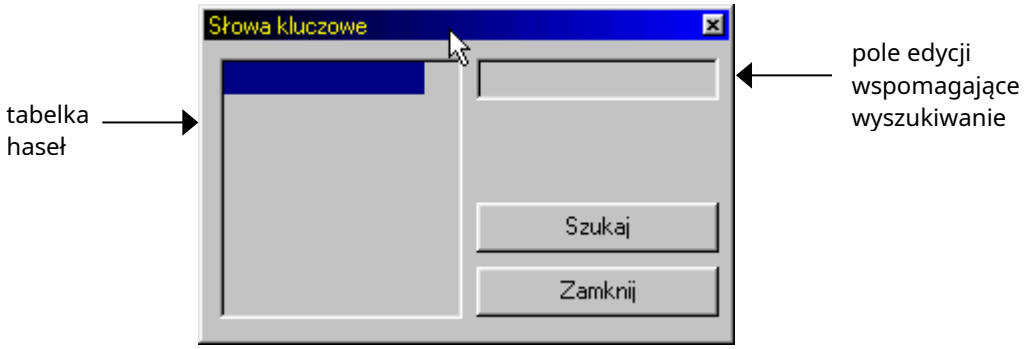
7.2.2 Wypożyczanie książek

To okno służy do ewidencjonowania wypożyczeń. Aby przeprowadzić wypożyczenie należy:

- Wybrać czytelnika, który ma pobrać książki. Wyszukiwanie czytelników wspomagają dwa pola edycji: „Nazwisko" i „Imię". Po wybraniu czytelnika w tabelce „Książki już wypożyczone" pojawi się lista książek, wcześniej wypożyczonych przez czytelnika. Poniżej wyświetlane są informacje o wysokości limitu i możliwości wypożyczania książek;
- Wybrać książki do wypożyczenia. W tym celu należy odszukać na liście „Księgozbiór" szukaną książkę i za pomocą przycisku „>" przesłać do sąsiedniej tabelki „Książki do wypożyczenia". Tę czynność należy powtórzyć dla każdej następnej książki. Wybór książki można anulować przyciskiem „<". Odnalezienie książki wspomagają pola edycji „Tytuł" i „Sygnatura". Wyszukiwanie wg haseł odbywa się z wykorzystaniem dodatkowego okna „Słowa kluczowe";
- Nacisnąć przycisk „Wypożycz książkę" – jeśli czytelnik ma mniejszy limit niż łączna ilość książek wypożyczonych i do wypożyczenia, program spyta czy ma odpowiednio zwiększyć limit.

Istnieje możliwość zmodyfikowania domyślnej daty wypożyczenia w polu edycji „Data wypożyczenia".

7.2.3 Słowa kluczowe



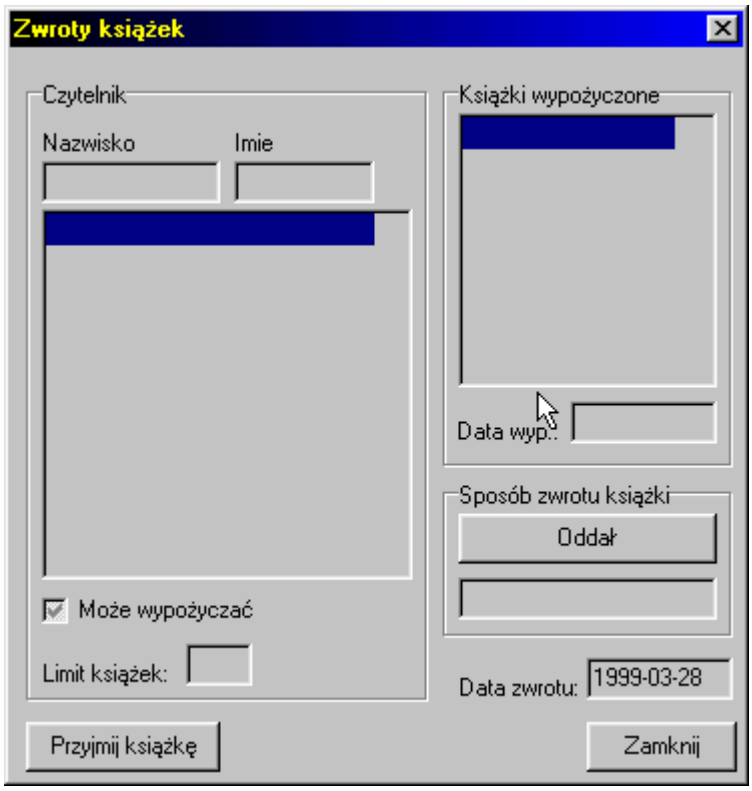
Okno wspomaga wyszukiwanie książek w księgozbiorze w oknie „Wypożyczanie książek". Wyszukiwanie wspomaga pole edycji. Po odnalezieniu hasła należy kliknąć przycisk „Szukaj". W oknie „Wypożyczanie książek" pojawią się wszystkie książki związane z tym hasłem. Okno może pozostawać otwarte podczas pracy w oknie „Wypożyczanie książek".

