II 14.10

Konsultacje s.237:

- poniedziałek 14:30-16:00

- czwartek 14:00-15:30

1. Informatyka – przetwarzanie informacji za pomocą automatycznych środków pomocniczych
2. Informacja – fakt (zbiór faktów), które można zakomunikować
3. Programowanie – umiejętność rozwiązania pewnego zadania (ale konkretnego) za pomocą wybranego języka programowania
4. Co należy umieć, by pisać programy:
   1. Znać język programowania
   2. Umieć rozwiązywać zadania
5. Pojęcia pierwotne:
   1. Rzeczywiste
      1. Akcja (instrukcje) – zdarzenie o skończonym czasie trwania i o zamierzonym dobrze określonym skutku (efekt akcji)
      2. Obiekt (zmienna, dana)– akcja jest wykonywana na pewnym obiekcie, wynik działania akcji rozpoznawany jest po zmianach stanu tego obiektu
   2. wirtualne
      1. Instrukcja – opisujemy akcje w jakimś języku
      2. Obiekty – to są dane do programu
6. Proces – jeżeli akacja składa się w kilku części składowych to jest to proces, w procesie poszczególne akcje najczęściej muszą następować w ściśle określonej kolejność, np. pracownik musi najpierw otrzymać kartkę z wynagrodzeniem, a potem dopiero może pójść do kasy
7. Program – instrukcja opisująca proces (najczęściej kilka instrukcji)
8. Algorytm – sposób rozwiązywania problemu, który potrzebuje dwóch elementów: algorytmu i maszyny wykonującej algorytm; zadanie 🡪 zestaw danych 🡪 operacje 🡪 dane wyjściowe
   1. Własności algorytmu:
      1. Algorytm wymaga skończonej liczby reguł postępowania i może zawierać tylko pewien zbiór czynności (nie ma pętli nieskończonych)
      2. Algorytm służy do rozwiązywania problemów tej samej klasy, używane dane powinny być sparametryzowane
      3. Algorytm projektuje się dla zadań, dla których istnieje rozwiązanie, gdy rozwiązanie nie istnieje należy określić moment przerwania algorytmu
      4. Algorytm powinien uwzględniać możliwe wszystkie sposoby rozwiązania zadania.
   2. Dopuszczalne działania w algorytmach:
      1. Maszyna cyfrowa np. operacje arytmetyczne, działania trygonometryczne
9. Procesor, komputer – algorytmy są wykonywane przez automaty, gdzie najważniejszym elementem jest procesor z elementami wspomagającymi takimi jak pamięć, urządzenie wejścia/wyjścia, pamięci zewnętrzne itd.
10. Pamięć – obiekty muszą być gdzieś umieszczanie, zawartość pamięci może być zmieniana
    1. Cechy pamięci: pojemność, szybkość pobierania danych i ich zapisywania
    2. Pamięć składa się z komórek, komórka odzwierciedla pewien kod obiektu (np. kod ASCII)
11. Kodowanie znaków – ASCII (American Standard Code for Information Interchange):
    1. 0-31 i 127 – kody sterujące
    2. 32-126 – litery, cyfry I znaki drukowane
    3. 128-255 – rozszerzenie (np. polskie znaki)

