MINISTERSTWO EDUKACJI
i NAUKI

Andrzej Krawczyk
Ewa Sromala

Programowanie w środowisku języka strukturalnego 312[01].Z2.01

Poradnik dla ucznia

Wydawca
Instytut Technologii Eksploatacji – Państwowy Instytut Badawczy
Radom 2005

„Projekt współfinansowany ze środków Europejskiego Funduszu Społecznego”

Recenzenci:
mgr Bogdan Kostrzewa
mgr inż. Andrzej Uzar

Opracowanie redakcyjne:
mgr inż. Katarzyna Maćkowska

Konsultacja:
dr inż. Bożena Zając

Korekta:
mgr inż. Tomasz Sułkowski

Poradnik stanowi obudowę dydaktyczną programu jednostki modułowej 312[01].Z2.01 Programowanie w środowisku języka strukturalnego zawartego w modułowym programie nauczania dla zawodu technik informatyk.

Wydawca
Instytut Technologii Eksploatacji – Państwowy Instytut Badawczy, Radom 2005

SPIS TREŚCI

1. WPROWADZENIE

Poradnik będzie Ci pomocny w kształtowaniu umiejętności:
- projektowania rozwiązań algorytmicznych,
- budowania optymalnych algorytmów rozwiązywania problemów,
- kodowania algorytmów za pomocą języka strukturalnego wysokiego poziomu,
- testowania algorytmów i aplikacji,
- pracy w grupie.

W poradniku zamieszczono:
- wykaz umiejętności, jakie powinieneś posiadać, by bez problemów korzystać z poradnika,
- cele kształcenia, które pozwolą Ci uświadomić sobie cel informacji i ćwiczeń zawartych w poradniku,
- materiał nauczania, czyli skondensowane wiadomości teoretyczne oraz wskazówki, które pomogą Ci zrozumieć i poprawnie wykonać zestaw ćwiczeń,
- zestaw pytań, który pozwoli Ci sprawdzić, czy zapoznałeś się i zrozumiałeś wszystkie wiadomości i możesz przystąpić do wykonania ćwiczeń,
- ćwiczenia, które pozwolą ukształtować praktyczne umiejętności programowania strukturalnego,
- sprawdzian osiągnięć, dzięki któremu ocenisz swoje wiadomości i umiejętności ukształtowane w trakcie pracy z poradnikiem,
- literaturę uzupełniającą.

Pracując z poradnikiem i wykonując ćwiczenia możesz napotkać trudności. Uwagi zapisane pod ćwiczeniami pozwolą Ci:
- prawidłowo rozwiązać zadania, chociaż nie są jedyną drogą prowadzącą do osiągnięcia rezultatu końcowego – Twoja inwencja będzie bardzo cenna,
- zwrócić uwagę na umiejętności, które mogą przydać się w przyszłości i które są kluczowe w obszarze kształcenia.

W razie wątpliwości zwróć się o pomoc do nauczyciela.

2. WYMAGANIA WSTĘPNE

Przystępując do realizacji programu nauczania jednostki modułowej powinieneś umieć:
- sprawnie posługiwać się aparatem matematycznym,
- zarządzać zasobami na nośnikach stałych,
- korzystać z zasobów sieci lokalnej,
- modyfikować dokument tekstowy,
- wyszukiwać informacje w różnych źródłach.

3. CELE KSZTAŁCENIA

Po zakończonym procesie kształcenia tej jednostki modułowej będziesz potrafił:

- określić pojęcia: algorytm, program, dane i wyniki,
- zilustrować sposoby opisywania algorytmów w językach programowania strukturalnego,
- wyjaśnić cel, budowę i zasadę działania programu w języku programowania,
- zastosować podstawowe elementy języków algorytmicznych,
- zastosować instrukcje złożone,
- zaprojektować strukturę programu w języku programowania,
- sprawdzić poprawność algorytmów,
- ocenić czasową i pamięciową złożoność algorytmów,
- określić specyfikację problemu (algorytmu),
- wybrać algorytm do rozwiązania problemu: metodą zstępującą i metodą wstępującą,
- zaplanować podział zadania na moduły,
- zastosować współpracę modułów w oparciu o zasady bezpiecznego programowania,
- zaplanować zmienne globalne i lokalne,
- zaprojektować i zastosować biblioteki podprogramów,
- przetworzyć tekst,
- zastosować podprogramy graficzne, zaprojektować wykresy, posłużyć się kolorem, wypełnieniem i linią,
- odczytać i zapisać rysunki w pamięci i pliku, zaprojektować animacje,
- zastosować strukturalne typy danych: tablice, napisy, pliki, rekordy, zbiory,
- zastosować typy wskaźnikowe do operowania na dynamicznych strukturach danych takich jak: stosy, kolejki, listy, drzewa,
- zastosować rachunek wektorowy i macierzowy do rozwiązywania problemów,
- posłużyć się narzędziami informatycznymi do gromadzenia, przetwarzania i prezentacji danych,
- wybrać sposób prezentacji danych statystycznych,
- rozwiązać równania przy użyciu metod komputerowych.

4. MATERIAŁ NAUCZANIA

4.1. Algorytmy

4.1.1. Materiał nauczania

Algorytm to zbiór prostych czynności niezbędnych do wykonania pewnego zadania. Przeprowadza system z określonego stanu początkowego do wymaganego stanu końcowego.

Definicja wymaga pewnych objaśnień. Nie można jednoznacznie określić, co to jest „prosta czynność". Czy czynność jest prosta, zależy od przygotowania i doświadczenia osoby wykonującej ją. Rozumie się jednak, że chodzi o czynności, których wykonanie w danym momencie nie wymaga dodatkowych wyjaśnień lub ćwiczeń. Nazywa się ją krokiem (akcją) algorytmu.

Nie można też jednoznacznie określić, ile prostych czynności powinno składać się na algorytm. Wiadomo jednak, że powinna to być skończona ilość. Teoretycy algorytmiki zajmują się między innymi odpowiedzią na pytanie: który algorytm wykona postawione zadanie najsprawniej, najszybciej, a więc za pomocą najmniejszej liczby czynności prostych.

Pojęcie stanu początkowego jest bardzo uniwersalne. Pozwala myśleć o algorytmie jako procedurze przetwarzającej. Często jednak trudno jest zdefiniować stan początkowy. Algorytm komputerowy obliczający wartość bezwzględną liczby czeka na dane wejściowe. Kiedy pojawią się (liczba, liczby) przechodzi w stan końcowy, którym jest informacja (dane wyjściowe) o ich wartości bezwzględnej. Nie każdy algorytm wymaga podania w sposób jawny danych wejściowych. Algorytm, którego zadaniem będzie podanie kilku liczb pierwszych wydaje się po prostu „produkować" te liczby sam z siebie. Nie jest to prawdą. Algorytm ma przecież określone ile liczb znaleźć, od jakiej wartości rozpocząć poszukiwanie, jaką zastosować metodę. Umiejętność jasnego określania danych i informacji pozwala zrozumieć wiele zjawisk.

Algorytmy są zwykle realizowane, implementowane w jakimś środowisku. Obecnie jest to najczęściej program komputerowy lub układ elektroniczny.

Słowo algorytm pochodzi prawdopodobnie od nazwiska arabskiego matematyka Muhammeda ibn Musa Alchwarizmiego (IX wiek), który wprowadzał cyfry arabskie i zajmował się systemem dziesiętnym. W historii cywilizacji algorytmizację zastosowano między innymi przy produkcji tkanin żakardowych. Do ogromnego postępu przyczynił się Charles Babbage, który na podstawie swoich doświadczeń sformułował w roku 1842 ideę maszyny analitycznej zdolnej do realizacji złożonych algorytmów matematycznych.

Wynalezienie kart perforowanych pozwoliło zapisać czynności proste na nośniku, a czytającym je elektromechanicznym maszynom realizować algorytmy przetwarzające ogromne zbiory danych. Kolejnym krokiem było przeniesienie czynności prostych do pamięci kalkulatorów i komputerów. Pojawiło się pojęcie operacji (zadania) lub instrukcji (programu komputerowego). Rozwinęła się też nauka badająca podstawowe prawa budowania i realizowania algorytmów. Istnieje kilka podejść do tworzenia algorytmów:

- dziel i zwyciężaj – problem jest dzielony na kilka mniejszych, te znów są dzielone, aż ich rozwiązania staną się oczywiste,
- programowanie dynamiczne – problem dzielony jest na kilka części, ważność każdej z nich jest oceniana, a wyniki analizy prostszych zagadnień wykorzystuje się do rozwiązania głównego problemu,
- metoda zachłanna – wybiera się najbardziej obiecującą w danym momencie drogę rozwiązania bez głębszej analizy,

- programowanie liniowe – ocenia się rozwiązanie problemu za pomocą pewnej funkcji i szuka jej wartości minimalnej,
- poszukiwanie i wyliczanie – zbiór danych jest przeszukiwany aż do znalezienia rozwiązania,
- metody probabilistyczne – algorytm działa poprawnie, ale wynik uzyskiwany jest z zadaną niepewnością,
- metody heurystyczne – człowiek na podstawie własnych doświadczeń tworzy algorytm, który działa w najbardziej prawdopodobnych warunkach; rozwiązanie zawsze jest przybliżone.

W trakcie budowania algorytmicznego, komputerowego rozwiązania problemu korzysta się z kilku podstawowych technik implementacyjnych:
- proceduralność – algorytm jest dzielony na szereg podstawowych procedur (modułów); wiele algorytmów korzysta ze wspólnej biblioteki standardowych procedur, z których są w miarę potrzeby wywoływane (procedury graficzne, matematyczne itp),
- praca sekwencyjna – wykonywane są kolejne procedury algorytmu; na raz pracuje tylko jedna procedura,
- praca równoległa – wiele procedur wykonywanych jest w tym samym czasie, wymieniają się one danymi,
- rekurencja – procedura wywołuje sama siebie, aż do uzyskania wyniku,
- obiektowość – procedury i dane są łączone w obiekty reprezentujące najważniejsze elementy algorytmu oraz stan wewnętrzny wykonującego je urządzenia.

Bardzo wiele problemów algorytmicznych zostało już rozwiązanych i przeanalizowanych. Warto zapoznać się z tymi standardowymi algorytmami, żeby korzystać z optymalnych rozwiązań i ukształtować umiejętność budowania dobrych własnych rozwiązań.

Zwykle analizuje się algorytmy:
- obliczające (NWD, pierwiastek liczby, pierwiastki równania, liczby pierwsze, silnia, ciąg Fibonacciego, pi, Horner, MonteCarlo),
- sortujące (minimum, wyszukiwanie binarne, wstawianie, wybór, bąbelkowe, szybkie),
- graficzne (fraktale),
- kodujące (szyfr Cezara, systemy liczbowe),
- inne (kompresujące, kwantowe, AI).

