Informatyka Stosowana

Kurs: Język C

Wtorki, 12-14

Sala A-1-06, budynek WFAIS

(III Kampus)

Egzamin: pisemny (test wyboru)

Cel dydaktyczny: Nauka czytelnego stylu programowania z jednoczesnym przedstawieniem wszystkich elementów standardowego języka programowania ANSI C

Wymagania wstępne: Znajomość matematyki (przede wszystkim logiki) na poziomie szkoły średniej

Zagadnienia: deklaracje i wyrażenia, tablice, kwalifikatory, instrukcje sterujące i warunkowe, zmienne, zakres zmiennych, funkcje, zakres funkcji, instrukcje preprocesora, operacje bitowe, wskaźniki, pliki, operacje wejścia/wyjścia buforowane i niebuforowane, wskaźniki, struktury, dynamiczna alokacja pamięci, wybrane algorytmy sortowania, programowanie wykorzystujące język C w środowisku systemu operacyjnego LINUX/UNIX

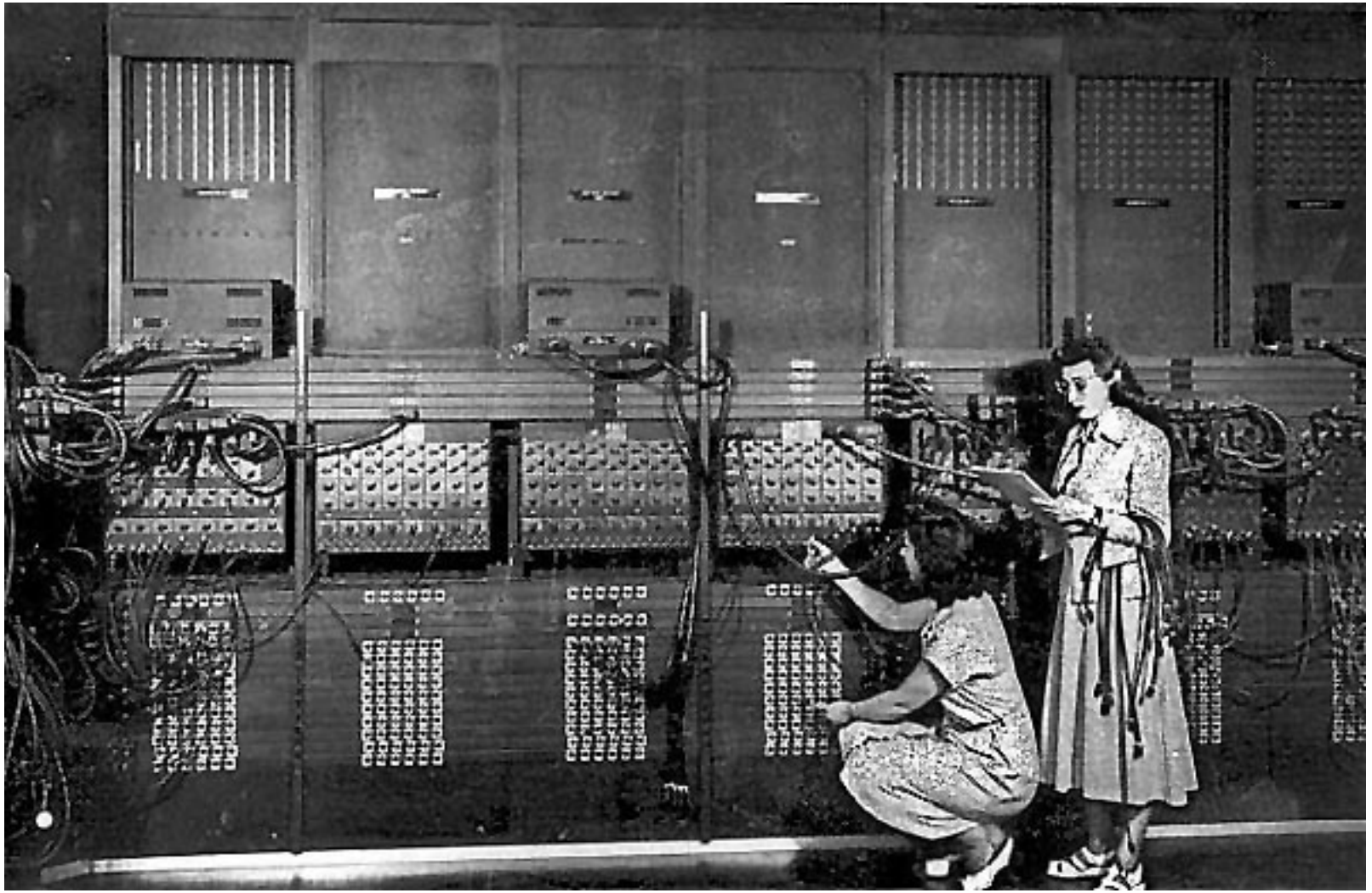
Mamy komputer….

