POLITECHNIKA
LUBELSKA
WYDZIAŁ MATEMATYKI
I INFORMATYKI TECHNICZNEJ

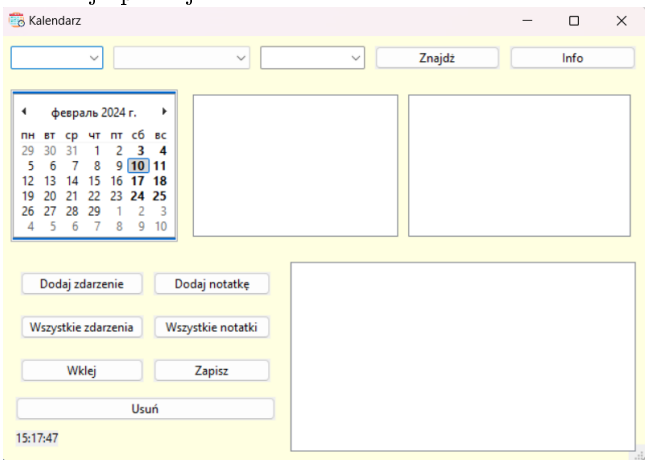
Projekt z zakresu programowania

Temat projektu: „Kalendarz”

Autorzy:
Davyd Antoniuk

Opis Interfejsu

Interfejs naszej aplikacji do tworzenia kalendarza został zaprojektowany w sposób intuicyjny i przyjazny dla użytkownika (patrz Rysunek 1). Główne funkcje aplikacji są łatwo dostępne i umożliwiają szybkie tworzenie, przeglądanie oraz zarządzanie kalendarzami. Interfejs został zaprojektowany z myślą o wygodnym korzystaniu zarówno na komputerach stacjonarnych, jak i urządzeniach mobilnych. Zastosowaliśmy czytelne ikony, przejrzyste menu oraz responsywne rozwiązania, aby zapewnić użytkownikom komfortowe doświadczenie podczas korzystania z naszej aplikacji.



Rysunek 1: Interfejs

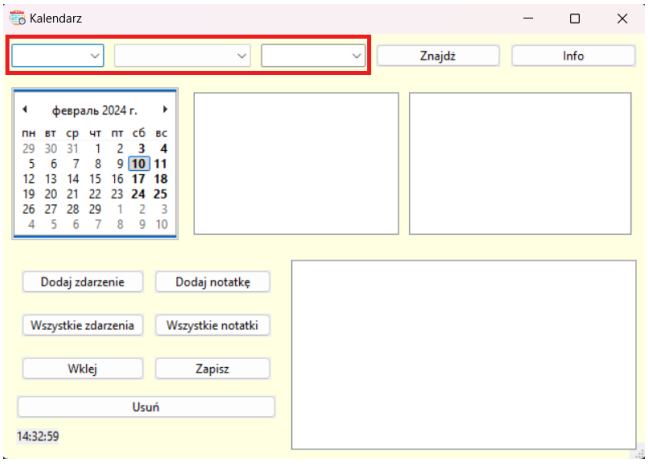
Cel Projektu

Celem naszego projektu jest stworzenie wszechstronnej aplikacji do tworzenia kalendarzy, która spełnia różnorodne potrzeby użytkowników. Chcemy zapewnić użytkownikom prosty i efektywny sposób zarządzania swoimi terminami, wydarzeniami oraz zadaniami. Nasza aplikacja ma być użytecznym narzędziem dla osób prywatnych, które potrzebują skutecznego sposobu organizacji czasu. Dążymy do stworzenia aplikacji, która będzie łatwa w obsłudze, ale jednocześnie oferuje szeroki zakres funkcji i możliwości personalizacji, aby każdy użytkownik mógł dostosować ją do swoich indywidualnych potrzeb.

Inicjalizacja listy

W kodzie inicjalizowane są trzy wysuwane listy(patrz Rysunek 2) (combo box):

- **A**: **ComboBox1**: Ta lista zawiera dni miesiąca od 1 do 31.
- **B**: **Choice1**: Druga lista zawiera nazwy miesięcy po polsku (np. „Styczeń" dla stycznia).
- **C**: **ComboBox2**: Trzecia lista zawiera lata od 2000 do bieżącego roku + 25.

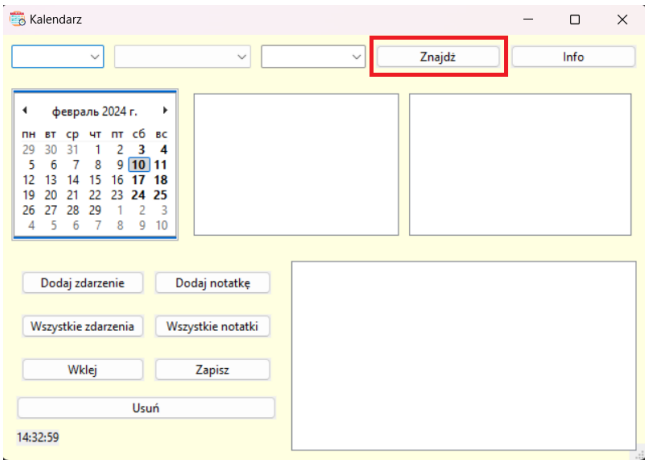


Rysunek 2: List

Dla każdego dnia miesiąca (od 1 do 31) tworzony jest ciąg znaków z liczbą, który jest dodawany do ComboBox1.
Miesiące (po polsku) dodawane są do Choice1 z tablicy months.
Lata (od 2000 do bieżącego roku + 25) dodawane są do ComboBox2.
Następnie wywoływana jest funkcja UpdateDaysList(), która prawdopodobnie aktualizuje listę dni w zależności od wybranego miesiąca i roku.