Podstawową metodą oceny poprawnie działającego algorytmu jest jego złożoność. Złożoność obliczeniowa jest to zależność liczby operacji, które musi wykonać algorytm od liczebności przetwarzanego zbioru. Złożoność zapisuje się zwykle w postaci O(n log n), $O(n^2)$, O(n), $O(2^n)$.

Wygodnie byłoby brać pod uwagę czas realizacji algorytmu, jednak zależałby on od komputera, na którym jest realizowany. Dlatego zamiast czasu bada się ilość wykonywanych operacji elementarnych, zakładając jednakowy czas ich wykonania. Za operacje elementarne można uznać: operacje arytmetyczne, logiczne, relacyjne:
- podstawienie pod zmienną prostą lub wskaźnikową,
- indeksowanie, odwołanie do pola rekordu,
- inicjalizacja wywołania procedury,
- przekazanie wartości parametru aktualnego,
- instrukcja skoku wejścia / wyjścia.

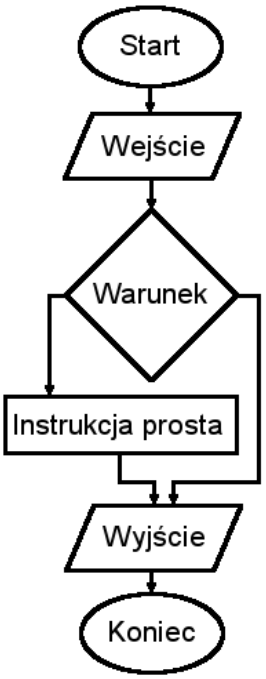
Poza tym bada się złożoność pamięciową, czyli ilość zasobów niezbędnych do wykonania algorytmu.

Opisanie zadania, które ma zostać rozwiązane za pomocą algorytmu, czyli poszukiwanie zależności pomiędzy danymi, a wynikami jest nazywane specyfikacją zadania. Rozwiązuje się je w kilku krokach:

1) Sformułowanie zadania.

2) Określenie danych wejściowych.
3) Określenie celu.
4) Poszukiwanie optymalnego algorytmu.
5) Zapisanie algorytmu (lista kroków albo schemat blokowy).
6) Analiza poprawności rozwiązania.
7) Testowanie rozwiązania i ocena efektywności.

Najbardziej czytelnym sposobem zapisywania algorytmów jest schemat blokowy. Korzysta się w nim z kilku rodzajów kształtów oznaczających podstawowe operacje:



- Owal:

a) znaczenie: początek i koniec algorytmu,
b) cechy: blok początkowy nie ma wejść, ma dokładnie jedno wyjście, zawiera nazwę algorytmu; blok końcowy może mieć wiele wejść, nie ma wyjścia, zawiera słowo „koniec”,

- Prostokąt:

c) znaczenie: instrukcja prosta,
d) cechy: wiele wejść, dokładnie jedno wyjście, zawiera instrukcję podstawienia,

- Kwadrat obrócony o 45°:

e) znaczenie: instrukcja warunkowa,
f) cechy: wiele wejść, dokładnie dwa wyjścia (podpisane TAK i NIE), zawiera warunek logiczny,

- Równoległobok:

g) znaczenie: operacja wejścia/wyjścia,
h) cechy: wiele wejść, dokładnie jedno wyjście, zawiera: przypisanie.

Algorytm zapisany w postaci schematu blokowego jest łatwy do zapisania, odczytania i analizowania. Niestety, nie posiada jednoznacznego przełożenia na kod programowania strukturalnego.

Algorytmy można budować i analizować za pomocą kartki i ołówka. Istnieją jednak narzędzia umożliwiające realizowanie algorytmów. Pozwalają one złożyć schemat blokowy z elementów składowych, wpisać operacje wejścia i wyjścia, zdefiniować podstawienia i warunki, ostatecznie uruchomić algorytm i krok po kroku obserwować sterowanie procesami i zmianę wartości zmiennych.

4.1.2. Pytania sprawdzające

Odpowiadając na pytania sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.

1) Podaj definicję algorytmu.
2) Skąd pochodzi nazwa „algorytm”?
3) Wyszukaj przykłady korzystania z algorytmów przed rokiem 1900.
4) Wymień cechy algorytmów.
5) Jakie znasz podejścia, techniki stosowane przy tworzeniu algorytmów?
6) Wymień algorytmy, z których ludzie korzystają na co dzień.
7) W jaki sposób można zapisać algorytm?
8) Co to jest złożoność obliczeniowa algorytmu?
9) Odszukaj program SETI@Home. Spróbuj zaklasyfikować ten algorytm.
10) Co to jest specyfikacja zadania?
11) Wylicz fazy rozwiązywania problemów.
12) Z jakich elementów buduje się schematy blokowe?

4.1.3. Ćwiczenia

Ćwiczenie 1

Zapisz w postaci algorytmu dzielenie liczb za pomocą kalkulatora.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) wyszukać w literaturze lub sieci WWW informacje o notacjach stosowanych w kalkulatorach,
2) przeanalizować sytuacje, z którymi może spotkać się użytkownik kalkulatora,
3) zapisać czynności użytkownika kalkulatora zgodnie z zasadami algorytmiki.
4) wykonać test algorytmu.

Wyposażenie stanowiska pracy:

Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 2

Zapisz algorytm przekształcania liczby całkowitej składającej się z kilku cyfr do postaci naukowej z dwoma miejscami po przecinku.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) wskazać dane,
2) omówić notację naukową,
3) zapisać algorytm,
4) wykonać test.

Wyposażenie stanowiska pracy:

Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 3

Zapisz w postaci algorytmu przeliczanie koni mechanicznych na waty i odwrotnie.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zaproponować rozwiązanie problemu zamiany jednostek w obie strony,
2) zapisać algorytm,
3) wykonać test.

Wyposażenie stanowiska pracy:

Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 4

Zapisz w postaci algorytmu obliczanie wartości bezwzględnej liczby.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) podać definicję wartości bezwzględnej,
2) zaproponować rozwiązanie problemu wprowadzania wartości całkowitej lub rzeczywistej,
3) zapisać algorytm,
4) wykonać test.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 5

Sprawdź poprawność algorytmów przedstawionych w postaci schematu blokowego.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) wymienić cechy poprawności algorytmów,
2) skorzystać ze schematów przygotowanych przez kolegów w trakcie wykonywania ćwiczeń 1-4,
3) sprawdzić poprawność kilku algorytmów,
4) przekazać swoje uwagi innym uczniom w grupie.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 6

Przedstaw w postaci schematu blokowego wybrany algorytm z życia codziennego.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) wybrać przykładową sytuację,
2) uzasadnić wybór,
3) zapisać algorytm,
4) zaproponować sposób sprawdzenia i wykonać sprawdzenie algorytmu.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 7

Zbuduj algorytm znajdowania wartości najmniejszej w zbiorze liczb całkowitych.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) uruchomić aplikację umożliwiającą testowanie algorytmów,
2) zbudować algorytm,
3) przetestować algorytm na zbiorze kilku liczb,
4) przetestować algorytm na zbiorze o liczebności trzech, dwóch i mniej liczb,
5) zapisać i przedyskutować zachowanie algorytmu w każdym przypadku.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 8
Zbuduj algorytm znajdujący w zbiorze liczb całkowitych wartość najmniejszą, ale większą niż zadana granica.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zmodyfikować algorytm z poprzedniego ćwiczenia,
2) zaproponować test dla różnych wartości granicznych: mniejsza niż najmniejsza liczba w zbiorze, większa niż największa liczba w zbiorze, pośrednia,
3) przedyskutować zachowanie algorytmu.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 9
Zbuduj algorytm znajdowania wartości największej w zbiorze liczb całkowitych.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) przeanalizować możliwość wykorzystania algorytmu z ćwiczenia 7,
2) zbudować algorytm,
3) przetestować rozwiązanie.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 10
Zbuduj algorytm znajdujący w zbiorze liczb całkowitych wartość największą, ale mniejszą niż zadana.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zmodyfikować algorytm z poprzedniego ćwiczenia,

2) zaproponować test dla różnych wartości granicznych: mniejsza niż najmniejsza liczba w zbiorze, większa niż największa liczba w zbiorze, pośrednia,
3) przedyskutować zachowanie algorytmu.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 11

Zbuduj algorytm zamieniający miejscami dwa wskazane wyrazy ciągu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować wykonanie ćwiczenia uwzględniając również indeksy wskazujące wyrazy ciągu leżące poza zakresem ciągu,
2) zbudować algorytm,
3) zaplanować i wykonać test.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 12

Zbuduj algorytm, który przesunie wskazane, leżące kolejno elementy ciągu o jedną pozycję w prawo.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować wykonanie ćwiczenia (jak określić wyrazy ciągu, które zostaną przesunięte?),
2) zaplanować sposób przedstawienia wyniku działania algorytmu,
3) zbudować algorytm,
4) zaplanować i wykonać test.

Wyposażenie stanowiska pracy:
Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 13

Zbuduj algorytm wyszukujący zadaną wartość w zbiorze.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) uwzględnić fakt, że zbiór niekoniecznie jest uporządkowany,
2) uwzględnić fakt, że wartość poszukiwana niekoniecznie musi występować w zbiorze,
3) zbudować algorytm,
4) zaplanować i wykonać test.

Wyposażenie stanowiska pracy:

Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów.

Ćwiczenie 14

Zbuduj algorytm, który obliczy pole trójkąta o zadanych bokach. Wykorzystaj arkusz kalkulacyjny do wykonania aplikacji, w której będzie można obliczyć pole trójkąta o podanych bokach.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) przeanalizować, czy z podanych długości boków można zbudować trójkąt,
2) zbudować algorytm,
3) przemyśleć, jak zachowa się algorytm dla nieprawidłowych wartości,
4) zaplanować i wykonać test,
5) zaplanować interfejs użytkownika w arkuszu kalkulacyjnym,
6) określić składnię wyrażenia, które obliczy pole trójkąta, gdy podane liczby są długościami boków trójkąta,
7) sprawdzić działanie aplikacji dla różnych danych liczbowych.

Wyposażenie stanowiska pracy:

Komputer z zainstalowanym programem umożliwiającym rysowanie schematów blokowych i testowanie algorytmów oraz arkusz kalkulacyjny.

