1. test z PJC Jméno studenta:_____

Pocet bodu: ----

Varianta 37: příklad 1 (5 bodů)

Určete hodnotu výrazu: a=1,2,3

(a) 1 (b) 2 (c) 3

Varianta 37: příklad 2 (5 bodů)

Jednoduchá implementace datového typu zásobník se opírá o pole celých čísel, v jehož nultém prvku se pamatuje počet položek na zásobníku. Samotný zásobník pak zaplňuje další buňky pole (od první buňky dále). Jaký výraz použijete k zařazení nového záznamu na vrchol zásobníku?

```
void push(int *stack, int value)
{
  ...
}
```

(a) stack[++*stack] = value; (c) stack[*stack++] = value;
(b) stack[1 + ++*stack] = value; (d) stack[++stack] = value;

Varianta 37: příklad 3 (5 bodů)

Jaká bude po skončení programu hodnota proměnné z?

```
int x = 3, *y = &x, z = 5;
x += z++;
z += x + ++*y;
```

(a) 23 (b) 24 (c) 18 (d) nedefinovaná

Varianta 37: příklad 4 (5 bodů)

Uvažme následující deklaraci:

```
struct tnode { int x:4; unsigned y:6; };
```

Tato deklarace popisuje:
(a) typ struktury tnode s položkami x a y s počátečními hodnotami 4 a 6
(b) typ struktury tnode s položkami x a y s délkou 4 a 6
(c) uživatelsky typ tnode s položkami x a y s počátečními hodnotami 4 a 6
(d) uživatelsky typ tnode s položkami x a y s délkou 4 a 6

Varianta 37: příklad 5 (5 bodů)

Jakou funkci má následující procedura?

```
void c(char *s)
{
  while(*s)
  {
    if(*s >= 'a' && *s <= 'z')
      *s -= 'a' - 'A';
      s++;
  }
}
```

(a) Odstraní z řetězce písmena.
(b) Z písmen v řetězci udělá čísla.
(c) Z malých písmen udělá velká.
(d) Prohodí velká a malá písmena.
(e) V programu je chyba, a proto zřejmě skončí v nekonečné smyčce.

Varianta 37: příklad 6 (5 bodů)

Vytvořte program, který vypíše převodní tabulku mezi stupni Fahrenheita a Celsia. Tabulka bude mít tvar:

```
50 F = 10.0 C
55 F = 12.8 C
60 F = 15.6 C
...
100 F = 37.8 C
```

Na přírůstek 9 F připadá přírůstek 5 C.

Varianta 37: příklad 7 (5 bodů)

Funkce for_each zavolá zadanou funkci s jedním číselným parametrem pro všechna čísla ze zadaného pole. Jak bude vypadat hlavička této funkce za předpokladu, ze tělo funkce vypadá takto:

```
void for_each(...)
{
  int i;
  for(i = 0; i < n; i++)
    fn(*pole++);
}
```

(a) void for_each(const int pole[], int n, void (*fn)(int));
(b) void for_each(const int *pole, int n, void fn(int));
(c) void for_each(const int *pole, int n, void (*fn)(int));
(d) void for_each(const int *pole[], int n, void (*fn)(int));

Varianta 37: příklad 8 (5 bodů)

```
#define MAX2(X,Y) X > Y ? X : Y
#define MAX3(X,Y,Z) MAX2(X,MAX2(Y,Z))
int a = 8,b = 7, c = 9, d;
d = MAX3 ( a,++b,--c);
```

V proměnné d bude hodnota:
(a) 7 (b) 8 (c) 9 (d) 10

Varianta 37: příklad 9 (10 bodů)

Je definováno pole 'int a[100]'. Pole obsahuje neseřazené hodnoty. Napište úryvek programu, který pole seřadí Vzestupně..