Przycisk Znajdź

Metoda OnButton3Click jest wywoływana po naciśnięciu przycisku(patrz Rysunek 3) w aplikacji „KalendarzDialog". Poniżej znajduje się opis tego, co dzieje się w tej metodzie:

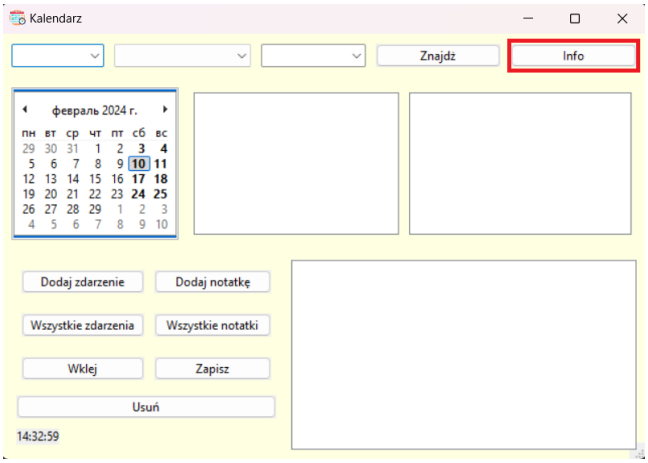


Rysunek 3: Przycisk „Znajdź"

1. Najpierw następuje aktualizacja daty w kalendarzu, listy z flagami i listy.
2. Następnie ListBox2 jest czyszczony.
3. Następnie pobierana jest bieżąco wybrana data z CalendarCtrl1 i formatowana jako ciąg znaków w formacie „dzień-miesiąc".
4. Następnie metoda przegląda listę świąt i porównuje daty każdego święta z wybraną datą w kalendarzu:
   - Jeśli znajdzie dopasowanie, dodaje to święto do ListBox2 i oznacza, że święta zostały znalezione.
   - Jeśli dla tej daty nie ma żadnych świąt, dodaje komunikat „Nie ma świąt" do ListBox2.

```
void KalendarzDialog::OnButton3Click(wxCommandEvent& event)
{
UpdateCalendarDate();
UpdateCheckListBox();
UpdateListBox();
ListBox2->Clear();
wxDateTime currentDate = CalendarCtrl1->GetDate();
wxString dateString = currentDate.Format("%d-%m");

bool holidaysFound = false;
for (const auto& holiday : holidays)
{
wxString holidayDate = wxString::Format("%02d-%02d", holiday.second.day,
holiday.second.month);

if (dateString == holidayDate)
{
wxString displayString = wxString::Format("%s - %s", holiday.first,
 holiday.second.name);
ListBox2->Append(displayString);
selectedHolidays.push_back(holiday.second);
holidaysFound = true;
}
}

if (!holidaysFound)
{
ListBox2->Append(_("Nie ma świąt"));
}
}
```

Opis przycisku info:

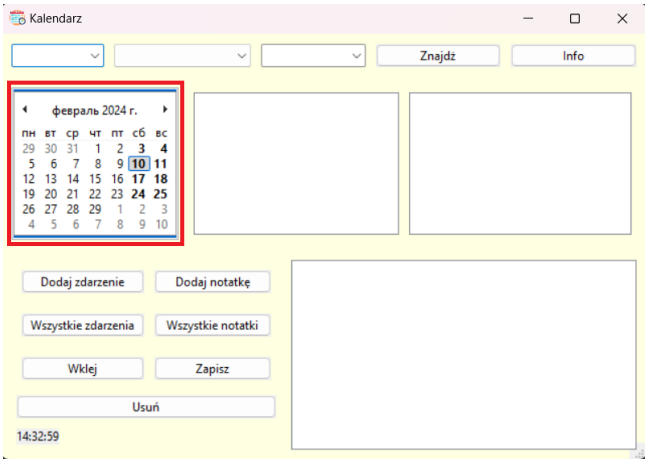


Rysunek 4: Przycisk „Info”

Nazwa: Wyświetl strony internetowe świąt
Funkcja: Otwiera strony internetowe związane z wybranymi świętami w domyślnej przeglądarce użytkownika.(patrz Rysunek 4) Przycisk jest aktywny tylko wtedy, gdy co najmniej jedno święto zostało wybrane na liście. Jeśli nie wybrano żadnych świąt, wyświetlany jest komunikat o błędzie.

```
void KalendarzDialog::OnButton9Click(wxCommandEvent& event)
{
if (selectedHolidays.empty())
{
wxMessageBox(_("Brak świąt do wyświetlenia"),
_("Błąd"), wxICON_ERROR | wxOK, this);
return;
}

for (const auto& holiday : selectedHolidays)
{
wxString url = wxString::FromUTF8(holiday.href.c_str());
wxLaunchDefaultBrowser(url);
}
}
```