ENIAC (zbudowany 1943 – 1945)

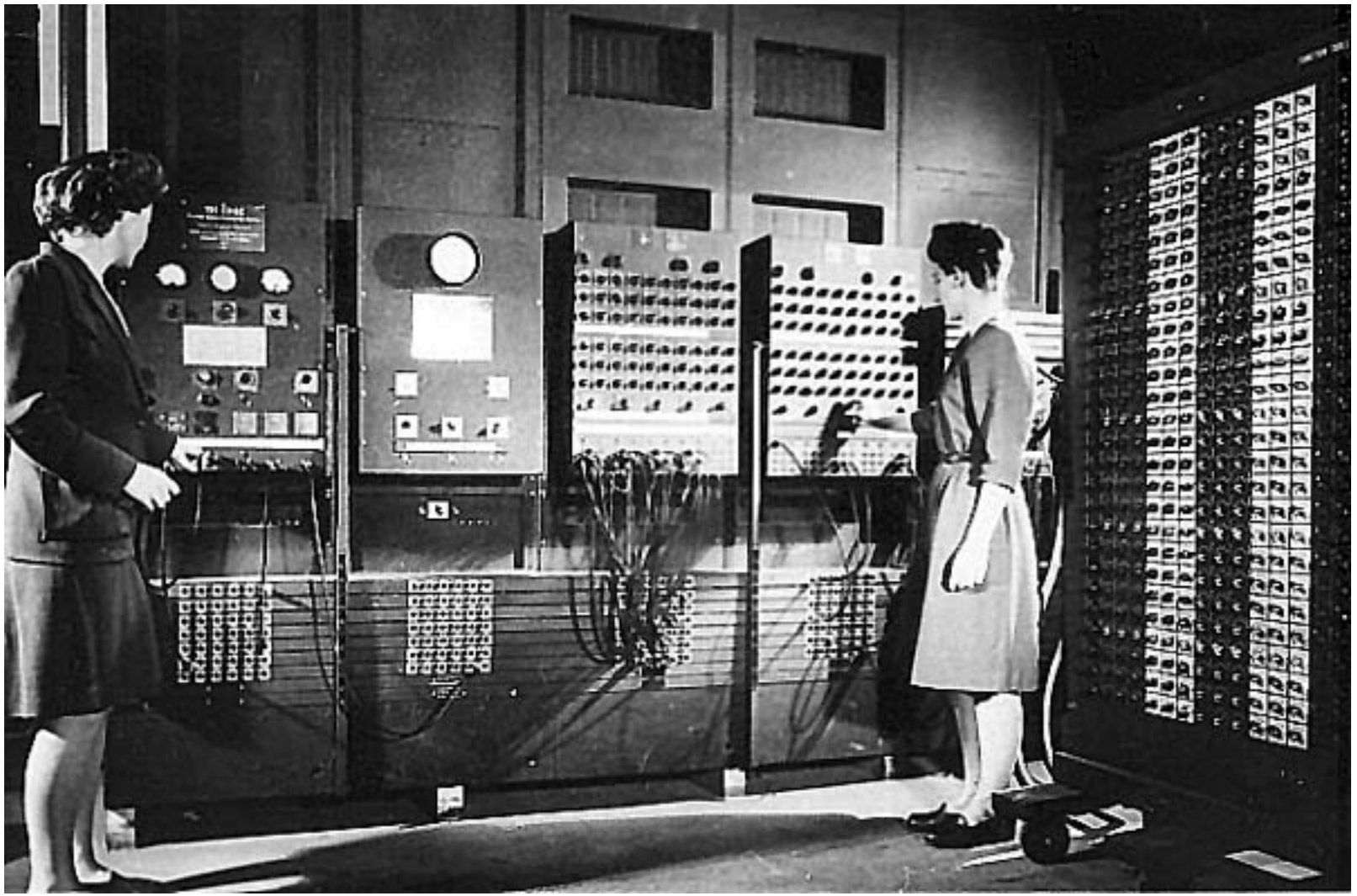
Electronic **N**umerical **I**ntegrator **A**nd **C**omputer

(chyba wcześniej skonstruował działający komputer Konrad Zuse w Niemczech)