1. test z PJC Jméno studenta:_____

Pocet bodu: ----

Varianta 57: příklad 1 (5 bodů)

```
int a = 2, b = 3, c;
c = --a + b;
b = c || b / --a;
```

V proměnné b bude:
(a) 1 (b) 0 (c) chyba, dělení 0 (d) nedefinovaná hodnota

Varianta 57: příklad 2 (5 bodů)

Standardní funkce memmove zkopíruje daný počet bytu ze zadané paměťové adresy na jinou adresu. Kopírování překrývajících se bloku je korektně ošetřeno. Jak použijete funkci memmove k vytvoření funkce vymaz, která ze zadaného pole vymaže první prvek (posune zbylé prvky pole na jeho místo)?

```
void vymaz(int *pole, int počet)
{
  if(počet > 0)
    memmove(...)
}
```

(a) memmove(pole, pole + 1, počet);
(b) memmove(pole, pole + 1, sizeof(int) * počet);
(c) memmove(pole, pole + 1, sizeof(int) * (počet - 1));
(d) memmove(pole, pole + sizeof(int), sizeof(int) * (počet - 1));

Varianta 57: příklad 3 (5 bodů)

Jaká je hodnota následujícího výrazu, pokud int a=4, int b=1?

```
b+=2, a/3 << a >> 2
```

(a) 3 (b) syntakticky špatně (c) 4 (d) 7

Varianta 57: příklad 4 (5 bodů)

Uvažme následující deklaraci:

```
struct tnode { int x:4; unsigned :6; int z:4; };
```

Tato deklarace popisuje:
(a) typ struktury tnode s položkami x a z s počátečními hodnotami 4 a 6
(b) uživatelsky typ tnode s položkami x a z s počátečními hodnotami 4 a 6
(c) typ struktury tnode s položkami x a z s délkou 4 a 4 a dírou 6 bitu
(d) chybně zapsaná deklarace struktury tnode

Varianta 57: příklad 5 (5 bodů)

Standardní knihovna <string.h> definuje funkci strcpy, která zkopíruje řetězec a vrátí ukazatel na začátek cílového retezce. Která z následujících implementací této funkce je správná?

(a)
```
char *strcpy
  (char *dest, const char *src)
  {
    char *ptr = dest;
    while(*ptr++ = *src++)
      ;
    return ptr;
  }
```

(c)
```
char *strcpy
  (char *dest, const char *src)
  {
    char *ptr = dest;
    while(*src)
      *ptr++ = *src++;
    return dest;
  }
```

(b)
```
char *strcpy
  (char *dest, const char *src)
  {
    char *ptr = dest;
    while(*ptr++ == *src++)
      ;
    return dest;
  }
```

(d)
```
char *strcpy
  (char *dest, const char *src)
  {
    char *ptr = dest;
    while(*dest++ = *src++)
      ;
    return ptr;
  }
```

Varianta 57: příklad 6 (5 bodů)

Implementujte standardní knihovní funkci strdup, která zkopíruje zadaný řetězec, ukončený znakem '\0', do paměťového místa, které dynamicky alokuje pomocí funkce malloc. Funkce malloc naalokuje zadaný počet bytů a vrátí netypový ukazatel na začátek naalokované paměťové oblasti, nebo NULL, pokud dojde k chybě (nedostatek paměti). Ke kopírování řetězce můžete využít buď knihovní funkci memcpy, která zkopíruje zadaný počet bytů z paměťové adresy src na adresu dest, nebo knihovní funkci strcpy, která zkopíruje rovnou celý řetězec ukončený znakem '\0'. Co bude rychlejší, strcpy, nebo memcpy? Která varianta vytvoří kratší kód programu? Proč?
Navrhněte a implementujte vhodné ošetření chybového stavu - nedostatek paměti. Pokud je funkce vyvolána s parametrem NULL, nic neprovede a vrátí NULL.

```
char *strdup(const char *src);

/* můžete využít */
void *malloc(unsigned počet_bytu);
void *memcpy(void *dest, const void *src, unsigned počet_bytu);
char *strcpy(char *dest, const char *src);
```

Varianta 57: příklad 7 (5 bodů)

Funkce for_each zavolá zadanou funkci s jedním číselným parametrem pro všechna čísla ze zadaného pole. Jak bude vypadat hlavička této funkce za předpokladu, ze tělo funkce vypadá takto:

```
void for_each(...)
{
  int i;
  for(i = 0; i < n; i++)
    fn(*pole++);
}
```

(a) void for_each(const int pole[], int n, void (*fn)(int));
(b) void for_each(const int *pole, int n, void fn(int));
(c) void for_each(const int *pole, int n, void (*fn)(int));
(d) void for_each(const int *pole[], int n, void (*fn)(int));

Varianta 57: příklad 8 (5 bodů)

Zapište fragment hlavičkového souboru ctype.h, který zajisti, aby nevadilo jeho opakované vkládání.