7.2.4 Zwroty książek

W tym oknie są ewidencjonowane zwroty książek przeczytanych przez czytelników.

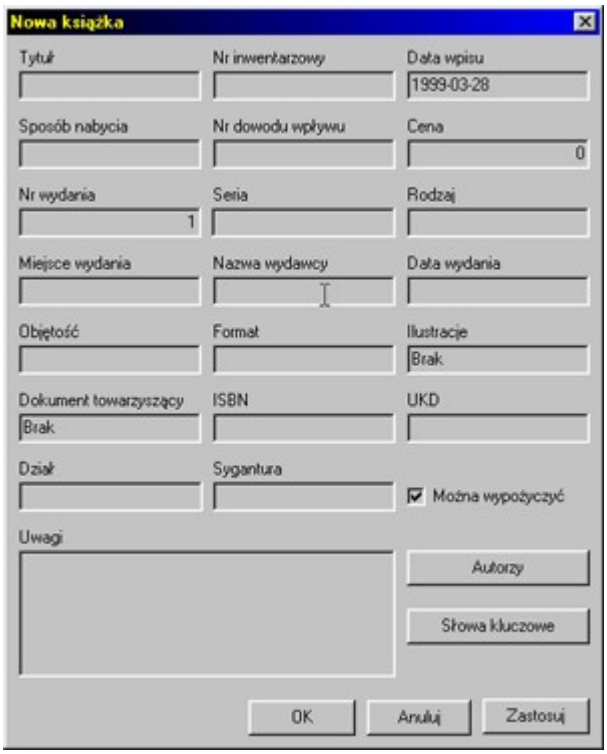
Aby przeprowadzić operację zwrotu książki należy:

- Wybrać czytelnika (w analogiczny sposób jak przy wypożyczaniu) z tabelki „Czytelnik". Wtedy w tabelce „Książki wypożyczone" pojawią się wszystkie nieoddane książki przez czytelnika. Poniżej tabelki czytelników wyświetlane są informacje o wysokości limitu i możliwości wypożyczania książek przez wybranego czytelnika;
- Wybrać książkę do zwrotu z tabelki „Książki wypożyczone". W polu edycji „Data wyp." Pojawi się data wypożyczenia książki;
- Określić sposób zwrotu książki. Jeśli czytelnik po prostu ją oddaje można kliknąć klawisz „Oddał" i program sam dokona tego wpisu.



Istnieje możliwość zmodyfikowania domyślnej daty zwrotu w polu edycji „Data zwrotu”.

7.2.5 Nowa książka lub Edycja książki



W tym oknie odbywa się wprowadzanie nowej książki. W tym celu należy uzupełnić wszystkie pola edycji odpowiednimi danymi (są one wymagane przez każdą bibliotekę). Autorów (kliknij przycisk „Autorzy”) wybiera się w oknie „Autorzy”.

Następnie można:

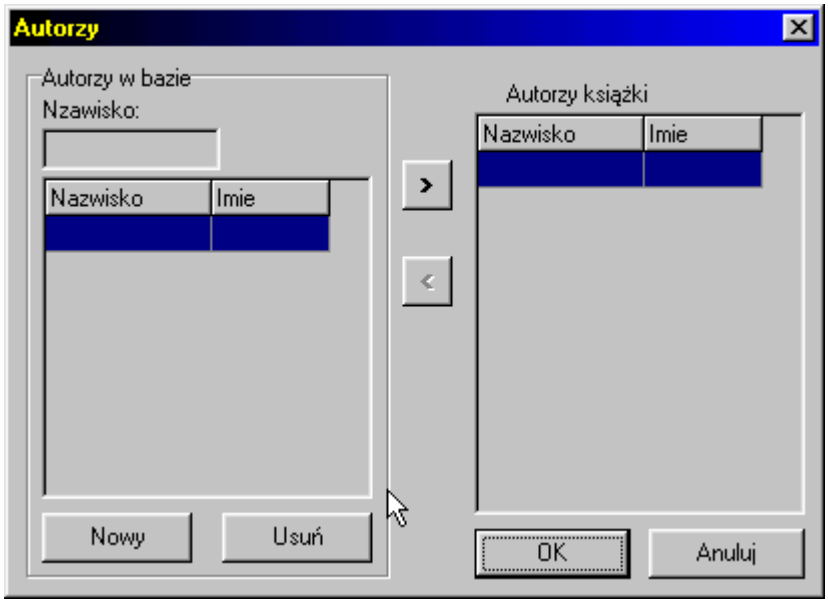
- Określić czy książka ma być dostępna dla czytelników;

- Wpisać uwagi o tej książce;
- Wybrać z jakimi hasłami ma być powiązana ta książka (kliknij przycisk „Słowa kluczowe”) w oknie „Słowa kluczowe”.

To okno można wywołać z okna „Księgozbiór” Wtedy będzie ono nosiło nazwę „Edycja książki”, a w odpowiednich polach edycji będą dane książki wybranej w oknie księgozbiór. W czasie edycji książki zmiany wprowadzane w zawartości okien „Autorzy” i „Słowa kluczowe” są nie odwołalne.

7.2.6 Autorzy

W tym oknie wybieramy autorów książki z dostępnej listy lub dodajemy nowych (kliknij przycisk „Nowy”). Autora można usunąć z listy (kliknij „Usuń”) jedynie w przypadku gdy autor nie jest związany z żadną książką. Wyszukiwanie autorów wspomaga pole edycji „Nazwisko”.



7.2.7 Nowy autor

W tym oknie wpisujemy nazwisko i imię nowego autora. Po wprowadzeniu nowego autora automatycznie jest on umieszczany w obu tabelkach okna „Autorzy” („Autorzy w bazie” i „Autorzy książki”).