Wybór daty



Rysunek 5: wxCalendar

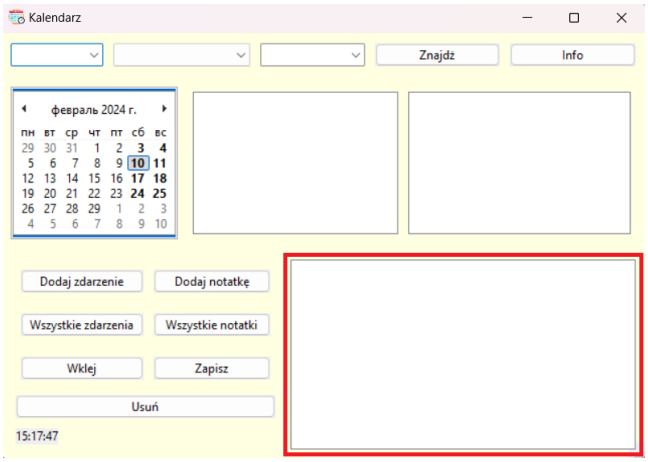
Użytkownik ma możliwość wyboru daty z kalendarza, klikając na nią.(patrz Rysunek 5) Po dwukrotnym kliknięciu prycisku datę:

- Wybrana data zostanie wyświetlona w polu „Dzień”.
- Miesiąc zostanie wybrany z listy rozwijanej „Miesiąc”.
- Rok zostanie wyświetlony w polu „Rok”.
- Zostanie wywołana funkcja `OnButton3Click()`.

Dodatkowe informacje:

- Kalendarz może być używany do wyświetlania wydarzeń lub świąt.
- Użytkownik może przejść do poprzedniego lub następnego miesiąca za pomocą strzałek.
- Użytkownik może również wybrać datę z pola tekstowego.

Lista świąt



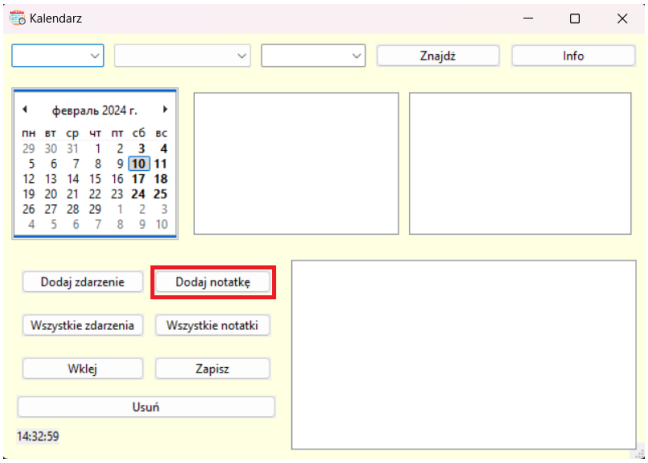
Rysunek 6: ListBox2(Święta)

Funkcja polega na wyświetlaniu listy świąt odpowiadających wybranej dacie w kalendarzu.(patrz Rysunek 6) Lista świąt jest wyświetlana po dwukrotnym kliknięciu daty w kalendarzu i naciśnięciu przycisku „3". Dodatkowo, przycisk „Informacja" pozwala otworzyć w przeglądarce dodatkowe informacje o wybranym święcie.

Opis

Pole, w którym wyświetlana jest lista świąt, jest polem tekstowym. Każde święto jest wyświetlane w osobnym wierszu, a jego nazwa może być uzupełniona krótkim opisem. Użytkownik może korzystać z przycisku „Informacja", aby uzyskać dodatkowe informacje o wybranym święcie w przeglądarce internetowej.

Opis przycisku Dodawania notatek:



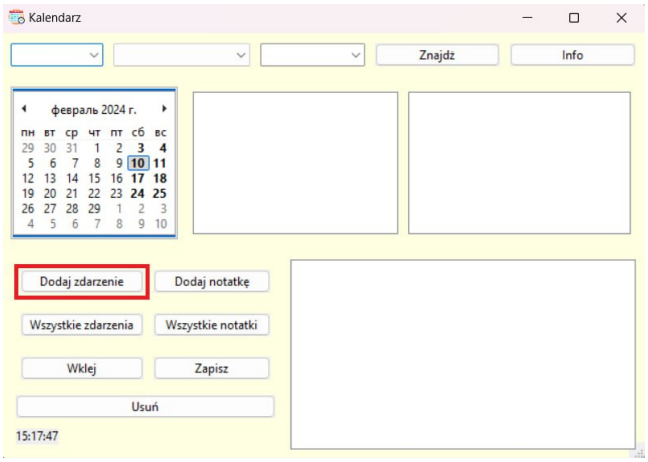
Rysunek 7: Przycisk „Dodaj notatkę”