Varianta 57: příklad 9 (10 bodů)

Napište program, který převádí stupně Fahrenheita na stupně Celsia. (F=1.8*C+32)

1. test z PJC Jméno studenta:_____

Pocet bodu: ----

Varianta 68: příklad 1 (5 bodů)

Jaká hodnota bude v proměnné a po proběhnutí programu?

```
int a, b;
b = 7;
a = (b++ - 2), b;
```

(a) 5 (b) 6 (c) 7 (d) 8

Varianta 68: příklad 2 (5 bodů)

V implementaci počítačové hry TETRIS reprezentujete herní plochu (šachtu) polem 10 x 20 buněk typu char, které obsahují informace o dříve spadlých tvarech. Jak použijete standardní funkci memmove, která zkopíruje zadaný počet bytu ze zadané paměťové adresy na jinou adresu (s korektním ošetřením překrývajících se polí), k implementaci funkce zaplněna, která odstraní n-tou řádku (posune pole směrem dolu) při zaplnění řádky?

```
void zaplněna(char tetris[20][10], int radek)
{
  memmove(...);
  memset(tetris, 0, 10); // vyprazdni horni radek
}
```

(a) memmove(tetris + 10, tetris, (radek - 1) * sizeof(*tetris));
(b) memmove(tetris + 1, tetris, (radek - 1) * sizeof(*tetris));
(c) memmove(tetris + radek + 1, tetris + radek,
(20 - radek) * sizeof(*tetris));
(d) memmove(tetris + radek, tetris, (radek - 1) * sizeof(*tetris));

Varianta 68: příklad 3 (5 bodů)

Muze kompilator jazyka C rozlisovat velikosti pismen?

(a) nemuze (b) podle nastaveni uzivatele
(c) zalezi na implementaci (d) musi podle normy

Varianta 68: příklad 4 (5 bodů)

Uvažme následující deklaraci:

```
struct tnode { int x:4; unsigned :6; int z:4; };
```

Tato deklarace popisuje:
(a) typ struktury tnode s položkami x a z s počátečními hodnotami 4 a 6
(b) uživatelsky typ tnode s položkami x a z s počátečními hodnotami 4 a 6
(c) typ struktury tnode s položkami x a z s délkou 4 a 4 a dírou 6 bitu
(d) chybně zapsaná deklarace struktury tnode

Varianta 68: příklad 5 (5 bodů)

Standardní knihovní funkce strcmp porovná dva retezce lexikograficky podle ASCII kódu jejich znaku. Vrátí záporné číslo, pokud první řetězec je v ASCII abecedě před druhým řetězcem, nulu v případe identické shody obou řetězců a kladné číslo v případe, ze první řetězec následuje v ASCII abecedě za druhým řetězcem.. Je tato implementace správná?

```
int strcmp(const char *s1, const char *s2)
{
  while(*s1 && *s1 == *s2)
    s1++, s2++;
  return *s1 - *s2;
}
```

(a) Implementace je chybná - přepíše první řetězec druhým.
(b) Implementace nefunguje pro prázdný řetězec v parametru s2
(c) Implementace funguje Správně..
(d) Implementace nefunguje, pokud je první řetězec kratší než druhý.

Varianta 68: příklad 6 (5 bodů)

Implementujte standardní knihovní funkci strrev, která obrátí řetězec ukončený znakem '\0' (prohodí první znak s posledním, druhý s předposledním atd.) a vrátí ukazatel na začátek tohoto řetězce. Korektně ošetřete případ, kdy zadaný řetězec je prázdný.

char *strrev(char *str);

Varianta 68: příklad 7 (5 bodů)

Jak byste deklarovali funkci integral, která pro zadaný funkcionál f (ukazatel na reálnou funkci jedné reálné proměnné) a zadaný interval x_min až x_max vypočte (numericky) určitý integrál této funkce pres zadaný interval?

(a) double integral(double f(double x), double x_min, double x_max);
(b) double integral(double *f(double x), double x_min, double x_max);
(c) double integral(double (*f)(double x), double x_min, double x_max);
(d) double integral(double *(*f)(double x), double x_min, double x_max);

Varianta 68: příklad 8 (5 bodů)

```
#define MIN(A,B) (A) < (B) ? (A) : (B)
int a=4, b=0, c;
c = !MIN(a,b) ? 4 : 5;
```

V proměnné a bude:
(a) 4 (b) nic, syntaktická chyba (c) 5 (d) nedefinovaná hodnota

---

Varianta 68: příklad 9 (10 bodů)

Napište úryvek programu, který vytiskne hexadecimálně hodnotu proměnné 'int number;'. Program nesmi použit konverzi %x, %X funkce printf, sprintf apod.