4.1.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) podać definicję algorytmu?	☐	☐
2) podać specyfikację przedstawionego zadania?	☐	☐
3) poprawnie rysować schematy blokowe?	☐	☐
4) znaleźć wartość najmniejszą w zbiorze? największą?	☐	☐
5) zamienić miejscami dwa wyrazy ciągu?	☐	☐
6) wyszukać wartość w zbiorze?	☐	☐

4.2. Podstawy programowania

4.2.1. Materiał nauczania

Język programowania to usystematyzowany sposób przekazywania komputerowi poleceń do wykonania.

Język programowania pozwala programiście na precyzyjne przekazanie maszynie skąd pobrać dane, jak je przetworzyć, jakie czynności podjąć w określonych warunkach i w jaki sposób zaprezentować informacje.

Języki programowania klasyfikuje się zależnie od tego, do jakich zastosowań najlepiej się nadają:

- strukturalne (program składa się ze zmiennych oraz modyfikujących je operacji, z jawnie określonym przepływem sterowania: C, Pascal),
- obiektowe (zbiór obiektów komunikujących się pomiędzy sobą w celu wykonywania zadań; ułatwia pisanie, konserwację i wielokrotne użycie programów lub ich fragmentów: C++, Java),
- funkcyjne (filozofia programowania, w której funkcje należą do wartości podstawowych, a nacisk kładzie się na wartościowanie funkcji, a nie na wykonywanie poleceń: Haskell),
- logiczne (program podawany jest jako pewien zestaw zależności, a obliczenia są dowodem pewnego twierdzenia w oparciu o te zależności: Prolog),
- inne (w sieci można znaleźć listę ponad 2 tysięcy języków programowania).

Jedną z technik programowania imperatywnego jest programowanie strukturalne. Polega na stosowaniu jedynie pewnych wyróżnionych struktur algorytmicznych. Są to:

- sekwencja (polecenia wykonywane są w kolejności zapisanej w algorytmie),
- selekcja (w zależności od wartości pewnego wyrażenia, realizowana jest jedna z dwóch możliwości),
- iteracja (można wprowadzić licznik określający ilość powtórzeń polecenia),
- rekursja (wywołanie procedury przez siebie samą),
- modułowa struktura.

Dzięki takiemu ograniczeniu zbioru możliwych konstrukcji algorytmicznych mogą powstawać programy w dużym stopniu niezawodne i przyjazne zarówno dla użytkownika, jak i dla programisty. Łatwo jest też o teoretyczną analizę poprawności i złożoności algorytmu zapisanego strukturalnie. Mimo iż każdy program da się napisać w sposób strukturalny, nie zawsze będzie to optymalne ze względu na zużyty czas lub pamięć.

Istnieją dwa podstawowe style programowania:

- zstępujący (Top-down, czyli z góry do dołu): najpierw planuje się rozwiązanie całości zagadnienia, a potem dochodzi do szczegółów,
- wstępujący (Bottom-up, czyli z dołu w górę): z zaprojektowanych i rozwiązanych elementów składa się całość.

Programowanie obiektowe posiada następujące cechy charakterystyczne:

- abstrakcja (każdy obiekt w systemie stanowi model, który może wykonywać pracę, opisywać i zmieniać swój stan oraz komunikować się z innymi obiektami w systemie, bez ujawniania, w jaki sposób zaimplementowano te cechy),
- enkapsulacja (ukrywanie implementacji; obiekt nie może zmieniać stanu wewnętrznego innych obiektów w nieoczekiwany sposób. Tylko wewnętrzne metody obiektu są uprawnione do zmiany jego stanu),
- polimorfizm (referencje i kolekcje obiektów mogą dotyczyć obiektów różnego typu, a wywołanie metody dla referencji spowoduje zachowanie odpowiednie dla pełnego typu obiektu wywoływanego),

- dziedziczenie (umożliwia definiowanie i tworzenie specjalizowanych obiektów na podstawie bardziej ogólnych. Dla obiektów specjalizowanych nie trzeba ponownie definiować całej funkcjonalności, lecz tylko uzupełnić tę, której nie ma obiekt ogólniejszy).

Każdy program operuje na zbiorze danych. Potrzebne jest dla nich miejsce w pamięci. Projektant, a w jego ślad programista, powinien zaplanować odpowiednią dla każdej danej ilość miejsca w pamięci. Do dyspozycji ma zbiór typów zmiennych (i stałych), spośród których dokonuje wyboru:

- typ całkowity,
- typ zmiennoprzecinkowy ,
- typ logiczny,
- typ łańcuchowy,
- typ znakowy,
- typ wskaźnikowy,
- typ rekordowy,
- typ tablicowy,
- typ obiektowy.

W celu optymalnego i poprawnego korzystania z typów zmiennych, programista powinien zapoznać się z ich właściwościami w konkretnych warunkach realizacji.

Za pomocą zmiennych przechowuje się dane wejściowe, wartości tymczasowe powstające w trakcie realizacji programu i informacje przed ich zaprezentowaniem. Tworzy się z nich wyrażenia, które składają się ze zmiennych, operatorów i innych struktur.

Programista ma do dyspozycji następujące operatory, za pomocą których buduje wyrażenia i warunki logiczne w kodzie programu:

- arytmetyczne,
- logiczne,
- relacyjne,
- bitowe.

Polecenia, z których składa się program wykonywane są przez procesor sekwencyjnie. w każdym poleceniu najpierw obliczana jest jego prawa strona, a następnie przypisywana do zmiennej występującej po lewej stronie znaku równości. Algorytm liniowy nie nadaje się jednak do rozwiązywania problemów, chyba że bardzo prostych (np. fragment większego zadania). Programista ma do dyspozycji narzędzia, które pozwalają rozgałęzić (zmienić kierunek realizacji) program:

- instrukcja warunkowa (realizuje gałąź TAK albo NIE),
- pętla (powtarza instrukcje pod warunkiem spełnienia podanego warunku).

Ponadto powtarzające się fragmenty kodu można zapisać w postaci procedury lub funkcji. Kod programu zyskuje na czytelności. Program można pisać i modyfikować szybciej i łatwiej.

Na powstanie programu, czyli jednoznacznie sformułowanego zadania wraz ze sposobem jego rozwiązania za pomocą komputera składa się:

- sformułowanie problemu,
- zaprojektowanie programu:
  a) określenie danych wejściowych
  b) wybranie algorytmów przetwarzania
  c) określenie sposobu prezentowania wyniku
- zbudowanie algorytmu:
  d) schemat blokowy
  e) testowanie

- napisanie kodu programu (implementacja algorytmu):
  f) wybór narzędzia,
  g) sposób nazywania zmiennych,
  h) komentarze,
  i) wcięcia,
  j) bloki,
- skompilowanie kodu,
- testowanie programu,
- tworzenie dokumentacji.

Praktycy programowania polecają korzystanie z następujących zasad ułatwiających nie tyle pisanie programu, co jego modyfikowanie i testowanie:
- kod programu będzie czytany przez innych ludzi,
- dodawaj więcej komentarzy niż będzie ci, jak sądzisz, potrzeba,
- stosuj komentarze wstępne,
- stosuj przewodniki w długich programach,
- komentarz ma dawać coś więcej niż tylko parafrazę tekstu programu,
- stosuj do komentarzy takie same wcięcia jak w tekście, który komentują,
- błędne komentarze to gorzej niż zupełny brak komentarzy,
- stosuj odstępy dla poprawienia czytelności,
- używaj dobrych nazw mnemonicznych,
- wystarczy jedna instrukcja w wierszu,
- porządkuj listy według alfabetu,
- nawiasy kosztują mniej niż błędy,
- stosuj wcięcia dla uwidocznienia struktury programu,
- stosuj wcięcia dla uwidocznienia struktury danych.

Być może dzięki tym regułom prawo informatyczne „Pisanie programu jest rozkoszą, uruchamianie – zmorą" nie będzie brzmiało aż tak groźnie.

4.2.2. Pytania sprawdzające

Odpowiadając na pytania sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.
1) Co to jest język programowania?
2) Zdefiniuj programowanie strukturalne.
3) Opisz metodę wstępującą w programowaniu.
4) Opisz metodę zstępującą w programowaniu.
5) Co nazywa się blokiem programu?
6) Jakie rodzaje zmiennych można wykorzystywać?
7) Wymień operatory
8) Naszkicuj etapy rozwiązywania problemów za pomocą komputera.
9) Omów rolę komentarzy w kodzie programu.
10) Omów rolę i zasady stosowania wcięć w kodzie programu.
11) Od czego zależy kolejność wykonywania operacji?

4.2.3. Ćwiczenia

Ćwiczenie 1

Napisz program wyświetlający na ekranie imię i nazwisko.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm realizacji zadania,
2) wpisać kod programu w wybranym edytorze,
3) skompilować program,
4) uruchomić program wykonywalny i przetestować działanie.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 2

Napisz program wyświetlający na ekranie wynik dowolnego działania arytmetycznego na liczbach naturalnych.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić program i przetestować.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 3

Napisz program, który pobierze od użytkownika dwie liczby naturalne i wyświetli na ekranie cztery działania arytmetyczne wykonane na tych liczbach. Zastosuj pełny zapis (np.: 3+5=8).

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić program i przetestować,
4) sprawdzić, co stanie się, gdy wprowadzone zostaną wartości innego typu?
5) zaproponować rozwiązanie problemu z punktu 4 w programach tworzonych w przyszłości.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 4

Zbadaj własności zmiennych typu całkowitego: rozmiar i maksymalna wartość.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) wyszukać w dostępnych źródłach informacje o zakresie zmiennych,
2) zaplanować sposób znalezienia wartości maksymalnej,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić program i porównać wyniki ze znalezionymi informacjami.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 5

Zbadaj rozmiar zmiennych typu zmiennoprzecinkowego.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) wyszukać w dostępnych źródłach informacje o zakresie zmiennych,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program na dowolnej wartości i na ułamku okresowym.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 6

Napisz program obliczający średnią arytmetyczną kilku liczb.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób wprowadzania liczb,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program dla kilku liczb. Dla jednej liczby.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 7

Napisz program obliczający powierzchnię koła o promieniu podanym przez użytkownika.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program. Sprawdzić zachowanie dla liczb ujemnych.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 8

Napisz program obliczający powierzchnię trapezu. Długości podstaw i wysokość podaje użytkownik.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program. Sprawdzić zachowanie dla liczb ujemnych.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 9

Napisz program zamieniający radiany na stopnie.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program. Sprawdzić zachowanie dla liczb ujemnych.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 10

Napisz program zamieniający stopnie kąta podane w postaci liczby rzeczywistej na stopnie, minuty i sekundy łuku.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program. Sprawdzić zachowanie dla liczb ujemnych.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 11

Połącz ćwiczenie 9 i 10. Zamieniaj radiany na stopnie, minuty i sekundy łuku.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) skopiować niezbędne linie kodu,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

4.2.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) skompilować program?	☐	☐
2) interpretować komunikaty o błędach?	☐	☐
3) pobierać dane od użytkownika?	☐	☐
4) wyświetlać informacje na ekranie?	☐	☐
5) budować proste wyrażenia arytmetyczne?	☐	☐
6) dobrać typ zmiennej do problemu?	☐	☐
7) zapisać fragment kodu w postaci procedury?	☐	☐
8) pobrać kilka wartości do zmiennych?	☐	☐

4.3. Struktury i metody programowania

4.3.1. Materiał nauczania

Instrukcja warunkowa może przyjmować różne formy. W zależności od potrzeb programista w celu zrealizowania algorytmu może zastosować jej mniej lub bardziej złożoną postać:

- instrukcja warunkowa prosta (wykonaj, jeśli spełniony jest warunek),
- instrukcja warunkowa pełna (jeśli spełniony jest warunek wykonaj instrukcję1 w przeciwnym razie wykonaj instrukcję2),
- instrukcja warunkowa złożona (jeśli ... w przeciwnym razie jeśli ...),
- instrukcja warunkowa przełączająca (oblicz wartość wyrażenia i wykonaj odpowiadającą mu instrukcję z listy).