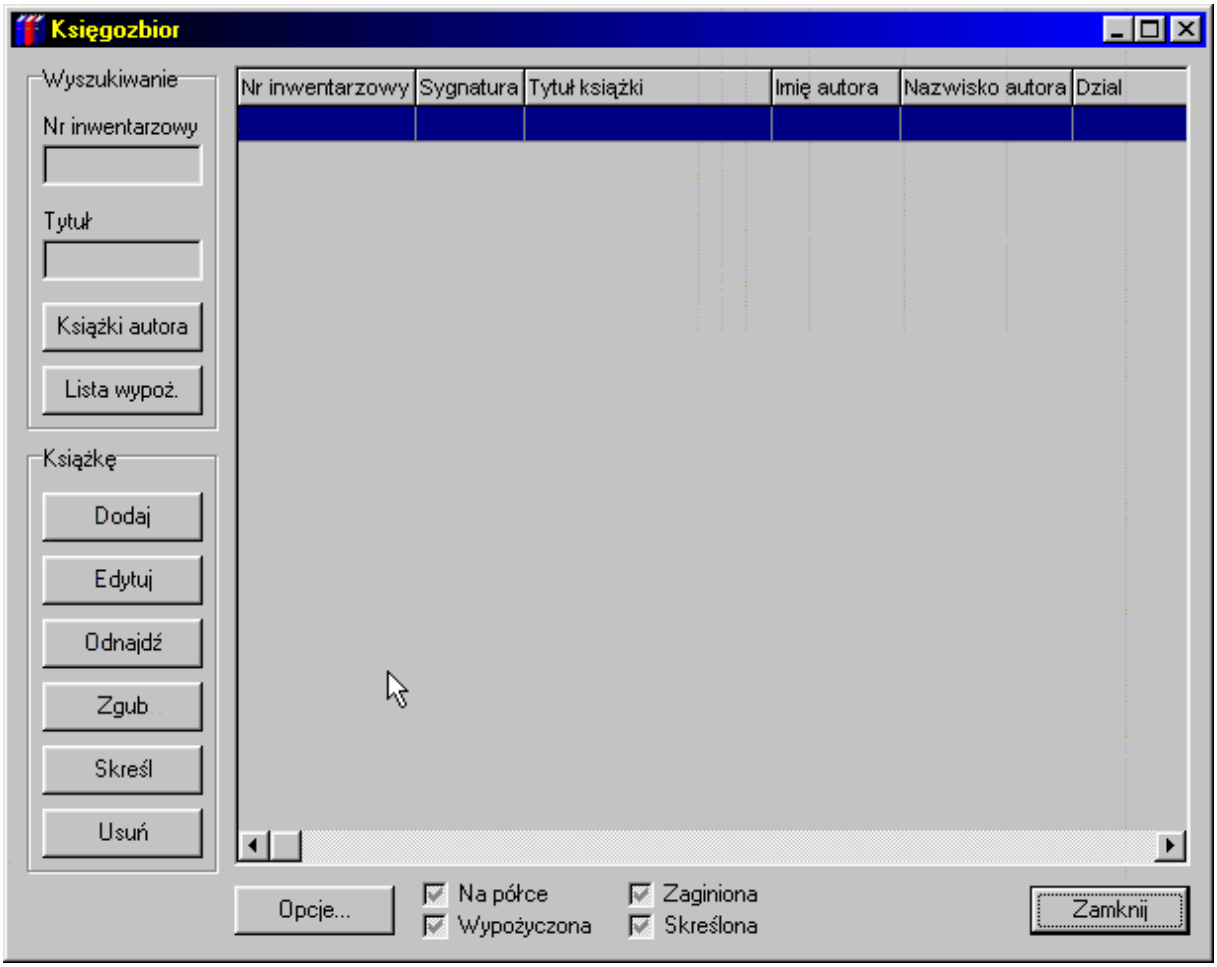
7.2.8 Słowa kluczowe

W tym oknie można wybrać z dostępnej listy haseł te z którymi książka ma być związana. Można też dodać nowe (kliknij „Dodaj”) lub usunąć stare hasła (kliknij „Usuń”).

7.2.9 Nowe słowo kluczowe

Służy do wprowadzania nowych słów kluczowych. Po wprowadzeniu nowego hasła automatycznie jest ono umieszczane w obu tabelkach okna „Słowa kluczowe” („Hasła bazy” i „Hasła książki”).

7.2.10 Księgozbiór



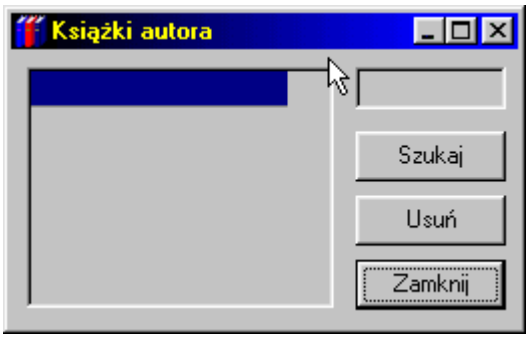
Okno „Księgozbiór" umożliwia przeprowadzenie większości operacji na książkach (bez wypożyczeń i zwrotów). Okno to otwiera się na całym ekranie. Po niżej tabelki jest przycisk „Opcje" jego kliknięcie spowoduje wyświetlenie okna „Konfiguracja". Obok tego przycisku znajduje się kilka „wskaźników" określających stan książki wskazywanej niebieskim kursorem w tabelce. Po lewej stronie okna są dwie grupy przycisków:

„Wyszukiwanie" - wspomagane polami edycji „Nr inwentarzowy" i „Tytuł". Posiada dwa przyciski:

- „Książki autora": wyświetla dodatkowe okno „Książki autora". Po wybraniu w tym oknie autora zostanie wyświetlona lista jego książek;
- „Lista wypożyczeń": wyświetla listę wypożyczeń wybranej książki;
- „Książkę": służy do działań na wybranej książce. Posiada następujące przyciski:
- „Dodaj": wyświetlenie okna „Nowa książka";
- „Edytuj": wyświetlenie okna „Edycja książki" - możliwość edycji książki;
- „Odnajdź": po odnalezieniu wcześniej zaginionej książki można ją przywrócić do normalnego funkcjonowania;
- „Zgub": jeśli książka zginęła to można ja zaznaczyć jako zaginioną;
- „Skreśl": skreślenie książki, nie można jej wypożyczać pozostaje jedynie informacja, że istniała;
- „Usuń": całkowite usunięcie książki nie pozostaje po niej żaden ślad.