1. test z PJC Jméno studenta:_____

Pocet bodu: ----

Varianta 83: příklad 1 (5 bodů)

Tri z uvedených výrazu mají stejný výsledek (předpokládejte rozumné deklarace). Označte výraz, který dává jiný výsledek.

(a) *(**p).polozka (c) *p -> polozka[0]
(b) p[0] -> polozka[0] (d) *(*p) -> polozka

Varianta 83: příklad 2 (5 bodů)

Standardní funkce memmove zkopíruje daný počet bytů ze zadané paměťové adresy na jinou adresu. Kopírování překrývajících se bloků je korektně ošetřeno. Jak použijete funkci memmove k vytvoření funkce vymaz, která ze zadaného pole vymaže první prvek (posune zbylé prvky pole na jeho místo)?

```
void vymaz(int *pole, int počet)
{
  if(počet > 0)
    memmove(...)
}
```

(a) memmove(pole, pole + 1, počet);
(b) memmove(pole, pole + 1, sizeof(int) * počet);
(c) memmove(pole, pole + 1, sizeof(int) * (počet - 1));
(d) memmove(pole, pole + sizeof(int), sizeof(int) * (počet - 1));

Varianta 83: příklad 3 (5 bodů)

Kolik je hodnota následujícího výrazu, pokud x = 3 a y = 1?

```
9 / 1 << x - y
```

(a) 71 (b) 36 (c) 2 (d) 1

Varianta 83: příklad 4 (5 bodů)

Zapište deklaraci struktury hilo, která umožni využit nejnižší 2 a nejvyšší dva bity slabiky (bytu) jako malá cela čísla.

Varianta 83: příklad 5 (5 bodů)

Standardní knihovní funkce strstr hledá v řetězci podřetězec. Předvedená implementace k tomu využívá funkci memcmp, která porovná zadaný počet znaků na dvojici zadaných paměťových adres. Jaká je časová náročnost této funkce v nejhorším případě?

```
int strstr(const char *str, const char *substr)
{
  int len = strlen(str);
  int sublen = strlen(substr);
  const char *p, *e;
  if(sublen > len)
    return -1;
  for(e = (p = str) + (len - sublen); p <= e; p++)
    if(!memcmp(p, substr, sublen))
      return p - str;
  return -1;
}
```

(a) O(m.n) řádu součinu délky řetězce a podřetězce
(b) O(m) řádu délky řetězce
(c) O(n) řádu délky podřetězce
(d) O(log m) řádu logaritmu délky řetězce

Varianta 83: příklad 6 (5 bodů)

Implementujte standardní knihovní funkci strcpy, která zkopíruje řetězec src (ukončený znakem '\0') do paměti na adresu dest a vrátí ukazatel na začátek tohoto cílového řetězce dest. Neošetřujte případné přetečení cílového pole (předpokládejte, že volající funkce ví, kolik místa pro řetězec vyhradila a jak dlouhý řetězec do něj tedy smí zkopírovat).

```
char *strcpy(char *dest, const char *src);
```

Varianta 83: příklad 7 (5 bodů)

V programu používáte pro razení číselných hodnot strom. Strom je vytvořen tak, ze každý jeho uzel má v levém podstromu všechny uzly s menšími hodnotami, v pravém podstromu všechny uzly s vyššími hodnotami. Jak bude vypadat funkce pro tisk seřazeného pole?

```
struct node
{
  struct node *left, *right;
  int         value;
};

void tisk(const struct node *n)
{
  if(!n)
    return;
  ...
}
```

(a) tisk(n -> left);
    printf("%d\n", n -> value);
    tisk(n -> right);

(b) tisk(n -> left);
    tisk(n -> right);
    printf("%d\n", n -> value);

(c) printf("%d\n", n.value);
    tisk(n.left);
    tisk(n.right);

(d) tisk(n.left);
    printf("%d\n", n.value);
    tisk(n.right);

Varianta 83: příklad 8 (5 bodů)

```
#define MAX2(X,Y) X > Y ? X : Y
#define MAX3(X,Y,Z) MAX2(X,MAX2(Y,Z))
int a = 8,b = 7, c = 9, d;
d = MAX3 ( a,++b,--c);
```

V proměnné d bude hodnota:
(a) 7 (b) 8 (c) 9 (d) 10

Varianta 83: příklad 9 (10 bodů)

Je definováno pole 'int a[100]'. Pole obsahuje neseřazené hodnoty. Napište úryvek programu, který pole seřadí Vzestupně..