Często zdarzającym się błędem jest nieprawidłowe zapisanie warunku logicznego. Warunkiem logicznym mogą być w najprostszej sytuacji dwa wyrażenia połączone operatorem relacyjnym. Należy zwrócić uwagę na sposób zapisywania operacji przypisania (podstawienia) i porównania (relacja) w wybranym języku programowania.

Warunek logiczny może mieć postać złożoną, polegającą na łączeniu warunków logicznych prostych operatorami logicznymi.

W językach programowania wykorzystuje się kilka rodzajów pętli, określanych niezbyt precyzyjnymi, za to krótkimi terminami:

- skończona (z góry wiadomo ile razy zostanie wykonana),
- nieskończona:
  a. ze sprawdzaniem warunku wykonania na początku (możliwy brak wykonań),
  b. ze sprawdzaniem warunku wykonania na końcu (przynajmniej jedno wykonanie).

Z definicji algorytmu wynika, że pętla (program) muszą wykonać zawarte w nich instrukcje skończoną ilość razy. Nie zawsze jednak wiadomo ile to będzie razy, ponieważ zostanie określone dopiero w trakcie wykonywania programu.

Warunek logiczny kończący lub umożliwiający wykonanie pętli może być złożony (może zawierać warunki logiczne połączone operatorem logicznym). Pętla może zawierać instrukcję będącą pętlą. Mówi się wówczas o zagnieżdżaniu pętli.

Zastosowanie pętli do wykonania powtarzającej się operacji dla wielu zmiennych pozwala zaoszczędzić czas i uprościć kod, czyli również zmniejszyć ryzyko wprowadzenia błędu. Technika ta nazywa się iteracją, a zmienna zliczająca kolejne pętle zwyczajowo oznaczana jest literą „i”. Należy bardzo uważnie określać ilość powtórzeń (pętli) lub warunek kończący ich wykonywanie, ponieważ MUSI on być spełniony w skończonym czasie.

W programie mogą występować grupy instrukcji, które powtarzają się w wielu miejscach. Warto wydzielić te grupy nadając im unikatowe nazwy i wykorzystywać jak instrukcję prostą. Nazywa się ją procedurą lub funkcją w zależności od języka programowania.

W wielu językach programowania procedura wywołuje samą siebie. Technika ta nazywana jest rekurencją. Znacznie upraszcza (skraca) kod programu, nie oznacza jednak szybszego wykonania, ponieważ kolejne rezultaty wywołania są zapamiętywane jako tymczasowe, a następnie wykorzystywane w odwrotnej kolejności.

Często można wydzielić grupy instrukcji nie identycznych, ale wykonujących podobne zadanie. Na przykład program może obliczać funkcję sinus dla kąta 1°, 2°, 3°, ... w takiej sytuacji buduje się procedurę z parametrem. Parametr może być stałą lub zmienną o wybranej nazwie, za pomocą której wykonywane są obliczenia wewnątrz procedury. W zależności od wybranego języka programowania procedury mogą przyjmować różne postaci, na przykład

mogą zachowywać się jak zmienne i stanowić fragment wyrażeń matematycznych pisanych po prawej stronie instrukcji podstawienia. Zmienne zadeklarowane wewnątrz procedury obowiązują tylko wewnątrz niej. Nazywa się to zachowanie zakresem zmiennej.

Jeżeli kolejne rezultaty obliczeń muszą być zapamiętywane lub należy wykonać obliczenia dla wielu wartości tego samego typu (np. liczb, łańcuchów), wygodnie jest nazywać wszystkie te zmienne jedną wspólną nazwą i ponumerować. Można kojarzyć ten sposób z matematycznym zapisem x_1, x_2, ..., chociaż pojęcie tablicy (macierzy) jest w matematyce znane i stosowane. Powstaje w ten sposób (jest deklarowana) zmienna tablicowa. Zmienna tablicowa musi mieć określony rozmiar (twórcy języków programowania starają się czasami obejść to ograniczenie, na przykład udostępniając możliwość zmieniania rozmiaru tablicy w trakcie wykonywania kodu). W zależności od języka programowania tablice mogą być numerowane od wartości zero (najczęściej) albo od dowolnej wartości.

Tablica może mieć więcej niż jeden wymiar (przynajmniej w sposobie zapisywania). Wypełnianie i odczytywanie tablicy jednowymiarowej odbywa się w pętli. Tablice o liczbie wymiarów n obsługiwane są zwykle przez n zagnieżdżonych pętli.

Skoro zmienna jest zapisywana w pamięci, to można ją odczytać podając jej adres w pamięci. Należałoby jednak pamiętać, jaki jest rozmiar zmiennej, czyli na ilu bajtach zapisywana jest jej wartość. Wygodniejszym sposobem jest korzystanie ze wskaźników. Wskaźnik, pointer, jest typem zmiennych, który przechowuje adres zmiennej. Dzięki wskaźnikom można przyspieszyć wykonywanie programu i umożliwić procedurom korzystanie ze zmiennych (zamiast tylko wartości zmiennych) i poszerzyć możliwości korzystania z pamięci. Wskaźniki sprawdzają się również przy przetwarzaniu tablic.

Typowymi zadaniami algorytmicznymi, rozwiązanymi, przetestowanymi i opisanymi w literaturze naukowej są:

- znajdowanie wartości minimalnej i maksymalnej,
- sortowanie zbiorów (różne metody),
- wyszukiwanie informacji w zbiorach,
- obliczanie wartości (silnia, NWD, pierwiastek, liczby pierwsze),
- usprawnienia wykonywania obliczeń (Horner, NWD),
- metody przybliżone (pi, pierwiastek, MonteCarlo),
- szyfrowanie,
- i wiele innych.

Trzeba poznać te metody, żeby uniknąć „wyważania otwartych drzwi” i nabrać wprawy w optymalnym rozwiązywaniu problemów.

4.3.2. Pytania sprawdzające

Odpowiadając na pytania sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.

1) Jakie znasz sposoby rozgałęziania kodu programu?
2) Omów budowę instrukcji warunkowej.
3) Omów rodzaje pętli stosowanych w programach.
4) Podaj powody stosowania podprogramów.
5) Wyjaśnij pojęcie parametru.
6) Co oznacza pojęcie iteracji w programowaniu? Kiedy występuje iteracja (po czym ją poznasz)?
7) Podaj definicję metody rekurencyjnej.
8) Jak zbudowane są tablice?
9) Co to są tablice wielowymiarowe?
10) Wymień metody sortowania.

11) Czego dotyczy metoda:
    a. Hornera,
    b. Monte Carlo,
    c. sita Eratostenesa?
12) Na czym polega szyfr Cezara?
13) Jak powstaje ciąg Fibonacciego?
14) Do czego służą wskaźniki?

4.3.3. Ćwiczenia

Ćwiczenie 1

Połącz ćwiczenia 9 i 10 z poprzedniego rozdziału.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zamienić kod z ćwiczenia 9 na procedurę,
2) zamienić kod z ćwiczenia 10 na procedurę,
3) uzupełnić kod w taki sposób, by możliwa była zamiana radianów na stopnie, minuty i sekundy łuku,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 2

Napisz program, który wyświetli na ekranie tyle gwiazdek (lub innych znaków), ile lat ma użytkownik.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 3

Napisz program, który pobierze od użytkownika liczbę i wyświetli pierwszych 10 wielokrotności tej liczby.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować czytelny sposób zaprezentowania wyników,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 4

Napisz program, który zsumuje liczby całkowite wprowadzane przez użytkownika. Wyświetl wynik dopiero po zakończeniu wprowadzania.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób zakończenia wprowadzania liczb. Użytkownik nie wie, ile będzie liczb,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 5

Napisz program, który pobiera od użytkownika znaki z klawiatury i wypełnia nimi całą następną linię ekranu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób zakończenia wprowadzania znaków,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 6

Napisz program, który pobierze od użytkownika rok urodzenia i wyświetli na ekranie tyle gwiazdek, ile użytkownik ma lat. Zamiast każdej dziesiątej gwiazdki wstaw znak #. Możesz wykorzystać dwa inne znaki.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 7

Napisz program, który pobierze od użytkownika liczbę, a następnie wyświetli w kolejnych liniach ekranu rosnącą liczbę gwiazdek od jednej do podanej przez użytkownika, a następnie malejąco do jednej gwiazdki.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 8

Napisz program, który pobierze od użytkownika liczbę, a następnie wyświetli kwadrat złożony z takiej liczby gwiazdek, jaką podał użytkownik, ale z pustymi znakami na przekątnej. Narysuj dwa kwadraty z przekątnymi poprowadzonymi od lewego górnego i od prawego górnego rogu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 9

Napisz program, który pobiera od użytkownika ocenę szkolną i wyświetla jej słowny odpowiednik.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób kontrolowania poprawności danych,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 10

Napisz program, który znajdzie metodą Euklidesa największy wspólny dzielnik dwóch liczb wprowadzonych przez użytkownika.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) przemyśleć, w jaki sposób na wynik mogą wpłynąć liczby ujemne,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 11

Napisz program, który iteracyjnie obliczy silnię liczby wprowadzonej przez użytkownika.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program. Jak duża może być liczba? Sprawdzić za pomocą kalkulatora.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 12

Napisz program, który rekurencyjnie obliczy silnię liczby wprowadzonej przez użytkownika.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
- komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 13