III 21.10

* **Języki programowania**
  + Samo ułożenie algorytmu jest niewystarczające, ponieważ układa go człowiek a wykonuje coś innego. Z tego powodu musi być możliwość zakomunikowania co należy wykonać.
  + Algorytm pisany przez człowieka jest opisowy i nie jest możliwe wykonanie go przez urządzenie. Należy go sformalizować.
  + Algorytm jest ideą rozwiązania, dlatego zaprezentowany algorytm nie nadaje się do wykorzystania poprzez komputer, gdyż mogą wystąpić np. symbole graficzne.
  + Dlatego algorytm musi być zapisany w języku programowania, określa on jakie symbole mogą być użyte do zapisu programu.
* Podział języków programowania
  + **Języki niskiego poziomu**:
    - Obejmują one różnego rodzaju asemblery, które pozwalają zapisywać instrukcje języka maszynowego w postaci symboli zwanych mnemonikami.
    - Mnemoniki używają liczb, które mogą być zapisywane binarnie, dziesiętnie lub szesnastkowo. Składnia asemblera zależy od konkretnego typu komputera, a tekst programu musi być przetłumaczony na język maszynowy za pomocą programu zwanego asemblerem.
    - Język maszynowy składa się z ciągu cyfr 0 i 1, co eliminuje potrzebę dalszej translacji. Programy systemowe często tworzy się w asemblerze, ponieważ umożliwia to maksymalne wykorzystanie zasobów sprzętowych.
  + **Języki wysokiego poziomu**:
    - Zbliżone do naturalnych języków, pozwalają tworzyć programy przy pomocy prostych zdań połączonych wyrażeniami arytmetycznymi. Program napisany w języku wysokiego poziomu jest kompilowany, a następnie łączony z procedurami bibliotecznymi, tworząc program wynikowy. Przykłady takich języków to: C++, Java, Visual Basic.
* Generacje języków programowania  
  Generacja języka określa stopień zaawansowania jego struktury:
  + 1GL (język maszynowy): Pierwsza generacja, bezpośrednie instrukcje dla procesora w postaci cyfr binarnych.
  + 2GL (język asemblera): Symboliczne nazwy instrukcji procesora (mnemoniki), które są przekształcane na kod maszynowy.
  + 3GL (język trzeciej generacji): Proceduralne języki programowania, bardziej zrozumiałe dla użytkownika. Przykłady: C++, Turbo Pascal, Java.
  + 4GL (język czwartej generacji): Specjalistyczne języki, umożliwiające tworzenie programów za pomocą krótkich instrukcji, np. SQL. Charakteryzują się efektywnością w określonych zastosowaniach.
* Translacja, kompilacja, interpretacja
  + **Kompilacja**: Proces tłumaczenia całego programu na język maszynowy, co tworzy plik wynikowy. Zaletą kompilacji jest szybkie działanie programu, jednak każda zmiana wymaga ponownej kompilacji.
  + **Interpretacja**: Polega na tłumaczeniu i wykonywaniu programu krok po kroku, bez tworzenia pliku wynikowego. Wadą jest konieczność ponownego tłumaczenia każdej instrukcji przy każdym jej wykonaniu, co spowalnia program.
  + **Translacja**: Może oznaczać zarówno proces kompilacji, jak i interpretacji, w zależności od przyjętego podejścia.

IV 28.10

* Tworzenie programów w języku programowania:
  + Języki wysokiego poziomu przypominają proste zdania w języku naturalnym połączone wyrażeniami arytmetycznymi z matematyki.
  + Napisany program zostaje skompilowany
  + W fazie łączenia łączony zostaje tekst programu z procedurami bibliotecznymi, realizują to linkery lub konsolidatory.
  + Efektem końcowym jest program wynikowy, może być zapisany na dysku i uruchamiany przez system operacyjny.
  + Języki wysokiego poziomu: Turbo Pascal, C i C++, Java, Visual Basic i inne.
* Etapy rozwiązywania zadań:
  + Sformułowanie zadania z wyróżnieniem informacji wejściowych i wyjściowych
  + Opracowanie algorytmu – kroków rozwiązania zadania
  + Sprawdzenie działania algorytmu
  + Opracowanie kodu programu według zasad i symboliki narzuconej przez konkretny język programowania.
  + Sprawdzenie poprawności działania programu, skompilowanie programu i przetestowanie. W przypadku niewłaściwych wyników – powtórzyć wszystkie kroki.
* Schematy blokowe
  + Poglądowa forma graficznego przedstawienia algorytmu, stworzony ze ściśle określonego zbioru figur geometrycznych i ustalonych form łączenia. We wnętrzu bloków w umowny sposób zapisuje się występujące w algorytmie operacje arytmetyczne, logiczne, operacje wejścia i wyjścia oraz warunki, od których należą decyzje co do kolejności wykonywania obliczeń. Polski zbiór --PN-75E-01226.
  + Elementy bloków:

|  |  |
| --- | --- |
| Początek lub koniec |  |
| Przetwarzanie – grupa operacji, w wyniku których ulega zmianie wartość, postać lub miejsce zapisu danych |  |
| Operacje wejścia i wyjścia - wprowadzanie danych lub wyprowadzanie wyników |  |
| Decyzja – operacja określająca wybór jednej z alternatywnych dróg działania |  |
| Proces zdefiniowany (podprogram) – proces (ciąg instrukcji) zdefiniowany poza programem |  |
| Łącznik stronnicowy – wejście lub wyjście z wyodrębnionych fragmentów schematu znajdujących się na różnych stronach |  |
| Kontynuacja schematu blokowego |  |
| Droga przepływu danych – więź operacyjna między poszczególnymi operacjami procesu przetwarzania, mogą być skierowane |  |
| Skrzyżowanie dróg przepływu danych bez powiązania – przecięcie więzi operacyjnych nie związanych ze sobą; mogą być skierowane |  |