7.2.11 Książki autora

Okno wywoływane z okna „Księgozbiór” służy do określenia jakiego autora mają być wyświetlane książki. Wyszukiwanie wspomaga pole edycji w które należy wpisać pierwsze litery nazwiska szukanego autora. Po kliknięciu „Szukaj” w oknie „Księgozbiór” wyświetlona zostanie lista książek tego autora. W oknie „Książki autora” można usunąć autora z bazy (o ile nie ma jego książek w bazie) klikając przycisk „Usuń”.



7.2.12 Konfiguracja

Służy do ustalania parametrów wyświetlania tabelek w oknach „Księgozbiór” i „Kartoteka czytelników”. Każdemu z tych okien poświęcona jest jedna „zakładka”.

Opis „zakładek”:

„Księgozbiór”: można ustalić według jakiej kolumny w tabelce mają być wyświetlane

książki („Wyświetlaj książki według”) i które książki mają być wyświetlane („Wyświetlaj książki”) – domyślnie wszystkie książki są wyświetlane.

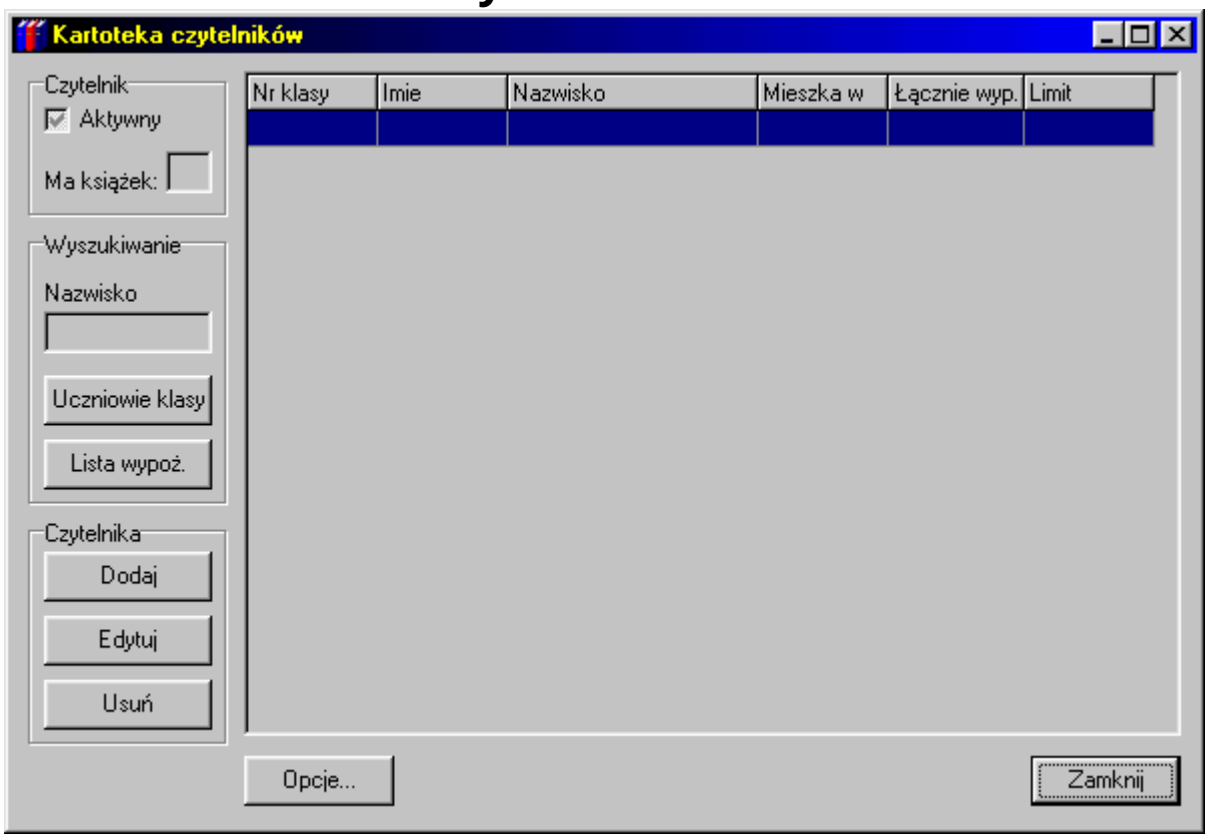
„Kartoteka”: można wyświetlać czytelników wg:

- Nazwisk;
- Klas;
- Miejsca zamieszkania;
- Limitów książek;
- Ilości wypożyczeń.