Nazwa: Dodaj notatkę
Funkcja: obsługuje zdarzenie kliknięcia przycisku „Dodaj notatkę”(patrz Rysunek 7) w interfejsie użytkownika. Po kliknięciu tego przycisku wywoływane jest okno dialogowe, które pozwala użytkownikowi dodać nową notatkę do kalendarza.
Główny kod funkcji wykonuje następujące czynności:

- Tworzy nowe okno dialogowe za pomocą klasy wxDialog, które zawiera pola do wprowadzenia danych dotyczących nowej notatki, takie jak treść notatki i data.
- Definiuje etykiety i pola tekstowe dla treści notatki oraz pola wyboru daty.
- Tworzy przyciski „Zapisz” i „Anuluj” do zatwierdzania i anulowania dodawania notatki.
- Wiąże zdarzenia przycisków „Zapisz” i „Anuluj” z ich odpowiednimi funkcjami obsługi zdarzeń.
- Sprawdza poprawność wprowadzonych danych, takich jak czy data jest poprawna.
- Dodaje nową notatkę do listy notatek notes, jeśli data jest poprawna.
- Zamyka okno dialogowe po zapisaniu lub anulowaniu operacji.

```
wxDialog* dialog = new wxDialog(this, wxID_ANY, wxT("Dodaj notatkę"),
wxDefaultPosition,
wxSize(300, 150));
wxIcon icon;
icon.LoadFile("note.ico", wxBITMAP_TYPE_ICO);
dialog->SetIcon(icon);
wxStaticText* labelEvent = new wxStaticText(dialog, wxID_ANY, wxT("Notatka:"),
wxPoint(10, 10), wxSize(80, 20));
wxTextCtrl* textEvent = new wxTextCtrl(dialog, wxID_ANY, wxT(""),
wxPoint(100, 10), wxSize(180, 20));
wxStaticText* labelDate = new wxStaticText(dialog, wxID_ANY, wxT("Data:"),
wxPoint(10, 40), wxSize(80, 20));
wxDatePickerCtrl* datePicker = new wxDatePickerCtrl
(dialog, wxID_ANY, wxDefaultDateTime,
wxPoint(100, 40), wxSize(200, 20), wxDP_DEFAULT);
wxButton* btnSave = new wxButton(dialog, wxID_ANY, wxT("Zapisz"), wxPoint(10, 80),
wxSize(80, 25));
wxButton* btnCancel = new wxButton(dialog, wxID_ANY, wxT("Anuluj"), wxPoint(100, 80),
wxSize(80, 25));
btnSave->Bind(wxEVT_BUTTON, [this, dialog, textEvent, datePicker]
(wxCommandEvent& event) {
wxString eventName = textEvent->GetValue();
wxDateTime selectedDate = datePicker->GetValue();
if (!selectedDate.IsValid()) {
wxMessageBox(wxT("Nieprawidłowa data."), wxT("Błąd"), wxOK | wxICON_ERROR);
return;
}
Note newNote(selectedDate, eventName);
notes.push_back(newNote);
dialog->EndModal(wxID_OK);
});
btnCancel->Bind(wxEVT_BUTTON, [dialog](wxCommandEvent& event) {
dialog->EndModal(wxID_CANCEL);
});
wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);
sizer->Add(labelEvent, 0, wxALL, 5);
sizer->Add(textEvent, 0, wxALL, 5);
sizer->Add(labelDate, 0, wxALL, 5);
sizer->Add(datePicker, 0, wxALL, 5);
sizer->AddSpacer(10);
sizer->Add(btnSave, 0, wxALL, 5);
sizer->Add(btnCancel, 0, wxALL, 5);
dialog->SetSizerAndFit(sizer);
if (dialog->ShowModal() == wxID_OK) {
}
dialog->Destroy();
```

Opis przycisku Dodawania zdarzeń:



Rysunek 8: Przycisk „Dodaj zdarzenie"

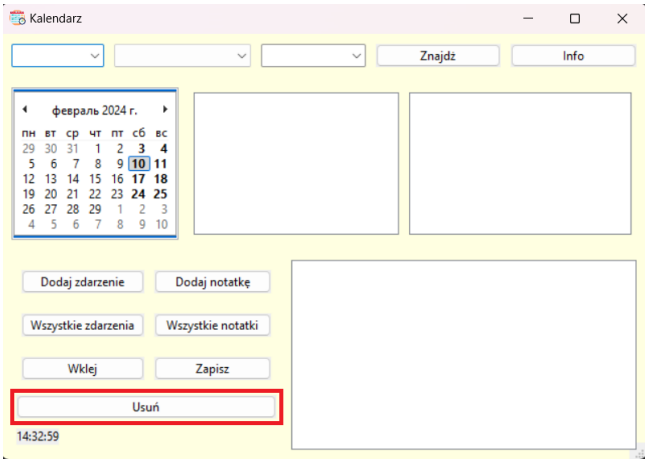
Nazwa: Dodaj zdarzenie w przyszłości
Funkcja: obsługuje zdarzenie kliknięcia przycisku „Dodaj zdarzenie" w interfejsie użytkownika.(patrz Rysunek 8) Po kliknięciu tego przycisku, wywoływane jest okno dialogowe, które pozwala użytkownikowi dodać nowe zdarzenie do kalendarza.
Główny kod funkcji wykonuje następujące czynności:

- Tworzy nowe okno dialogowe za pomocą klasy wxDialog, które zawiera pola do wprowadzenia danych dotyczących nowego zdarzenia, takie jak nazwa zdarzenia i data czas.
- Definiuje etykiety i pola tekstowe dla nazwy zdarzenia i daty czas.
- Tworzy przyciski „Zapisz" i „Anuluj" do zatwierdzania i anulowania dodawania zdarzenia.
- Wiąże zdarzenia przycisków „Zapisz" i „Anuluj" z ich odpowiednimi funkcjami obsługi zdarzeń.
- Sprawdza poprawność wprowadzonych danych, takich jak czy pola nie są puste, czy data jest w przyszłości, oraz czy format daty jest poprawny.
- Dodaje nowe zdarzenie do listy zdarzeń eventsList, jeśli dane są poprawne.
- Zamyka okno dialogowe po zapisaniu lub anulowaniu operacji.

```
void KalendarzDialog::OnButton1Click(wxCommandEvent& event)
{
wxDialog* dialog = new wxDialog(this, wxID_ANY, wxT("Dodaj zdarzenie"),
wxDefaultPosition, wxSize(300, 150));
wxIcon icon;
icon.LoadFile("event.ico", wxBITMAP_TYPE_ICO);
dialog->SetIcon(icon);

wxStaticText* labelEvent = new wxStaticText(dialog, wxID_ANY, wxT("Zdarzenie:"),
wxPoint(10, 10), wxSize(80, 20));
wxTextCtrl* textEvent = new wxTextCtrl(dialog, wxID_ANY, wxT(""), wxPoint(100, 10),
wxSize(180, 20));

wxStaticText* labelDateTime = new wxStaticText(dialog, wxID_ANY,
wxT("Data i godzina(lub data):"), wxPoint(10, 40), wxSize(150, 20));
wxTextCtrl* textDateTime = new wxTextCtrl(dialog, wxID_ANY, wxT(""),
wxPoint(100, 40), wxSize(180, 20));
textDateTime->SetHint(wxT("DD-MM-YYYY HH:MM"));

wxButton* btnSave = new wxButton(dialog, wxID_ANY, wxT("Zapisz"), wxPoint(10, 80),
wxSize(80, 25));
wxButton* btnCancel = new wxButton(dialog, wxID_ANY, wxT("Anuluj"), wxPoint(100, 80),
wxSize(80, 25));

btnSave->Bind(wxEVT_BUTTON, [this, dialog, textEvent, textDateTime]
(wxCommandEvent& event){
wxString eventName = textEvent->GetValue();
wxString eventDateTime = textDateTime->GetValue();

if (eventName.IsEmpty() || eventDateTime.IsEmpty()) {
wxMessageBox(_("Wprowadź wszystkie dane."), _("Błąd"), wxICON_ERROR | wxOK, this);
return;
}

wxDateTime currentTime = wxDateTime::Now();
wxDateTime enteredDateTime;
```

```
if (enteredDateTime.ParseFormat(eventDateTime, "%d-%m-%Y %H:%M")) {
if (enteredDateTime < currentTime) {
wxMessageBox(_("Nieprawidłowa data. Wybierz datę w przyszłości."), _("Błąd :("),
wxICON_ERROR | wxOK, this);
} else {
EventWithTime eventWithTime(eventName, enteredDateTime);
eventsList.push_back(eventWithTime);
dialog->EndModal(wxID_OK);
}
}
else if (enteredDateTime.ParseFormat(eventDateTime, "%d-%m-%Y")) {
enteredDateTime.SetHour(0);
enteredDateTime.SetMinute(0);
EventWithTime eventWithTime(eventName, enteredDateTime);
eventsList.push_back(eventWithTime);
dialog->EndModal(wxID_OK);
} else {
wxMessageBox(_("Format daty i godziny jest nieprawidłowy.
Wprowadź format DD-MM-YYYY HH:MM
lub DD-MM-YYYY."), _("Błąd :("), wxICON_ERROR | wxOK, this);
}
});
btnCancel->Bind(wxEVT_BUTTON, [dialog](wxCommandEvent& event) {
dialog->EndModal(wxID_CANCEL);
});

wxBoxSizer* sizer = new wxBoxSizer(wxVERTICAL);
sizer->Add(labelEvent, 0, wxALL, 5);
sizer->Add(textEvent, 0, wxALL, 5);
sizer->Add(labelDateTime, 0, wxALL, 5);
sizer->Add(textDateTime, 0, wxALL, 5);
sizer->AddSpacer(10);
sizer->Add(btnSave, 0, wxALL, 5);
sizer->Add(btnCancel, 0, wxALL, 5);

dialog->SetSizerAndFit(sizer);

if (dialog->ShowModal() == wxID_OK) {
}

dialog->Destroy();
}
```

Opis przycisku usuwania:



Rysunek 9: Przycisk „Usuń”

Nazwa: Usuń notatkę lub zdarzenie
Funkcja: obsługuje zdarzenie kliknięcia przycisku, który usuwa wybrane zdarzenia lub notatki z kalendarza.(patrz Rysunek 9)
Usuwanie notatek:

- Sprawdza indeks zaznaczonego elementu w kontrolce ListBox1 (prawdopodobnie zawierającej notatki).
- Jeśli jakiś element jest zaznaczony, wyodrębnia datę z zaznaczonego elementu.
- Usuwa zaznaczony element z ListBox1.