Napisz program, który zamieni liczbę naturalną wprowadzoną przez użytkownika na postać binarną.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sprawdzenie, czy liczba należy do zbioru liczb naturalnych,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 14

Napisz program, który będzie zamieniał liczby naturalne wpisywane przez użytkownika na wybrany system liczbowy od binarnego do dziesiętnego.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób testowania poprawności działania algorytmu,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 15

Napisz program, który oblicza wskazane przez użytkownika wyrazy ciągu Fibonacciego i wypisuje je na ekranie wraz z komentarzem. Zwróć uwagę na poprawność gramatyczną.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) znaleźć w dostępnych źródłach informacje na temat ciągu Fibonacciego,
2) zaplanować sposób poprawnego wypisywania liczników porządkowych,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 16

Napisz program, który oblicza wartość stałej pi/4 na podstawie sumy szeregu 1–1/3+1/5–1/7+... Oszacuj dokładność.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób oszacowania dokładności przybliżenia,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 17

Napisz program, który obliczy pierwiastek dowolnego stopnia z liczby podanej przez użytkownika, z zadaną dokładnością na podstawie wzoru Herona.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować zabezpieczenie przed wpisaniem (podaniem) niepoprawnej wartości liczby podpierwiastkowej,
2) zaplanować wyświetlanie informacji o ilości iteracji,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program. Sprawdź za pomocą kalkulatora.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 18

Napisz program, który przedstawi ułamek dziesiętny właściwy w postaci binarnej z zadaną dokładnością.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób sprawdzenia wyniku,
2) zaplanować sposób oszacowania dokładności,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 19

Napisz program, który ciąg znaków podany przez użytkownika posortuje przez wstawianie. Wyświetl kolejne etapy porządkowania.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób pobierania liczb,
2) zamienić wczytywanie danych stałym zbiorem liczb na czas testowania programu. wielokrotne wpisywanie może być nużące,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 20

Napisz program, który ciąg znaków podany przez użytkownika posortuje przez wybór. Wyświetl kolejne etapy porządkowania.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować zademonstrowanie ciągu znaków po każdej iteracji,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 21

Napisz program, który ciąg znaków podany przez użytkownika posortuje metodą bąbelkową. Wyświetl kolejne etapy porządkowania.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować zademonstrowanie ciągu znaków po każdej iteracji,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 22

Napisz program, który ciąg znaków podany przez użytkownika posortuje metodą quick-sort. Wyświetl kolejne etapy porządkowania.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować zademonstrowanie ciągu po każdej iteracji,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 23

Napisz program, który w uporządkowanym zbiorze liczb podanym przez użytkownika znajdzie zadaną liczbę. Porównaj metodę sekwencyjną z metodą dziel i rządź.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować pokazanie liczby kroków dla każdej z metod,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 24

Napisz program, który obliczy wartość wielomianu stopnia najwyżej dziewiątego metodą Hornera. Porównaj ilość operacji z metodą tradycyjną (ze wzoru).
Zbadaj dla x=-1, x=0 i x=1.

Sposób wykonania ćwiczenia

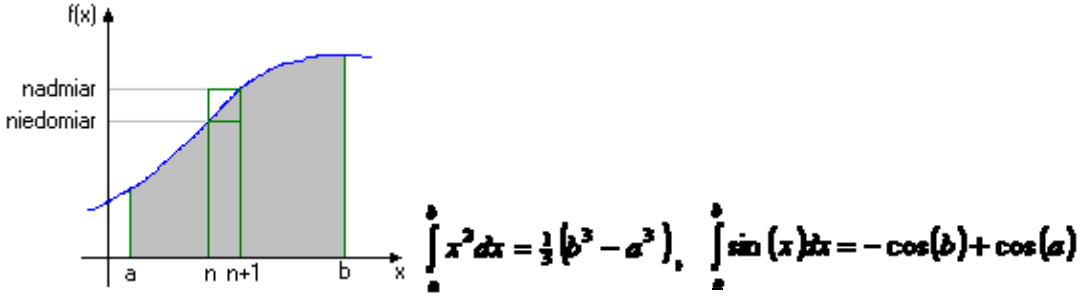
Aby wykonać ćwiczenie powinieneś:
1) zaplanować pokazanie liczby kroków dla każdej z metod,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 25

Napisz program, który obliczy powierzchnię pod krzywą w zadanym przedziale wartości x metodą przybliżoną za pomocą prostokątów z zadaną ilością przedziałów. Sprawdź na funkcji stałej, liniowej, kwadratowej i sinusoidalnej.

Wskazówka:



Rysunek 1. Obliczanie powierzchni pod krzywą

Powierzchnię można aproksymować prostokątami. Warianty z prostokątami budowanymi na początku (n) i na końcu przedziału (n+1) można obliczyć w jednym przebiegu pętli. Warto porównać wyniki dla różnych ilości przedziałów. Funkcja stała i liniowa pozwolą sprawdzić poprawność obliczeń.

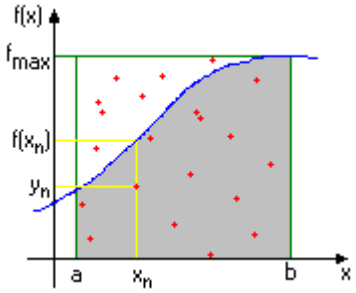
Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować pokazanie wyniku dla różnej liczby przedziałów,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 26

Napisz program, który obliczy powierzchnię pod krzywą metodą MonteCarlo. Sprawdź na funkcji stałej, liniowej, kwadratowej i sinusoidalnej.
Wskazówka:



Rysunek 2. Obliczanie powierzchni pod krzywą metodą MonteCarlo

Powierzchnię oblicza się jako stosunek trafień do ilości rzutów (losowań), gdzie rzuty obejmują obszar o znanej powierzchni (np. prostokąt), a trafienia – obszar, którego powierzchnię należy określić. Punkt trafia na powierzchnię, jeżeli $y<f(x_n)$.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować pokazanie wyniku dla różnej liczby losowań,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 27

Napisz program, który zaszyfruje i odszyfruje krótki tekst wprowadzony z klawiatury za pomocą szyfru Cezara. Klucz (przesunięcie) niech będzie wprowadzany przed szyfrowaniem.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować wykorzystanie zaszyfrowanego tekstu jako danych wejściowych do rozszyfrowania – sposób na sprawdzenie poprawności,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 28

Napisz program, który znajdzie liczby pierwsze (mniejsze niż granica podana przez użytkownika) metodą sita Eratostenesa.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zastanowić się lub znaleźć w dostępnych źródłach, do jakiej liczby należy wykonywać sprawdzanie,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program,
5) porównać metody zastosowane przez innych uczniów i oszacować, która metoda w najmniejszym stopniu zaangażuje pamięć.

Wyposażenie stanowiska pracy:

– komputer z przeglądarką i zainstalowanym kompilatorem języka programowania, edytor nieformatujący.

4.3.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) zdefiniować programowanie strukturalne?	☐	☐
2) budować złożone warunki ograniczające pętle?	☐	☐
3) lokalizować błędy na podstawie komunikatów?	☐	☐
4) zbudować procedurę?	☐	☐
5) opisać metodę rekurencyjną?	☐	☐
6) zapisywać dane do tablicy?	☐	☐
7) odczytać informacje z tablicy dwuwymiarowej?	☐	☐
8) wymienić metody sortowania?	☐	☐
9) obliczyć n-ty wyraz ciągu Fibonacciego?	☐	☐
10) pobierać dane od użytkownika?	☐	☐
11) skierować informacje na ekran monitora?	☐	☐
12) zastosować instrukcję wyboru?	☐	☐
13) zbudować algorytm Euklidesa?	☐	☐
14) napisać kod obliczający silnię?	☐	☐
15) zapisywać liczby w różnych systemach?	☐	☐
16) zakończyć obliczenia przybliżone po osiągnięciu zadanej dokładności?	☐	☐
17) wyszukać wartość metodą binarną?	☐	☐
18) obliczyć wartość wielomianu metodą Hornera?	☐	☐

4.4. Typy strukturalne

4.4.1. Materiał nauczania

Każdy język programowania umożliwia definiowanie zmiennych wielu różnych typów. Jeżeli połączyć tę różnorodność z tablicami, które mogą przechowywać dane pozostałych typów, otrzymuje się bogate narzędzie do przetwarzania danych. Istnieją także inne typy danych o specyficznych zastosowaniach.

W zależności od języka programowania można korzystać z następujących typów:
- tablice (zob. uwagi w 4.3.),
- struktury (C), rekordy (Pascal),
- unie,
- zbiory,
- pliki.

Tablica nie musi mieć rozmiaru określonego w kodzie programu. Istnieje możliwość tworzenia tablic w trakcie realizacji kodu. Można je także usuwać przed zakończeniem działania programu.

Rekordem nazywa się typ danych składający się ze skończonej liczby danych innych (prostych) typów nazywanych polami. Deklaracja polega na nadaniu nazwy rekordowi oraz wymienieniu nazw i typów zmiennych wchodzących w jego skład. Zapisywanie i odczytywanie wartości rekordu odbywa się poprzez konstrukcję:

nazwarekordu-separator-nazwaskładowej=wartość-typu-składowej

Zwykle separatorem jest kropka. Odwołanie do pól rekordu odbywa się analogicznie. Po nazwie rekordu i kropce podaje się nazwę pola.

Unią nazywa się zmienną, która może zawierać pola różnych typów i rozmiarów w różnych momentach. Jej rozmiar musi zapewnić przechowanie najobszerniejszej reprezentacji składowych. Deklarowanie odbywa się podobnie, a korzystanie jest analogiczne jak dla rekordu.

Zbiór jest zmienną składającą się z typów porządkowych. Jest to zbiór w sensie teoriomnogościowym wszystkich możliwych podzbiorów wartości typu porządkowego, włączając w to zbiór pusty. Na zbiorach można wykonywać następujące operacje:
- dodawanie,
- odejmowanie,
- mnożenie (iloczyn zbiorów),
- sprawdzanie przynależności (IN),
- porównywanie,
- sprawdzanie relacji inkluzji zbiorów (<=,>=).

Dotychczas przedstawione typy danych odnoszą się do danych przechowanych w pamięci operacyjnej komputera. Dostęp do danych przechowywanych w pamięciach masowych takich jak dyski możliwy jest poprzez zmienne typu plikowego. Typ plikowy określa zbiór danych sekwencyjnych o zmiennej liczbie elementów tego samego typu. Pozwala pracować z plikami na nośniku zewnętrznym. Praca ze zmienną typu plikowego obejmuje następujące instrukcje:
- deklarowanie typu,
- skojarzenie z plikiem,
- otwarcie w jednym z trybów (odczyt, zapis, dołączenie),
- czytanie lub modyfikowanie,
- zamknięcie skojarzenia.