Można również ustalić którzy czytelnicy mają być wyświetlani:

- Aktywni;
- Zawieszeni;
- Uczniowie (innej klasy niż pierwsza wprowadzona);
- Nie uczniowie.

7.2.13 Kartoteka czytelników



Okno „Kartoteka czytelników” służy do obsługi czytelników. Po niżej tabelki jest przycisk „Opcje” jego kliknięcie spowoduje wyświetlenie okna „Konfiguracja”.

U góry po lewej stronie w ramce „Czytelnik” wyświetlane są: wysokość limitu i informacja o możliwości wypożyczania książek przez czytelnika wskazywanego niebieskim kursorem w tabelce.

W ramce „Wyszukiwanie” jest pole edycji umożliwiające wyszukiwanie wg nazwisk. Są też dwa przyciski:

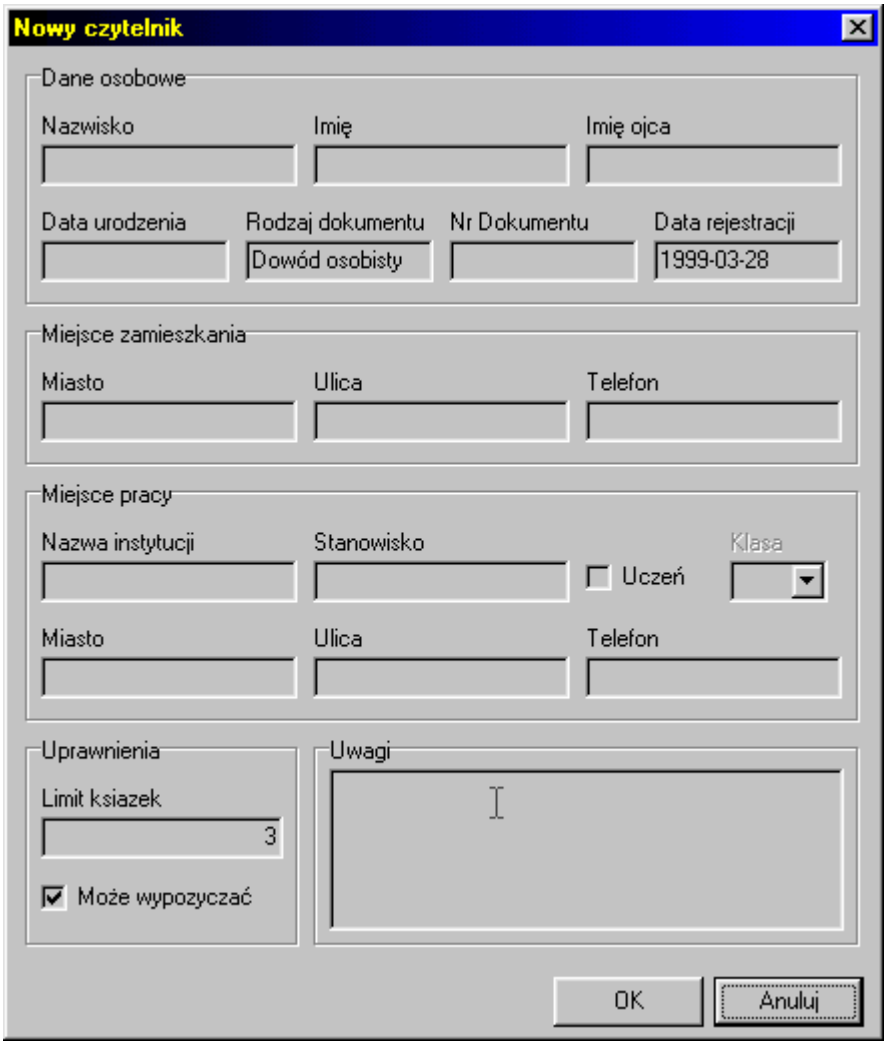
- „Uczniowie klasy” – wyświetla okno „Uczniowie klasy”, gdy wybierzemy w nim klasę to jej uczniowie zostaną wyświetleni w tabelce (okna „Kartoteka czytelników”);
- „Lista wypoż.” – umożliwia wyświetlenie wszystkich dotychczasowych wypożyczeń wybranego czytelnika.

W ramce „Czytelnika” są trzy przyciski:

- Dodaj – dodanie nowego czytelnika (okno „Nowy czytelnik”);
- Edytuj – edycja danych czytelnika (okno „Edycja czytelnika”);
- Usuń – usunięcie czytelnika z bazy danych.