* Opis dowolnego zadania może być przestawiony jako połączenie następujących bloków:
  + Przetwarzania,
  + Warunkowych,
  + Organizacyjnych.
* Bloki przetwarzania i organizacyjne nazywane są blokami wykonawczymi.
* Blok warunkowy nazywany jest blokiem kierującym lub decyzyjnym.
* Każdej czynności z algorytmu przyporządkowuje się blok opisany ściśle określoną figurą geometryczną, wewnątrz, której umieszcza się definicję tej czynności. Definicja wykorzystuje słowną lub za pomocą symboli postać zapisu danej czynności, np. wzór matematyczny można zapisać za pomocą odpowiednich symboli, takich jak znak całki, sumy.
* Natomiast blok warunkowy umożliwia zapis dowolnego warunku logicznego, od którego wartości zależy wybór jednej z dróg realizacji algorytmu.
* Wejście do bloków oznacza się za pomocą strzałki skierowanej do wnętrza bloku, a wyjście za pomocą strzałki skierowanej na zewnątrz bloku.
* Każdy schemat blokowy musi być spójny tzn., że od bloku typu początek do bloku typu koniec musi prowadzić przynajmniej jedna droga.

V 04.11

1. Elementy języka
   1. Alfabet – zbiór symboli z kodu ASCII (małe i duże litery, operatory arytmetyczne, operatory relacyjne, nawiasy, separatory, znaki specjalne, słowa kluczowe np. for, do, while itd.
   2. Identyfikatory – dowolny ciąg znaków zaczynający się od litery, nie może być słowem kluczowym, jako identyfikatory powinny być dobierane słowa coś mówiące o danej – nazwy powinny być mnemotechniczne.
   3. Zmienne – zmienne przechowują informacje o stanie maszyny cyfrowej (nazwa zmiennej, typ zmiennej, wartość zmiennej, położenie zmiennej w pamięci operacyjnej)
   4. Liczby – w postaci naturalnej (1,2,3) lub postaci wykładniczej/naukowe (1e12, 1.3e-3)
   5. Znaki – pojedynczy znak ASCII w apostrofie ‘a’
   6. Teksty – napisy ujęte w cudzysłowie „aaaa”
   7. Komentarze – komentarze znaczone znakiem /\*…\*/ lub //…, komentarze są ignorowane przez kompilator i nie wpływają na szybkość działania programu, jednak są ważne dla autora programu
   8. Wyrażenia – kombinacja wartości, zmiennych, operatorów, funkcji i nawiasów, obliczana zgodnie z regułami tego języka, który po wykonaniu zwraca wartość. Proces wartościowania nazywamy "wartościowaniem wyrażenia"; potocznie mówi się też, że "wyrażenie ewaluuje do tej wartości".
   9. Słowa kluczowe – wyrazy języka, które mają swoje przeznaczenie i nie mogą być wykorzystane w inny sposób
   10. Typy danych – określenie rodzaju danych
   11. Instrukcje – umożliwiają wykonanie określonych operacji
   12. Procedury, funkcje – grupują instrukcje
   13. Biblioteki – grupują procedury i funkcje
2. Typy danych
   1. Algorytm nie jest sam w sobie programem, do algorytmu7 trzeba dołączyć dane wejściowe lub wyjściowe. Wejściowe dane nadają nowe wartości zmiennym wykorzystywanym w programie. Zmienna może przyjmować dowolne wartości, ale zbiór wartości zmiennej powinien być ustalony w specyfikacji programu. To samo dotyczy operacji jakie można wykonywać na zmiennych dlatego w językach programowania pojęcie typu zmiennej.
   2. Rodzaje typów danych:
      1. Ciągi zakończone stoperem
      2. Ciągi n-elementowe
   3. Powody wprowadzania typów danych
      1. W tworzeniu programu uczestniczy kilka jednostek:
         1. programista – pisząc tekst – typy danych pozwala programiście na logiczną jasność wskazując czym jest dana zmienna i jakie operacje można na niej wykonywać, pozwala to na uniknięcie błędów typu dodawanie samochodów do owoców czy wykonywanie operacji charakterystycznych dla liczb całkowitych na liczbach rzeczywistych
         2. kompilator – tłumacząc tekst na język wewnętrzny – każdy język programowania ma swój repertuar podstawowych instrukcji testujących wartości pewnych zmiennych, w wyniku otrzymujemy wartość logiczną, lub przekształcających wartości zmiennych aby otrzymać nowe wartości, wymaga to jasnej specyfikacji typu danych dla zmiennych uczestniczących w tych operacjach