1. test z PJC Jméno studenta:_____

Pocet bodu: ----

Varianta 91: příklad 1 (5 bodů)

Jaké budou hodnoty proměnných po provedení následujícího programu:

```
int i = 5, j = i++;
i += j++; j += i++;
```

(a) i = 13, j = 20 (c) i = 10, j = 15
(b) i = 12, j = 17 (d) i = 11, j = 17

Varianta 91: příklad 2 (5 bodů)

Následující program má za cíl naplnit pole a hodnotou jeho prvního prvku. V programu je ovsem chyba. Opravte tuto chybu co nejmenším zásahem (tj. přepsáním, smazáním, resp. vložením co nejmenšího poctu znaku).

```
int a[10], *i;

for(i = a; i < a + 10;)
  *++i = *a;
```

Varianta 91: příklad 3 (5 bodů)

Napište výslednou hodnotu výrazů, jsou-li dány definice int a = 2, b = 2, c = 1, d = 4.

a) -a-- - - --d
b) a++ / ++c * --d
c) --b * c++ - a
d) -b - --c

Varianta 91: příklad 4 (5 bodů)

Zapište deklaraci struktury midd, která umožni využit prostředni 4 bity spodní slabiky celého čísla jako čísla bez znaménka.

Varianta 91: příklad 5 (5 bodů)

Standardní knihovní funkce strdup zkopíruje zadaný řetězec do paměťového místa, které dynamicky naalokuje pomocí funkce malloc. Jaké výrazy je třeba dosadit za vyraz1, vyraz2, aby funkce pracovala správně? Funkce memcpy zkopíruje zadaný počet bytu z jedné paměťové adresy na druhou.

```
char *strdup(const char *s)
{
  int len = strlen(s);
  char *result = malloc(vyraz1);
  if(result)
    memcpy(result, s, vyraz2);
  return result;
}
```

(a) vyraz1 = len, vyraz2 = len
(b) vyraz1 = len, vyraz2 = len + 1
(c) vyraz1 = len + 1, vyraz2 = len
(d) vyraz1 = len + 1, vyraz2 = len + 1

Varianta 91: příklad 6 (5 bodů)

Implementujte standardní knihovní funkci strdup, která zkopíruje zadaný řetězec, ukončený znakem '\0', do paměťového místa, které dynamicky alokuje pomocí funkce malloc. Funkce malloc naalokuje zadaný počet bytu a vrátí netypový ukazatel na začátek naalokované paměťové oblasti, nebo NULL, pokud dojde k chybě (nedostatek paměti). Ke kopírování retezce můžete využít buď knihovní funkci memcpy, která zkopíruje zadaný počet bytu z paměťové adresy src na adresu dest, nebo knihovní funkci strcpy, která zkopíruje rovnou celý řetězec ukončený znakem '\0'. Co bude rychlejší, strcpy, nebo memcpy? Která varianta vytvoří kratší kód programu? Proč?
Navrhnete a implementujte vhodné ošetření chybového stavu - nedostatek Paměti.. Pokud je funkce vyvolána s parametrem NULL, nic neprovede a vrátí NULL.

```
char *strdup(const char *src);

/* můžete využít */
void *malloc(unsigned počet_bytu);
void *memcpy(void *dest, const void *src, unsigned počet_bytu);
char *strcpy(char *dest, const char *src);
```

Varianta 91: příklad 7 (5 bodů)

Funkce for_each zavolá zadanou funkci s jedním číselným parametrem pro všechna čísla ze zadaného pole. Jak bude vypadat hlavička této funkce za předpokladu, že tělo funkce vypadá takto:

```
void for_each(...)
{
  int i;
  for(i = 0; i < n; i++)
    fn(*pole++);
}
```

(a) void for_each(const int pole[], int n, void (*fn)(int));
(b) void for_each(const int *pole, int n, void fn(int));
(c) void for_each(const int *pole, int n, void (*fn)(int));
(d) void for_each(const int *pole[], int n, void (*fn)(int));

Varianta 91: příklad 8 (5 bodů)

```
#define MAX  10000 ;
int a = 10000, b, c;
b = MAX + MAX - 40;
c = a < b ? 20 : 30;
```

V proměnné c bude:
(a) nic, syntaktická chyba (b) 30 (c) 20 (d) nedefinovaná hodnota

---

Varianta 91: příklad 9 (10 bodů)

Napište úryvek programu, který vytiskne počet znaku, slov a řádků obsažených v řetězci 'char *str;' Slova jsou oddělena jednim nebo více znaky ' ', '\t', '\n', řádky jsou odděleny znaky '\n'.