7.2.14 2.7.1 Uczniowie klasy

Umożliwia wybór klasy której uczniowie mają być wyświetleni w tabelce okna „Kartoteka czytelników”. W tym celu należy wybrać z tabelki po lewej stronie okna klasę i kliknąć przycisk „Szukaj”.



7.2.15 Nowy czytelnik lub Edycja czytelnika

Okno „Nowy czytelnik” umożliwia wprowadzenie nowego czytelnika do bazy danych. Okno „Edycja czytelnika” umożliwia edycję wcześniej wprowadzonego czytelnika. Przy wprowadzaniu nowego czytelnika należy uzupełnić pola edycji wymagane przez każdą bibliotekę.

Dodatkowo można:

- Zaznaczyć czytelnika jako ucznia (klikając obok „Uczeń”) wtedy pole edycji z listą rozwijaną zostanie odblokowane i można wybrać klasę. Nawet gdy nie oznaczymy czytelnika jako uczeń będzie on członkiem pierwszej wprowadzonej klasy – jest to konieczne ze względów technicznych;
- W ramce uprawnienia określić przysługujący mu limit książek (domyślnie 3);

- W ramce uwagi można wpisać np. uzasadnienie zwiększenia limitu temu czytelnikowi, lub powody dla których nie może wypożyczać książek.

7.2.16 Klasy

Okno służy do obsługi klas. W tabelce jest wyświetlana nazwa klasy i ilość jej uczniów.

Okno „Klasy” ma następujące przyciski:

- „Nowa klasa”: otwiera okno „Nowa klasa”, które umożliwia wprowadzenia nowej klasy;
- „Zmień nazwę”: umożliwia zmianę nazwy istniejącej klasy;
- „Usuń klasę”: usuwa klasę z bazy danych.

7.2.17 Nowa klasa

Okno umożliwia wprowadzanie nowych klas.

7.2.18 O programie

Wyświetla podstawowe informacje o programie „Bibliotekarz”.

7.3 Usuwanie programu

Procedura usuwania Bibliotekarza jest również standardowa i przebiega automatycznie. Aby od instalować program Bibliotekarz należy wykonać następujące czynności:

- z paska zadań Windows wybrać przycisk „Start”;
- najechać kursorem myszy na menu „Ustawienia”;
- kliknąć „Panel sterowania”;
- w wyświetlonym oknie kliknąć ikonę „Dodaj/Usuń programy”;
- na liście zainstalowanych programów należy odnaleźć program Bibliotekarz i zaznaczyć go (niebieskim kursorem);
- kliknąć przycisk „Dodaj/Usuń”;
- potwierdzić chęć de instalacji programu;
- po zakończeniu usuwania programu, na wyświetlonym oknie kliknąć przycisk „OK”.

8 Ocena projektu Bibliotekarz

Projekt Bibliotekarz stanowił dla mnie wyzwanie. Jego realizacja opierała się w całości na wiadomościach nie-objętych programem nauczania, jednak stanowiących jego logiczne rozwinięcie. Pokonywałem małe (np. wstawienie domyślnej daty) i wielkie przeszkody (np. relacja wiele-do-wielu), które pojawiały się na każdym kroku. Ich drobiazgowy opis bez szczegółowego wyjaśnienia terminów i niuansów systemu Builder jest trudny, obszerny i raczej bez celowy.

Ponieważ wcześniej poznałem teorię dotyczącą języka C++, realizując projekt, chciałem poznać konkretne zastosowanie mechanizmu klas. Rozwiązania przyjęte przez firmę Borland (poza komponentami z Delphi) uznałem za dobre i warte stosowania. Jeszcze w czasie realizacji projektu Bibliotekarz, rozpocząłem pracę nad kolejnym projektem: „Parallel Port Lecture" (PPL). Przy jego realizacji zastosowałem całą zdobytą wiedzę z zakresu języków C++ i asemblera. PPL to program dydaktyczny tworzony dla potrzeb laboratorium szkolnego. Ma działać w DOS, więc interfejs użytkownika musiałem stworzyć sam - zrobiłem go wzorując się na rozwiązaniach systemu C++ Builder.