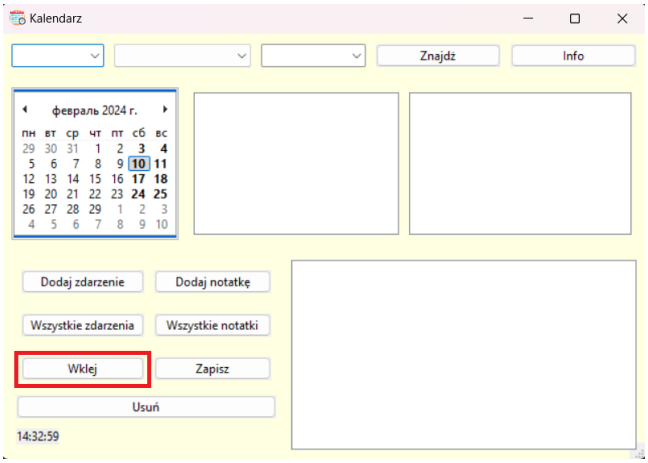
Usuwanie zdarzeń:

- Pobiera indeksy zaznaczonych elementów w kontrolce CheckListBox1.
- Przechodzi przez zaznaczone elementy i dodaje ich indeksy do wektora indicesToRemove.
- Usuwa zdarzenia z listy eventsList na podstawie indeksów przechowywanych w indicesToRemove.
- Usuwa zaznaczone elementy z CheckListBox1.
- Jeśli żadne zdarzenie nie było zaznaczone (licznik count == 0), to sprawdza, czy jest wybrany inny element w CheckListBox1 i usuwa go, jeśli jest.

```
void KalendarzDialog::OnButton4Click(wxCommandEvent& event)
{
int selectedIndexListBox = ListBox1->GetSelection();
int selectedIndCheckListBox = CheckListBox1->GetSelection();
if (selectedIndexListBox != wxNOT_FOUND && selectedIndCheckListBox != wxNOT_FOUND) {
wxMessageDialog dialog(nullptr,
wxT("Nie jest możliwe usunięcie dwóch elementów jednocześnie."),
wxT("Błąd"),
wxOK | wxICON_ERROR);
dialog.ShowModal();
}
else
{
if (selectedIndexListBox != wxNOT_FOUND) {
wxString selectedItemText = ListBox1->GetString(selectedIndexListBox);
auto it = std::remove_if(notes.begin(), notes.end(),
[selectedItemText](const Note& note) {
return note.eventName == selectedItemText;
});
notes.erase(it, notes.end());
ListBox1->Delete(selectedIndexListBox);
}

if (selectedIndCheckListBox != wxNOT_FOUND) {
wxString selectedCheckListBoxText = CheckListBox1->GetString(selectedIndCheckListBox);
size_t pos = selectedCheckListBoxText.find_first_of
(wxString::Format(wxT("0123456789")));
wxString onlyText;
if (pos != wxString::npos) {
onlyText = selectedCheckListBoxText.SubString(0, pos - 2);
} else {
onlyText = selectedCheckListBoxText;
}
if(!(CheckListBox1->IsChecked(selectedIndCheckListBox)))
{
wxMessageDialog dialog(nullptr,
wxT("Wydarzenie nie zostało ukończone. Czy na pewno chcesz je usunąć?"),
wxT("Potwierdź usunięcie"),
wxYES_NO |wxNO_DEFAULT| wxICON_QUESTION);
dialog.SetYesNoLabels(wxGetTranslation("Nie"), wxGetTranslation("Tak"));
int result = dialog.ShowModal();
```

```
if (result != wxID_YES) {
auto it1 = std::remove_if(eventsList.begin(), eventsList.end(), [onlyText]
(const EventWithTime& eventsList) {
return eventsList.eventName == onlyText;
});
eventsList.erase(it1, eventsList.end());
CheckListBox1->Delete(selectedIndCheckListBox);
}
else {}
}
else
{
auto it1 = std::remove_if(eventsList.begin(), eventsList.end(), [onlyText]
(const EventWithTime& eventsList) {
return eventsList.eventName == onlyText;
});
eventsList.erase(it1, eventsList.end());
CheckListBox1->Delete(selectedIndCheckListBox);
}
}
}
}
```

Opis przycisku wklejania:



Rysunek 10: Przycisk „Wklej”