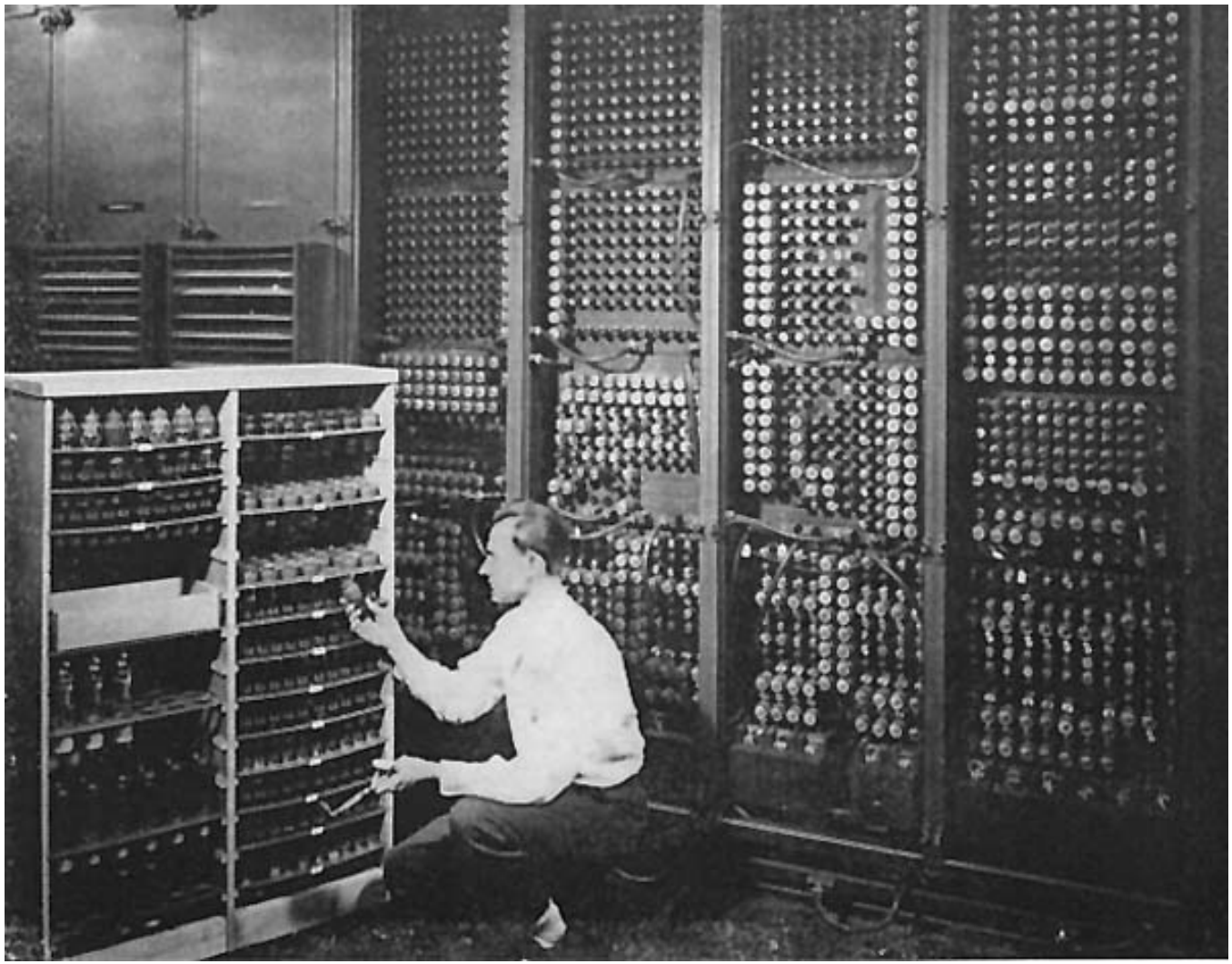
Mamy komputer….



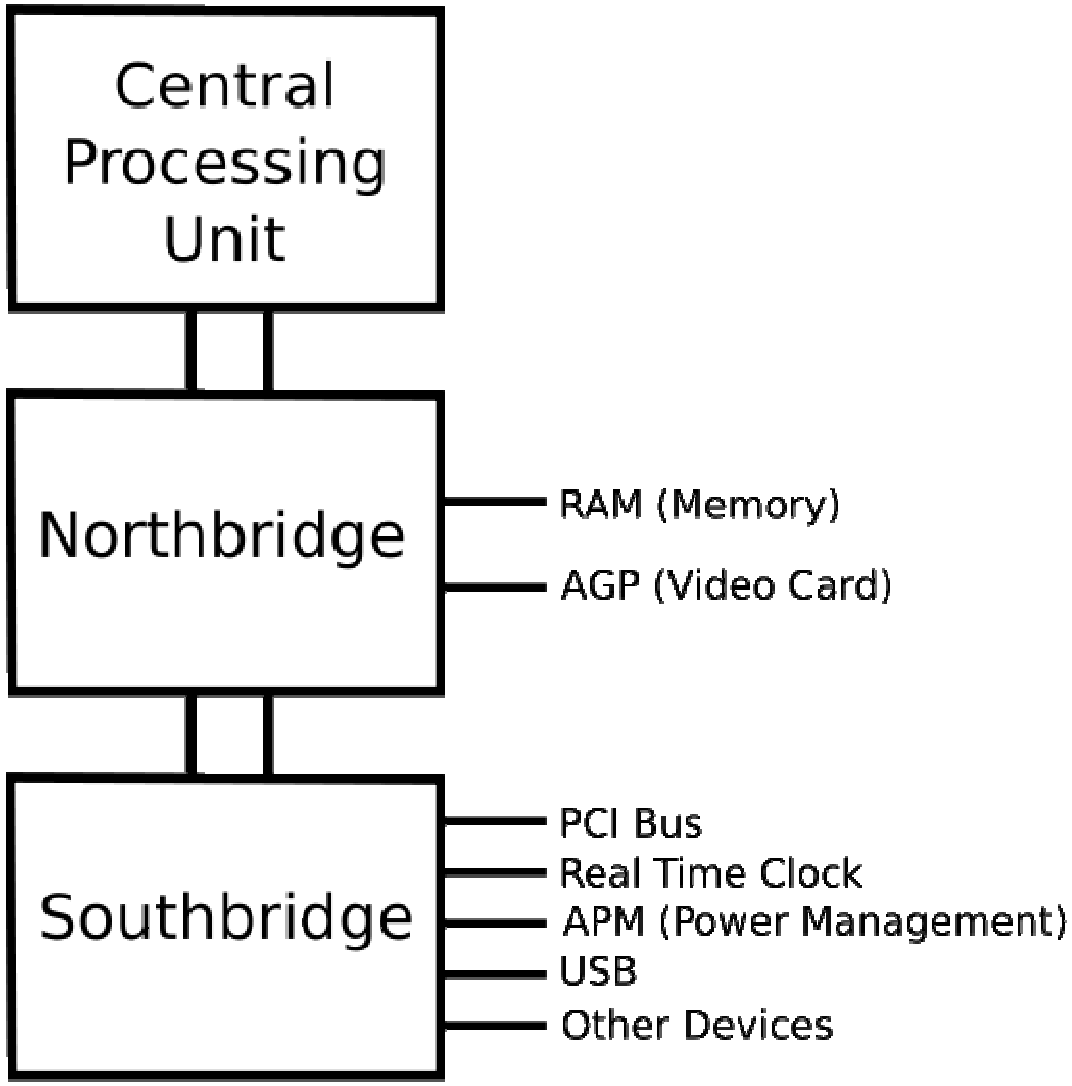
Mamy komputer….