         3. system operacyjny – wykonując program – podczas wykonywania programu, z różnych powodów, bieżąca wartość zmiennej może być zapamiętywana w kilku komórkach, typ danych pozwala translatorowi na zarezerwowanie niezbędnej liczby komórek z przeznaczeniem na przechowywanie wartości zmiennych oraz określenie jakie procedury mogą służyć do kodowania lub dekodowania wartości
   4. Program oraz dane znajdują się w pamięci operacyjnej komputera, jest to ciąg bitów, bit zawiera 0 lub 1 co odpowiada fizycznemu zjawisku włączenia lub wyłączenia napięcia. 0101011 – taki ciąg nie posiada żadnej struktury, strukturę tworzy się dopiero grupując bity w bajty lub słowa, rozmiar bajtu zależy od maszyny cyfrowej, zwykle jest to 8 bitów, każdy bajt na swój adres, dlatego w językach programowania można się odwoływać do bajtów lub do grup bajtów, jednak sama zawartość pamięci operacyjnej niewiele nam powiem ponieważ może on oznaczać różne rzeczy np. liczbę, ASCII litery/znaku, element tablicy znaków…
   5. Typ obiektu (zmiennej, stałej) to zespół niezmienników czyli własności, które są charakterystyczne dla wartości, jakie ten obiekt może przyjąć. Przykładowo typ int oznacza zakres wartości jako całkowitoliczbowe z zakresu -32767 do +32767 oraz dopuszczalne operacje + - \* / % oraz relacje: >, >= itd.
   6. W wypadku przekroczenia zakresu (np. w dodawaniu) nie następuje warunek błędu. Natomiast typ float (“rzeczywistoliczbowy”) nie dopuszcza operacji % (modulo). W wypadku przekroczenia zakresu (np. w dodawaniu) program przerywa pracę. Typem charakteryzuje się zmienne, stałe, wyrażenia, funkcje, parametry funkcji itd.
   7. Const int liczba = 203 - kompilator wykryje każdą próbę zmiany wartości symbolu i potraktuje jako błąd. Stałe nazywane są stałymi tylko do czytania. Stała symboliczna musi mieć nadaną wartość
   8. Literały – stałe, które oznaczają „zmienną” które jest niezmienna czyli stała, które mają swoje typy:
      1. 7 - literał typu int
      2. ‘v’ - literał typu char
      3. 7L - literał typu long
      4. 7.0 - literał typu float
      5. 7.0L - literał typu double
      6. “dzien dobry” - literał typu char[]
   9. Typy proste:
      1. Znakowe – char (od -128 do +127)(typ całkowity, można go używać w działaniach), unsigned char (od 0 do 255) – przechowuje jeden znak, znaki polskie są kodowane w zakresie: -128 do -1 dla char, czyli 128-255 dla unsigned char
      2. Całkowito-liczbowe – wynikiem operacji na liczbach całkowitych jest liczba całkowita
         1. całkowito-liczbowe ze znakiem
            1. char / signed char: <-128, 127>,
            2. short int / signed short int: <-128, 127>,
            3. int / signed int: <-32768,32767>
            4. long int / signed long int: <-2.147.483.648, 2.147.483.647>,
            5. bool – wielkości logiczne: true (prawda, gdy !=0), false (fałsz, gdy ==0).
         2. całkowito-liczbowe nieujemne
            1. unsigned char: <0, 255>,
            2. unsigned short int: <0, 255>,
            3. unsigned int: <0, 65535>,
            4. unsigned long int: <0,4.294.967.295>.
      3. Rzeczywisto-liczbowe
   10. Zakresy typów:
       1. Zakresy zależą od systemów komputerowych. Można odczytać je w dokumentacji systemowej oraz w pliku limits.h. Przekroczenie zakresu nie jest sygnalizowane.
       2. Limity (zakresy) nazywane:
          1. CHAR\_MIN, CHAR\_MAX, SHRT\_MIN, SHRT\_MAX,
          2. INT\_MIN, INT\_MAX, LONG\_MIN, LONG\_MAX, SCHAR\_MIN, SCHAR\_MAX,
          3. UCHAR\_MAX, USHRT\_MAX, UINT\_MAX, ULONG\_MAX
       3. Liczba bajtów na reprezentacj:
          1. char - 1 bajt,
          2. short - 2 bajty,
          3. int - 2 lub 4 bajty
          4. long int - 4 bajty.
   11. Typy wyliczeniowe umożliwiają nadawanie stałym całkowitymi indywidualnych nazw. Definiowanie typów wyliczeniowych: enum rodzajTypu{lista-elementów-wyliczeniowych} np. enum dnitygodnia {pon, wt, sr, czw, pt sb, nd};

50 slajd

1. Ogólne uwagi
   1. Język C/C++ ma wiele typów standardowych, tj. takich które nie wymagają definiowania, oraz daje możliwości zdefiniowania dowolnego potrzebnego typu.
   2. Każdy typ, standardowy lub zdefiniowany musi mieć określony w sposób jednoznaczny dopuszczalny zbiór wartości (można to zamieścić w specyfikacji programu).