Pomimo, iż możliwości i obsługa programu nieco odbiegają od zakładanej, udało mi się zrealizować wszystkie badawcze cele projektu. Jest to o tyle istotne, że doświadczenia zdobyte w czasie prac nad Bibliotekarzem od razu zaczęły procentować i dają wymierne efekty (PPL). Zdobyte wiadomości z zakresu baz danych i systemu C++ Builder zamierzam wkrótce pogłębić i wykorzystać przy realizacji kolejnego projektu „Gospodarz". Będzie to program wspomagający prowadzenie rachunkowości domowej.

Uważam, że projekt Bibliotekarz zakończył się sukcesem, ponieważ zdobyłem konkretną i niebanalną wiedzę, którą potrafię zastosować w praktyce.

9 Bibliografia

1. Jerzy Gręboszˇ : „Symfonia C++”, Oficyna Kallimach Kraków, 1996;
2. Jerzy Gręboszˇ : „Pasja C++”, Oficyna Kallimach Kraków, 1997;
3. Kent Reisdorph, Ken Henderson : „C++ Builder”, Helion Gliwice, 1997;
4. Jim Mischel, Jeff Duntemann : „Borland C++ Builder” , Mikom W-wa, 1997;
5. Martin Gruber : „SQL”, Helion Gliwice, 1996;
6. Michael J. Hernandez : „Bazy danych...”, Mikom W-wa, 1998;
7. Pomoc systemu C++ Builder.

10 Dodatek1. Historia dokumentu

1. Wiosna 1999r.: Wersja finalna przedstawiona do oceny.
2. 2022-11-11:

   2.1. Zmieniłem dane autora: zamiast Jacka Jaworskiego podałem Jacka Marcina Jaworskiego;

   2.2. Zmieniłem "Zespół Szkół Łączności" na "Technikum Łączności";

   2.3. Poprawiłem formatowanie w całym dokumencie;

   2.4. Diagramy przerobione na obrazki *.png;

   2.5. Treść pracy zostawiłem bez zmian.
3. 2022-12-04:

   W celu zapewnienia oryginalnej treści tekstu:

   3.1. Przywróciłem oryg. roz. "Statystyka";

   3.2. Wyodrębniłem roz. "Dodatek 2: Prawdziwe statystyki z 2022r.".

11 Dodatek 2: Prawdziwe statystyki z 2022r.

```
$ licznik.py --włącz '.*[.]h'
Lini         Plik
          47 ./AB.cpp
          46 ./AB.h
          58 ./Abox11.cpp
          38 ./Abox11.h
         147 ./B.cpp
          71 ./B.h
         769 ./BDel.cpp
         105 ./BDel.h
          57 ./Bibliotekarz.cpp
         203 ./KA1.cpp
          57 ./KA1.h
         464 ./Karto.cpp
          84 ./Karto.h
         202 ./Klasy1.cpp
          44 ./Klasy1.h
         222 ./NC1.cpp
          92 ./NC1.h
         577 ./NK1.cpp
```

```
         196 ./NK1.h
         347 ./NKA.cpp
          81 ./NKA.h
         102 ./NKANA1.cpp
          44 ./NKANA1.h
         323 ./NKSK1.cpp
          79 ./NKSK1.h
         165 ./NKSKNSK1.cpp
          54 ./NKSKNSK1.h
         116 ./NKlasy1.cpp
          41 ./NKlasy1.h
         162 ./SKF1.cpp
          46 ./SKF1.h
         151 ./SZK1.cpp
          53 ./SZK1.h
         886 ./Wyp1.cpp
         118 ./Wyp1.h
         334 ./Zwrot1.cpp
          88 ./Zwrot1.h
          48 ./k1.cpp
          69 ./k1.h
-------< Podsumowanie Skanowania >===---
Licznik plików:                      39
Licznik Lini:                      6786
Licznik Znaków UTF-8:            199731
Średnia Il. Lini W Pliku:           174
Średnia Il. zn. w Lini:              29
Największy Plik:
         886 ./Wyp1.cpp


$ licznik.py -c --włącz '.*[.]sql' --włącz '.*[.]SQL'
Lini         Plik
          17 ./QCShow.SQL
          30 ./QDBShow.SQL
          31 ./QDBShow1.SQL
          13 ./WypKCzQ.SQL
          15 ./ZwrotKCzQ.SQL
-------< Podsumowanie Skanowania >===---
Licznik plików:                       5
Licznik Lini:                       106
Licznik Znaków UTF-8:              2066
Średnia Il. Lini W Pliku:            21
Średnia Il. zn. w Lini:              19
Największy Plik:
          31 ./QDBShow1.SQL
```