Mamy komputer….



Mamy komputer….

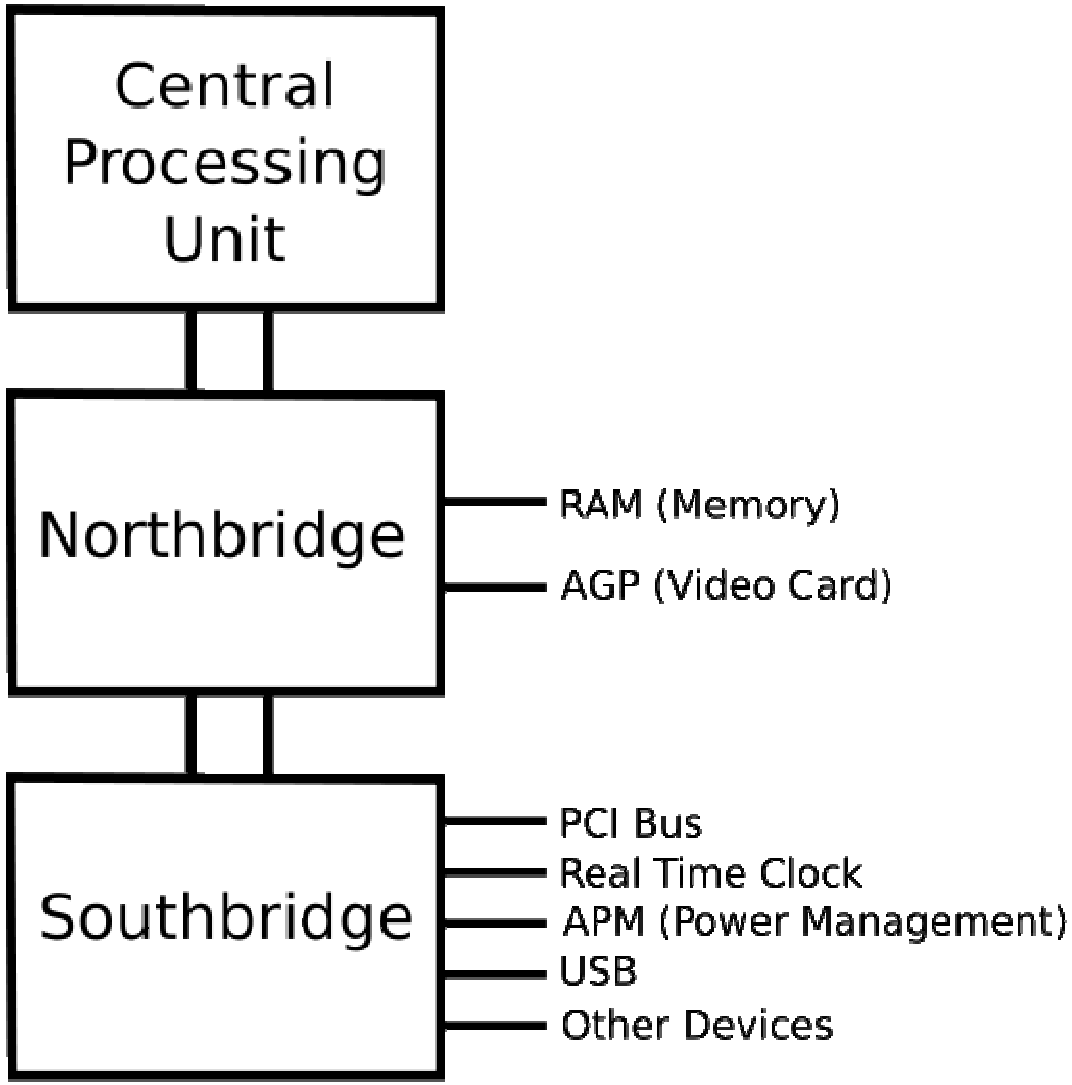


Mamy komputer….