1. test z PJC Jméno studenta:_____

Pocet bodu: ----

Varianta 105: příklad 1 (5 bodů)

Určete hodnotu výrazu: a=1,2,3

(a) 1 (b) 2 (c) 3

Varianta 105: příklad 2 (5 bodů)

Jakými hodnotami vyplní následující funkce zadané pole:

```
void funkce(int *pole, int maxpocet)
{
  int i;
        int *p, *dest = pole, *limit = pole + maxpocet;
        for(i = 2; dest < limit; i++)
        {
          for(p = pole; p < dest && i % *p; p++)
            ;
          if(p == dest)
            *dest++ = i;
        }
}
```

(a) Funkce uvízne v nekonečné smyčce.
(b) Funkce zaplní pole prvními maxpocet prvočísly.
(c) Funkce zaplní pole hodnotami 2, 3, ... maxpocet + 1.
(d) Funkce zaplní pole lichými čísly 3, 5, ... 2 * maxpocet + 1

Varianta 105: příklad 3 (5 bodů)

Jaká bude hodnota proměnné i po skončení programu ?

```
int i=2;
i=i++,++i,i++,++i,i++;
```

(a) 5 (b) 6 (c) 7 (d) 3

Varianta 105: příklad 4 (5 bodů)

Uvažme následující deklaraci:

```
struct tnode { int x:4; unsigned y:6; };
```

Tato deklarace popisuje:
(a) typ struktury tnode s položkami x a y s počátečními hodnotami 4 a 6
(b) typ struktury tnode s položkami x a y s délkou 4 a 6
(c) uživatelsky typ tnode s položkami x a y s počátečními hodnotami 4 a 6
(d) uživatelsky typ tnode s položkami x a y s délkou 4 a 6

Varianta 105: příklad 5 (5 bodů)

Co provede se zadaným řetězcem následující funkce:

```
void funkce(char *s)
{
  if(*s)
  {
    char *p = s, *q = s, t;
    while(*++p)
      (*p < p[-1] ? (t = *p, *p = *(q = p - 1), *q = t) : 0);
    while(q > s)
    {
      char *a = s;
      for(p = s; ++p <= q;)
        (*p < p[-1] ? (t = *p, *p = *(a = p - 1), *a = t) : 0);
      q = a;
    }
  }
}
```

(a) seřadí znaky v řetězci sestupně podle jejich ASCII kódu
(b) obrátí řetězec (od posledního znaku k prvnímu)
(c) seřadí znaky v řetězci vzestupně podle jejich ASCII kódu
(d) přepíše celý řetězec tím znakem původního retezce, který má nejnižší ASCII kód

---
Varianta 105: příklad 6 (5 bodů)

Implementujte standardní knihovní funkci strcat, která připojí zadaný
řetězec src (ukončený znakem '\0') na konec zadaného cílového retezce
dest (také ukončeného znakem '\0') a vrátí ukazatel na začátek
cílového retezce dest. Neošetřujte případné přetečení cílového pole
(předpokládejte, ze volající funkce ví, kolik místa pro cílový
řetězec vyhradila, a jak dlouhé retezce tedy smí spojovat).

char *strcat(char *dest, const char *src);

Varianta 105: příklad 7 (5 bodů)

Jak byste deklarovali funkci integral, která pro zadaný funkcionál f
(ukazatel na reálnou funkci jedné reálné proměnné) a zadaný interval
x_min až x_max vypočte (numericky) určitý integrál této funkce pres
zadaný interval?

(a) double integral(double f(double x), double x_min, double x_max);
(b) double integral(double *f(double x), double x_min, double x_max);
(c) double integral(double (*f)(double x), double x_min, double x_max);
(d) double integral(double *(*f)(double x), double x_min, double x_max);

Varianta 105: příklad 8 (5 bodů)

Napište makro SWAP, které prohodí obsahy
dvou proměnných.

Varianta 105: příklad 9 (10 bodů)

Je definováno pole 'int a[100]'. Pole obsahuje neseřazené hodnoty. Napište
úryvek programu, který pole seřadí Vzestupně..