Istnieje możliwość odczytywania zarówno pojedynczych znaków, jak i całych wierszy. Korzystanie z plików powinno odbywać się pod kontrolą poprawności wykonania operacji.

4.4.2. Pytania sprawdzające

Odpowiadając na pytania sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.
1) Jak definiuje się w programie typ tablicowy?
2) W jaki sposób czyta się wartości pól rekordu?
3) Jaka jest różnica pomiędzy rekordem, a unią?
4) Jakie operacje na zbiorach można wykonywać w programie?
5) W jaki sposób w programie korzysta się z plików?

4.4.3. Ćwiczenia

Ćwiczenie 1

W klasie jest 32 uczniów. Każdy może otrzymać maksymalnie 8 ocen. Zbuduj narzędzie do przechowywania ocen cząstkowych. Wprowadź dane próbne. Oblicz średnią ocenę ucznia wybranego poprzez numer kolejny.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) dokonać analizy możliwości przechowywania danych, o których mowa w ćwiczeniu,
2) zapisać na kartce, jakie dane zostaną wprowadzone. Ułatwi to sprawdzenie poprawności działania programu. Programy bardzo często wyświetlają dane. Nie zawsze są to te dane, o które chodzi,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 2

Zbuduj i zachowaj w pamięci heksadecymalną tabliczkę mnożenia. Wyświetl rezultat na ekranie.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) rozważyć różne możliwości przechowywania danych,
2) zwrócić uwagę na symetrię zagadnienia i zastanowić się, czy ten fakt może wpłynąć na szybkość obliczeń,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 3

Do testowania programów sortujących dane będą potrzebne ciągi liczbowe różnej długości. Napisz program, który wylosuje długość ciągu (liczba z przedziału od 0 do 9), a następnie utworzy tablicę o odpowiednim rozmiarze i wypełni ją losowymi liczbami.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować utworzenie i usunięcie tablicy z pamięci,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 4

Przechowaj w pamięci imię, datę urodzenia i wiek kilku osób. Wyświetl dane wybranej osoby.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować odpowiedni typ danych,
2) zaplanować sposób sprawdzenia działania programu,
3) naszkicować algorytm wykonania zadania,
4) wpisać kod, skompilować program i poprawić ewentualne błędy,
5) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 5

Oblicz średni wzrost osób z poprzedniego ćwiczenia.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 6

Sprawdź, ile znaków w zdaniu podanym przez użytkownika należy do 26-elementowego alfabetu łacińskiego.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować sposób sprawdzenia,
2) naszkicować algorytm wykonania zadania,
3) wpisać kod, skompilować program i poprawić ewentualne błędy,
4) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 7

Dopisz do pliku w jednym wierszu 6 liczb podanych przez użytkownika. Dokonaj kilku wpisów.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program. Wynik działania można sprawdzić podglądając plik zapisany na dysku. Sprawdza się w ten sposób zarówno istnienie pliku, jak i jego zawartość.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 8

Pomnóż wszystkie dane w pliku przez 10.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania uwzględniając pobranie danych z pliku i zapisanie wyniku do innego pliku,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 9

Wyświetl na ekranie 6 liczb z drugiego wiersza pliku tekstowego, który utworzyłeś w poprzednim ćwiczeniu..

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) naszkicować algorytm wykonania zadania,
2) wpisać kod, skompilować program i poprawić ewentualne błędy,
3) uruchomić i przetestować program.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

4.4.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) budować tablice wielowymiarowe?	☐	☐
2) tworzyć tablice w trakcie wykonywania kodu?	☐	☐
3) zdefiniować rekord?	☐	☐
4) przechować dane w zmiennej rekordowej?	☐	☐
5) zapisać dane do pliku na dysku?	☐	☐
6) odczytać dane z pliku na dysku?	☐	☐

4.5. Interfejs graficzny

4.5.1. Materiał nauczania

Programowanie w systemie graficznym ma charakter zdarzeniowy. Programista nie musi poświęcać uwagi tworzeniu obiektów interfejsu graficznego. Korzysta z gotowych kontrolek określając ich właściwości i programując zdarzenia.

Właściwości poszczególnych obiektów są uzależnione od ich przeznaczenia. Najczęściej określa się:

1) nazwę,
2) rozmiar,
3) położenie,
4) kolory,
5) opis (tekst, czcionka),
6) widoczność.

Każdy obiekt może mieć zaprogramowane różne, charakterystyczne zdarzenia. Najczęściej są to:

1) kliknięcie,
2) zmiana wartości,
3) zdarzenia związane z fokusem,
4) zdarzenia związane z przeciąganiem.

Na formie (formularzu) można umieszczać wszystkie dostępne kontrolki znane ze środowiska okienkowego. Najczęściej są to:

1) etykieta,
2) pole tekstowe,
3) ramka,
4) przycisk,
5) pole wyboru,
6) przycisk opcji,
7) lista,
8) obiekt rysunkowy,
9) lista obiektów dyskowych,
10) tablica,
11) menu,
12) obraz,
13) zegar (niewidoczny).

Środowisko graficzne pozwala też na wygodne umieszczanie i modyfikowanie obiektów. Większość operacji można wykonać metodą „przeciągnij i upuść” ze wspomaganiem w postaci siatki. Projektant może jednak określać położenia i rozmiary obiektów wpisując odpowiednie wartości liczbowe.

Duże znaczenia dla estetycznego odbioru aplikacji ma rozmieszczenie obiektów. Można je modyfikować i przemieszczać grupowo korzystając z klawiszy rozszerzających (shift, alt, ctrl).

Znacznym ułatwieniem dla programisty jest również rozbudowany system debugowania. Oferuje następujące usprawnienia:

1) obserwowanie wartości zmiennych,
2) wykonywanie kodu krok po kroku,
3) rozpoczęcie debugowania od wybranego miejsca,
4) ustawianie punktów zatrzymania wykonywania kodu,
5) wejście do podprogramu.

Projektant może określać zachowanie całego formularza takie jak:
1) wyświetlanie w określonym miejscu ekranu,
2) możliwość zmiany rozmiaru,
3) położenie „zawsze na wierzchu”,
4) relacje z formularzami podrzędnymi,
5) zakończenie pracy.

Programista może tworzyć i dodawać nowe obiekty (kontrolki).

Programowanie w środowisku graficznym pozwala szybko uzyskiwać zadowalające rozwiązania graficzne. Aby wykorzystać w pełni możliwości narzędzia potrzebna jest znajomość programowania.

4.5.2. Pytania sprawdzające

Odpowiadając na pytania sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.
1) Z jakich elementów może składać się środowisko programistyczne?
2) Co daje korzystanie z siatki na formie?
3) W jaki sposób można przesuwać i wyrównywać obiekty?
4) Wymień kilka charakterystycznych właściwości:
   - przycisku
   - pola tekstowego
   - pola wyboru
5) Wymień zdarzenia charakterystyczne dla:
   - przycisku
   - listy
   - obiektu graficznego
6) Jakie narzędzia oferuje debugger?
7) W jaki sposób można zablokować możliwość zmiany rozmiaru formy przez użytkownika?
8) Kiedy przydatna jest opcja „zawsze na wierzchu”?

4.5.3. Ćwiczenia

Ćwiczenie 1

Umieść na formie napis zawierający imię i nazwisko według specyfikacji: czcionka Arial 32 pt, kolor niebieski, centralnie w poziomie.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować parametry formy,
2) zaplanować rozmieszczenie kontrolek stosując jednostki formy lub określony typ wyrównania i zapisać w dokumentacji projektu,
3) ustawić atrybuty wszystkich obiektów,
4) uruchomić kompilator i usunąć ewentualne błędy,
5) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 2

Umieść na formie z ćwiczenia pierwszego przycisk, którego naciśnięcie przesunie napis o 10 pikseli w prawo.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować miejsce umieszczenia przycisku oraz jego wygląd i zapisać w dokumentacji projektu,
2) ustawić atrybuty nowego obiektu,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 3

Umieść na formie z ćwiczenia drugiego dodatkowy przycisk, którego naciśnięcie zmieni kolor napisu w sekwencji: niebieski, czerwony, zielony.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować miejsce umieszczenia przycisku, ewentualnie uwzględnić zmianę parametrów formy i zapisać w dokumentacji projektu,
2) zaplanować sposób cyklicznej zmiany koloru,
3) ustawić atrybuty obiektów,
4) uruchomić kompilator i usunąć ewentualne błędy,
5) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 4

Umieść na formie dowolną grafikę według specyfikacji: rozmiar 4×5 cm, odstęp od lewego brzegu 1,2 cm, odstęp od góry formy 0,5 cm. Dodaj przycisk powiększający obraz dwukrotnie.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować rozmieszczenie obiektów i zapisać w dokumentacji projektu,
2) ustawić atrybuty obiektów,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 5

Zbuduj na formie sygnalizator świetlny. Zmieniaj jego stan przyciskami „Stój", „Uwaga", „Idź".

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować rozmieszczenie i wygląd obiektów i zapisać w dokumentacji projektu,
2) przemyśl sposób wizualizacji zmiany świateł i zapisać w dokumentacji projektu,
3) ustawić atrybuty obiektów,
4) uruchomić kompilator i usunąć ewentualne błędy,
5) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 6

Umieść na formie obiekt combo zawierający kilka słów w języku obcym. Kliknięcie dowolnego słowa powinno wyświetlić jego polski odpowiednik w polu tekstowym.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować rozmieszczenie i wygląd obiektów i zapisać w dokumentacji projektu,
2) ustawić atrybuty obiektów,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 7

Umieść na formie krótki wiersz oraz przyciski radiowe pozwalające wyświetlić tekst za pomocą kilku różnych czcionek.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować rozmieszczenie i wygląd obiektów i zapisać w dokumentacji projektu,
2) ustawić atrybuty obiektów,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 8

Umieść na formie obiekt geometryczny odbijający się od lewej i prawej krawędzi formy.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) przemyśleć sposób zmiany kierunku ruchu obiektu, zapisać w postaci schematu blokowego i dołączyć do dokumentacji projektu,
2) zaplanować rozmieszczenie i wygląd obiektów i zapisać w dokumentacji projektu,
3) ustawić atrybuty obiektów,
4) uruchomić kompilator i usunąć ewentualne błędy,
5) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:

– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 9

Narysuj wykres funkcji drugiego stopnia w taki sposób, by widoczne były oba ramiona paraboli.