mostek północny (ang. *north bridge*)
• wymiana danych między pamięcią a procesorem, np. sterowanie magistralą AGP i PCI

mostek południowy (ang. *south bridge*)
• współpraca z urządzeniami we/wy, takimi jak dysk, inna pamięć masowa, karty rozszerzeń (PCI), port szeregowy, port równoległy, złącze klawiatury, złącze myszy, modem, interfejs FDD, wyjście audio itp.

….to po co potrzebny jeszcze kompilator?



Informatyka to przechowywanie oraz przetwarzanie informacji

Definicja informacji:

np. w pracy Claude Elwood Shannon (inżynier z Bell Lab)

„A Mathematical Theory of Communication” 1948;
http://plan9.bell-labs.com/cm/ms/what/shannonday/shannon1948.pdf

Jak przechowywanie i przetwarzanie zrealizować?

program i dane (mogą przebywać na tym samym nośniku!)

(program może się zmieniać, dane mogą się zmieniać)

„Cegiełki” z których zbudowany jest program:

algorytmy i struktury danych

Informatyka to przechowywanie oraz przetwarzanie informacji

Algorytmy można budować nie posiadając dostępu do maszyn cyfrowych (czy w ogóle ich nie używając);

-Algorytm rozbijania namiotu, algorytm gotowania ziemniaków…

-Matematyka, zwłaszcza matematyka stosowana oraz nauki przyrodnicze stosują całą masę algorytmów, o ich jakości stanowi zbieżność, uniwersalność zastosowania, koszt stosowania itp..

Definicja algorytmu dla potrzeb bieżącego wykładu: *algorytm to skończony zespół jednoznacznie zdefiniowanych czynności, koniecznych do wykonania postawionego zadania*.

Dlaczego *skończony*?

Dlaczego *zespół*, a nie *ciąg* ?

Informatyka to przechowywanie oraz przetwarzanie informacji

Algorytm a opisujący go język: algorytm można przedstawić w postaci słownej, można w postaci schematu blokowego.

Schemat blokowy ma jeden blok START, co najmniej jeden blok STOP, każda operacja zaznaczona jest jako BLOK, może być jeden lub wiele BLOKÓW WARUNKOWYCH, przy użyciu których opisuje się rozgałęzienia i pętle algorytmu.

Przykłady algorytmów:

Algorytm Euklidesa

Algorytmy sortowania

Algorytmy kompresji

Algorytmy przeszukiwania drzew; itd..

Wieże Hanoi

Tak nazywa się problem, polegający na odbudowaniu z zachowaniem kształtu, wieży zbudowanej z krążków o różnych średnicach, przy czym są utrudnienia: na raz wolno przełożyć tylko jeden krążek; możliwe jest tylko przekładanie krążków znajdujących się najwyżej, przy czym nie wolno w żadnym momencie położyć większego krążka na mniejszy. Wszystkie krążki (całą wieżę) należy przełożyć krążek po krążku z słupka pierwszego **A** na słupek drugi **B**, wykorzystując przy tym pomocniczo słupek trzeci **C**.

Wieże Hanoi

Za Wikipedią:

Zagadka Wież Hanoi stała się znana w XIX wieku dzięki matematykowi o nazwisku Edouard Lucas, który proponował zagadkę dla 8 krążków. Do sprzedawanego zestawu była dołączona (prawdopodobnie wymyślona przez Lucasa) tybetańska legenda, według której mnisi w świątyni Brahmy rozwiązują tę łamigłówkę dla 64 złotych krążków. Legenda mówi, że gdy mnisi zakończą zadanie, nastąpi koniec świata. Zakładając, że wykonują 1 ruch na sekundę, ułożenie wieży zajmie 2**64−1 = 18 446 744 073 709 551 615 (blisko 18 i pół tryliona) sekund, czyli około 584 542 miliardów lat. Dla porównania: Wszechświat ma około 13,7 mld lat.

Wieże Hanoi

Algorytm rekurencyjny (ilość krążków wynosi n)

1. Przenieś (n-1) krążków ze słupka A na słupek C

2. Przenieś jeden krążek (ten największy) ze słupka A na B

3. Przenieś (n-1) krążków ze słupka C na słupek B

Problem ulega uproszczeniu.

A jak dojdziemy do 2 krążków, to rozwiązanie jest trywialne.