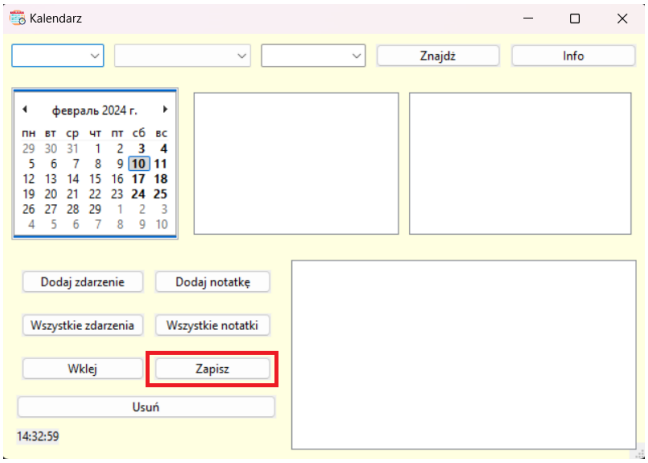
Nazwa: Wklej zdarzenia i notatki
Funkcja: Przycisk wklej służy do wczytywania zdarzeń lub notatek z pliku tekstowego do aplikacji.(patrz Rysunek 10)
Główny kod funkcji wykonuje następujące czynności:

- Funkcja przyjmuje argument filename, który jest nazwą pliku, z którego mają zostać wczytane zdarzenia lub notatki.
- Tworzy obiekt wxTextFile o nazwie file.
- Sprawdza, czy udało się otworzyć plik o nazwie filename do odczytu.
- Jeśli plik został otwarty pomyślnie:
  - Czyści zawartość kontrolki ListBox1, która prawdopodobnie zawiera notatki.
  - Czyści zawartość kontrolki CheckListBox1, która prawdopodobnie zawiera zdarzenia.
  - Iteruje przez wszystkie linie w pliku.
  - Jeśli linia jest pusta, oznacza to separator między notatkami a zdarzeniami.
  - Dla każdej linii, dzieli ją na tokeny za pomocą |. Pierwszy token jest nazwą notatki lub zdarzenia, a drugi token może zawierać informację o zaznaczeniu lub dodatkowych informacjach (np. flagi).
  - Jeśli separator został osiągnięty, dodaje nazwę zdarzenia do kontrolki CheckListBox1, w przeciwnym razie dodaje nazwę notatki do kontrolki ListBox1.
  - Po wczytaniu wszystkich linii, zamyka plik.

- Po wczytaniu zdarzeń lub notatek, wykonuje dodatkowe operacje (np. aktualizuje wektory lub aktualizuje widok aplikacji).
- Jeśli nie udało się otworzyć pliku do odczytu, wyświetla komunikat o błędzie za pomocą wxMessageBox.

```
void KalendarzDialog::LoadEventsFromFile(const wxString& filename) {
wxTextFile file;
if (file.Open(filename)) {
ListBox1->Clear();
CheckListBox1->Clear();

wxString line;
bool isSeparatorReached = false;

for (size_t i = 0; i < file.GetLineCount(); ++i) {
line = file.GetLine(i);
if (line.IsEmpty()) {
isSeparatorReached = true;
continue;
}

wxStringTokenizer tokenizer(line, "|");
wxString eventName = tokenizer.GetNextToken();
wxString isCheckedStr = tokenizer.GetNextToken();

if (isSeparatorReached) {
CheckListBox1->Append(eventName);
} else {
ListBox1->Append(eventName);
}
}

file.Close();


AddCheckListBoxItemsToVector();
AddListBoxItemsToVector();
UpdateDay();


} else {
wxMessageBox(wxT("Nie udało się otworzyć pliku do odczytu."),
 wxT("Błąd"), wxOK | wxICON_ERROR);
}
}
```

Opis przycisku zapisywania:



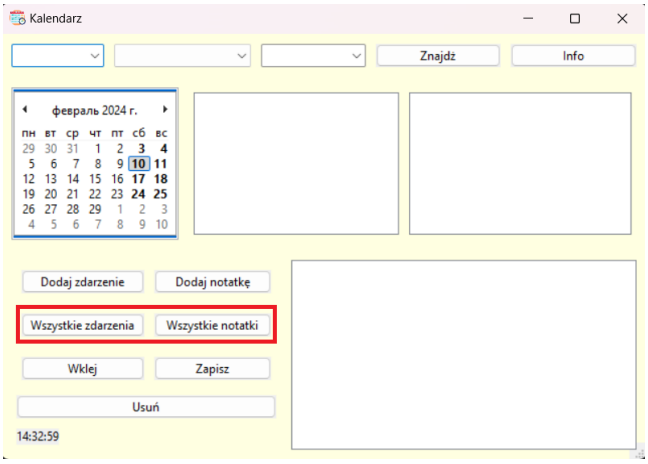
Rysunek 11: Przycisk „Zapisz”

Nazwa: Zapisz zdarzenie lub notatkę **Funkcja:** Przycisk zapisz służy do zapisywania zdarzeń lub notatek do pliku tekstowego o podanej nazwie.(patrz Rysunek 11)
Główny kod funkcji wykonuje następujące czynności:

- Funkcja przyjmuje argument filename, który jest nazwą pliku, do którego mają zostać zapisane zdarzenia lub notatki.
- Tworzy obiekt wxTextFile o nazwie file.
- Sprawdza, czy udało się utworzyć plik o nazwie filename.
- Jeśli plik został utworzony pomyślnie:
  - Iteruje przez wszystkie notatki w notes i dla każdej notatki tworzy tekstową linię zawierającą nazwę notatki oraz jej sformatowaną datę. Następnie dodaje tę linię do pliku.
  - Dodaje pustą linię oddzielającą notatki od zdarzeń.
  - Iteruje przez wszystkie zdarzenia w eventsList i dla każdego zdarzenia tworzy tekstową linię zawierającą nazwę zdarzenia, jego sformatowaną datę i czas oraz wartość 0 (to może być oznaczenie jakiegoś statusu lub flaga, która będzie interpretowana po odczycie pliku).
  - Następnie dodaje tę linię do pliku.
  - Zapisuje zawartość pliku.
  - Zamyka plik.
- Jeśli nie udało się otworzyć pliku do zapisu, wyświetla komunikat o błędzie za pomocą wxMessageBox.

```
void KalendarzDialog::SaveEventsToFile(const wxString& filename) {
wxTextFile file;
if (file.Create(filename)) {
for (const Note& note : notes) {
wxString line = wxString::Format("%s - %s",note.eventName,
note.GetFormattedDate());
file.AddLine(line);
}

file.AddLine(wxEmptyString);

for (const EventWithTime& event : eventsList) {
wxString line = wxString::Format("%s - %s|0",event.eventName,
event.eventDateTime.Format("%d-%m-%Y %H:%M"));
file.AddLine(line);
}
file.Write();
file.Close();
} else {
wxMessageBox(wxT("Nie udało się otworzyć pliku do zapisu."),
wxT("Błąd"), wxOK | wxICON_ERROR);
}
}
```

Wszystkie zdarzenia



Rysunek 12: Wszystkie zdarzenia\notatki

Po kliknięciu przycisków „Wszystkie zdarzenia", „Wszystkie notatki"(patrz Rysunek 12) wywoływane są okna dialogowe, które prezentują wszystkie zdarzenia lub notatki zapisane w kalendarzu.

Opis

Wszystkie zdarzenia: Po kliknięciu przycisku „Wszystkie zdarzenia", funkcja OnButton5Click wywołuje okno dialogowe, w którym wyświetlane są wszystkie zdarzenia z WxCheckListBox1. Okno dialogowe prezentuje zdarzenia w formie listy, posortowanej według daty i godziny. Użytkownik ma możliwość sortowania zdarzeń według daty za pomocą przycisku „Sortuj(wg dat)". Po zakończeniu przeglądania zdarzeń, użytkownik może zamknąć okno dialogowe za pomocą przycisku „Zamknij". Wszystkie notatki: Po kliknięciu przycisku „Wszystkie notatki", funkcja OnButton6Click wywołuje okno dialogowe, w którym wyświetlane są wszystkie notatki zapisane w CheckListBox1. Okno dialogowe prezentuje notatki w formie listy, gdzie każda notatka zawiera jej treść oraz datę. Po zakończeniu przeglądania notatek, użytkownik może zamknąć okno dialogowe za pomocą przycisku „Zamknij". Obie te funkcje umożliwiają użytkownikowi przeglądanie wszystkich zdarzeń lub notatek z kalendarza poprzez wyświetlenie ich w odpowiednich oknach dialogowych.

Wnioski z projektu

Podczas realizacji projektu udało się pomyślnie zaimplementować funkcje wyszukiwania daty w kalendarzu, dodawania wydarzeń na przyszłe daty oraz dodawania notatek, co znacząco zwiększa funkcjonalność systemu.
Projekt został wzbogacony o możliwość wyświetlania zarówno wydarzeń, jak i notatek w kalendarzu, co przyczynia się do zwiększenia czytelności i użyteczności interfejsu.
Dodatkowo, wprowadzono funkcję zapisu notatek i wydarzeń do pliku tekstowego oraz możliwość wczytywania danych z pliku, co umożliwia użytkownikom przechowywanie i przenoszenie danych między różnymi środowiskami.
Projekt został uzupełniony o funkcję usuwania notatek, co daje użytkownikom kontrolę nad swoimi danymi.
Warto podkreślić, że dodano funkcję wyświetlania wszystkich świąt oraz możliwość przejścia do opisu danego święta, co czyni kalendarz nie tylko praktycznym narzędziem, ale również źródłem wiedzy o świętach i tradycjach.
Możliwe dodatkowe funkcje, które można rozważyć do dodania w przyszłości, to:

- Powiadomienia o nadchodzących wydarzeniach lub świętach.
- Możliwość ustawienia powtarzalności dla wydarzeń.
- Integracja z innymi kalendarzami online.
- Wersja mobilna aplikacji dla większej wygody użytkowników.
- Personalizacja interfejsu i motywów kalendarza.

Dodatkowe funkcje te mogą dalszym stopniu zwiększyć użyteczność i atrakcyjność projektu dla użytkowników.