Wskazówka

Istnieją obiekty, na których można rysować proste obiekty geometryczne za pomocą wbudowanych funkcji. Aby narysować krzywą zmieniaj wartość zmiennej niezależnej o stałą wartość i obliczaj za każdym razem wartość zmiennej zależnej. Krzywa może powstać przez rysowanie punktów, prostokątów o bardzo krótkim boku lub linii prowadzonych od poprzedniego miejsca do właśnie obliczonego.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zaplanować rozmieszczenie i wygląd obiektów i zapisać w dokumentacji projektu,
2) ustawić atrybuty obiektów,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:

– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 10

Zmodyfikuj formę z ćwiczenia 9 w taki sposób, by użytkownik mógł wybierać zakres wartości zmiennej niezależnej, dla których chce oglądać wykres.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zaplanować rozmieszczenie i wygląd nowych obiektów i zapisać w dokumentacji projektu,
2) ustawić atrybuty obiektów,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 11

Zmodyfikuj formę z ćwiczenia 10 w taki sposób, by użytkownik mógł wybrać jedną z kilku zaproponowanych funkcji.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować rozmieszczenie i wygląd nowych obiektów i zapisać w dokumentacji projektu,
2) ustawić atrybuty obiektów,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

Ćwiczenie 12

Rozwiąż równanie $y=x^3+8$ metodą bisekcji.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować rozmieszczenie i wygląd obiektów i zapisać w dokumentacji projektu,
2) ustawić atrybuty obiektów,
3) uruchomić kompilator i usunąć ewentualne błędy,
4) sprawdzić działanie aplikacji.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym graficznym środowiskiem projektowym.

4.5.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) ustawić rozmiar i położenie formy na ekranie?	☐	☐
2) umieścić napis we wskazanym miejscu formy?	☐	☐
3) sterować właściwościami obiektów za pomocą przycisku?	☐	☐
4) umieścić obiekt graficzny o określonym rozmiarze?	☐	☐
5) ukrywać i przywracać obiekty?	☐	☐
6) określić, która pozycja z listy została wybrana?	☐	☐
7) utworzyć grupę przycisków radiowych?	☐	☐
8) animować obiekt na formie?	☐	☐
9) rysować punkty i linie na formie?	☐	☐

4.6. Struktury dynamiczne

4.6.1. Materiał nauczania

Analityczne podejście do typów zmiennych pozwala wyliczyć cztery typy podstawowe:

1) liczby całkowite,
2) liczby rzeczywiste,
3) wartości logiczne,
4) wewnętrzne reprezentacje znaków pisarskich i specjalnych.

Poza typami podstawowymi definiuje się typy złożone, które można wykorzystać w zastosowaniach specjalistycznych.

Stos jest strukturą liniowo uporządkowanych danych, z których jedynie ostatni element, zwany *wierzchołkiem*, jest w danym momencie dostępny. W wierzchołku odbywa się dołączanie nowych elementów. Jedynie wierzchołek można usunąć. Obsługuje dwie podstawowe operacje: odłóż (*push*) i pobierz (*pop*). Zasada działania stosu jest zatem następująca: ostatni wszedł, pierwszy wyszedł LIFO (*Last-In-First-Out*).

Stosowi musi towarzyszyć dodatkowa zmienna będąca jego licznikiem, która określa dokładny adres pamięci pod który należy odkładać i z którego należy pobierać dane

Stos jest dość często wykorzystywaną strukturą danych – na jego obsłudze opiera się działanie procesora. Działanie na nim jest często porównywane do stosu talerzy: nie można usunąć talerza znajdującego się na dnie stosu nie usuwając wcześniej wszystkich innych. Nie można także dodać nowego talerza gdzie indziej, niż na samą górę.

Kolejka jest strukturą liniowo uporządkowanych danych w której dołączać nowe dane można jedynie na koniec kolejki a usuwać z początku. Kolejka obsługuje dwie podstawowe operacje: dołącz i obsłuż. Operacja dołącz dopisuje do kolejki pojedynczy element; operacja obsłuż usuwa z kolejki element, który był do niej dołączony jako pierwszy. Zasada działania kolejki jest zatem następująca: pierwszy wszedł, pierwszy wyszedł FIFO (*First-In-First-Out*). Procedura usunięcia danych z końca kolejki jest taka sama, jak w przypadku stosu, z tą różnicą, że usuwamy dane od początku a nie od końca.

Pierwszy element (a dokładniej wskaźnik do jego miejsca w pamięci) musi zostać zapamiętany, by możliwe było usuwanie pierwszego elementu w czasie stałym O(1).W przeciwnym razie, aby dotrzeć do pierwszego elementu należałoby przejść wszystkie od elementu aktualnego (czyli ostatniego), co wymaga czasu O(n).

Działanie na kolejce jest intuicyjnie jasne, gdy skojarzymy ją z kolejką ludzi np. w sklepie. Każdy nowy klient staje na jej końcu, obsługa odbywa się jedynie na początku.

Lista jest to liniowo uporządkowany zbiór elementów, z których dowolny element można usunąć oraz dodać w dowolnym miejscu. Pierwszy i ostatni element listy nazywamy *końcami listy*. Często pierwszy element listy nazywany jest głową. Szczególnym przypadkiem listy może być stos (gdy pobrać, odczytać i wstawić element można tylko na końcu listy) lub kolejka (pobrać i odczytać element można tylko na początku listy, a dodać na końcu).

Listy pozwalają na znacznie efektywniejsze wykorzystywanie pamięci: zużywa się jej dokładnie tyle ile w danej chwili potrzeba, a nie tyle, ile byłoby potrzebne w najgorszym razie. Przykładowo, jeżeli na liście obecności znajduje się pięć osób, a maksymalnie mogłoby być ich dwadzieścia, to chcąc zapisać ją w tablicy należałoby zarezerwować pamięć dla 20 wpisów; zapisując ją na liście wystarczy 5 rzeczywiście wykorzystanych elementów.

W liście jednokierunkowej każdy element zna adres następnego elementu, o ile ten istnieje. Elementami listy muszą więc być struktury zawierające pole wskazujące na następny element. W przypadku braku wystarczającej ilości pamięci nowy element nie zostanie utworzony. Trzeba koniecznie korzystać z metod obsługi wyjątków.

Podstawowymi operacjami, które wykonuje się na liście jednokierunkowej są wyszukanie elementu, wstawienie nowego elementu i usunięcie wybranego elementu.

Aby dodać element do jednokierunkowej listy posortowanej należy sprawdzić, w którym miejscu powinien się on znajdować. Sprawdza się od korzenia schodząc w dół. Jeśli element jest większy od badanego węzła i mniejszy od jego następnika, to należy umieścić go między nimi. Należy więc ustawić wskaźnik aktualnego węzła na dodawany element, a wskaźnik tego elementu na następnik. Ponieważ jest to lista jednokierunkowa, przeszukiwanie jej należy zawsze zaczynać od korzenia. Dodając więc pierwszy element do pustej listy należy zapamiętać jego wskaźnik, by później móc się do niego przenieść.

W przeciwieństwie do stosu i kolejki listy mogą zawierać dwa wskaźniki. Takie listy nazywa się *dwukierunkowymi.* Można się po nich poruszać w obydwu kierunkach, co przyspiesza wszystkie operacje.

Listę nazywa się cykliczną, jeżeli ostatni jej element powiązany jest z pierwszym. Lista taka nie posiada zatem ani początku, ani końca, i można po niej poruszać się w nieskończoność.

Drzewo jest bardziej skomplikowaną strukturą niż poprzednie. Dla każdego drzewa wyróżniony jest jeden, charakterystyczny element – *korzeń*. Korzeń jest jedynym elementem drzewa, który nie posiada elementów poprzednich. Dla każdego innego elementu określony jest dokładnie jeden element poprzedni. Dla każdego elementu oprócz ostatnich, tzw. *liści* istnieją co najmniej 2 elementy następne. Pojedynczy element drzewa nazywa się węzłem (gałęzią). Węzeł następujący po innym węźle nazywany jest jego potomkiem (dzieckiem). Węzeł poprzedzający inny węzeł nazywany jest jego przodkiem (rodzicem). Jeżeli liczba następnych elementów wynosi dokładnie 2, to drzewo nazywa się *binarnym*. Drzewo można zdefiniować, jako acykliczny graf.

Dostęp do danych przechowywanych w postaci drzewa binarnego jest znacznie szybszy, niż w przypadku listy. Dostęp do danych znajdujących się na n-tym poziomie drzewa wymaga sprawdzenia n węzłów. Na n-tym poziomie drzewa binarnego można umieścić 2^n elementów. Dla każdego drzewa można określić:

1) długość drogi u (głębokość) – liczba wierzchołków, przez które należy przejść od korzenia do wierzchołka u,
2) wysokość u – maksymalna liczba wierzchołków na drodze od u do pewnego liścia,
3) wysokość drzewa = głębokość = wysokość korzenia +1,
4) ścieżka z u do v – zbiór wierzchołków, przez które należy przejść z wierzchołka u do v,
5) droga – ścieżka skierowana,
6) stopień wierzchołka – liczba jego bezpośrednich następników,
7) stopień drzewa – maksymalny stopień wierzchołka.

Do struktur dynamicznych zalicza się również tablice. Można w trakcie realizacji programu określać ich rozmiar. Były już wykorzystane w rozdziale 4.4.

4.6.2. Pytania sprawdzające

Odpowiadając na pytania sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.

1) Co to jest stos?
2) Jakie operacje wykonuje się na stosie?
3) Co to jest kolejka?
4) Jakie operacje wykonuje się na kolejce?
5) Co to jest lista?
6) Jakie operacje wykonuje się na liście?
7) Co to jest drzewo?
8) Podaj zasady dynamicznego tworzenia tabel

4.6.3. Ćwiczenia

Ćwiczenie 1

Utwórz listę jednokierunkową i wypełnij ją kilkoma danymi. Nowe dane umieszczaj na początku kolejki

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować realizację zadania,
2) zdefiniować strukturę elementu listy pamiętając o adresie,
3) ustawić wskaźnik na pusty adres,
4) napisać kod dopisujący nowe wartości uwzględniając obsługę błędów,
5) wpisać kod i skompilować,
6) uruchomić i przetestować aplikację.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 2

Wyświetl na ekranie elementy listy utworzone w ćwiczeniu 1.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować realizację zadania uwzględniając zapisanie kodu w postaci procedury,
2) wpisać kod i skompilować,
3) uruchomić i przetestować aplikację.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 3

Zmodyfikuj wybrany element utworzonej w ćwiczeniu 1 listy i wyświetl na ekranie rezultat działania programu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) zaplanować realizację zadania,
2) wpisać kod i skompilować,
3) uruchomić i przetestować aplikację.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

Ćwiczenie 4

Usuń wybrany element utworzonej w ćwiczeniu 1 listy i wyświetl na ekranie rezultat działania programu.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:

1) zaplanować realizację zadania,
2) wpisać kod i skompilować,
3) uruchomić i przetestować aplikację.