Wieże Hanoi – algorytm iteracyjny

Definicja; na lewo od A jest C, na prawo od C jest A (cyklicznie)

0. sprawdź, czy n (ilość krążków) jest parzyste, czy nieparzyste

1. Zidentyfikuj największy krążek którym możesz wykonać poprawny ruch, ruch ten nie może być odwróceniem poprzedniego ruchu.
2. Jeśli z 1. wynika, że to najmniejszy ze wszystkich krążek, to jeśli liczba n jest parzysta przełóż go w lewo, a jeśli nieparzysta, to w prawo.
3. Powtarzaj 1. oraz 2. tak długo, aż wieża z krążków z położenia A przejdzie w położenie B.

Jak sprawdzić, czy algorytm działa?

Najlepiej go użyć i sprawdzić to doświadczalnie!

(potrzebna jest maszyna matematyczna, czyli komputer oraz kompilator)

Dygresja o początkach budowy komputerów

Jeśli przyjąć definicję, że komputer to maszyna zdolna do wykonywania dowolnych ciągów instrukcji oraz operowania na danych, to pierwszy komputer powstawał ~1830 roku.

Charles Babbage (1791-1871), maszyna różniczkowa („differential engine”). Komputer odtworzono ~1990, przy użyciu tolerancji toczenia (obrabiarki) jak w czasach autora. Działa, jest przechowywany w London Science Museum. Instalacja posiada także drukarkę (autorem projektu jest również Charles Babbage).

Ada Lovelace (córka poety, lorda Byrona); pracowała nad algorytmami dla powstającej maszyny różnicowej i dla drukarki; jeden z języków programowania nazwano ADA w uznaniu jej zasług jako pierwszej osoby piszącej programy komputerowe.

Można znaleźć darmowy kompilator C pracujacy w systemie WINDOWS

http://www.bloodshed.net

kompilator Dev-C++

http://prdownloads.sourceforge.net/dev-cpp/devcpp-4.9.9.2_setup.exe

Można znaleźć darmowy kompilator C pracujacy w systemie WINDOWS

Orwell Dev-C++

http://sourceforge.net/projects/orwelldevcpp/

(wersje 5.4.2, 5.5.0)

Krótka historia języka C

- system operacyjny UNIX ~1969 DEC PDP-7, w assemblerze
- język BCPL wspomagający programowanie systemowe
- język B, kolejna wersja BCPL, ~1970
- zupełnie nowy język C jako następca B, około 1971
- ~1973, system operacyjny UNIX przepisany w języku C

(więcej na stronie cm.bell-labs.com/cm/cs/who/dmr/chist.html
czy http://csapp.cs.cmu.edu/3e/docs/chistory.html

Charakterystyka języka C

- Mały rozmiar kompilatora
- Język strukturalny
- Programowanie niskiego poziomu (low level, bitwise)
- Implementacja wskaźników – wskaźniki do pamięci, macierzy, struktur, funkcji
- Ma jednocześnie cechy języka wysokiego poziomu

Charakterystyka języka C

- Daje efektywnie działające programy
- Kompilatory C są dostępne na praktycznie wszystkich typach komputerów

Podręcznik

- Steve Oualline ”Język C”, seria O’Reilly®,
  polskie wydanie wydawnictwo HELION
- Kernighan, Ritchie ”ANSI C”
- K. N. King „Język C. Nowoczesne programowanie.
  Wydanie II”

ftp://ftp.helion.pl/przyklady/cprpro.zip
(tu są przykładowe programy)

Styl programowania

- Optymalizacja: pamięci, czasu wykonania programu, kosztów opracowania programu, kosztów konserwacji programu
- Dąż do prostoty
- Program w trakcie opracowania jest podstawą do zmian
- Nie nadużywaj zmiennych globalnych

Styl programowania

- Używaj nawiasów w złożonych wyrażeniach
- Zamieszczaj dużo komentarzy
- Nie stosuj skomplikowanych warunków
- Korzystaj z funkcji bibliotecznych
- Używaj dobrze dobranych nazw zmiennych
- Zostaw oddzielny wiersz dla nawiasów klamrowych

Styl programowania

- Używaj stałych symbolicznych

  #define Dwa 2

  #define cisnienie P
- Staraj się, aby funkcje były małe
- Stosuj wcięcia w tekście dla uwypuklenia znaczenia
- Wyróżniaj koniec funkcji
- Nie nadużywaj instrukcji goto

Styl programowania

- if(i)
- if(i!=0) mają to samo znaczenie!
- if((i!=0)!=0)