Wyposażenie stanowiska pracy:

– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

4.6.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) utworzyć listę?	☐	☐
2) korzystać ze struktur obsługi wyjątków?	☐	☐
3) dopisać element listy?	☐	☐
4) zmodyfikować wybrany element listy?	☐	☐
5) usunąć wskazany element listy?	☐	☐

4.7. Sterowanie kompilacją

4.7.1. Materiał nauczania

Dyrektywa kompilatora to polecenie o charakterystycznej składni, narzucające kompilatorowi sposób, w jaki powinien dokonać kompilacji programu, jeżeli nie jest to sposób domyślny. Dyrektywy można podzielić na:
1) przełączające (włączają lub wyłączają określony sposób kompilacji, mogą mieć zasięg globalny lub lokalny),
2) parametryczne (specyfikują parametry kompilacji, np. nazwa pliku, rozmiar pamięci),
3) warunkowe.

W zależności od rodzaju środowiska programistycznego, programista może mieć możliwość korzystania z następujących narzędzi:
1) dołączanie wskazanego pliku. Plik zawiera zwykle narzędzia poszerzające funkcjonalność wykorzystywanego narzędzia,
2) definiowanie zmiennej, która będzie miała zasięg globalny,
3) umieszczenie bloku dyrektywy warunkowej. Pozwala sprawdzić dla jakiego systemu program jest kompilowany,
4) określenie sposobu przechowywania zmiennych w pamięci, co może przyspieszyć działanie programu,
5) określenie sposobu sprawdzania złożonych warunków logicznych, co może wpłynąć na szybkość wykonywania programu,
6) dołączenie do kodu informacji dla debuggera,
7) określenie zachowania programu w przypadku wystąpienia błędu lub wyjątku.

Z dyrektywami kompilatora można zapoznać się w systemie pomocy używanego środowiska programistycznego. Niektóre środowiska umożliwiają za pomocą menu dostęp do narzędzia pozwalającego ustawić przynajmniej podstawowe dyrektywy kompilacji.

4.7.2. Pytania sprawdzające

Odpowiadając na pytania sprawdzisz, czy jesteś przygotowany do wykonania ćwiczeń.
1) Co to są dyrektywy kompilatora?
2) Wymień metody, które mogą przyspieszyć działanie programu.
3) Podaj przykłady w jaki sposób programista może wpływać na proces kompilacji.

4.7.3. Ćwiczenia

Ćwiczenie 1

Wyszukaj informacje na temat dyrektyw kompilatora, które możesz wykorzystać w swoim środowisku programistycznym

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) wyszukać informacje w wybranym źródle (strony www, dokumentacja techniczna),
2) sporządzić notatkę,
3) na podstawie wyszukanych informacji usprawnić pracę programisty.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i wyszukiwarką.

Ćwiczenie 2

Wykorzystaj dyrektywę decydującą o sposobie traktowania wyjątków. Wygeneruj odpowiedni wyjątek w celu sprawdzenia zmiany zachowania aplikacji.

Sposób wykonania ćwiczenia

Aby wykonać ćwiczenie powinieneś:
1) ustawić dyrektywę tak, by pozwalała na obsługę wyjątków,
2) doprowadzić do powstania błędu, aby sprawdzić zachowanie aplikacji,
3) sformułować wniosek.

Wyposażenie stanowiska pracy:
– komputer z zainstalowanym kompilatorem języka programowania i edytor nieformatujący.

4.7.4. Sprawdzian postępów

Czy potrafisz:	**Tak**	**Nie**
1) wyszukać informacje o dyrektywach kompilatora?	☐	☐
2) zastosować dyrektywę obsługi wyjątków?	☐	☐

5. SPRAWDZIAN OSIĄGNIĘĆ

INSTRUKCJA DLA UCZNIA

1. Przeczytaj uważnie instrukcję.
2. Podpisz imieniem i nazwiskiem kartę odpowiedzi.
3. Zapoznaj się z zestawem pytań testowych.
4. Test zawiera 14 pytań testowych. Do każdego pytania dołączone są cztery możliwości odpowiedzi. Tylko jedna odpowiedź jest prawidłowa.
5. Udzielaj odpowiedzi wyłącznie na załączonej karcie odpowiedzi, stawiając w odpowiedniej rubryce znak **×.** W przypadku pomyłki należy błędną odpowiedź zaznaczyć kółkiem, a następnie ponownie zakreślić odpowiedź prawidłową.
6. Pracuj samodzielnie, bo tylko wtedy będziesz miał satysfakcję z wykonanego zadania..
7. Jeżeli udzielenie odpowiedzi będzie Ci sprawiało trudność, wtedy najlepiej odłóż jego rozwiązanie na później. Wróć do niego, gdy zostanie Ci wolny czas.
8. Na rozwiązanie testu masz 45 minut.

ZESTAW PYTAŃ TESTOWYCH

1) W przypadku kilkunastu liczb najdłużej będzie wykonywany algorytm o złożoności:
   a) O(n)
   b) $O(n^2)$
   c) O(nlogn)
   d) O(2n)

2) Obliczasz średni wzrost uczniów. Program pobiera wzrost każdego ucznia. Dobierz optymalny typ zmiennej przechowującej dane.
   a) całkowity, 1 bajt
   b) całkowity, 2 bajty
   c) całkowity, 3 bajty
   d) tekstowy

3) Warunek
   NOT(i<5) AND (j>=–5)
   jest prawdziwy dla:
   a) i=1, j=–1
   b) i=1, j=–10
   c) i=10, j=–1
   d) i=10, j=–10

4) W pewnym kodzie pętla zewnętrzna wykonywana jest dla indeksu i z przedziału od 0 do 9. Pętla wewnętrzna wykonywana jest dla indeksu j od wartości indeksu i do 9. Pętla wewnętrzna inkrementuje wyzerowaną początkowo zmienną suma. Wartość zmiennej suma wyniesie:
   a) 10
   b) 25
   c) 55
   d) 100

5) Wskaż ułamek binarny o wartości dziesiętnej 0,3125:
   a) 0.10101
   b) 0.01010
   c) 0.01100
   d) 0.01001

6) Dany jest ciąg 10–elementowy posortowany rosnąco. Użyto go jako dane wejściowe do optymalnego programu sortującego metodą bąbelkową. Ile standardowych operacji wykona program?
   a) zero
   b) 9
   c) 10
   d) 45

7) Dany jest wielomian x^4+x^2-3. Obliczana jest wartość wielomianu dla x=2 metodą Hornera. Ile wykonano operacji?
   a) 4 mnożenia i 2 dodawania
   b) 3 mnożenia i 2 dodawania
   c) 2 mnożenia i 2 dodawania
   d) 2 mnożenia i 1 dodawanie

8) Tabela numerowana od 1 służąca do znalezienia liczb pierwszych metodą Eratostenesa zawiera następujące wartości:

   1 1 1 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1 0 0 0 1 0 1

   Najbliższe sprawdzanie zostanie wykonane dla liczby:
   a) 3
   b) 5
   c) 7
   d) 9

9) Wyniki testowych obliczeń powierzchni metodą MonteCarlo przedstawia tabela:

losowane x	losowane y	obliczone y
1	50	3
2	22	5
5	1	27
1	81	3
9	12	83
7	60	51
3	32	11
5	34	27
4	6	18
4	49	18

Losowano wartości z przedziału od 0 do 9. Krzywa przyjmuje w tym przedziale wartości od 2 do 83. Powierzchnia pod krzywą wynosi więc (w przybliżeniu):

a) 83
b) 224
c) 523
d) 747

10) Forma ma szerokość 600px. Narysowano na niej centralnie prostokąt o szerokości 100px. Polecenie przesuwające prostokąt o wartość 400px w prawo spowoduje, że:
    a) szerokość formy zmieni się,
    b) prostokąt częściowo ukryje się za brzegiem formy,
    c) prostokąt przestanie być widoczny dla użytkownika,
    d) wygenerowany zostanie błąd.

11) Dana jest tablica dwuwymiarowa:

```
1 2 3 4 5
2 3 4 5 6
3 4 5 6 7
4 5 6 7 8
5 6 7 8 9
```

    Które polecenie wyzeruje wszystkie wartości na przekątnej?
    a) wiersz=i, kolumna=0
    b) wiersz=0, kolumna=i
    c) wiersz=0, kolumna=0
    d) wiersz=i, kolumna=4–i

12) Usuwając jedyną wartość listy jednokierunkowej, należy:
    a) usunąć listę
    b) wpisać dowolny adres
    c) ustawić wskaźnik na NULL
    d) niczego nie należy robić

13) Program ma wyświetlić na ekranie iloraz liczb wprowadzonych przez użytkownika. Programista ustawił dokładność wyświetlania na 20 miejsc po przecinku. Użytkownik wpisał dzielną 2 i dzielnik 3. Z jakich cyfr będzie się składał wynik wyświetlony na ekranie?
    a) zero i kilka 6
    b) zero, kilka 6 i 7
    c) zero, kilka 6 i 9
    d) zero, kilka 6 i różne cyfry

14) Która operacja NIE jest dozwolona na wskaźnikach?
    a) dodanie wartości całkowitej
    b) dekrementacja
    c) przypisanie wartości NULL
    d) zmiana typu wskazywanej zmiennej

KARTA ODPOWIEDZI

Imię i nazwisko .....

Programowanie w środowisku języka strukturalnego

Zakreśl poprawną odpowiedź.

Nr zadania	Warianty odpowiedzi				Punkty
1	a	b	c	d	
2	a	b	c	d	
3	a	b	c	d	
4	a	b	c	d	
5	a	b	c	d	
6	a	b	c	d	
7	a	b	c	d	
8	a	b	c	d	
9	a	b	c	d	
10	a	b	c	d	
11	a	b	c	d	
12	a	b	c	d	
13	a	b	c	d	
14	a	b	c	d	
				Razem:	

6. LITERATURA

1. Bujanowska I., Bujanowski I, Talaga Z., Informatyka, WSzPWN 2003
2. Liberty J., C++ dla każdego, Helion 2002
3. Praktyka Programowania, Wydawnictwo Naukowo-Technicznе, Warszawa 1982
4. Shtern V., C++. Inżynieria programowania, Helion 2003
5. Stroustrup B., Język C++, Wydawnictwa Naukowo-Techniczne
6. Sysło M. M., Algorytmy, WSIP 2002
7. http://eduseek.interklasa.pl
8. http://pl.wikipedia.org