Styl programowania

```
#include <stdio.h>
#include <stdlib.h>

#define   DIE   \
{fprintf(stderr, "Blad krytyczny: Zatrzymanie\n");exit(8);}

int main() {
/* wartosc losowa dla celow testowych */
int value;
value = 1; if (value < 0) DIE;
printf("To jeszcze nie koniec\n");
return (0);
} /*  koniec funkcji main()   */
```

```
#include <stdio.h> main(t,_,a) char *a; {return!0<t?t<3?main(-79,-
13,a+main(-87,1-_, main(-86, 0, a+1 )+a)):1,t<_?main(t+1, _, a ):3,main (
-94, -27+t, a )&&t == 2 ?_<13 ?main ( 2, _+1, "%s %d %d\n"
):9:16:t<0?t<-72?main(_,
t,"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l,+,/n{n+\
,/+#n+,/#;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l q#'+d'K#!/\
+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){)#}w'){){nl]'/+#n';d}rw' i;# ){n\ l]!/n{n#';
r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#\ n'wk nw' iwk{KK{nl]!/w{%'l##w#' i;
:{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \ ;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;\
#'rdq#w! nr'/ ') }+}{rl#'{n' ')# }'+}##(!!/") :t<-50?_==*a
?putchar(a[31]):main(-65,_,a+1):main((*a == '/')+t,_,a\ +1 ):0<t?main ( 2,
2 , "%s"):*a=='/'||main(0,main(-61,*a, "!ek;dc \ i@bK'(q)-
[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

”It will compile and run and produce meaningful output.”

The International Obfuscated C Code Contest

http://www.ioccc.org/

komentarz

/* to jest komentarz, jego długość może być dowolnie długa */

Słowa kluczowe

- Następujące słowa, zwane słowami kluczowymi, są zastrzeżone dla kompilatora języka C i **nie mogą być używane** jako nazwy zmiennych, funkcji lub etykiety:

Słowa kluczowe

auto	else	int	typedef
break	entry	long	switch
case	enum	register	union
char	extern	return	unsigned
continue	float	short	void
default	for	sizeof	while
do	goto	static	
double	if	struct	

standard ANSI: const signed volatile

```
#include <stdio.h>
#include <stdlib.h>
int main()
{
    printf("    Witaj swiecie\n");
    /*  printf(”    Hello, world\n”);  */
    return (0);
}
```

```
#-----------------------------------------#
#       Makefile for unix systems         #
#    using a GNU C compiler               #
#-----------------------------------------#
CC=gcc
CFLAGS=-g -Wall -D__USE_FIXED_PROTOTYPES__ -ansi
# Compiler flags:
#       -g      -- Enable debugging
#       -Wall   -- Turn on all warnings
#       -D__USE_FIXED_PROTOTYPES__
#               -- Force the compiler to use the correct headers
#       -ansi   -- Don't use GNU extensions.  Stick to ANSI
target1:
            $(CC) koral.c   $(CFLAGS)   -o    koral.exe
```

```
#-----------------------------------------#
#       Makefile for unix systems         #
#    using a GNU C compiler               #
#-----------------------------------------#
CC=gcc
CFLAGS=-g -Wall -D__USE_FIXED_PROTOTYPES__ -ansi  -pedantic
# Compiler flags:
#       -g      -- Enable debugging
#       -Wall   -- Turn on all warnings
#       -D__USE_FIXED_PROTOTYPES__
#               -- Force the compiler to use the correct headers
#       -ansi   -- Don't use GNU extensions.  Stick to ANSI
target1:
            $(CC) koral.c   $(CFLAGS)   -o    koral.exe
```

Operatory-1

Łączne lewostronnie

()	wywołanie funkcji	getc(stdin)
[]	odwołanie do elementu tablicy	i[17]
->	wskaźnik do elementu struktury	alt_ptr->c
	odwołanie do elementu struktury	alt.c

Operatory-2

Łączne prawostronnie

-	minus jednoargumentowy	-i
++	inkrementacja	i++
--	dekrementacja	i—
!	negacja logiczna	!znaleziony
~	uzupełnienie do 1, negacja bitowa	~0x7f
*	wskazanie pośrednie, dereferencja	*c_ptr
&	adres elementu	&i
sizeof	rozmiar zmiennej lub typu	sizeof(j)
(typ)	zmiana typu	(int)c

Operatory-3

łączne lewostronnie

*	mnożenie	i*j
/	dzielenie	i/j
%	operacja modulo	i%j

Operatory-4

łączne lewostronnie

+	dodawanie	i+j
-	odejmowanie	i-j

Operatory-5

łączne lewostronnie

<<	przesunięcie bitowe w lewo	i<<2
>>	przesunięcie bitowe w prawo	i>>2

Operatory-6

łączne lewostronnie

<	mniejszy niż	i<j
<=	mniejszy lub równy	i<=6
>	większy niż	i>2
>=	większy lub równy	i>=j

Operatory-7

łączne lewostronnie

==	równy	if(i==5)
!=	nierówny	if(i!=5)

Operatory-8

łączne lewostronnie

&	bitowy iloczyn logiczny	c & 033

Operatory-9

łączne lewostronnie

^	bitowa suma modulo 2	c ^ 0317

Operatory-10

łączne lewostronnie

| | bitowa suma logiczna | c | 0333

Operatory-11

łączne lewostronnie

&&	iloczyn logiczny	i == 5 && j == 6

Operatory-12

łączne lewostronnie

||	suma logiczna	i == 5 || j == 6

Operatory-13

łączne prawostronnie

?:	wyrażenie warunkowe	i>4 ? i : j

Użycie np. a = i>4 ? i : j

a = (i>4 ? i : j)

Operatory-14a

łączne prawostronnie

=	przypisanie	i = 7
*=	mnożenie, potem przypisanie	i*=3
/=	dzielenie, potem przypisanie	i/=4
%=	modulo, potem przypisanie	i%=4
+=	dodawanie,potem przypisanie	i+=5
-=	odejmowanie,potem przypisanie	i-=5
&=	iloczyn bitowy, potem przypisane	i &= 0333
^=	suma mod 2,potem przypisanie	i ^= 0317

Operatory-14b

|=	suma bitowa potem przypisanie	i |= 0177
<<=	przesunięcie w lewo, potem przypisanie	i <<= 2
>>=	przesunięcie w prawo, potem przypisanie	i >>= 3

Operatory-15

łączny lewostronnie

,	operator przecinkowy	printf(”Hello”),n=7;

Operatory-podsumowanie

- mnożenie i dzielenie przed dodawaniem i odejmowaniem
- kolejność innych operatorów najlepiej wymuszać przez umieszczanie w nawiasach okrągłych

Znaki specjalne

\b	backspace (cofacz)
\f	wysuw strony
\n	nowy wiersz
\r	powrót (początek aktualnego wiersza)
\t	tabulator

Znaki specjalne

\’	apostrof
\”	podwójny cudzysłów
\\	lewy ukośnik
\nnn	numer znaku (w zapisie ósemkowym)

wyrażenie

- nazwa zmiennej
- wywołanie funkcji
- nazwa tablicy
- stała
- nazwa funkcji
- odwołanie do elementu struktury
- odwołanie do elementu tablicy
- jedna z powyższych postaci z nawiasami i/lub operatorami

Instrukcja

wyrażenie zakończone średnikiem
jest **instrukcją**

Stałe

typ	**składnia**	**przykład**	
char	ujęte w apostrofy		‘a’
ciąg znaków	ujęte w cudzysłowy		”abc”
int	bez zera na początku		17
l.ósemkowa			015
l. szesnastkowa			0x2f
long int		123l	123L
float		3.24	3.2e7
double		3.2e7	3.2E7

Nazwy (identyfikatory)

Nazwy zmiennych i funkcji są to dowolne ciągi liter i cyfr – zaczynają się od litery. Można użyć znaku podkreślenia, niektóre kompilatory nie akceptują go na początku ciągu. Litery małe są uważane za różne od dużych.

Definicje danych

Typ	**Definicja**	**Typowy rozmiar**	
char	zmienna znakowa	8 bitów(bajt)	
short int	liczba całk. krótka	2 bajty	
int	liczba całkowita	4 bajty	!?
long int	liczba całk. długa	4 bajty	

Definicje danych

Typ	**Definicja**	**Typowy rozmiar**
float	liczba zmiennoprzecinkowa	4 bajty
double	liczba zmiennoprzecinkowa	8 bajtów
unsigned char	liczba znakowa bez znaku	1 bajt
unsigned int	liczba całkowita bez znaku	4 bajty ?!

Definicje danych

tablica wielokrotne wystąpienie identycznych zmiennych **tab[17]**

struktura, unia, wskaźnik, dane wyliczeniowe (enum),

ciąg znaków

Dodatkowe typy danych można definiować korzystając z instrukcji **typedef**

funkcja

klasa_pamięci typ_wyniku nazwa_funkcji(lista_arg_form)

lista_argumentów deklarowanych

{

definicje i deklaracje zmiennych

instrukcje

}

klasa_pamięci: **extern** (domyślna) **static**

funkcja - przykład

```
static  int  func(i, j, d, c)
double d;
char c;
int i,j;
{
        int k;
        k = 1;
        return( d*k);                    return( (int) d*k );
}
```

funkcja - przykład

```
static  int  func(int i, int j, double d, char c)
{
        int k;
        k = 1;
        return( d*k);           /*      return( (int) d*k );   */
}
```

podstawowa struktura programu

deklaracje danych, funkcje oraz komentarze

```
/**************************************
        Komentarz                    **
**************************************/
....deklaracje danych...
int main()
{
 ......instrukcje programu...
      return();
}
```

pętla while

```
while  (wyrażenie)  instrukcja
  while(i<k)
  {
        func_jeden(++i);
        func_dwa(++j);
   }
```

pętla for

```
for(wyrażenie1;  wyrażenie2;  wyrażenie3) instrukcja
```

```
 wyrażenie1;
 while (wyrażenie2)
 {
        instrukcja
        wyrażenie3
 }
```

```
n=0;
 while (n<=7)
 {
  a[n]=n;
   ++n;
  }
```

pętla do-while

```
do  instrukcja  while wyrażenie ;

        do
                {
                        funkcja1(n++);
                        funkcja2(++k);
                }  while (k<j) ;
```